

IKapC 用户开发手册

(V1.4.5.3 2023.02.11)



合肥埃科光电科技股份有限公司

<http://www.i-tek.cn/>

历史版本

版本	日期	更新内容描述
1.0.0.1	2017/07/02	<ul style="list-style-type: none"> ● 初始版本
1.0.1.1	2017/07/14	<ul style="list-style-type: none"> ● 增加函数接口获取 Camera Link 相机特征 ● 增加函数接口获取 Camera Link 相机特征的轮询时间
1.0.2.1	2017/07/28	<ul style="list-style-type: none"> ● 增加配置 GigEVision 相机的功能
1.0.3.1	2017/08/02	<ul style="list-style-type: none"> ● 增加采集图像、缓存管理模块
1.0.4.1	2017/09/20	<ul style="list-style-type: none"> ● 增加图像显示模块 ● 增加 Bayer 图像转换功能
1.0.5.1	2017/11/10	<ul style="list-style-type: none"> ● 增加函数接口获取相机特征可视权限 ● 增加函数接口获取相机特征表示方式
1.0.6.1	2017/11/29	<ul style="list-style-type: none"> ● 增加函数接口进行图像缓冲区的拷贝、保存、载入、读写等操作
1.1.1.4	2018/08/01	<ul style="list-style-type: none"> ● 增加 GigEVision 相机功能：ForceIP
1.1.1.5	2018/09/20	<ul style="list-style-type: none"> ● 增加接口：ItkViewRemoveAllBuffer()
1.2.0.1	2019/01/09	<ul style="list-style-type: none"> ● 增加配置 CoaXPress 相机的功能
1.3.4.0	2020/06/15	<ul style="list-style-type: none"> ● 增加对 USB3.0 相机的支持
1.4.3.1	2021/05/14	<ul style="list-style-type: none"> ● 增加 IKapC 函数
1.4.4.0	2021/07/21	<ul style="list-style-type: none"> ● 增加对 PCIe 相机的支持 ● 增加 IKapC 函数解析
1.4.5.0	2022/04/01	<ul style="list-style-type: none"> ● 修改文档格式
1.4.5.1	2022/04/25	<ul style="list-style-type: none"> ● 更新公司 logo
1.4.5.2	2022/07/20	<ul style="list-style-type: none"> ● 增加回调事件： ITKSTREAM_VAL_EVENT_TYPE_FRAME_LOST ● 增加函数接口： ItkManGetPrm(), ItkManSetPrm(), ItkBufferDenoise() ● 更新参数列表、错误处理和数据类型定义
1.4.5.3	2023/02/11	<ul style="list-style-type: none"> ● 更新公司地址

联系方式

警告

版权所有 © 2023 合肥埃科光电科技股份有限公司

本用户开发手册由合肥埃科光电科技股份有限公司编印，版权所有，翻版必究。

声明

本用户开发手册适用于合肥埃科光电科技股份有限公司面阵/线阵相机和 Vulcan 系列采集卡。在使用相机或者采集卡前，请仔细阅读本手册，并妥善保管，以便备用。合肥埃科光电科技股份有限公司保留对本手册中的打印错误、与最新资料不一致、软件升级及产品改进等解释权及随时进行改动的权利。这些更改恕不另行通知，将直接编入新版手册中。

合肥埃科光电科技股份有限公司

电话：+86-551-65318597

传真：+86-551-65318597

网址：www.i-tek.cn

地址：安徽省合肥市高新区望江西路中安创谷科技园二期 J2 栋 3F

邮编：230088

目 录

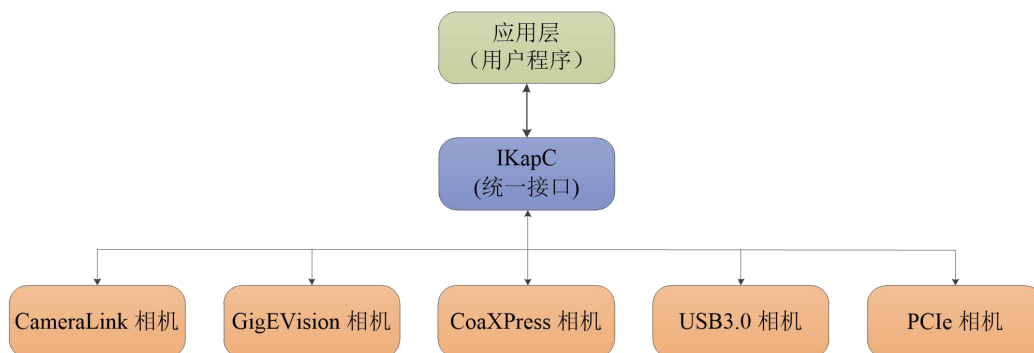
历史版本	2
联系方式	3
目 录	4
1 IKapC 概述	6
2 开始使用 IKapC	7
2.1 主要组成	7
2.2 适用对象	7
2.3 一般约定	7
2.4 开发环境	7
3 IKapC 工作流程	8
3.1 整体工作流程图	8
3.2 特征控制流程图	9
3.3 采集控制流程图	10
3.4 掉线事件获取流程图	11
4 创建 IKapC 应用程序	12
4.1 头文件和库文件	12
4.2 创建应用程序	12
5 使用千兆网相机调试程序	13
6 IKapC 使用流程	14
6.1 初始化 IKapC 运行环境	14
6.2 打开/关闭相机设备	14
6.2.1 枚举设备	14
6.2.2 打开/关闭相机设备	15
6.3 配置相机	15
6.3.1 获取相机特征的名称	15
6.3.2 收集访问特征所需要的资源	16
6.3.3 获取相机特征的访问权限	16
6.3.4 获取相机特征的数据类型	16
6.3.5 读取/配置相机特征	18
6.3.6 释放相机特征的资源	19
6.4 采集图像	19
6.4.1 初始化图像数据流资源	20
6.4.2 添加数据缓冲区	21
6.4.3 单帧图像采集	21
6.4.4 多帧图像采集	22
6.4.5 连续图像采集	22
6.4.6 释放图像数据流资源	24
6.5 缓存管理	24
6.5.1 图像缓冲区的状态	24
6.5.2 图像数据流的传输模式	25
6.6 显示图像	26
6.6.1 创建显示图像的窗口	26

6.6.2	添加/移除图像缓冲区	27
6.6.3	设置参数	28
6.6.4	显示图像	30
6.6.5	释放显示图像的窗口资源	31
6.7	获取相机掉线信息	31
7	API 函数说明	32
7.1	函数概览	32
7.2	函数说明	36
8	参数列表	125
8.1	全局参数	125
8.2	采集卡参数	125
8.3	设备参数	126
8.4	数据流参数	127
8.5	缓冲区参数	131
8.6	图像显示窗口参数	138
8.7	事件信息参数	142
9	错误处理	144
9.1	模块 ID	144
9.2	错误类别 ID	144
9.3	错误码 ID	144
10	数据类型定义	147
10.1	宏定义	147
10.2	结构体定义	147
10.3	句柄定义	148
10.4	类型定义	148
10.5	回调函数定义	149

1 IKapC 概述

IKapC 是合肥埃科光电科技股份有限公司针对面阵/线阵相机开发的 C 语言应用程序库。通过使用 IKapC，用户可以方便地管理和控制相机设备、配置相机参数，从而简洁快速地进行高性能机器视觉应用的设计、开发和部署。IKapC 支持合肥埃科光电科技股份有限公司生产的所有 Camera Link(CL) / Gigabit Ethernet(GigE) / CoaXPress(CXP) / USB3.0(U30) / PCIe 接口相机。

IKapC 兼容 GenICam 协议标准，为合肥埃科光电科技股份有限公司生产的工业相机定义了 XML 文件并提供了统一的访问接口，其基本框架如下图所示：



2 开始使用 IKapC

2.1 主要组成

- 创建 IKapC 应用程序
- 初始化 IKapC 运行环境
- 打开/关闭相机设备
- 配置相机
- 采集图像
- 缓存管理
- 显示图像
- 获取相机掉线信息

2.2 适用对象

本文档主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

2.3 一般约定

文件名、目录以及网站以粗体文字表示（例如：**IKapCViewer.exe**, **<http://www.i-tek.cn>**）。

函数参数以斜体文字表示（例如：*nHeight*）。

源码，示例代码以及文件列表以字符边框包含（例如：时间戳）。

2.4 开发环境

对于 32 位开发环境，支持下列编译平台：

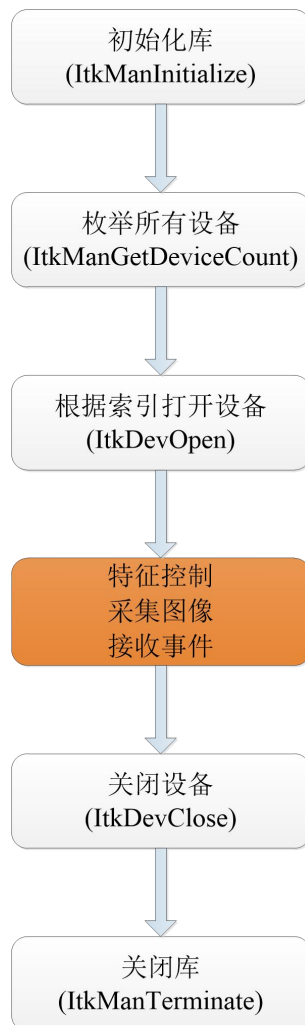
- Microsoft Visual C++ 2008 (with Service Pack 1)
- Microsoft Visual C++ 2010 (with Service Pack 1)
- Microsoft Visual C++ 2012
- Microsoft Visual C++ 2013
- Microsoft Visual C++ 2015

对于 64 位开发环境，支持下列编译平台：

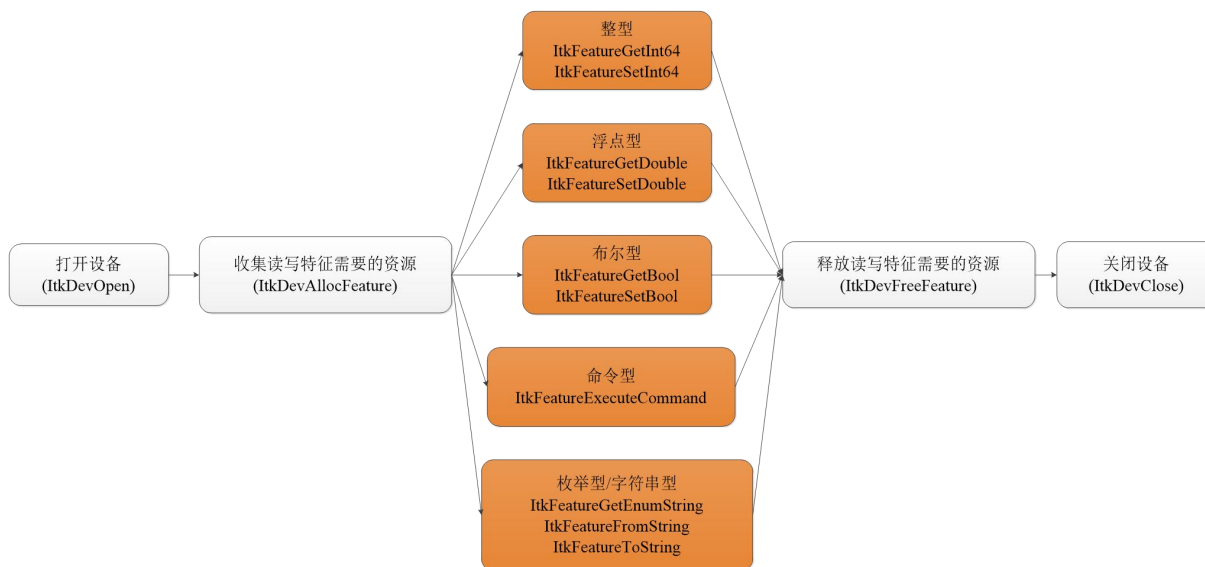
- Microsoft Visual C++ 2008 (with Service Pack 1)
- Microsoft Visual C++ 2010 (with Service Pack 1)
- Microsoft Visual C++ 2012
- Microsoft Visual C++ 2013
- Microsoft Visual C++ 2015

3 IKapC 工作流程

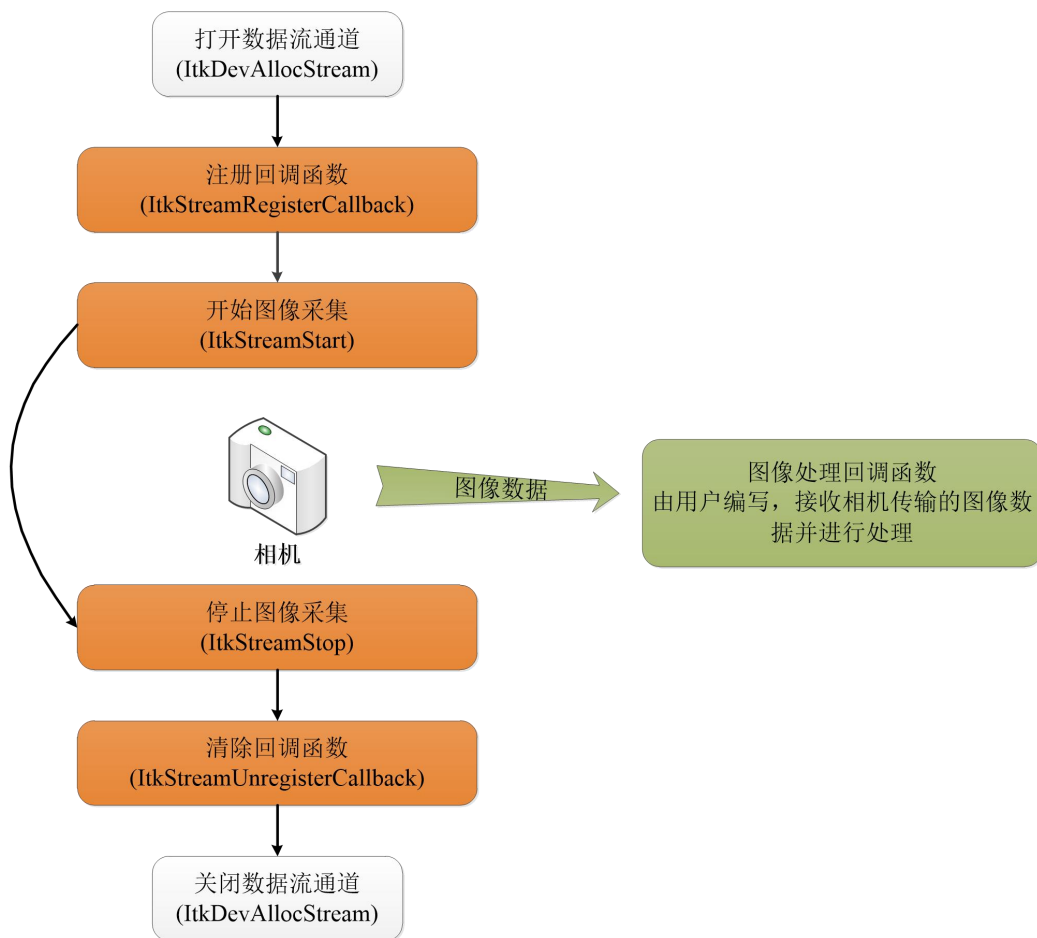
3.1 整体工作流程图



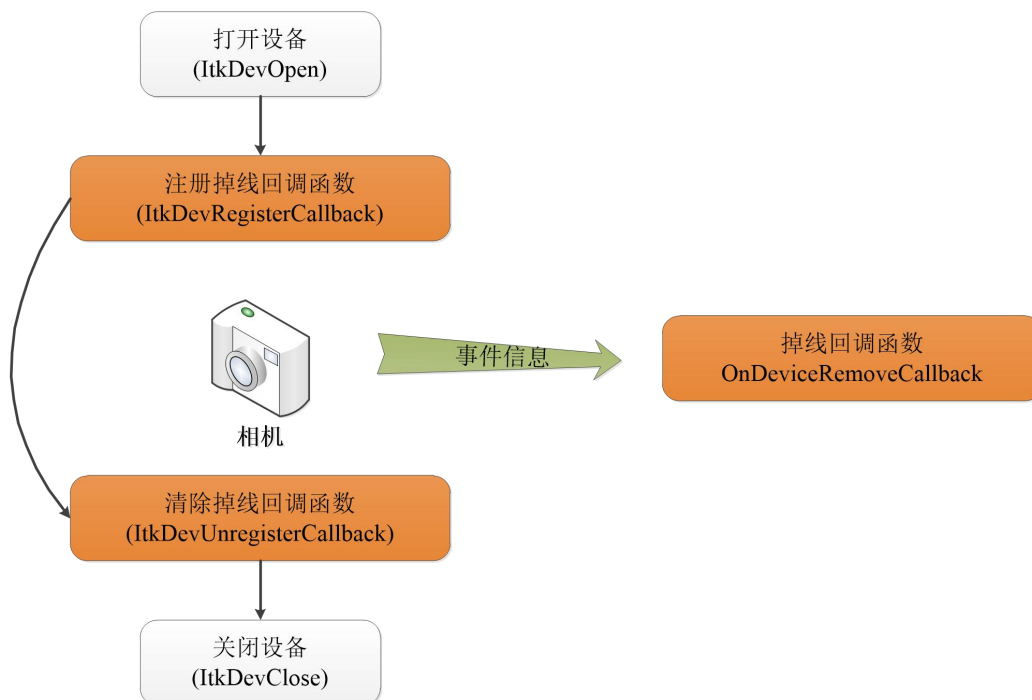
3.2 特征控制流程图



3.3 采集控制流程图



3.4 掉线事件获取流程图



4 创建 IKapC 应用程序

4.1 头文件和库文件

创建 IKapC 应用程序所需的头文件和库文件在安装 IKapCViewer 时会自动安装在目标计算机上。用户可以在 IKapCViewer 软件的安装位置找到相应的文件，具体如下表所示：

文件名	描述	位置
IKapC.h	Basic header file	\ I-TEK OptoElectronics \ IKapLibrary \ Include
IKapC.lib	x64 basic library for all Visual C++ version	\ I-TEK OptoElectronics \ IKapLibrary \ Lib \ x64
IKapC.lib	x86 basic library for all Visual C++ version	\ I-TEK OptoElectronics \ IKapLibrary \ Lib \ x86
IKapC.dll	32bit system basic DLL for Visual C++ 2008 and up	<windir> \ System32
IKapC.dll	64bit system basic DLL for Visual C++ 2008 and up	<windir> \ SysWOW64

4.2 创建应用程序

下面步骤描述了如何在 VS2010 创建一个使用 IKapC 的 C 应用程序：

- 在程序源代码中包含 **IKapC.h**（它会包含所有需要的头文件）
- 添加 **\$(ITEK_ROOT)\Include** 或 **\$(IKAPC_ROOT)\Include** 到 项目 - 属性 - 配置属性 - C/C++ - 常规 - 附加包含目录
- 如果是 x64 应用程序，添加 **\$(ITEK_ROOT) \ Lib \ x64 \ IKapC.lib** 或 **\$(IKAPC_ROOT)\Lib \ x64 \ IKapC.lib** 到 项目 - 添加现有项
- 如果是 x86 应用程序，添加 **\$(ITEK_ROOT) \ Lib \ x86 \ IKapC.lib** 或 **\$(IKAPC_ROOT)\Lib \ x86 \ IKapC.lib** 到 项目 - 添加现有项
- 项目 - 属性 - 配置属性 - C/C++ - 代码生成 - 运行库 选择 多线程 DLL(/MD) 或者 多线程调试 DLL(/MDd)

5 使用千兆网相机调试程序

Windows 用户通过 Visual Studio 开发平台在 Debug 模式下调试千兆网相机的时候可能会遇到由于心跳超时而导致设备掉线的情况。应用程序必须以固定的时间间隔发送心跳包到设备，如果设备没有接收到心跳包，则会认为当前连接已经断开，从而不会再接收从应用程序发送的任何命令。

当用户正常运行应用程序时，底层库会正常发送心跳包，保持和设备的连接状态。但是当用户在应用程序中设置断点调试时，程序运行到这些断点，调试器会暂停所有线程，包括发送心跳包的线程。所以当用户在 Debug 模式下单步调试代码时，底层库不会发送心跳包到设备。

可以通过增加心跳超时时间的方式来解决这个问题。在调试状态下，用户可以在打开设备后添加如下代码：

示例代码

```
// hDevice 为设备句柄，已经通过 ItkDevOpen()接口打开设备
// 设置心跳包超时时间为 5 分钟
uint32_t timeout = 300 * 1000;
ItkDevSerPrm(hDevice, ITKDEV_PRM_HEARTBEAT_TIMEOUT, &timeout);
```

6 IKapC 使用流程

6.1 初始化 IKapC 运行环境

用户在使用 IKapC 函数前，必须先调用 **ItkManInitialize()** 初始化 IKapC 的运行环境，并根据函数返回值确定是否成功初始化。在同一个进程中多次初始化 IKapC 环境不会影响 IKapC 的正常工作。

用户在结束访问 IKapC 库函数前，应该调用 **ItkManTerminate()** 释放 IKapC 运行环境收集的资源。注意在释放 IKapC 运行时环境后，除非再次初始化，否则无法调用其他 IKapC 库函数。

下列代码片段演示了如何初始化 IKapC 运行时环境。

示例代码

```
// 初始化 IKapC 运行环境
ITKSTATUS res = ItkManInitialize();
if (res == ITKSTATUS_OK)
{
    // 此处打开配置相机设备、采集图像等
}
ItkManTerminate();
```

6.2 打开/关闭相机设备

6.2.1 枚举设备

设备的枚举由两步组成：第一步，调用 **ItkManGetDeviceCount()** 函数，其返回参数 *pCount* 标识了当前电脑上连接的相机设备的数量。假设当前共有 N 个设备连接到电脑上，则用户可以访问索引为 [0, N-1] 范围内的相机设备。第二步，调用 **ItkManGetDeviceInfo()** 函数获取相机的基本信息，相机的基本信息将会存储在 **ITKDEV_INFO** 结构体中，用户可以通过该结构体中的字段区分不同种类和接口的相机。

下列代码演示了枚举相机设备的过程。

示例代码

```
// 初始化 IKapC 运行时环境
ITKSTATUS res = ItkManInitialize();
Check(res);
uint32_t count = 0;
res = ItkManGetDeviceCount(&count);
Check(res);
for (uint32_t i=0; i < count; ++i)
{
    ITKDEVICE_INFO di;
    res = ItkManGetDeviceInfo(i, &di);
    Check(res);
    // 此处处理获取的相机信息
}
ItkManTerminate();
```

6.2.2 打开/关闭相机设备

在控制相机参数和采集图像前，用户必须先打开设备。用户通过调用 **ItkDevOpen()** 打开相机设备，并根据返回值确定是否成功打开设备。用户可以选择访问设备的方式，例如独占式访问模式（仅允许一个进程访问相机设备）、只读式访问模式（仅允许从设备中读取信息而不允许写入信息）。关于设备访问模式的详细信息，参见 **ItkDevOpen()** 函数说明。

当用户希望关闭相机设备时，可以调用 **ItkDevClose()** 函数来关闭设备。注意当设备被关闭后，原来指向相机的设备句柄 **ITKDEVICE** 将不能再被其他函数调用。

注意：打开相机设备前，必须先调用 **ItkManGetDeviceCount()** 枚举当前电脑上可用相机的数量。

下列代码演示了打开/关闭相机设备的过程。

示例代码

```
// 初始化 IKapC 运行环境
ITKSTATUS res = ItkManInitialize();
Check(res);
ITKDEVICE hDev;    // 相机设备句柄
res = ItkDevOpen(0, ITKDEV_VAL_ACCESS_MODE_CONTROL, &hDev);
Check(res);
// 此处使用相机句柄配置相机参数、采集图像数据等
ItkDevClose(hDev);
ItkManTerminate();
```

6.3 配置相机

本节介绍了如何配置相机参数。在本手册中，我们将相机的每个可以独立配置的参数称为相机的特征，下文如无特别说明，我们将以特征指代相机的具体配置参数。

相机的所有配置信息都存储在 XML 文件中，用户无需关心 XML 文件存储的位置和格式，仅需要知道相机特征的名称就可以配置相机。当用户希望读取/配置相机特征时，通常需要遵循如下步骤：

- [获取相机特征的名称](#)
- [收集访问特征所需要的资源](#)
- [获取相机特征的访问权限](#)
- [获取相机特征的数据类型](#)
- [读取/配置相机特征](#)
- [释放相机特征的资源](#)

6.3.1 获取相机特征的名称

用户既可以查阅相机的使用手册直接获取其支持配置特征的参数名称，也可以通过 IKapC 的函数枚举设备所有支持配置特征的名称。调用 **ItkDevGetFeatureCount()** 可以获取当前打开设备支持配置特征的数量，假设相机支持配置 N 个特征，则用户可以访问范围在 [0, N-1] 范围内的相机特征。调用 **ItkDevGetFeatureName()** 可以获取指定索引的特征名称，使用该名称可以进一步控制相机特征。

下列代码演示了如何获取相机特征的名称。

示例代码

```
// 初始化 IKapC 运行环境
ITKSTATUS res = ItkManInitialize();
Check(res);
ITKDEVICE hDev;    // 相机设备句柄
res = ItkDevOpen(0, ITKDEV_VAL_ACCESS_MODE_CONTROL, &hDev);
Check(res);
// 此处使用相机句柄配置相机参数、采集图像数据等
uint32_t count;
res = ItkDevGetFeatureCount(hDev, &count);
for (uint32_t i=0; i < count; ++i)
{
    uint32_t nameSize = 0;
    res = ItkDevGetFeatureName(hDev, i, NULL, &nameSize);
    char *name = malloc(nameSize);
    res = ItkDevGetFeatureName(hDev, i, name, &nameSize);
    Check(res);
    free(name);
}
ItkDevClose(hDev);
ItkManTerminate();
```

6.3.2 收集访问特征所需要的资源

当用户知道其希望访问的特征的名称后，就可以收集配置该特征需要的资源，从而读取/配置该特征信息。所有的相机特征均有相机特征句柄 **ITKFEATURE** 统一管理。调用 **ItkDevAllocFeature()**根据特征名称返回特征句柄。

6.3.3 获取相机特征的访问权限

相机特征的访问权限决定了相机特征读写操作是否能正常进行。用户可以调用 **ItkFeatureGetAccessMode()**获取特征的访问权限。下面对于相机的访问权限进行说明：

值	说明
ITKFEATURE_VAL_ACCESS_MODE_RW	该特征具有读/写访问权限，可以完成读/写相机配置操作
ITKFEATURE_VAL_ACCESS_MODE_RO	该特征具有读访问权限，可以完成读取相机配置操作
ITKFEATURE_VAL_ACCESS_MODE_WO	该特征具有写访问权限，可以完成写入相机配置操作
ITKFEATURE_VAL_ACCESS_MODE_NI	该特征未实现，无法读取/写入相机配置
ITKFEATURE_VAL_ACCESS_MODE_NA	该特征在当前状态不可用，无法读取/写入相机配置

6.3.4 获取相机特征的数据类型

相机特征具有不同的数据类型，对于不同类型的相机特征应该使用对应的 API 进行访问，否则 IKapC 的库函数可能无法正常工作。用户可以调用 **ItkFeatureGetType()** 获取相机特征的具体数据类型。下面对于不同类型的相机特征进行说明。

- **整型**：整型类型的配置参数代表了可以使用 64 位整数(int64_t)存储的特征，例如采集图像的高度和宽度。每一个整型特征都具有如下属性：最大值 **maximum**，最小值 **minimum** 以及步长 **increment**。因此在设定整型特征的值时，其设定值 **x** 遵循如下规则：

$$x := \text{minimum} + N * \text{increment} \quad (N=0,1,2,\dots) \quad x \leq \text{maximum}$$

- **浮点型**：浮点型的配置参数代表了可以使用 64 位浮点数(double)存储的特征，例如相机传感器的温度。每一个浮点数类型特征都具有如下属性：最大值 **maximum**，最小值 **minimum**。因此在设定浮点数类型特征的值时，设定值 **x** 需要满足如下规则：

$$\text{minimum} \leq x \leq \text{maximum}$$

- **布尔类型**：布尔类型的配置参数代表了可以使用二值变量(true or false)存储的特征，例如相机是否开启坏点矫正。对于该类型参数，用户可以选择使能或者禁用该特征。
- **字符串类型**：字符串类型的配置参数代表可以存储在字符串中的特征，比如相机的模型名称和序列号。
- **枚举类型**：枚举类型的配置参数代表可以从预先定义好的集合中选择相应枚举项进行配置的特征，比如相机的通讯波特率。对于枚举类型，其每个枚举项均可以通过索引(32 位整数)和枚举值(字符串)唯一确定。
- **命令类型**：命令类型的配置参数代表了可执行特征，比如存储用户配置。对于该类型特征，命令执行过程可能需要耗费较长的时间，用户需要等待命令指令的执行完毕，函数才会返回。

下列代码演示了如何访问一个整型特征的最小值、最大值以及步长。

示例代码

```
// 初始化 IKapC 运行环境
ITKSTATUS res = ItkManInitialize();
Check(res);
ITKDEVICE hDev;    // 相机设备句柄
res = ItkDevOpen(0, ITKDEV_VAL_ACCESS_MODE_CONTROL, &hDev);
Check(res);
// 获取曝光时间的最大值，最小值以及步长
ITKFEATURE hFeature;
res = ItkDevAllocFeature(hDev, "ExposureTime", &hFeature);
Check(res);
uint64_t max = 0, min = 0, inc = 0;
res = ItkFeatureGetInt64Max(hFeature, &max);
Check(res);
res = ItkFeatureGetInt64Min(hFeature, &min);
Check(res);
res = ItkFeatureGetInt64Inc(hFeature, &inc);
Check(res);
res = ItkDevFreeFeature(hFeature);
ItkDevClose(hDev);
```

```
ItkManTerminate();
```

6.3.5 读取/配置相机特征

相机特征的读取和配置可以通过 IKapC 中相应函数完成，读取特征的函数一般遵循如下命名规则：**ItkFeatureGetXXX**；配置特征的函数一般遵循如下命名规则：**ItkFeatureSetXXX**。其中 XXX 是特征的数据类型，可以由 **ItkFeatureGetType()** 获得。

下列代码演示了如何读取/配置整数类型的相机特征：

示例代码

```
// 初始化 IKapC 运行环境
ITKSTATUS res = ItkManInitialize();
Check(res);
ITKDEVICE hDev;    // 相机设备句柄
res = ItkDevOpen(0, ITKDEV_VAL_ACCESS_MODE_CONTROL, &hDev);
Check(res);
// 获取曝光时间的最大值，最小值以及步长
ITKFEATURE hFeature;
res = ItkDevAllocFeature(hDev, "ExposureTime", &hFeature);
Check(res);
uint64_t max = 0, min = 0, inc = 0;
res = ItkFeatureGetInt64Max(hFeature, &max);
Check(res);
res = ItkFeatureGetInt64Min(hFeature, &min);
Check(res);
res = ItkFeatureGetInt64Inc(hFeature, &inc);
Check(res);
uint64_t val = 0;
// 获取整型特征值
res = ItkFeatureGetInt64(hFeature, &val);
Check(res);
// 设置整型特征值
val = (max+min)/2;
res = ItkFeatureSetInt64(hFeature, val);
Check(res);
res = ItkDevFreeFeature(hFeature);
ItkDevClose(hDev);
ItkManTerminate();
```

下列代码演示了如何读取/配置枚举类型的相机特征：

示例代码

```
// 初始化 IKapC 运行环境
ITKSTATUS res = ItkManInitialize();
Check(res);
ITKDEVICE hDev;    // 相机设备句柄
```

```

res = ItkDevOpen(0, ITKDEV_VAL_ACCESS_MODE_CONTROL, &hDev);
Check(res);
// 获取曝光时间的最大值，最小值以及步长
ITKFEATURE hFeature;
res = ItkDevAllocFeature(hDev, "DeviceSerialPortBaudRate", &hFeature);
Check(res);
// 获取枚举类型特征的枚举项数量
uint32_t enumCount;
res = ItkFeatureGetEnumCount(hFeature, &count);
Check(res);

// 获取枚举类型特征的枚举项
uint32_t enumStrSize = 0;
res = ItkFeatureGetEnumString(hFeature, 0, NULL, &enumStrSize);
char *enumStr = malloc(enumStrSize);
res = ItkFeatureGetEnumString(hFeature, 0, enumStr, &enumStrSize);
Check(res);

// 获取枚举类型特征的值
uint32_t enumValSize = 0;
res = ItkFeatureToString(hFeature, NULL, &enumValSize);
char* enumVal = malloc(enumValSize);
res = ItkFeatureToString(hFeature, enumVal, &enumValSize);
Check(res);

// 设置枚举类型特征的值
res = ItkFeatureFromString(hFeature, enumStr);
Check(res);

free(enumVal);
free(enumStr);
res = ItkDevFreeFeature(hFeature);
ItkDevClose(hDev);
ItkManTerminate();

```

6.3.6 释放相机特征的资源

当用户完成相机特征的读取/配置操作后，应该释放相机特征使用过程中申请的资源。调用 **ItkDevFreeFeature()** 可以释放相关资源。注意在释放特征资源后，除非再次收集相机特征资源，否则原先指向相机特征的句柄 **ITKFEATURE** 将无法再被其他 IKapC 函数调用。

6.4 采集图像

本节介绍了如何从相机设备中采集图像。对于同一个相机设备，其可能支持多路数据通道同时传输图像数据，用户可以选择其中的一路或者多路进行图像数据采集，各路图像数据相互独立

并维护自己的采集资源。

6.4.1 初始化图像数据流资源

图像数据流资源被相机设备统一管理，用户可以通过 **ItkDevGetStreamCount()** 函数获取相机设备支持的图像数据流个数。注意对于某些类型的相机，比如 Camera Link 接口相机，其支持的图像数据流个数可能为 0。

假设用户通过 **ItkDevGetStreamCount()** 获得当前可以采集的图像数据流个数为 $N(N>0)$ ，则用户可以选择其中一路或者多路完成图像采集，在采集图像前必须调用函数 **ItkDevAllocStream()** 函数完成数据流的初始化操作。

IKapC 中的每个数据采集流都会维护自己的缓冲区链表，当图像数据被采集后，新采集得到的图像数据将会顺序写入缓冲区链表中，关于缓冲区的建立和销毁参见 **ItkBufferNew()** 和 **ItkBufferFree()**。在初始化图像数据流过程中需要指定一个缓冲区作为图像传输的默认地址。

注意：每次采集操作 **ItkStreamStart()** 都会清空缓冲区链表中所有缓冲区的状态并从缓存区链表头部写入图像数据。

下列代码演示了如何初始化图像数据流资源。

示例代码

```
// 初始化 IKapC 运行环境
ITKSTATUS res = ItkManInitialize();
Check(res);
ITKDEVICE hDev;    // 相机设备句柄
// 打开索引为 0 的相机设备
res = ItkDevOpen(0, ITKDEV_VAL_ACCESS_MODE_CONTROL, &hDev);
Check(res);
// 获取相机支持采集的数据流数量
uint32_t streamCount = 0;
res = ItkDevGetStreamCount(hDev, &streamCount);
Check(res);
if (streamCount > 0)
{
    // 申请宽 2048，高 2048，图像格式为 ITKBUFFER_VAL_FORMAT_MONO8 的缓冲区
    // 该缓冲区作为图像数据的默认传输地址
    ITKBUFFER hBuffer;
    res = ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hBuffer);
    Check(res);
    ITKSTREAM hStream;
    // 申请图像数据流资源
    res = ItkDevAllocStream(hDev, 0, hBuffer, &hStream);
    Check(res);
    /* 此处处理数据流 */

    // 释放图像数据流资源
    ItkDevFreeStream(hStream);
}
```

```
// 释放图像资源
ItkBufferFree(hBuffer);
}
// 关闭相机设备
ItkDevClose(hDev);
// 释放 IKapC 初始化环境
ItkManTerminate();
```

6.4.2 添加数据缓冲区

IKapC 为每个独立的数据采集流维护一个缓冲区链表，新的图像数据被采集后将会顺序填入缓冲区链表的缓冲区中，关于缓冲区的管理方法参见“[6.5 缓存管理](#)”。

当连续采集或者采集图像序列时，相机采集帧率过快可能会导致新采集得到图像数据丢失或者覆盖原有图像数据。通过增加缓冲区链表中缓冲区的个数可以避免这种情况的发生。

注意：

- (1) 同一个数据采集流不会重复添加相同的图像缓冲区。
- (2) 当调用 **ItkDevFreeStream()** 释放数据采集流资源时会自动清空内部缓冲区链表，但是并不会销毁图像缓冲区资源，用户需要调用 **ItkBufferFree()** 释放申请的图像缓冲区资源。

下列代码演示了如何添加数据缓冲区到数据流。

示例代码

```
// 申请 10 个图像缓冲区并添加到数据采集流中
uint32_t bufCount = 10;
for (uint32_t i=0; i < bufCount; ++i)
{
    // 申请宽 2048，高 2048，图像格式为 ITKBUFFER_VAL_FORMAT_MONO8 的缓冲区
    // 该缓冲区作为图像数据的默认传输地址
    ITKBUFFER hBuffer;
    res = ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hBuffer);
    Check(res);
    // 添加图像缓冲区到数据流资源中
    res = ItkStreamAddBuffer(hStream, hBuffer);
    Check(res);
}
```

6.4.3 单帧图像采集

初始化图像数据流后，用户可以通过调用 **ItkStreamStart()** 函数采集图像数据。对于单帧/多帧图像数据采集，用户可以选择以阻塞式或者非阻塞式方式采集图像数据。当用户选择阻塞式方式采集图像数据时，直到采集的图像数量 *count* 全部采集完毕或者采集超时状况下 **ItkStreamStart()** 才会返回；当用户选择非阻塞式采集图像数据时，**ItkStreamStart()** 不等待采集过程结束并离开返回。

下列代码演示了如何以阻塞式方式采集单帧图像数据。

示例代码

```
// 设置采集模式为阻塞式
uint32_t blockMode = ITKSTREAM_VAL_STATR_MODE_BLOCK;
ITKSTATUS res = ItkStreamSetPrm(hStream, ITKSTREAM_PRM_START_MODE, &blockMode);
Check(res);
// 采集单帧图像
res = ItkStreamStart(hStream, 1);
Check(res);
```

6.4.4 多帧图像采集

当用户采集多帧图像数据时，建议通过增加数据采集流中图像缓冲区的数量来保证采集图像不会被覆盖或丢失，关于如何添加图像缓冲区参见“[6.4.2 添加数据缓冲区](#)”。

下列代码演示了如何以非阻塞式方式采集多帧图像数据。

示例代码

```
// 设置采集模式为非阻塞式
uint32_t blockMode = ITKSTREAM_VAL_STATR_MODE_NON_BLOCK;
ITKSTATUS res = ItkStreamSetPrm(hStream, ITKSTREAM_PRM_START_MODE, &blockMode);
Check(res);
// 添加 10 个图像缓冲区并采集 10 帧图像数据
uint32_t count = 10;
for (uint32_t i=0; i < count; ++i)
{
    // 申请宽 2048，高 2048，图像格式为 ITKBUFFER_VAL_FORMAT_MONO8 的缓冲区
    ITKBUFFER hBuffer;
    res = ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hBuffer);
    Check(res);

    // 添加图像缓冲区到数据流资源中
    res = ItkStreamAddBuffer(hStream, hBuffer);
    Check(res);
}
res = ItkStreamStart(hStream, 10);
Check(res);
// 等待采集结束
res = ItkStreamWait(hStream);
```

6.4.5 连续图像采集

当连续采集图像时，**ItkStreamStart()**始终以非阻塞方式启动采集，采集得到的图像将会顺序写入数据流的缓冲区链表中。用户可以通过数据流的采集状态判断当前是否正在进行采集或者采集已经终止。为了处理采集到的图像数据，IKapC 提供了多种触发事件，用户可以通过注册回调函数来处理相应的事件以及图像数据。

● 数据流采集状态

用户可以通过 **ItkStreamGetPrm()**函数并设置参数为 **ITKSTREAM_PRM_STATUS** 来获取数据流的传输状态，其状态可能是如下值：

状态	说明
ITKSTREAM_VAL_STATUS_STOPPED	数据采集已经停止
ITKSTREAM_VAL_STATUS_ACTIVE	数据采集正在进行
ITKSTREAM_VAL_STATUS_PENDING	数据采集被挂起
ITKSTREAM_VAL_STATUS_ABORTED	数据采集过程异步停止
ITKSTREAM_VAL_STATUS_TIMEOUT	数据采集超时

● 事件和回调函数

IKapC 提供了多种触发事件作为用户进行图像处理的时机。用户可以通过调用 **ItkStreamRegisterCallback()**为事件注册回调函数，事件被触发时，会调用相应的回调函数。

IKapC 支持如下触发事件：

事件	说明
ITKSTREAM_VAL_EVENT_TYPE_START_OF_STREAM	数据采集开始时调用
ITKSTREAM_VAL_EVENT_TYPE_END_OF_STREAM	数据采集结束时调用
ITKSTREAM_VAL_EVENT_TYPE_START_OF_FRAME	开始采集一帧图像数据时调用
ITKSTREAM_VAL_EVENT_TYPE_END_OF_FRAME	结束采集一帧图像数据时调用
ITKSTREAM_VAL_EVENT_TYPE_END_OF_LINE + n	第 n 行图像采集完成时调用
ITKSTREAM_VAL_EVENT_TYPE_TIME_OUT	采集超时调用
ITKSTREAM_VAL_EVENT_TYPE_FRAME_LOST	采集丢帧时调用

注意： 对于尺寸较大、需要较长采集时间的图像数据，可以通过增加参数 **ITKSTREAM_PRM_TIME_OUT** 来保证采集操作不会超时结束。

● 结束采集

用户可以选择调用 **ItkStreamStop()**同步结束采集过程或者 **ItkStreamAbort()**异步结束采集过程。对于同步结束采集操作，IKapC 将会等待正在采集的图像采集完毕才停止采集操作，因此可以保证最后一帧图像的完整性。对于异步结束采集操作，IKapC 立刻结束采集过程而不等待最后一帧图像采集完毕，因此无法保证最后一帧图像的完整性。

下列代码演示了如何注册回调函数并连续采集图像数据。

示例代码
<pre> void IKAPC_CC cbEndOfFrame(uint32_t type, uint32_t count, void* context) { /* 此处处理图像数据 */ printf("End of frame\n"); } // 注册回调函数 ITKSTATUS res = ItkStreamRegisterCallback(hStream, ITKSTREAM_VAL_EVENT_TYPE_END_OF_FRAME, cbEndOfFrame, NULL); Check(res); // 开始连续采集 </pre>


```
res = ItkStreamStart(hStream, ITKSTREAM_CONTINUOUS);
Check(res);
// 等待 5000ms 并停止采集
Sleep(5000);
ItkStreamStop(hStream);
```

6.4.6 释放图像数据流资源

用户采集完成后应该调用 **ItkDevFreeStream()** 释放采集过程中申请的资源。

注意：**ItkDevFreeStream()** 仅会清空缓冲区链表而不会销毁缓冲区数据，用户需要调用 **ItkBufferFree()** 函数销毁缓冲区数据。

下列代码演示了如何释放数据流资源。

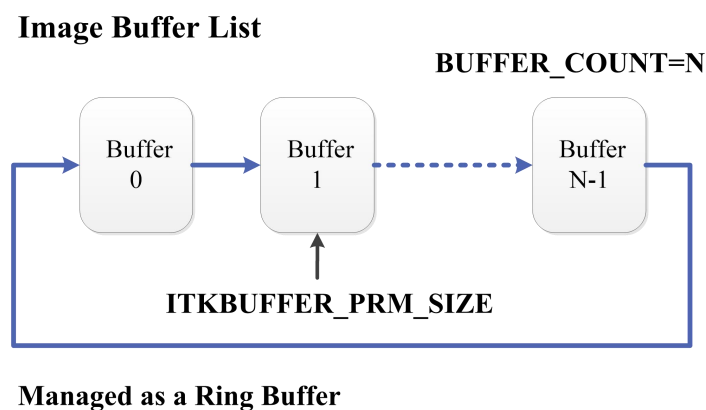
示例代码

```
ItkDevFreeStream(hStream);
```

6.5 缓存管理

相机采集到的图像数据将会顺序存储到图像数据流的缓存链表中。**IKapC** 提供了不同的方法来控制图像的传输流程。通过设置合理的图像传输机制，可以更好的保证图像数据的安全，提高应用程序性能。

如下图所示，在数据流采集图像过程中，图像数据会被顺序的写入缓存链表中，**IKapC** 按照环形缓冲区的方式存储图像数据。通过增加数据流的缓冲区链表长度，可以避免相机帧率过快导致的图像丢失或覆盖，详细信息参见“[6.4.2 添加数据缓冲区](#)”。



6.5.1 图像缓冲区的状态

- 图像缓冲区的四种状态
 - **ITKBUFFER_VAL_STATE_EMPTY**: 空状态，意味着当前缓冲区没有相机采集到的图像数据
 - **ITKBUFFER_VAL_STATE_FULL**: 满状态，意味着当前缓冲区已经被相机采集到的图像数据填满

- **ITKBUFFER_VAL_STATE_OVERFLOW:** 覆盖状态，意味缓冲区在清空前被新到来的图像数据覆盖
 - **ITKBUFFER_VAL_STATE_UNCOMPLETED:** 缓冲区非满，在采集千兆网相机时可能因为网络丢包导致无法采集完整的一帧图像
- 用户可以通过 **ItkBufferGetPrm()** 并设置参数为 **ITKBUFFER_PRM_STATE** 来获取指定传输缓冲区的状态。
- 图像缓冲区的状态切换

图像缓冲区的状态会在如下时刻切换：

 - 数据流启动图像采集时，该数据流缓冲区链表中的所有缓冲区被设置为 **ITKBUFFER_VAL_STATE_EMPTY**
 - 当图像缓冲区被相机数据填满时被设置为 **ITKBUFFER_VAL_STATE_FULL**
 - 当图像缓冲区为 **ITKBUFFER_VAL_STATE_FULL** 状态且再次写入图像数据时，设置缓冲区状态为 **ITKBUFFER_VAL_STATE_OVERFLOW**
 - 当图像数据流触发 **ITKSTREAM_VAL_EVENT_TYPE_END_OF_FRAME** 并且执行完成用户设置的回调函数后（无论是否设置），都会自动设置图像缓冲区状态为 **ITKBUFFER_VAL_STATE_EMPTY**
 - 图像缓冲区的自动清空机制

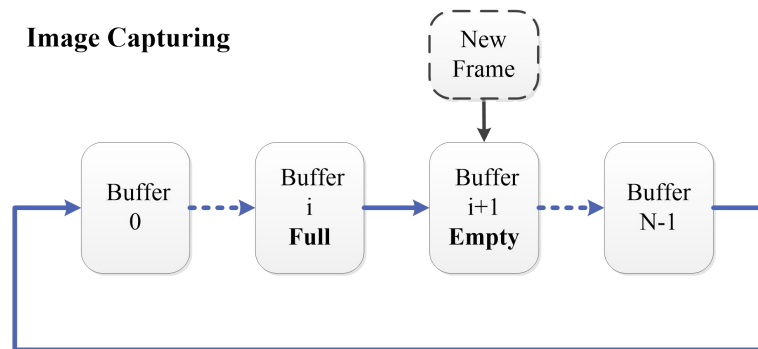
当图像数据流触发 **ITKSTREAM_VAL_EVENT_TYPE_END_OF_FRAME** 并且执行完成用户设置的回调函数后（无论是否设置），都会自动设置图像缓冲区状态为 **ITKBUFFER_VAL_STATE_EMPTY**。

注意：

- （1）用户需要在回调函数中读取所有状态为 **ITKBUFFER_VAL_STATE_FULL** 的缓冲区，否则新的图像数据有可能会覆盖旧数据。
- （2）如果用户需要手动控制缓存区域的状态，可以调用 **ItkStreamSetPrm()** 和参数 **ITKSTREAM_PRM_AUTO_CLEAR** 关闭 IKapC 的自动清空机制。注意该函数应该在采集过程开始前调用。
- （3）当用户关闭自动清空机制后，需要手动清空状态为 **ITKBUFFER_VAL_STATE_FULL** 的图像数据。通过调用 **ItkBufferSetPrm()** 并设置参数为 **ITKBUFFER_PRM_STATE** 来设置传输缓存区域状态为 **ITKBUFFER_VAL_STATE_EMPTY**，新的图像数据将会再次写入状态为 **ITKBUFFER_VAL_STATE_EMPTY** 的缓冲区中。

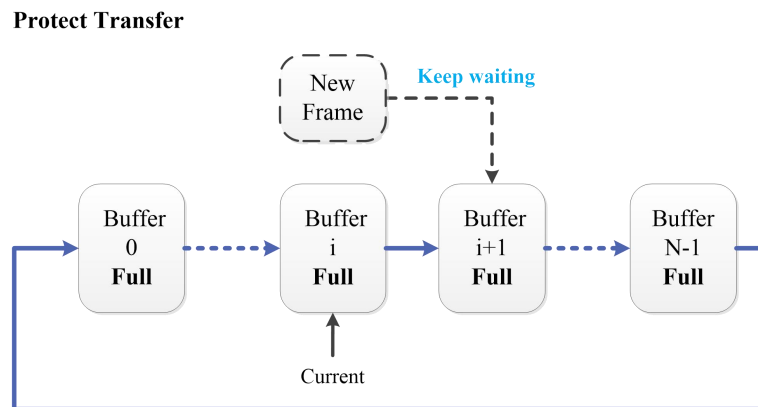
6.5.2 图像数据流的传输模式

IKapC 支持数据流的同步保护模式，用户可以通过 **ItkStreamSetPrm()** 和参数 **ITKSTREAM_PRM_TRANSFER_MODE** 设置数据传输模式。



● 同步保护模式

在这种模式下，IKapC 不会写入图像数据，直到下一帧缓存状态为空。注意在这种模式下，原有图像数据的安全性可以得到保证，但是可能会导致新到来图像数据的丢失。



注意：

- (1) 当用户禁用 IKapC 自动清空机制后，必须手动调用 **ItkBufferSetPrm()** 和参数 **ITKBUFFER_PRM_STATE** 来清空缓冲区。
- (2) 图像数据将会从缓冲区链表的第一个缓冲区开始顺序写入。为了避免采集图像乱序，用户应该保存访问的图像编号。

6.6 显示图像

本章节介绍如何显示存储在 **ITKBUFFER** 缓存中的图像数据。IKapC 支持在显示图像的过程中对图像数据进行进一步处理，如翻转、缩放、截取图像 ROI（感兴趣区域）等。

6.6.1 创建显示图像的窗口

在显示图像数据前，用户必须先调用 **ItkViewNew()** 申请建立窗口的资源。每个图像显示窗口都会维护独立的缓冲区链表，该链表默认存储一个图像缓冲区数据，即通过 **ItkViewNew()** 新建窗口时候传入的缓冲区。

下面代码演示了如何创建图像显示窗口。

示例代码

```
// 初始化 IKapC 运行环境
```

```
ITKSTATUS res = ItkManInitialize();
Check(res);
// 申请宽 2048, 高 2048, 图像格式为 ITKBUFFER_VAL_FORMAT_MONO8 的缓冲区
// 该缓冲区作为图像数据的默认传输地址
ITKBUFFER hBuffer;
res = ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hBuffer);
Check(res);
ITKVIEW hView;
// 申请图像显示窗口
res = ItkViewNew(hBuffer, NULL, &hView);
Check(res);
/* 此处显示图像 */
// 释放图像显示窗口
ItkViewFree(hView);
// 释放图像资源
ItkBufferFree(hBuffer);
// 释放 IKapC 初始化环境
ItkManTerminate();
```

6.6.2 添加/移除图像缓冲区

每个图像显示窗口都会维护独立的数据缓冲区链表，当用户调用 **ItkViewShow()** 或者 **ItkViewShowNext()** 时，会从缓冲区链表中取出一个或者多个图像数据进行显示。

注意：

- (1) 同一个图像显示窗口不会重复添加相同的图像缓冲区。
- (2) 当调用 **ItkViewFree()** 释放图像显示窗口时，会自动清空内部缓冲区链表，但是并不会销毁图像缓冲区资源。用户需要调用 **ItkBufferFree()** 释放申请的图像缓冲区资源。
- (3) 用户添加的缓冲区类型应该和建立图像窗口时传入的缓冲区类型保持一致，即保持相同的宽度、高度和像素格式。

下列代码演示了如何添加数据缓冲区到图像显示窗口。

示例代码

```
// 申请 10 个图像缓冲区并添加到图像缓冲区中
uint32_t bufCount = 10;
for (uint32_t i=0; i < bufCount; ++i)
{
    // 申请宽 2048,高 2048,图像格式为 ITKBUFFER_VAL_FORMAT_MONO8 的缓冲区
    ITKBUFFER hBuffer;
    res = ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hBuffer);
    Check(res);
    // 添加图像缓冲区到数据流资源中
    res = ItkViewAddBuffer(hView, hBuffer);
    Check(res);
}
```

}

6.6.3 设置参数

- 设置图像翻转参数

IKapC 支持在显示图像的过程中对图像进行水平或垂直翻转操作。用户可以调用 **ItkViewSetPrm()** 设置显示图像是否沿水平或者垂直翻转。

下列代码演示了如何水平翻转显示图像。

示例代码

```
// 申请宽 2048,高 2048,图像格式为 ITKBUFFER_VAL_FORMAT_MONO8 的缓冲区
// 该缓冲区作为图像数据的默认传输地址
ITKBUFFER hBuffer;
res = ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hBuffer);
Check(res);
ITKVIEW hView;
// 申请图像显示窗口
res = ItkViewNew(hBuffer, NULL, &hView);
Check(res);
// 水平翻转图像
int32_t bFlipX = ITKVIEW_VAL_FLIP_X_ENABLE;
res = ItkViewSetPrm(hView, ITKVIEW_PRM_FLIP_X, &bFlipX);
Check(res);
/* 此处显示图像 */
// 释放图像显示窗口
ItkViewFree(hView);
// 释放图像资源
ItkBufferFree(hBuffer);
```

- 设置图像感兴趣区域参数

IKapC 支持在显示图像的过程中仅显示用户选择的感兴趣区域, 用户感兴趣区域的大小取决于如下四个参数, 用户可以通过 **ItkViewSetPrm()** 设置这些参数。

参数	说明
ITKVIEW_PRM_BUFFER_ROI_HEIGHT	用户感兴趣区域的高度
ITKVIEW_PRM_BUFFER_ROI_WIDTH	用户感兴趣区域的宽度
ITKVIEW_PRM_BUFFER_ROI_TOP	用户感兴趣区域的上边缘坐标
ITKVIEW_PRM_BUFFER_ROI_LEFT	用户感兴趣区域的下边缘坐标

注意: 用户在创建显示窗口的 ROI 区域时候, 应该先设置 ROI 区域的高度和宽度, 再设置 ROI 区域的上边缘坐标和左边缘坐标。

下列代码演示了如何创建 ROI 区域并进行显示。

示例代码

```
// 申请宽 2048,高 2048,图像格式为 ITKBUFFER_VAL_FORMAT_MONO8 的缓冲区
```

```
// 该缓冲区作为图像数据的默认传输地址
ITKBUFFER hBuffer;
res = ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hBuffer);
Check(res);
ITKVIEW hView;
// 申请图像显示窗口
res = ItkViewNew(hBuffer, NULL, &hView);
Check(res);
// 设置图像 ROI 区域，ROI 区域的高度为 500，宽度为 500，左上角坐标为(10,10)
int32_t val = 500;
res = ItkViewSetPrm(hView, ITKVIEW_PRM_BUFFER_ROI_HEIGHT, &val);
Check(res);
res = ItkViewSetPrm(hView, ITKVIEW_PRM_BUFFER_ROI_WIDTH, &val);
Check(res);
val = 10;
res = ItkViewSetPrm(hView, ITKVIEW_PRM_BUFFER_ROI_LEFT, &val);
Check(res);
res = ItkViewSetPrm(hView, ITKVIEW_PRM_BUFFER_ROI_TOP, &val);
Check(res);
/* 此处显示图像 */
// 释放图像显示窗口
ItkViewFree(hView);
// 释放图像资源
ItkBufferFree(hBuffer);
```

● 设置图像缩放参数

IKapC 在显示图像时可以通过鼠标滚轮控制图像的缩小和放大，当前 IKapC 支持图像最大放大比例是 128，最小缩小比例是 1/128，同时用户可以通过 **ItkViewSetPrm()**选择不同的算法来缩放图像，可选算法如下所示：

参数	说明
ITKVIEW_VAL_ZOOM_NN	最邻近插值算法
ITKVIEW_VAL_ZOOM_LINEAR	线性插值算法
ITKVIEW_VAL_ZOOM_CUBIC	立方插值算法
ITKVIEW_VAL_ZOOM_AREA	重采样插值算法

下列代码演示了如何选择缩放算法。

示例代码

```
// 申请宽 2048，高 2048，图像格式为 ITKBUFFER_VAL_FORMAT_MONO8 的缓冲区
// 该缓冲区作为图像数据的默认传输地址
ITKBUFFER hBuffer;
res = ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hBuffer);
Check(res);
ITKVIEW hView;
```

```
// 申请图像显示窗口
res = ItkViewNew(hBuffer, NULL, &hView);
Check(res);
// 设置图像缩放算法
int32_t nZoomMethod = ITKVIEW_VAL_ZOOM_LINEAR;
res = ItkViewSetPrm(hView, ITKVIEW_PRM_ZOOM_METHOD, &nZoomMethod);
Check(res);
/* 此处显示图像 */
// 释放图像显示窗口
ItkViewFree(hView);
// 释放图像资源
ItkBufferFree(hBuffer);
```

6.6.4 显示图像

在显示图像的过程中，用户有两种方法可以调用：**ItkViewShow()**和**ItkViewShowNext()**。

ItkViewShow()将会遍历整个显示窗口的内部缓冲区链表，只要数据缓冲区采集完毕就会在窗口上显示图像，同时该方法会在不影响图像数据传输的情况下显示尽可能多的图像数据，这意味着如果显示图像的操作过于耗时，可能会跳过部分图像的显示过程。如果用户在连续采集的过程中希望实时显示采集到的图像数据，可以在事件**ITKSTREAM_VAL_EVENT_TYPE_END_OF_FRAME**的回调函数中调用该函数。

ItkViewShowNext()将会显示下一帧图像缓冲区数据，每次调用该函数都仅显示一帧图像数据，这意味着该方法不会跳过任何图像的显示过程，然而在图像显示操作过于耗时且频繁调用该函数的情况下可能会导致图像显示内部线程阻塞。如果用户在单次采集的过程中希望显示得到的图像数据，可以在采集图像完成后调用该函数。

下列代码演示了如何显示图像。

示例代码

```
void IKAPC_CC cbEndOfFrame(uint32_t type, uint32_t count, void* context)
{
    /* 此处显示图像 */
    ITKVIEW hView = (ITKVIEW)context;
    ItkViewShow(hView);
}
// 申请图像窗口资源
ITKVIEW hView;
ItkViewNew(hBuffer, NULL, &hView);
// 注册回调函数
ITKSTATUS res = ItkStreamRegisterCallback(hStream,
    ITKSTREAM_VAL_EVENT_TYPE_END_OF_FRAME, cbEndOfFrame, hView);
Check(res);
// 开始连续采集
res = ItkStreamStart(hStream, ITKSTREAM_CONTINUOUS);
Check(res);
// 等待 5000ms 并停止采集
```

```
Sleep(5000);
ItkStreamStop(hStream);
```

6.6.5 释放显示图像的窗口资源

当用户不需要显示图像时，应该调用 **ItkViewFree()** 释放图像窗口资源。

注意：即使用户已经点击“关闭”按钮关闭图像显示窗口，仍然需要调用 **ItkViewFree()** 释放窗口资源。

下列代码演示了如何释放图像窗口资源。

示例代码

```
ItkViewFree(hView);
```

6.7 获取相机掉线信息

用户可以通过注册回调函数来获取相机掉线信息，具体使用方式如下：

- 在成功打开设备后注册相机掉线回调函数

示例代码

```
res = ItkDevRegisterCallback(hDev, "DeviceRemove", removalCallbackFunction, context);  
Check(res);
```

- 在关闭设备前清除掉线回调函数

示例代码

```
res = ItkDevUnregisterCallback(hDev, "DeviceRemove");
```

- 当相机掉线事件发生后会触发如下的回调函数

示例代码

```
void _stdcall removalCallbackFunction(void* context, ITKEVENTINFO eventInfo)  
{  
    /* 检索事件类型 */  
    uint32_t type = 0;  
    uint64_t countstamp = 0;  
    ITKDEVICE hDev = (ITKDEVICE)context;  
  
    ITKSTATUS res = ItkEventInfoGetPrm (eventInfo, ITKEVENTINFO_PRM_TYPE, &type);  
    Check(res);  
  
    callbackCounter++;  
}
```

7 API 函数说明

7.1 函数概览

函数	说明
<u>ItkManInitialize</u>	初始化 IKapC 开发环境
<u>ItkManTerminate</u>	释放 IKapC 初始化过程中申请的资源
<u>ItkManGetDeviceCount</u>	获取当前连接到 PC 上相机设备的数量
<u>ItkManGetDeviceInfo</u>	获取相机具体设备信息
<u>ItkManGetBoardCount</u>	获取当前连接到 PC 上采集卡设备的数量
<u>ItkManGetBoardInfo</u>	获取采集卡具体设备信息
<u>ItkManGetGigEDeviceInfo</u>	获取千兆网相机专有设备信息
<u>ItkManGetCLDeviceInfo</u>	获取 Camera Link 相机专有信息
<u>ItkManGetCXPDeviceInfo</u>	获取 CoaXPress 相机专有信息
<u>ItkManGetU3VDeviceInfo</u>	获取 USB3.0 相机专有信息
<u>ItkManGetStatusText</u>	获取状态码的信息
<u>ItkManGetPrm</u>	获取全局参数
<u>ItkManSetPrm</u>	设置全局参数
<u>ItkDevOpen</u>	打开相机设备
<u>ItkDevClose</u>	关闭相机设备
<u>ItkDevLoadConfigurationFromFile</u>	加载相机设备配置文件
<u>ItkDevSaveConfigurationToFile</u>	保存相机设备配置文件
<u>ItkDevGetFeatureCount</u>	获取相机支持配置特征的数量
<u>ItkDevGetFeatureName</u>	获取相机特征的名称
<u>ItkDevAllocFeature</u>	申请配置相机特征需要的资源
<u>ItkDevFreeFeature</u>	释放配置相机特征的资源
<u>ItkDevGetStreamCount</u>	获取相机数据流的数量
<u>ItkDevAllocStream</u>	申请数据流采集过程中需要的资源
<u>ItkDevFreeStream</u>	释放数据流采集过程中申请的资源
<u>ItkDevGetEventCount</u>	获取相机支持触发事件的数量
<u>ItkDevGetEventName</u>	获取相机支持触发事件的名称
<u>ItkDevIsEventAvailable</u>	判断相机是否支持该触发事件
<u>ItkDevRegisterCallback</u>	为相机触发事件注册回调函数
<u>ItkDevUnregisterCallback</u>	清除注册的回调函数
<u>ItkDevGetPrm</u>	获取相机参数
<u>ItkDevSetPrm</u>	设置相机参数
<u>ItkDevPortRead</u>	直接读取寄存器
<u>ItkDevPortWrite</u>	直接写入寄存器
<u>ItkDevSerialPortRead</u>	从串口设备读取数据
<u>ItkDevSerialPortWrite</u>	向串口设备发送数据
<u>ItkGigEDevForceIp</u>	改变千兆网相机临时 IP 地址
<u>ItkGigEDevGetPersistentIpAddress</u>	获取千兆网相机静态 IP 地址

ItkGigEDevSetPersistentIpAddress	设置千兆网相机静态 IP 地址
ItkEventInfoGetPrm	获取事件参数信息
ItkFeatureGetAccessMode	获取相机特征的访问模式
ItkDevGetAccessMode	获取相机特征的访问模式
ItkFeatureGetCategory	获取相机特征的类别
ItkDevGetCategory	获取相机特征的类别
ItkFeatureGetType	获取相机特征的数据类型
ItkDevGetType	获取相机特征的数据类型
ItkFeatureGetDisplayName	获取相机特征的显示名称
ItkDevGetDisplayName	获取相机特征的显示名称
ItkFeatureGetTooltip	获取相机特征的提示信息
ItkDevGetTooltip	获取相机特征的提示信息
ItkFeatureGetDescription	获取相机特征的具体描述信息
ItkDevGetDescription	获取相机特征的具体描述信息
ItkFeatureGetNameSpace	获取相机特征的命名空间
ItkDevGetNameSpace	获取相机特征的命名空间
ItkFeatureGetVisibility	获取相机特征可视权限
ItkDevGetVisibility	获取相机特征可视权限
ItkFeatureGetRepresentation	获取相机特征表示方式
ItkDevGetRepresentation	获取相机特征表示方式
ItkFeatureGetInt32Min	获取 32 位整数类型特征的最小值
ItkDevGetInt32Min	获取 32 位整数类型特征的最小值
ItkFeatureGetInt32Max	获取 32 位整数类型特征的最大值
ItkDevGetInt32Max	获取 32 位整数类型特征的最大值
ItkFeatureGetInt32Inc	获取 32 位整数类型特征的步长
ItkDevGetInt32Inc	获取 32 位整数类型特征的步长
ItkFeatureGetInt32	获取 32 位整数类型特征的值
ItkDevGetInt32	获取 32 位整数类型特征的值
ItkFeatureSetInt32	设置 32 位整数类型特征的值
ItkDevSetInt32	设置 32 位整数类型特征的值
ItkFeatureGetInt64Min	获取 64 位整数类型特征的最小值
ItkDevGetInt64Min	获取 64 位整数类型特征的最小值
ItkFeatureGetInt64Max	获取 64 位整数类型特征的最大值
ItkDevGetInt64Max	获取 64 位整数类型特征的最大值
ItkFeatureGetInt64Inc	获取 64 位整数类型特征的步长
ItkDevGetInt64Inc	获取 64 位整数类型特征的步长
ItkFeatureGetInt64	获取 64 位整数类型特征的值
ItkDevGetInt64	获取 64 位整数类型特征的值
ItkFeatureSetInt64	设置 64 位整数类型特征的值
ItkDevSetInt64	设置 64 位整数类型特征的值
ItkFeatureGetDoubleMin	获取浮点数类型特征的最小值

<u>ItkDevGetDoubleMin</u>	获取浮点数类型特征的最小值
<u>ItkFeatureGetDoubleMax</u>	获取浮点数类型特征的最大值
<u>ItkDevGetDoubleMax</u>	获取浮点数类型特征的最大值
<u>ItkFeatureGetDoubleInc</u>	获取浮点数类型特征的步长
<u>ItkDevGetDoubleInc</u>	获取浮点数类型特征的步长
<u>ItkFeatureGetDouble</u>	获取浮点数类型特征的值
<u>ItkDevGetDouble</u>	获取浮点数类型特征的值
<u>ItkFeatureSetDouble</u>	设置浮点数类型特征的值
<u>ItkDevSetDouble</u>	设置浮点数类型特征的值
<u>ItkFeatureGetBool</u>	获取布尔数据类型特征的值
<u>ItkDevGetBool</u>	获取布尔数据类型特征的值
<u>ItkFeatureSetBool</u>	设置布尔数据类型特征的值
<u>ItkDevSetBool</u>	设置布尔数据类型特征的值
<u>ItkFeatureToString</u>	获取字符串类型或枚举类型特征的值
<u>ItkDevToString</u>	获取字符串类型或枚举类型特征的值
<u>ItkFeatureFromString</u>	设置字符串类型或枚举类型特征的值
<u>ItkDevFromString</u>	设置字符串类型或枚举类型特征的值
<u>ItkFeatureGetEnumCount</u>	获取枚举类型特征的枚举数量
<u>ItkDevGetEnumCount</u>	获取枚举类型特征的枚举数量
<u>ItkFeatureGetEnumString</u>	获取枚举类型特征的枚举名
<u>ItkDevGetEnumString</u>	获取枚举类型特征的枚举名
<u>ItkFeatureEnumIsAvailable</u>	判断枚举名是否存在
<u>ItkDevEnumIsAvailable</u>	判断枚举名是否存在
<u>ItkFeatureExecuteCommand</u>	根据相机特征执行相应命令
<u>ItkDevExecuteCommand</u>	根据相机特征执行相应命令
<u>ItkFeatureIsSelector</u>	判断特征是否为选择器
<u>ItkDevIsSelector</u>	判断特征是否为选择器
<u>ItkFeatureGetSelectedFeatureCounts</u>	获取被选择特征的数量
<u>ItkDevGetSelectedFeatureCounts</u>	获取被选择特征的数量
<u>ItkFeatureGetSelectedFeatureName</u>	获取被选择特征的名称
<u>ItkDevGetSelectedFeatureName</u>	获取被选择特征的名称
<u>ItkFeatureGetPollingTime</u>	获取相机特征的轮询周期
<u>ItkDevGetPollingTime</u>	获取相机特征的轮询周期
<u>ItkStreamAbort</u>	异步结束数据流采集
<u>ItkStreamAddBuffer</u>	向数据流添加缓冲区
<u>ItkStreamRemoveBuffer</u>	从数据流移除缓冲区
<u>ItkStreamClearBuffer</u>	清空数据流缓冲区
<u>ItkStreamGetPrm</u>	获取数据流参数
<u>ItkStreamSetPrm</u>	设置数据流参数
<u>ItkStreamStart</u>	开始数据流采集工作
<u>ItkStreamStop</u>	同步结束数据流采集工作

<u>ItkStreamWait</u>	等待数据流采集结束
<u>ItkStreamRegisterCallback</u>	注册数据流回调函数
<u>ItkStreamUnregisterCallback</u>	清空数据流回调函数
<u>ItkBufferNew</u>	新建缓冲区
<u>ItkBufferFree</u>	销毁缓冲区
<u>ItkBufferGetPrm</u>	获取缓冲区参数
<u>ItkBufferSetPrm</u>	设置缓冲区参数
<u>ItkBufferRead</u>	从缓冲区中读取数据
<u>ItkBufferWrite</u>	向缓冲区写入数据
<u>ItkBufferBayerConvert</u>	将 Bayer 图像转换成彩色图像
<u>ItkBufferClear</u>	清除缓冲区中的数据
<u>ItkBufferClearBlack</u>	将缓冲区中所有像素设为黑色
<u>ItkBufferCopy</u>	拷贝源缓冲区中的数据到目的缓冲区中
<u>ItkBufferCopyRect</u>	拷贝源缓冲区矩形区域的数据到目的缓冲区中
<u>ItkBufferLoad</u>	从文件中载入数据到缓冲区
<u>ItkBufferSave</u>	将缓冲区中的数据保存到文件
<u>ItkBufferReadElement</u>	读取缓冲区中的一个像素值
<u>ItkBufferReadLine</u>	读取缓冲区中直线上的像素值
<u>ItkBufferReadRect</u>	读取缓冲区中矩形区域的像素值
<u>ItkBufferWriteElement</u>	向缓冲区写入单个像素数据
<u>ItkBufferWriteLine</u>	向缓冲区写入一行像素数据
<u>ItkBufferWriteRect</u>	向缓冲区的矩形区域写入数据
<u>ItkBufferDenoise</u>	阈值影响降噪效果
<u>ItkViewNew</u>	新建图像显示窗口
<u>ItkViewFree</u>	释放图像显示窗口
<u>ItkViewAddBuffer</u>	图像显示窗口添加缓冲区
<u>ItkViewRemoveBuffer</u>	图像显示窗口移除缓冲区
<u>ItkViewRemoveAllBuffer</u>	图像显示窗口移除所有缓冲区
<u>ItkViewGetPrm</u>	获取图像显示窗口的参数
<u>ItkViewSetPrm</u>	设置图像显示窗口的参数
<u>ItkViewHide</u>	隐藏图像显示窗口
<u>ItkViewShow</u>	显示所有采集完毕的图像缓冲区数据
<u>ItkViewShowNext</u>	显示下一帧图像缓冲区数据

7.2 函数说明

ItkManInitialize 初始化 IKapC 开发环境

C/C++	ITKSTATUS ItkManInitialize();
参数说明	无
返回值	ITKSTAUTS_OK : 调用成功 ITKSTATUS_INSUFFICIENT_RESOURCES : 资源不足
说明	初始化 IKapC 运行环境，必须在调用其他 IKapC 函数前调用。
关联文件	IKapC.h
关联库	IKapC.lib
注意事项	本函数必须在调用其他 IKapC 函数前调用。
相关函数	ItkManTerminate()

示例代码

无

ItkManTerminate 释放 IKapC 初始化过程中申请的资源

C/C++	ITKSTATUS ItkManTerminate ();
参数说明	无
返回值	ITKSTAUTS_OK : 调用成功 ITKSTATUS_INSUFFICIENT_RESOURCES : 资源不足
说明	释放 IKapC 初始化开发环境中申请的资源。
关联文件	IKapC.h
关联库	IKapC.lib
注意事项	本函数调用后，除非再次调用 ItkManInitialize() 初始化 IKapC 运行环境， 否则所有 IKapC 函数都无法执行。
相关函数	ItkManInitilize()

示例代码

无

ItkManGetDeviceCount 获取当前连接到 PC 上相机设备的数量

C/C++	ITKSTATUS ItkManGetDeviceCount(uint32_t *pCount);	
参数说明	pCount	当前连接到 PC 上的相机的数量
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_ARG_NULL : 输入参数为空	
说明	获取当前连接到 PC 上的相机数量。	

关联文件	IKapC.h
关 联 库	IKapC.lib
注意事项	无
相关函数	ItkManGetDeviceInfo()

示例代码

```
ITKSTATUS res = ItkManInitilize();
if (res == ITKSTATUS_OK)
{
    uint32_t count;
    ItkManGetDeviceCount(&count);
    ItkManTerminate();
}
```

ItkManGetDeviceInfo 获取相机具体设备信息

C/C++	ITKSTATUS ItkManGetDeviceInfo (uint32_t index, PITKDEV_INFO pDI);	
参数说明	index	相机索引，有效范围[0, ItkManGetDeviceCount() - 1]
	pDI	相机描述信息结构体
返 回 值	ITKSTATUS_OK : 调用成功 ITKSTATUS_ARG_NULL : 输入参数为空 ITKSTATUS_DEVICE_ID_OUT_OF_RANGE : 相机索引越界	
说 明	获取由索引指定相机的基本信息。 相机基本信息结构体在下文中有详细描述。	
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	在调用本函数前，必须先调用 ItkManGetDeviceCount()	
相关函数	ItkManGetDeviceCount()	

示例代码

```
ITKSTATUS res = ItkManInitilize();
if (res == ITKSTATUS_OK)
{
    uint32_t count;
    ItkManGetDeviceCount(&count);
    for (uint32_t i=0; i < count; ++i)
    {
        ITKDEV_INFO DI;
        ItkManGetDevInfo(i, &DI);
    }
    ItkManTerminate();
}
```

ItkManGetBoardCount 获取当前连接到 PC 上采集卡设备的数量

C/C++	ITKSTATUS ItkManGetBoardCount(uint32_t *pCount);	
参数说明	pCount	当前连接到 PC 上的采集卡的数量
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_ARG_NULL : 输入参数为空	
说明	获取当前连接到 PC 上采集卡数量。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkManGetBoardInfo()	

示例代码

```
ITKSTATUS res = ItkManInitilize();
if (res == ITKSTATUS_OK)
{
    uint32_t count;
    ItkManGetBoardCount(&count);
    ItkManTerminate();
}
```

ItkManGetBoardInfo 获取采集卡具体设备信息

C/C++	ITKSTATUS ItkManGetBoardInfo (uint32_t index, PITKBOARD_INFO pDI);	
参数说明	index	采集卡索引，有效范围[0,ItkManGetBoardCount() - 1]
	pDI	采集卡描述信息结构体
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_ARG_NULL : 输入参数为空 ITKSTATUS_DEVICE_ID_OUT_OF_RANGE : 索引越界	
说明	获取由索引指定的采集卡基本信息。 采集卡基本信息结构体在下文中有详细描述。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	在调用本函数前，必须先调用 ItkManGetBoardCount()	
相关函数	ItkManGetBoardCount()	

示例代码

```
ITKSTATUS res = ItkManInitilize();
if (res == ITKSTATUS_OK)
{
```

```

uint32_t count;
ItkManGetBoardCount(&count);
for (uint32_t i=0; i < count; ++i)
{
    ITKBOARD_INFO DI;
    ItkManGetBoardInfo(i, &DI);
}
ItkManTerminate();
}

```

ItkManGetGigEDeviceInfo 获取千兆网相机专有设备信息

C/C++	ITKSTATUS ItkManGetGigEDeviceInfo(uint32_t index, PITKGIGEDEV_INFO pDI);	
参数说明	index	相机索引，有效范围[0,ItkManGetDeviceCount() - 1]
	pDI	千兆网相机专有设备信息
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_ARG_NULL : 输入参数为空 ITKSTATUS_DEVICE_ID_OUT_OF_RANGE : 相机索引越界 ITKSTATUS_NOT_IMPLEMENT : 设备接口不是千兆以太网	
说明	获取千兆网相机专有设备信息，对于非千兆网相机设备，如 Camera Link 接口相机，本函数返回值是 ITKSTATUS_NOT_IMPLEMENT 。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	在调用本函数前，必须先调用 ItkManGetDeviceCount()	
相关函数	ItkManGetDeviceCount()	

示例代码

```

ITKSTATUS res = ItkManInitilize();
if (res == ITKSTATUS_OK)
{
    uint32_t count;
    ItkManGetDeviceCount(&count);
    for (uint32_t i=0; i < count; ++i)
    {
        ITKGIGEDEV_INFO DI;
        ItkManGetGigEDevInfo(i, &DI);
    }
    ItkManTerminate();
}

```

ItkManGetCLDeviceInfo 获取 Camera Link 相机专有信息

C/C++	ITKSTATUS ItkManGetCLDeviceInfo(uint32_t index, PITK_CL_DEV_INFO pDI);	
参数说明	index	相机索引，有效范围[0, ItkManGetDeviceCount() - 1]
	pDI	Camera Link 相机专有设备信息
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_ARG_NULL : 输入参数为空 ITKSTATUS_DEVICE_ID_OUT_OF_RANGE : 相机索引越界 ITKSTATUS_NOT_IMPLEMENT : 设备接口不是 Camera Link	
说明	获取 Camera Link 相机专有设备信息，对于非 Camera Link 相机设备，如 CoaXPress 接口相机，本函数返回值是 ITKSTATUS_NOT_IMPLEMENT 。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	在调用本函数前，必须先调用 ItkManGetDeviceCount()	
相关函数	ItkManGetDeviceCount()	

示例代码

```
ITKSTATUS res = ItkManInitilize();
if (res == ITKSTATUS_OK)
{
    uint32_t count;
    ItkManGetDeviceCount(&count);
    for (uint32_t i=0; i < count; ++i)
    {
        ITK_CL_DEV_INFO DI;
        ItkManGetCLDevInfo(i, &DI);
    }
    ItkManTerminate();
}
```

ItkManGetCXPDeviceInfo 获取 CoaXPress 相机专有信息

C/C++	ITKSTATUS ItkManGetCXPDeviceInfo(uint32_t index, PITK_CXP_DEV_INFO pDI);	
参数说明	index	相机索引，有效范围[0, ItkManGetDeviceCount() - 1]
	pDI	CoaXPress 相机专有设备信息
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_ARG_NULL : 输入参数为空 ITKSTATUS_DEVICE_ID_OUT_OF_RANGE : 相机索引越界 ITKSTATUS_NOT_IMPLEMENT : 设备接口不是 CoaXPress	
说明	获取 CoaXPress 相机专有设备信息，对于非 CoaXPress 相机设备，如 Camera Link	

	接口相机，本函数返回值是 ITKSTATUS_NOT_IMPLEMENT 。
关联文件	IKapC.h
关联库	IKapC.lib
注意事项	在调用本函数前，必须先调用 ItkManGetDeviceCount()
相关函数	ItkManGetDeviceCount()

示例代码

```
ITKSTATUS res = ItkManInitilize();
if (res == ITKSTATUS_OK)
{
    uint32_t count;
    ItkManGetDeviceCount(&count);
    for (uint32_t i=0; i < count; ++i)
    {
        ITK_CXP_DEV_INFO DI;
        ItkManGetCXPDevInfo(i, &DI);
    }
    ItkManTerminate();
}
```

ItkManGetU3VDeviceInfo 获取 USB3.0 相机专有信息

C/C++	ITKSTATUS ItkManGetU3VDeviceInfo(uint32_t index, PITK_U3V_DEV_INFO pDI);	
参数说明	index	相机索引，有效范围[0, ItkManGetDeviceCount() - 1]
	pDI	USB3.0 相机专有设备信息
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_ARG_NULL : 输入参数为空 ITKSTATUS_DEVICE_ID_OUT_OF_RANGE : 相机索引越界 ITKSTATUS_NOT_IMPLEMENT : 设备接口不是 USB	
说明	获取 USB3.0 相机专有设备信息，对于非 USB3.0 相机设备，如 Camera Link 接口相机，本函数返回值是 ITKSTATUS_NOT_IMPLEMENT 。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	在调用本函数前，必须先调用 ItkManGetDeviceCount()	
相关函数	ItkManGetDeviceCount()	

示例代码

```
ITKSTATUS res = ItkManInitilize();
if (res == ITKSTATUS_OK)
{
    uint32_t count;
```

```

ItkManGetDeviceCount(&count);
for (uint32_t i=0; i < count; ++i)
{
    ITK_U3V_DEV_INFO DI;
    ItkManGetU3VDevInfo(i, &DI);
}
ItkManTerminate();
}
    
```

ItkManGetStatusText 获取状态码的信息

C/C++	ITKSTATUS ItkManGetStatusText(ITKSTATUS status, char * idBuf, uint32_t * idBufSize, char * levelBuf, uint32_t * levelBufSize, char * moduleBuf, uint32_t * moduleBufSize);	
参数说明	status	状态码
	idBuf	错误类别信息
	idBufSize	作为输入，指明 <i>idBuf</i> 的最大长度；作为输出，指明 <i>idBuf</i> 的有效长度
	levelBuf	错误级别信息
	levelBufSize	作为输入，指明 <i>levelBuf</i> 的最大长度；作为输出，指明 <i>levelBuf</i> 的有效长度
	moduleBuf	错误模块信息
返回值	moduleBufSize	作为输入，指明 <i>moduleBuf</i> 的最大长度；作为输出，指明 <i>moduleBuf</i> 的有效长度
	ITKSTATUS_OK : 调用成功	
	ITKSTATUS_ARG_OUT_OF_RANGE : 参数越界	
说明	ITKSTATUS_BUFFER_TOO_SMALL : 输入缓冲区太小	
	获取状态码的详细信息。	
	关联文件	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	

示例代码

无

ItkManGetPrm 获取全局参数

C++/C	ITKSTATUS ItkManGetPrm(uint32_t prm, void* pValue);	
参数说明	Prm	参数索引
	pValue	参数值
返回值	ITKSTATUS_OK : 调用成功	

	ITKSTATUS_INVALID_ARG: 无效的输入参数 ITKSTATUS_ARG_OUT_OF_RANGE: 参数越界
说 明	获取全局参数信息。
关联文件	IKapC.h
关 联 库	IKapC.lib
注意事项	无
相关函数	ItkManSetPrm()

示例代码

无

ItkManSetPrm 设置全局参数

C++/C	ITKSTATUS ItkManSetPrm(uint32_t prm, const void* value);	
参数说明	Prm	参数索引
	pValue	参数值
返 回 值	ITKSTATUS_OK: 调用成功 ITKSTATUS_INVALID_ARG: 无效的输入参数 ITKSTATUS_ARG_OUT_OF_RANGE: 参数越界	
说 明	设置全局参数信息。	
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	无	
相关函数	ItkManGetPrm()	

示例代码

无

ItkDevOpen 打开相机设备

C/C++	ITKSTATUS ItkDevOpen (uint32_t index, int accessMode, ITKDEVICE *phDev);	
参数说明	Index	相机索引，有效范围[0,ItkManGetDeviceCount() - 1]
	accessMode	相机控制模式，可以是下述任意值或者它们的组合
	phDev	相机设备句柄
返 回 值	ITKSTATUS_OK: 调用成功 ITKSTATUS_INVALID_ARG: 无效的输入参数 ITKSTATUS_DEVICE_ID_OUT_OF_RANGE: 相机索引越界 ITKSTATUS_DEVICE_PERMISSION_DENY: 设备访问拒绝（权限不足） ITKSTATUS_DEVICE_NOT_ACCESSABLE: 设备无法访问（可能正在被其他进程访问）	

说 明	<p>打开指定索引和访问模式的相机设备。注意，如果当前设备已经被打开且该设备支持多个程序同时读写操作，则可以以只读模式打开该设备。多个应用程序无法同时访问独占类型的设备。</p> <p>打开相机设备前，必须先调用 ItkManGetDeviceCount()枚举当前电脑上可用相机的数量。</p>	
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	相机访问模式 <i>accessMode</i> 可以是下列任意值或者它们的组合。	
	ITKDEV_VAL_ACCESS_MODE_CONTROL	允许读/写相机配置
	ITKDEV_VAL_ACCESS_MODE_STREAM	允许读取相机采集的图像数据
	ITKDEV_VAL_ACCESS_MODE_EXCLUSIVE	独占式访问设备。当设备被独占后，其他应用程序无法访问该设备
	ITKDEV_VAL_ACCESS_MODE_MONITOR	只读式访问设备参数
相关函数	ItkManGetDeviceCount(); ItkDevClose()	

示例代码

```

ITKSTATUS res = ItkManInitilize();
if (res == ITKSTATUS_OK)
{
    uint32_t count;
    ItkManGetDeviceCount(&count);
    if (count > 0)
    {
        ITKDEVICE hDev;
        ItkDevOpen(0, ITKDEV_VAL_ACCESS_MODE_CONTROL, &hDev);
        // 此处配置相机
        ItkDevClose(hDev);
    }
    ItkManTerminate();
}

```

ItkDevClose 关闭相机设备

C/C++	ITKSTATUS ItkDevClose(ITKDEVICE hDev);	
参数说明	hDev	相机设备句柄
返 回 值	ITKSTATUS_OK: 调用成功 ITKSTATUS_INVALID_HANDLE: 无效的句柄	
说 明	关闭相机设备。当设备关闭后除非被再次打开，否则无法被访问。	
关联文件	IKapC.h	
关 联 库	IKapC.lib	

注意事项	无
相关函数	ItkDevOpen()

示例代码

```
ITKSTATUS res = ItkManInitilize();
if (res == ITKSTATUS_OK)
{
    uint32_t count;
    ItkManGetDeviceCount(&count);
    if (count > 0)
    {
        ITKDEVICE hDev;
        ItkDevOpen(0, ITKDEV_VAL_ACCESS_MODE_CONTROL, &hDev);
        // 此处配置相机
        ItkDevClose(hDev);
    }
    ItkManTerminate();
}
```

ItkDevLoadConfigurationFromFile 加载相机设备配置文件

C/C++	ITKSTATUS ItkDevLoadConfigurationFromFile(ITKDEVICE hDev, char* lpFileName);	
参数说明	hDev	相机设备句柄
	lpFileName	相机设备文件名称，注意所有相机配置文件都是以.ccf 为后缀
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄	
说明	加载相机设备配置文件。 本函数必须要指明配置文件的全部路径。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevSaveConfigurationToFile()	

示例代码

无

ItkDevSaveConfigurationToFile 保存相机设备配置文件

C/C++	ITKSTATUS ItkDevSaveConfigurationToFile(ITKDEVICE hDev,char*
-------	--

	lpFileName);	
参数说明	hDev	相机设备句柄
	lpFileName	相机设备文件名称，注意所有相机配置文件都是以.ccf 为后缀
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄	
说明	保存相机设备配置文件。 本函数必须要指明配置文件的全部路径。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevLoadConfigurationFormFile()	

示例代码

无

ItkDevGetFeatureCount 获取相机支持配置特征的数量

C/C++	ITKSTATUS ItkDevGetFeatureCount(ITKDEVICE hDev, uint32_t* pCount);	
参数说明	hDev	相机设备句柄
	pCount	相机支持的特征数量
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄	
说明	获取相机支持配置特征的数量。 对于不同型号的相机，其支持的特征数量可能不同，请查阅相关相机手册获取其支持的具体特征类型。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevGetFeatureName()	

示例代码

```
ITKSTATUS res = ItkManInitilize();
if (res == ITKSTATUS_OK)
{
    uint32_t count;
    ItkManGetDeviceCount(&count);
    if (count > 0)
    {
        ITKDEVICE hDev;
        ItkDevOpen(0, ITKDEV_VAL_ACCESS_MODE_CONTROL, &hDev);
        // 此处配置相机
    }
}
```

```

uint32_t count;
ItkDevGetFeatureCount(hDev, &count);
ItkDevClose(hDev);
}
ItkManTerminate();
}

```

ItkDevGetFeatureName 获取相机特征的名称

C/C++	ITKSTATUS ItkDevGetFeatureName(ITKDEVICE hDev, uint32_t index, char* name, uint32_t* pNameSize);	
参数说明	hDev	相机设备句柄
	index	特征索引
	name	特征名称
	pNameSize	作为输入，该参数指明 <i>name</i> 的缓冲区最大长度；作为输出，该参数指明返回特征名称的有效长度，包括结尾处的空字符
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄 ITKSTATUS_BUFFER_TOO_SMALL : 输入缓冲区太小	
说明	获取相机支持配置特征名称。 当用户获取特征名称后，可以通过相机特征名称来控制相机参数。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevGetFeatureCount()	

示例代码

```

ITKSTATUS res = ItkManInitilize();
if (res == ITKSTATUS_OK)
{
    uint32_t count;
    ItkManGetDeviceCount(&count);
    if (count > 0)
    {
        ITKDEVICE hDev;
        ItkDevOpen(0, ITKDEV_VAL_ACCESS_MODE_CONTROL, &hDev);
        // 此处配置相机
        uint32_t featureCount;
        ItkDevGetFeatureCount(hDev, &featureCount);
        for (uint32_t i=0; i < featureCount; i++)
        {

```

```

uint32_t nameSize;
ItkDevGetFeatureName(hDev, i, NULL, &nameSize);
char* name = malloc(nameSize);
ItkDevGetFeatureName(hDev, i, name, &nameSize);
free(name);
}
ItkDevClose(hDev);
}
ItkManTerminate();
}

```

ItkDevAllocFeature 申请配置相机特征需要的资源

C/C++	ITKSTATUS ItkDevAllocFeature(ITKDEVICE hDev, const char* name, ITKFEATURE* phFeature);	
参数说明	hDev	相机设备句柄
	name	特征名称
	phFeature	相机特征句柄
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数	
说明	申请读/写相机特征所需要的资源。 用户可以通过返回的句柄控制相机的指定特征。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevFreeFeature()	

示例代码

```

ITKSTATUS res = ItkManInitilize();
if (res == ITKSTATUS_OK)
{
    uint32_t count;
    ItkManGetDeviceCount(&count);
    if (count > 0)
    {
        ITKDEVICE hDev;
        ItkDevOpen(0, ITKDEV_VAL_ACCESS_MODE_CONTROL, &hDev);
        // 此处配置相机
        ITKFEATURE hFeature;
        ItkDevAllocFeature(hDev, "ExposureTime", &hFeature);
        // 此处配置曝光时间特征
        ItkDevFreeFeature(hFeature);
    }
}

```



```

        ItkDevClose(hDev);
    }
    ItkManTerminate();
}

```

ItkDevFreeFeature 释放配置相机特征的资源

C/C++	ITKSTATUS ItkDevFreeFeature(ITKFEATURE* phFeature);	
参数说明	phFeature	相机特征句柄
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数	
说明	释放相机特征占用的资源。 当释放句柄后，则无法在通过该句柄配置相机参数。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevAllocFeature()	

示例代码

```

ITKSTATUS res = ItkManInitilize();
if (res == ITKSTATUS_OK)
{
    uint32_t count;
    ItkManGetDeviceCount(&count);
    if (count > 0)
    {
        ITKDEVICE hDev;
        ItkDevOpen(0, ITKDEV_VAL_ACCESS_MODE_CONTROL, &hDev);
        // 此处配置相机
        ITKFEATURE hFeature;
        ItkDevAllocFeature(hDev, "ExposureTime", &hFeature);
        // 此处配置曝光时间特征
        ItkDevFreeFeature(hFeature);
        ItkDevClose(hDev);
    }
    ItkManTerminate();
}

```

ItkDevGetStreamCount 获取相机数据流的数量

C/C++	ITKSTATUS ItkDevGetStreamCount(ITKDEVICE hDev, uint32_t* pCount);
-------	---

参数说明	hDev	相机设备句柄
	pCount	相机可用数据流的数量
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_IVNALID_HANDLE : 无效的句柄	
说明	获取相机可用数据流的数量。对于同一个相机，可能支持多个数据流同时传输图像数据。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevAllocStream(); ItkDevFreeStream()	

示例代码

```

ITKSTATUS res = ItkManInitilize();
if (res == ITKSTATUS_OK)
{
    uint32_t count;
    ItkManGetDeviceCount(&count);
    if (count > 0)
    {
        ITKDEVICE hDev;
        ItkDevOpen(0, ITKDEV_VAL_ACCESS_MODE_CONTROL, &hDev);
        // 获取数据流数量
        uint32_t streamCount;
        ItkDevGetStreamCount(hDev, &streamCount);
        // 创建缓冲区
        ITKBUFFER hBuffer;
        ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hBuffer);
        // 申请数据流资源
        ITKSTREAM hStream;
        ItkDevAllocStream(hDev, 0, hBuffer, &hStream);
        // 释放数据流资源
        ItkDevFreeStream(hStream);
        // 释放缓冲区
        ItkBufferFree(hBuffer);
        ItkDevClose(hDev);
    }
    ItkManTerminate();
}

```

ItkDevAllocStream 申请数据流采集过程中需要的资源

C/C++	ITKSTATUS ItkDevAllocStream(ITKDEVICE hDev, uint32_t index, ITKBUFFER hBuffer, ITKSTREAM* phStream);	
参数说明	hDev	相机设备句柄
	Index	数据流索引，有效范围[0, ItkDevGetStreamCount()-1]
	hBuffer	图像缓冲区用来存放采集过程中的图像数据
	phStream	数据流句柄
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INVALID_HANDLE : 无效的句柄	
说明	收集相机数据流采集过程中需要的资源。 对于同一个相机可能存在多个数据流同时采集数据，通过索引 <i>index</i> 来指明用户希望采集的数据流。 用户收集数据流资源过程中必须指定一个数据缓冲区，其将用来存放采集过程中的图像数据。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevFreeStream()	

示例代码

参见 “[ItkDevGetStreamCount](#)”

ItkDevFreeStream 释放数据流采集过程中申请的资源

C++/C	ITKSTATUS ItkDevFreeStream(ITKSTREAM hStream);	
参数说明	hStream	数据流句柄
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄	
说明	释放相机数据流采集过程中需要的资源。 如果存在正在进行的数据流采集过程，本函数将会停止数据采集并释放资源。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	本函数不会释放数据传输过程中的缓冲区资源，用户需要调用 ItkBufferFree() 释放缓冲区。	
相关函数	ItkDevAllocStream()	

示例代码

参见 “[ItkDevGetStreamCount](#)”

ItkDevGetEventCount 获取相机支持触发事件的数量

C++/C	ITKSTATUS ItkDevGetEventCount(ITKDEVICE hDev, uint32_t* pCount);	
参数说明	hDev	相机设备句柄
	pCount	相机支持触发事件的数量
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数	
说明	获取相机支持触发事件种类的数量。 用户可以为感兴趣的触发时间注册回调函数进行进一步的处理。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevGetEventName()	

示例代码

无

ItkDevGetEventName 获取相机支持触发事件的名称

C++/C	ITKSTATUS ItkDevGetEventName(ITKDEVICE hDev, uint32_t index, char* pEventName, uint32_t* pNameSize);	
参数说明	hDev	相机设备句柄
	index	相机支持触发事件的索引
	pEventName	相机触发事件的名称
	pNameSize	作为输入，该参数指明 <i>pEventName</i> 的最大有效长度； 作为输出，该参数指明 <i>pEventName</i> 的输出有效长度
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 IKTSTATUS_BUFFER_TOO_SMALL : 输入缓冲区太小	
说明	获取相机支持触发事件的名称。 用户可以为感兴趣的触发时间注册回调函数进行进一步的处理。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevGetEventName()	

示例代码

```
ITKSTATUS res = ItkManInitilize();
if (res == ITKSTATUS_OK)
{
    uint32_t count;
    ItkManGetDeviceCount(&count);
}
```

```

if (count > 0)
{
    ITKDEVICE hDev;
    ItkDevOpen(0, ITKDEV_VAL_ACCESS_MODE_CONTROL, &hDev);
    // 此处配置相机
    uint32_t eventCount;
    ItkDevGetEventCount(hDev, &eventCount);
    for (uint32_t i=0; i < eventCount; i++)
    {
        uint32_t nameSize;
        ItkDevGetEventName(hDev, i, NULL, &nameSize);
        char* name = malloc(nameSize);
        ItkDevGetEventName (hDev, i, name, &nameSize);
    }
    ItkDevClose(hDev);
}
ItkManTerminate();
}

```

ItkDevIsEventAvailable 判断相机是否支持该触发事件

C++/C	ITKSTATUS ItkDevIsEventAvailable(ITKDEVICE hDev, const char* pEventName, _Bool* bAvailable);	
参数说明	hDev	相机设备句柄
	pEventName	相机触发事件的名称
	bAvailable	0：相机不支持该事件触发 1：相机支持该事件触发
返回值	ITKSTATUS_OK ：调用成功 ITKSTATUS_INVALID_ARG ：无效的输入参数	
说明	获取相机是否支持指定名称的触发事件。 用户可以为感兴趣的触发时间注册回调函数进行进一步的处理。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevGetEventName()	

示例代码

无

ItkDevRegisterCallback 为相机触发事件注册回调函数

C++/C	ITKSTATUS ItkDevRegisterCallback(ITKDEVICE hDev, const char* pEventName, PITKEVENTINFOCALLBACK callback, void *pContext);	
参数说明	hDev	相机设备句柄
	pEventName	相机触发事件的名称
	callback	用户回调函数地址，其满足如下形式： void IKAPC_CC MyCallback(void *context, ITKEVENTINFO hEventInfo) { } }
	pContext	用户回调函数参数，可以为 NULL
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数	
说明	用户为相机事件注册回调函数前，请先调用 ItkDevIsEventAvailable() 判断该事件是否支持触发。 多次为相同事件注册回调函数，IKapC 仅会调用最近一次注册的回调函数。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevIsEventAvailable() ; ItkDevRegisterCallback()	

示例代码

无

ItkDevUnregisterCallback 清除注册的回调函数

C++/C	ITKSTATUS ItkDevRegisterCallback(ITKDEVICE hDev, const char* pEventName);	
参数说明	hDev	相机设备句柄
	pEventName	相机触发事件的名称
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数	
说明	清除用户注册的回调函数。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevIsEventAvailable() ; ItkDevRegisterCallback()	

示例代码

无

ItkDevGetPrm 获取相机参数

C++/C	ITKSTATUS ItkDevGetPrm(ITKDEVICE hDev, uint32_t prm, void* pValue);	
参数说明	hDev	相机设备句柄
	Prm	设备参数索引
	pValue	参数值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_ARG_OUT_OF_RANGE : 参数越界	
说明	获取设备参数信息。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevSetPrm()	

示例代码

```
// 获取设备心跳包时间长度
uint32_t heartbeat_timeout = 0;
ItkDevGetPrm(hDev,ITKDEV_PRM_HEARTBEAT_TIMEOUT,&heartbeat_timeout);
```

ItkDevSetPrm 设置相机参数

C++/C	ITKSTATUS ItkDevSetPrm(ITKDEVICE hDev, uint32_t prm, void* pValue);	
参数说明	hDev	相机设备句柄
	Prm	设备参数索引
	pValue	参数值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_ARG_OUT_OF_RANGE : 参数越界	
说明	设置设备参数信息。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevGetPrm()	

示例代码

```
// 设置设备心跳包时间长度为 5 分钟
uint32_t heartbeat_timeout = 300 * 1200;
ItkDevSetPrm(hDev,ITKDEV_PRM_HEARTBEAT_TIMEOUT,&heartbeat_timeout);
```

ItkDevPortRead 直接读取寄存器

C++/C	ITKSTATUS ItkDevPortRead(ITKDEVICE hDev, void* pBuffer, uint64_t address, uint32_t length);	
参数说明	hDev	相机设备句柄
	pBuffer	输出缓冲区
	address	寄存器地址
	Length	输出缓冲区长度
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_TIMEOUT : 操作超时	
说明	直接读取设备寄存器。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevPortWrite()	

示例代码

无

ItkDevPortWrite 直接写入寄存器

C++/C	ITKSTATUS ItkDevPortWrite(ITKDEVICE hDev, const void* pBuffer, uint64_t address, uint32_t length);	
参数说明	hDev	相机设备句柄
	pBuffer	输入缓冲区
	address	寄存器地址
	Length	输入缓冲区长度
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_TIMEOUT : 操作超时	
说明	直接写入设备寄存器。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevPortRead()	

示例代码

无

ItkDevSerialPortRead 从串口设备读取数据

C++/C	ITKSTATUS ItkDevSerialPortRead(ITKDEVICE hDev, void* pBuffer, uint32_t length, uint32_t timeout);	
参数说明	hDev	相机设备句柄
	pBuffer	输出串口缓冲区
	length	输出串口缓冲区长度
	timeout	读取等待时间，如从串口设备读取正确值或超时则退出
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_TIMEOUT : 操作超时	
说明	从串口设备读取数据，只对 Camera Link 相机有效。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevSerialPortWrite()	

示例代码

无

ItkDevSerialPortWrite 向串口设备发送数据

C++/C	ITKSTATUS ItkDevSerialPortWrite(ITKDEVICE hDev, const void* pBuffer, uint32_t length, uint32_t timeout);	
参数说明	hDev	相机设备句柄
	pBuffer	输入串口缓冲区
	length	输入串口缓冲区长度
	timeout	读取等待时间，如从串口设备读取正确值或超时则退出
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_TIMEOUT : 操作超时	
说明	向串口设备发送数据，只对 Camera Link 相机有效。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevSerialPortWrite()	

示例代码

无

ItkGigEDevForceIp 改变千兆网相机临时 IP 地址

C++/C	ITKSTATUS ItkGigEDevForceIp(const char* pMacAddress, const char* pIpAddress, const char* pSubnetMask, const char* pDefaultGateway);	
参数说明	pMacAddress	设备 MAC 地址
	pIpAddress	待设置 IP 地址
	pSubnetMask	待设置子网掩码
	pDefaultGateway	待设置默认网关
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_TIMEOUT : 操作超时	
说明	设置千兆网相机临时 IP 地址，相机断电重启后将会恢复原有 IP 地址。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	

示例代码

无

ItkGigEDevGetPersistentIpAddress 获取千兆网相机静态 IP 地址

C++/C	ITKSTATUS ItkGigEDevGetPersistentIpAddress(ITKDEVICE hDev, char* pIpAddress, uint32_t* pIpAddressLen, char* pSubnetMask, uint32_t* pSubnetMaskLen, char* pDefaultGateway, uint32_t* pDefaultGatewayLen);	
参数说明	hDev	设备句柄
	pIpAddress	设备静态 IP 地址
	pIpAddressLen	静态 IP 地址长度
	pSubnetMask	设备静态子网掩码
	pSubnetMaskLen	静态子网掩码长度
	pDefaultGateway	设备默认网关
	pDefaultGatewayLen	默认网关长度
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_NOT_IMPLEMENT : 设备不是千兆网接口设备 ITKSTATUS_TIMEOUT : 操作超时 ITKSTATUS_BUFFER_TOO_SMALL : 输入缓冲区太小	
说明	获取千兆网相机静态 IP 地址。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkGigEDevSetPersistentIp()	

示例代码

```
uint32_t ip_len = 0, subnet_len = 0, gateway_len = 0;
ItkGigEDevGetPersistentIpAddress(hDev, NULL, &ip_len, NULL, &subnet_len, NULL,
    &gateway_len);
char* ip = malloc(ip_len);
char* subnet = malloc(subnet_len);
char* gateway = malloc(gateway_len);
ItkGigEDevGetPersistentIpAddress(hDev, ip, &ip_len, subnet, &subnet_len, gateway,
&gateway_len);
```

ItkGigEDevSetPersistentIpAddress 设置千兆网相机静态 IP 地址

C++/C	ITKSTATUS ItkGigEDevSetPersistentIpAddress(ITKDEVICE hDev, const char* pIpAddress, const char* pSubnetMask, const char* pDefaultGateway);	
参数说明	hDev	设备句柄
	pIpAddress	带设置设备静态 IP 地址
	pSubnetMaskLen	带设置静态子网掩码长度
	pDefaultGateway	带设置设备默认网关
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_NOT_IMPLEMENT : 设备不是千兆网接口设备 ITKSTATUS_TIMEOUT : 操作超时	
说明	设置千兆网相机静态 IP 地址。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkGigEDevGetPersistentIp()	

示例代码

```
ItkGigEDevGetPersistentIpAddress(hDev, "192.168.1.12", "255.255.255.0", "192.168.1.1");
```

ItkEventInfoGetPrm 获取事件参数信息

C++/C	ITKSTATUS ItkEventInfoGetPrm(ITKEVENTINFO hEventInfo, uint32_t prm, void* value);	
参数说明	hEventInfo	事件信息句柄
	prm	事件信息索引
	Value	事件信息输出缓冲区
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_NOT_IMPLEMENT : 设备不是千兆网接口设备	
说明	获取事件参数信息。	
关联文件	IKapC.h	

关 联 库	IKapC.lib
注意事项	无
相关函数	ItkDevOpen()

示例代码

```
void __stdcall featureValueChangeCallback(void* context, ITKEVENTINFO eventInfo)
{
    uint32_t type = 0, feature_index = 0, feature_name_length = 64;
    uint64_t countstamp = 0;
    char feature_name[64] = { 0 };
    ITKDEVICE hDev = (ITKDEVICE)context;

    ITKSTATUS res = ItkEventInfoGetPrm(eventInfo, ITKEVENTINFO_PRM_TYPE,
&type);
    CHECK(res);

    res = ItkEventInfoGetPrm(eventInfo, ITKEVENTINFO_PRM_FEATURE_INDEX,
&feature_index);
    CHECK(res);

    res = ItkEventInfoGetPrm(eventInfo, ITKEVENTINFO_PRM_HOST_TIME_STAMP,
&countstamp);
    CHECK(res);

    res = ItkDevGetFeatureName(hDev, feature_index, feature_name, &feature_name_length);
    CHECK(res);

    #if __STDC__ VERSION__ >= 199901L
    printf("OnFeatureValueChangeCallback, feature: %s, type: %08X, time: %lld.\n", feature_name,
        type, countstamp);
    #else
    printf("OnFeatureValueChangeCallback, feature: %s, type: %08X, time: %l64d.\n",
        feature_name, type, countstamp);
    #endif
}
```

ItkFeatureGetAccessMode 获取相机特征的访问模式

C++/C	ITKSTATUS ItkFeatureGetAccessMode(ITKFEATURE hFeature, uint32_t* pAccessMode);	
参数说明	hFeature	相机特征句柄
	pAccessMode	相机特征的访问模式
返 回 值	ITKSTATUS_OK: 调用成功	

	ITKSTATUS_INVALID_ARG: 无效的输入参数	
说 明	获取相机特征的访问模式，可以是下列任意值或者它们的结合。	
	ITKFEATURE_VAL_ACCESS_MODE_UNDEFINED	访问模式未定义
	ITKFEATURE_VAL_ACCESS_MODE_RW	读写访问模式
	ITKFEATURE_VAL_ACCESS_MODE_RO	只读访问模式
	ITKFEATURE_VAL_ACCESS_MODE_WO	只写访问模式
	ITKFEATURE_VAL_ACCESS_MODE_NI	该特征未实现，无法访问
	ITKFEATURE_VAL_ACCESS_MODE_NA	该特征当前状态下不可用，无法访问
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	无	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevGetAccessMode 获取相机特征的访问模式

C++/C	ITKSTATUS ItkDevGetAccessMode (ITKDEVICE hDev, const char* featureName, uint32_t* pAccessMode);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	pAccessMode	相机特征的访问模式
返 回 值	ITKSTATUS_OK: 调用成功 ITKSTATUS_INVALID_ARG: 无效的输入参数	
说 明	获取设备特征的访问模式，可以是下列任意值或者它们的结合。	
	ITKFEATURE_VAL_ACCESS_MODE_UNDEFINED	访问模式未定义
	ITKFEATURE_VAL_ACCESS_MODE_RW	读写访问模式
	ITKFEATURE_VAL_ACCESS_MODE_RO	只读访问模式
	ITKFEATURE_VAL_ACCESS_MODE_WO	只写访问模式
	ITKFEATURE_VAL_ACCESS_MODE_NI	该特征未实现，无法访问
	ITKFEATURE_VAL_ACCESS_MODE_NA	该特征当前状态下不可用，无法访问
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	无	

示例代码

无

ItkFeatureGetCategory 获取相机特征的类别

C++/C	ITKSTATUS ItkFeatureGetCategory (ITKFEATURE hFeature, char* pCategory, uint32_t* pBufferSize);	
参数说明	phFeature	相机特征句柄
	pCategory	相机特征的所属类型
	pBufferSize	作为输入，该参数指明 <i>pCategory</i> 的最大长度；作为输出，该参数指明 <i>pCategory</i> 返回值的有效长度
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数	
说明	获取相机特征的所属类型，不同的相机特征可能属于同一类型。比如：“DigitalOffset”和“DigitalGain”都归属于“DigitalControl”类型。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevGetCategory 获取相机特征的类别

C++/C	ITKSTATUS ItkDevGetCategory (ITKDEVICE hDev, const char* featureName, char* pCategory, uint32_t* pBufferSize);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	pCategory	相机特征的所属类型
	pBufferSize	作为输入，该参数指明 <i>pCategory</i> 的最大长度；作为输出，该参数指明 <i>pCategory</i> 返回值的有效长度
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数	
说明	获取相机特征的所属类型，不同的相机特征可能属于同一类型。比如：“DigitalOffset”和“DigitalGain”都归属于“DigitalControl”类型。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	

示例代码

无

ItkFeatureGetType 获取相机特征的数据类型

C++/C	ITKSTATUS ItkFeatureGetType(ITKFEATURE hFeature, uint32_t type);	
参数说明	phFeature	相机特征句柄
	type	相机特征的数据类型
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数	
说明	获取相机特征的数据类型。对于不同数据类型的相机特征，用户应该选择相应的 API 进行访问。相机的数据类型可以是一下任意值：	
	ITKFEATURE_VAL_TYPE_UNDEFINED	数据类型未定义
	ITKFEATURE_VAL_TYPE_INT32	32 位整数类型
	ITKFEATURE_VAL_TYPE_INT64	64 位整数类型
	ITKFEATURE_VAL_TYPE_FLOAT	32 位浮点数类型
	ITKFEATURE_VAL_TYPE_DOUBLE	64 位浮点数类型
	ITKFEATURE_VAL_TYPE_BOOL	布尔数据类型
	ITKFEATURE_VAL_TYPE_ENUM	枚举数据类型
	ITKFEATURE_VAL_TYPE_STRING	字符串数据类型
	ITKFEATURE_VAL_TYPE_COMMAND	命令数据类型
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevGetType 获取相机特征的数据类型

C++/C	ITKSTATUS ItkDevGetType (ITKDEVICE hDev, const char* featureName, uint32_t type);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	type	相机特征的数据类型
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数	
说明	获取相机特征的数据类型。对于不同数据类型的相机特征，用户应该选择相应的 API 进行访问。相机的数据类型可以是一下任意值：	
	ITKFEATURE_VAL_TYPE_UNDEFINED	数据类型未定义
	ITKFEATURE_VAL_TYPE_INT32	32 位整数类型

	ITKFEATURE_VAL_TYPE_INT64	64 位整数类型
	ITKFEATURE_VAL_TYPE_FLOAT	32 位浮点数类型
	ITKFEATURE_VAL_TYPE_DOUBLE	64 位浮点数类型
	ITKFEATURE_VAL_TYPE_BOOL	布尔数据类型
	ITKFEATURE_VAL_TYPE_ENUM	枚举数据类型
	ITKFEATURE_VAL_TYPE_STRING	字符串数据类型
	ITKFEATURE_VAL_TYPE_COMMAND	命令数据类型
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	无	

示例代码

无

ItkFeatureGetDisplayName 获取相机特征的显示名称

C++/C	ITKSTATUS ItkFeatureGetDisplayName (ITKFEATURE hFeature, char* featureName, uint32_t* pNameSize);	
参数说明	phFeature	相机特征句柄
	featureName	相机特征的显示名称
	pBufferSize	作为输入，该参数指明 <i>featureName</i> 的最大长度；作为输出，该参数指明 <i>featureName</i> 返回值的有效长度
返 回 值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数	
说 明	获取相机特征的显示名称。注意相机特征的显示名称和名称可能有所不同，前者用于在用户 GUI 界面中显示特征，后者则唯一标示了相机特征。	
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	无	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevGetDisplayName 获取相机特征的显示名称

C++/C	ITKSTATUS ItkDevGetDisplayName (ITKDEVICE hDev, const char* featureName, char* featureDisplayName, uint32_t* pNameSize);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称

	featureName	相机特征的显示名称
	pBufferSize	作为输入，该参数指明 <i>featureName</i> 的最大长度；作为输出，该参数指明 <i>featureName</i> 返回值的有效长度
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数	
说明	获取相机特征的显示名称。注意相机特征的显示名称和名称可能有所不同，前者用于在用户 GUI 界面中显示特征，后者则唯一标示了相机特征。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	

示例代码

无

ItkFeatureGetTooltip 获取相机特征的提示信息

C++/C	ITKSTATUS ItkFeatureGetTooltip (ITKFEATURE hFeature, char* pBuff, uint32_t* pBufLen);	
参数说明	phFeature	相机特征句柄
	pBuff	相机特征的提示信息
	pBufLen	作为输入，该参数指明 <i>pBuff</i> 的最大长度；作为输出，该参数指明 <i>pBuff</i> 返回值的有效长度
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数	
说明	获取相机特征的提示信息。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevGetTooltip 获取相机特征的提示信息

C++/C	ITKSTATUS ItkDevGetTooltip (ITKDEVICE hDev, const char* featureName, char* pBuff, uint32_t* pBufLen);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	pBuff	相机特征的提示信息

	pBufLen	作为输入，该参数指明 <i>pBuff</i> 的最大长度；作为输出，该参数指明 <i>pBuff</i> 返回值的有效长度
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数	
说明	获取相机特征的提示信息。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	

示例代码

无

ItkFeatureGetDescription 获取相机特征的具体描述信息

C++/C	ITKSTATUS ItkFeatureGetDescription (ITKFEATURE hFeature, char* pBuff, uint32_t* pBufLen);	
参数说明	phFeature	相机特征句柄
	pBuff	相机特征的详细说明信息
	pBufLen	作为输入，该参数指明 <i>pBuff</i> 的最大长度；作为输出，该参数指明 <i>pBuff</i> 返回值的有效长度
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数	
说明	获取相机特征的详细说明信息。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevGetDescription 获取相机特征的具体描述信息

C++/C	ITKSTATUS ItkDevGetDescription (ITKDEVICE hDev, const char* featureName, char* pBuff, uint32_t* pBufLen);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	pBuff	相机特征的详细说明信息
	pBufLen	作为输入，该参数指明 <i>pBuff</i> 的最大长度；作为输出，该参数指明 <i>pBuff</i> 返回值的有效长度

返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数
说明	获取相机特征的详细说明信息。
关联文件	IKapC.h
关联库	IKapC.lib
注意事项	无

示例代码

无

ItkFeatureGetNamespace 获取相机特征的命名空间

C++/C	ITKSTATUS ItkFeatureGetNamespace (ITKFEATURE hFeature, uint32_t* pNamespace);	
参数说明	phFeature	相机特征句柄
	pNamespace	相机特征的命名空间
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数	
说明	获取相机特征的名称空间，可能是下列值：	
	ITKFEATURE_VAL_NAME_SPACE_UNDEFINED	命名空间未定义
	ITKFEATURE_VAL_NAME_SPACE_CUSTOM	自定义命名空间
	ITKFEATURE_VAL_NAME_SPACE_STANDARD	标准命名空间
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevGetNamespace 获取相机特征的命名空间

C++/C	ITKSTATUS ItkDevGetNamespace (ITKDEVICE hDev, const char* featureName, uint32_t* pNamespace);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	pNamespace	相机特征的命名空间
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数	
说明	获取相机特征的命名空间，可能是下列值：	

	ITKFEATURE_VAL_NAME_SPACE_UNDEFINED	命名空间未定义
	ITKFEATURE_VAL_NAME_SPACE_CUSTOM	自定义命名空间
	ITKFEATURE_VAL_NAME_SPACE_STANDARD	标准命名空间
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	无	

示例代码

无

ItkFeatureGetVisibility 获取相机特征可视权限

C++/C	ITKSTATUS ItkFeatureGetVisibility(ITKFEATURE hFeature, uint32_t* pVis);	
参数说明	hFeature	相机特征句柄
	pVis	相机特征的可视性
返 回 值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数	
说 明	获取相机特征的可视性，可能是下列值。	
	ITKFEATURE_VAL_VISIBILITY_UNDEFINED	可视性未定义
	ITKFEATURE_VAL_VISIBILITY_BEGINNER	所有用户可见
	ITKFEATURE_VAL_VISIBILITY_EXPERT	专业人士可见
	ITKFEATURE_VAL_VISIBILITY_GURU	专家可见
	ITKFEATURE_VAL_VISIBILITY_INVISIBLE	特征不可见
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	无	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevGetVisibility 获取相机特征可视权限

C++/C	ITKSTATUS ItkDevGetVisibility (ITKDEVICE hDev, const char* featureName, uint32_t* pVis);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	pVis	相机特征的可视性
返 回 值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数	

说 明	获取相机特征的可视性，可能是下列值。	
	ITKFEATURE_VAL_VISIBILITY_UNDEFINED	可视性未定义
	ITKFEATURE_VAL_VISIBILITY_BEGINNER	所有用户可见
	ITKFEATURE_VAL_VISIBILITY_EXPERT	专业人士可见
	ITKFEATURE_VAL_VISIBILITY_GURU	专家可见
	ITKFEATURE_VAL_VISIBILITY_INVISIBLE	特征不可见
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	无	

示例代码

无

ItkFeatureGetRepresentation 获取相机特征表示方式

C++/C	ITKSTATUS ItkFeatureGetRepresentation(ITKFEATURE hFeature, uint32_t* pRep);	
参数说明	hFeature	相机特征句柄
	pRep	相机特征的表示方式
返 回 值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数	
说 明	获取相机特征的可视性，可能是下列值：	
	ITKFEATURE_VAL_REPRESENTATION_UNDEFINED	方式未定义
	ITKFEATURE_VAL_REPRESENTATION_LINEAR	线性
	ITKFEATURE_VAL_REPRESENTATION_LOGARITHMIC	对数
	ITKFEATURE_VAL_REPRESENTATION_BOOLEAN	布尔型
	ITKFEATURE_VAL_REPRESENTATION_PURENUMBER	十进制数
	ITKFEATURE_VAL_REPRESENTATION_HEXNUMBER	十六进制数
	ITKFEATURE_VAL_REPRESENTATION_IPV4ADDRESS	IPV4 地址
	ITKFEATURE_VAL_REPRESENTATION_MACADDRESS	物理地址
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	只对整数特征类型有效，对于其它特征，表示方式取值为 ITKFEATURE_VAL_REPRESENTATION_UNDEFINED	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevGetRepresentation 获取相机特征表示方式

C++/C	ITKSTATUS ItkDevGetRepresentation (ITKDEVICE hDev, const char* featureName, uint32_t* pRep);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	pRep	相机特征的表示方式
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数	
说明	获取相机特征的可视性，可能是下列值：	
	ITKFEATURE_VAL_REPRESENTATION_UNDEFINED	方式未定义
	ITKFEATURE_VAL_REPRESENTATION_LINEAR	线性
	ITKFEATURE_VAL_REPRESENTATION_LOGARITHMIC	对数
	ITKFEATURE_VAL_REPRESENTATION_BOOLEAN	布尔型
	ITKFEATURE_VAL_REPRESENTATION_PURENUMBER	十进制数
	ITKFEATURE_VAL_REPRESENTATION_HEXNUMBER	十六进制数
	ITKFEATURE_VAL_REPRESENTATION_IPV4ADDRESS	IPV4 地址
	ITKFEATURE_VAL_REPRESENTATION_MACADDRESS	物理地址
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	只对整数特征类型有效，对于其它特征，表示方式取值为 ITKFEATURE_VAL_REPRESENTATION_UNDEFINED	

示例代码

无

ItkFeatureGetInt32Min 获取 32 位整数类型特征的最小值

C++/C	ITKSTATUS ItkFeatureGetInt32Min(ITKFEATURE hFeature, int32_t* pValue);	
参数说明	phFeature	相机特征句柄
	pValue	32 位整数类型的最小值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说明	获取 32 位整数类型的最小值。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	

相关函数 **ItkDevAllocFeature()**

示例代码

无

ItkDevGetInt32Min 获取 32 位整数类型特征的最小值

C++/C	ITKSTATUS ItkDevGetInt32Min (ITKDEVICE hDev, const char* featureName, int32_t* pValue);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	pValue	32 位整数类型的最小值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说明	获取 32 位整数类型的最小值。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	

示例代码

无

ItkFeatureGetInt32Max 获取 32 位整数类型特征的最大值

C++/C	ITKSTATUS ItkFeatureGetInt32Max(ITKFEATURE hFeature, int32_t* pValue);	
参数说明	phFeature	相机特征句柄
	pValue	32 位整数类型的最大值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说明	获取 32 位整数类型的最大值。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevGetInt32Max 获取 32 位整数类型特征的最大值

C++/C	ITKSTATUS ItkDevGetInt32Max (ITKDEVICE hDev, const char* featureName, int32_t* pValue);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	pValue	32 位整数类型的最大值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说明	获取 32 位整数类型的最大值。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	

示例代码

无

ItkFeatureGetInt32Inc 获取 32 位整数类型特征的步长

C++/C	ITKSTATUS ItkFeatureGetInt32Inc (ITKFEATURE hFeature, int32_t* pValue);	
参数说明	phFeature	相机特征句柄
	pValue	32 位整数类型的单位增量
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说明	获取 32 位整数类型的单位增量。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevGetInt32Inc 获取 32 位整数类型特征的步长

C++/C	ITKSTATUS ItkDevGetInt32Inc (ITKDEVICE hDev, const char* featureName, int32_t* pValue);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	pValue	32 位整数类型的单位增量
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说明	获取 32 位整数类型的单位增量。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	

示例代码

无

ItkFeatureGetInt32 获取 32 位整数类型特征的值

C++/C	ITKSTATUS ItkFeatureGetInt32 (ITKFEATURE hFeature, int32_t* pValue);	
参数说明	phFeature	相机特征句柄
	pValue	32 位整数类型的相机特征的值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说明	获取 32 位整数类型的单位增量。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	对于访问模式不是 ITKFEATURE_VAL_ACCESS_MODE_RW 或者 ITKFEATURE_VAL_ACCESS_MODE_RO 的相机特征，无法调用该函数获取特征值。	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevGetInt32 获取 32 位整数类型特征的值

C++/C	ITKSTATUS ItkDevGetInt32 (ITKDEVICE hDev, const char* featureName, int32_t* pValue);	
参数说明	hDev	相机设备的句柄
	featureName	相机特征的名称
	pValue	32 位整数类型的相机特征的值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说明	获取 32 位整数类型的单位增量。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	对于访问模式不是 ITKFEATURE_VAL_ACCESS_MODE_RW 或者 ITKFEATURE_VAL_ACCESS_MODE_RO 的相机特征，无法调用该函数获取特征值。	

示例代码

无

ItkFeatureSetInt32 设置 32 位整数类型特征的值

C++/C	ITKSTATUS ItkFeatureSetInt32 (ITKFEATURE hFeature, int32_t value);	
参数说明	phFeature	相机特征句柄
	value	32 位整数类型的相机特征的值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说明	设置 32 位整数类型特征的值。 设定值的取值范围一定是在[ItkFeatureGetInt32Min() , ItkFeatureGetInt32Max()] 之间。 设定值的取值同时也应该满足: $value = ItkFeatureGetInt32Min() + N * ItkFeatureGetInt32Inc(), N=0,1,2,\dots$	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	对于访问模式不是 ITKFEATURE_VAL_ACCESS_MODE_RW 或者	

	ITKFEATURE_VAL_ACCESS_MODE_WO 的相机特征，无法调用该函数设置特征值。
相关函数	ItkDevAllocFeature()

示例代码

无

ItkDevSetInt32 设置 32 位整数类型特征的值

C++/C	ITKSTATUS ItkDevSetInt32 (ITKDEVICE hDev, const char* featureName, int32_t value);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	value	32 位整数类型的相机特征的值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说明	设置 32 位整数类型特征的值。 设定值的取值范围一定是在[ItkFeatureGetInt32Min() , ItkFeatureGetInt32Max()] 之间。 设定值的取值同时也应该满足: $value = ItkFeatureGetInt32Min() + N * ItkFeatureGetInt32Inc(), N=0,1,2,\dots$	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	对于访问模式不是 ITKFEATURE_VAL_ACCESS_MODE_RW 或者 ITKFEATURE_VAL_ACCESS_MODE_WO 的相机特征，无法调用该函数设置特征值。	

示例代码

无

ItkFeatureGetInt64Min 获取 64 位整数类型特征的最小值

C++/C	ITKSTATUS ItkFeatureGetInt64Min(ITKFEATURE hFeature, int64_t* pValue);	
参数说明	phFeature	相机特征句柄
	pValue	64 位整数类型的最小值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数	

	ITKSTATUS_NOT_IMPLEMENT: 相机特征读写方法未实现
说 明	获取 64 位整数类型的最小值。
关联文件	IKapC.h
关 联 库	IKapC.lib
注意事项	无
相关函数	ItkDevAllocFeature()

示例代码

无

ItkDevGetInt64Min 获取 64 位整数类型特征的最小值

C++/C	ITKSTATUS ItkDevGetInt64Min (ITKDEVICE hDev, const char* featureName, int64_t* pValue);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	pValue	64 位整数类型的最小值
返 回 值	ITKSTATUS_OK: 调用成功 ITKSTATUS_INVALID_ARG: 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE: 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT: 相机特征读写方法未实现	
说 明	获取 64 位整数类型的最小值。	
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	无	

示例代码

无

ItkFeatureGetInt64Max 获取 64 位整数类型特征的最大值

C++/C	ITKSTATUS ItkFeatureGetInt64Max(ITKFEATURE hFeature, int64_t* pValue);	
参数说明	phFeature	相机特征句柄
	pValue	64 位整数类型的最大值
返 回 值	ITKSTATUS_OK: 调用成功 ITKSTATUS_INVALID_ARG: 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE: 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT: 相机特征读写方法未实现	

说 明	获取 64 位整数类型的最大值。
关联文件	IKapC.h
关 联 库	IKapC.lib
注意事项	无
相关函数	ItkDevAllocFeature()

示例代码

无

ItkDevGetInt64Max 获取 64 位整数类型特征的最大值

C++/C	ITKSTATUS ItkDevGetInt64Max (ITKDEVICE hDev, const char* featureName, int64_t* pValue);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	pValue	64 位整数类型的最大值
返 回 值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说 明	获取 64 位整数类型的最大值。	
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	无	

示例代码

无

ItkFeatureGetInt64Inc 获取 64 位整数类型特征的步长

C++/C	ITKSTATUS ItkFeatureGetInt64Inc (ITKFEATURE hFeature, int64_t* pValue);	
参数说明	phFeature	相机特征句柄
	pValue	64 位整数类型的单位增量
返 回 值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说 明	获取 64 位整数类型的单位增量。	

关联文件	IKapC.h
关 联 库	IKapC.lib
注意事项	无
相关函数	ItkDevAllocFeature()

示例代码

无

ItkDevGetInt64Inc 获取 64 位整数类型特征的步长

C++/C	ITKSTATUS ItkDevGetInt64Inc (ITKDEVICE hDev, const char* featureName, int64_t* pValue);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	pValue	64 位整数类型的单位增量
返 回 值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说 明	获取 64 位整数类型的单位增量。	
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	无	

示例代码

无

ItkFeatureGetInt64 获取 64 位整数类型特征的值

C++/C	ITKSTATUS ItkFeatureGetInt64 (ITKFEATURE hFeature, int64_t* pValue);	
参数说明	phFeature	相机特征句柄
	pValue	64 位整数类型的相机特征的值
返 回 值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说 明	获取 64 位整数类型特征的值。	
关联文件	IKapC.h	
关 联 库	IKapC.lib	

注意事项	对于访问模式不是 ITKFEATURE_VAL_ACCESS_MODE_RW 或者 ITKFEATURE_VAL_ACCESS_MODE_RO 的相机特征，无法调用该函数获取特征值。
相关函数	ItkDevAllocFeature()

示例代码

无

ItkDevGetInt64 获取 64 位整数类型特征的值

C++/C	ITKSTATUS ItkDevGetInt64 (ITKDEVICE hDev, const char* featureName, int64_t* pValue);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	pValue	64 位整数类型的相机特征的值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说明	获取 64 位整数类型特征的值。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	对于访问模式不是 ITKFEATURE_VAL_ACCESS_MODE_RW 或者 ITKFEATURE_VAL_ACCESS_MODE_RO 的相机特征，无法调用该函数获取特征值。	

示例代码

无

ItkFeatureSetInt64 设置 64 位整数类型特征的值

C++/C	ITKSTATUS ItkFeatureSetInt64 (ITKFEATURE hFeature, int64_t value);	
参数说明	phFeature	相机特征句柄
	value	64 位整数类型的相机特征的值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说明	设置 64 位整数类型特征的值。 设定值的取值范围一定是在 [ItkFeatureGetInt64Min() , ItkFeatureGetInt64Max()] 之间。	

	设定值的取值同时也应该满足： $value = \text{ItkFeatureGetInt64Min}() + N * \text{ItkFeatureGetInt64Inc}()$, $N=0,1,2,\dots$
关联文件	IKapC.h
关 联 库	IKapC.lib
注意事项	对于访问模式不是 ITKFEATURE_VAL_ACCESS_MODE_RW 或者 ITKFEATURE_VAL_ACCESS_MODE_WO 的相机特征，无法调用该函数设置特征值。
相关函数	ItkDevAllocFeature()

示例代码

无

ItkDevSetInt64 设置 64 位整数类型特征的值

C++/C	ITKSTATUS ItkDevSetInt64 (ITKDEVICE hDev, const char* featureName, int64_t value);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	value	64 位整数类型的相机特征的值
返 回 值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说 明	设置 64 位整数类型特征的值。 设定值的取值范围一定是在 $[\text{ItkFeatureGetInt64Min}(), \text{ItkFeatureGetInt64Max}()]$ 之间。 设定值的取值同时也应该满足： $value = \text{ItkFeatureGetInt64Min}() + N * \text{ItkFeatureGetInt64Inc}()$, $N=0,1,2,\dots$	
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	对于访问模式不是 ITKFEATURE_VAL_ACCESS_MODE_RW 或者 ITKFEATURE_VAL_ACCESS_MODE_WO 的相机特征，无法调用该函数设置特征值。	

示例代码

无

ItkFeatureGetDoubleMin 获取浮点数类型特征的最小值

C++/C	ITKSTATUS ItkFeatureGetDoubleMin(ITKFEATURE hFeature, double* pValue);	
参数说明	phFeature	相机特征句柄

	pValue	64 位浮点数类型的最小值
返 回 值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说 明	获取 64 位浮点数类型的最小值。	
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	无	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevGetDoubleMin 获取浮点数类型特征的最小值

C++/C	ITKSTATUS ItkDevGetDoubleMin (ITKDEVICE hDev, const char* featureName, double* pValue);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	pValue	64 位浮点数类型的最小值
返 回 值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说 明	获取 64 位浮点数类型的最小值。	
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	无	

示例代码

无

ItkFeatureGetDoubleMax 获取浮点数类型特征的最大值

C++/C	ITKSTATUS ItkFeatureGetDoubleMax(ITKFEATURE hFeature, double* pValue);	
参数说明	phFeature	相机特征句柄
	pValue	64 位浮点数类型的最大值

返回 值	ITKSTATUS_OK : 调用成功
	ITKSTATUS_INVALID_ARG : 无效的输入参数
	ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数
	ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现
说 明	获取 64 位浮点数类型的最大值。
关联文件	IKapC.h
关 联 库	IKapC.lib
注意事项	无
相关函数	ItkDevAllocFeature()

示例代码

无

ItkDevGetDoubleMax 获取浮点数类型特征的最大值

C++/C	ITKSTATUS ItkDevGetDoubleMax (ITKDEVICE hDev, const char* featureName, double* pValue);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	pValue	64 位浮点数类型的最大值
返回 值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说 明	获取 64 位浮点数类型的最大值。	
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	无	

示例代码

无

ItkFeatureGetDoubleInc 获取浮点数类型特征的步长

C++/C	ITKSTATUS ItkFeatureGetDoubleInc(ITKFEATURE hFeature, double* pValue);	
参数说明	phFeature	相机特征句柄
	pValue	64 位浮点数类型的增量值
返回 值	ITKSTATUS_OK : 调用成功	

	ITKSTATUS_INVALID_ARG: 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE: 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT: 相机特征读写方法未实现
说 明	获取 64 位浮点数类型的增量值。
关联文件	IKapC.h
关 联 库	IKapC.lib
注意事项	无
相关函数	ItkDevAllocFeature()

示例代码

无

ItkDevGetDoubleInc 获取浮点数类型特征的步长

C++/C	ITKSTATUS ItkDevGetDoubleInc (ITKDEVICE hDev, const char* featureName, double* pValue);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	pValue	64 位浮点数类型的增量值
返 回 值	ITKSTATUS_OK: 调用成功 ITKSTATUS_INVALID_ARG: 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE: 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT: 相机特征读写方法未实现	
说 明	获取 64 位浮点数类型的增量值。	
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	无	

示例代码

无

ItkFeatureGetDouble 获取浮点数类型特征的值

C++/C	ITKSTATUS ItkFeatureGetDouble (ITKFEATURE hFeature, double* pValue);	
参数说明	phFeature	相机特征句柄
	pValue	64 位浮点数类型的相机特征的值
返 回 值	ITKSTATUS_OK: 调用成功 ITKSTATUS_INVALID_ARG: 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE: 相机特征不匹配当前输入	

	参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现
说 明	获取 64 位浮点数类型的值。
关联文件	IKapC.h
关 联 库	IKapC.lib
注意事项	对于访问模式不是 ITKFEATURE_VAL_ACCESS_MODE_RW 或者 ITKFEATURE_VAL_ACCESS_MODE_RO 的相机特征，无法调用该函数获取特征值。
相关函数	ItkDevAllocFeature()

示例代码

无

ItkDevGetDouble 获取浮点数类型特征的值

C++/C	ITKSTATUS ItkDevGetDouble (ITKDEVICE hDev, const char* featureName, double* pValue);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	pValue	64 位浮点数类型的相机特征的值
返 回 值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说 明	获取 64 位浮点数类型的值。	
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	对于访问模式不是 ITKFEATURE_VAL_ACCESS_MODE_RW 或者 ITKFEATURE_VAL_ACCESS_MODE_RO 的相机特征，无法调用该函数获取特征值。	

示例代码

无

ItkFeatureSetDouble 设置浮点数类型特征的值

C++/C	ITKSTATUS ItkFeatureSetDouble (ITKFEATURE hFeature, double value);	
参数说明	phFeature	相机特征句柄
	value	64 位浮点数类型的相机特征的值
返 回 值	ITKSTATUS_OK : 调用成功	

	ITKSTATUS_INVALID_ARG: 无效的输入参数 ITKSTATUS_NOT_IMPLEMENT: 相机特征读写方法未实现
说 明	设置 64 位浮点数类型特征的值。 设定值的取值范围一定是在[ItkFeatureGetDoubleMin() , ItkFeatureGetDoubleMax()] 之间。
关联文件	IKapC.h
关 联 库	IKapC.lib
注意事项	对于访问模式不是 ITKFEATURE_VAL_ACCESS_MODE_RW 或者 ITKFEATURE_VAL_ACCESS_MODE_WO 的相机特征，无法调用该函数设置特征值。
相关函数	ItkDevAllocFeature()

示例代码

无

ItkDevSetDouble 设置浮点数类型特征的值

C++/C	ITKSTATUS ItkDevSetDouble (ITKDEVICE hDev, const char* featureName, double value);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	value	64 位浮点数类型的相机特征的值
返 回 值	ITKSTATUS_OK: 调用成功 ITKSTATUS_INVALID_ARG: 无效的输入参数 ITKSTATUS_NOT_IMPLEMENT: 相机特征读写方法未实现	
说 明	设置 64 位浮点数类型特征的值。 设定值的取值范围一定是在[ItkFeatureGetDoubleMin() , ItkFeatureGetDoubleMax()] 之间。	
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	对于访问模式不是 ITKFEATURE_VAL_ACCESS_MODE_RW 或者 ITKFEATURE_VAL_ACCESS_MODE_WO 的相机特征，无法调用该函数设置特征值。	

示例代码

无

ItkFeatureGetBool 获取布尔数据类型特征的值

C++/C	ITKSTATUS ItkFeatureGetBool (ITKFEATURE hFeature, _Bool* pValue);	
参数说明	phFeature	相机特征句柄

	pValue	布尔类型的相机特征的值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说明	获取布尔数据类型特征的值。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	对于访问模式不是 ITKFEATURE_VAL_ACCESS_MODE_RW 或者 ITKFEATURE_VAL_ACCESS_MODE_RO 的相机特征，无法调用该函数获取特征值。	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevGetBool 获取布尔数据类型特征的值

C++/C	ITKSTATUS ItkDevGetBool (ITKDEVICE hDev, const char* featureName, _Bool* pValue);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	pValue	布尔类型的相机特征的值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说明	获取布尔数据类型特征的值。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	对于访问模式不是 ITKFEATURE_VAL_ACCESS_MODE_RW 或者 ITKFEATURE_VAL_ACCESS_MODE_RO 的相机特征，无法调用该函数获取特征值。	

示例代码

无

ItkFeatureSetBool 设置布尔数据类型特征的值

C++/C	ITKSTATUS ItkFeatureSetBool (ITKFEATURE hFeature, _Bool* value);	
参数说明	phFeature	相机特征句柄
	value	布尔类型的相机特征的值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说明	设置布尔类型特征的值。 设定值的取值只能是 0 或 1。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	对于访问模式不是 ITKFEATURE_VAL_ACCESS_MODE_RW 或者 ITKFEATURE_VAL_ACCESS_MODE_WO 的相机特征，无法调用该函数设置特征值。	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevSetBool 设置布尔数据类型特征的值

C++/C	ITKSTATUS ItkDevSetBool (ITKDEVICE hDev, const char* featureName, _Bool* value);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	value	布尔类型的相机特征的值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说明	设置布尔类型特征的值。 设定值的取值只能是 0 或 1。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	对于访问模式不是 ITKFEATURE_VAL_ACCESS_MODE_RW 或者 ITKFEATURE_VAL_ACCESS_MODE_WO 的相机特征，无法调用该函数设置特征值。	

示例代码

无

ItkFeatureToString 获取字符串类型或枚举类型特征的值

C++/C	ITKSTATUS ItkFeatureToString (ITKFEATURE hFeature, char* pBuff, uint32_t* pBuffLen);	
参数说明	phFeature	相机特征句柄
	pBuff	字符串类型或者枚举类型的相机特征的值
	pBuffLen	作为输入，该参数指明 <i>pBuff</i> 的最大长度；作为输出，该参数指明 <i>pBuff</i> 的有效长度
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说明	获取字符串类型或者枚举类型相机特征的值。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	对于访问模式不是 ITKFEATURE_VAL_ACCESS_MODE_RW 或者 ITKFEATURE_VAL_ACCESS_MODE_RO 的相机特征，无法调用该函数获取特征值。	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevToString 获取字符串类型或枚举类型特征的值

C++/C	ITKSTATUS ItkDevToString (ITKDEVICE hDev, const char* featureName, char* pBuff, uint32_t* pBuffLen);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	pBuff	字符串类型或者枚举类型的相机特征的值
	pBuffLen	作为输入，该参数指明 <i>pBuff</i> 的最大长度；作为输出，该参数指明 <i>pBuff</i> 的有效长度
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说明	获取字符串类型或者枚举类型相机特征的值。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	对于访问模式不是 ITKFEATURE_VAL_ACCESS_MODE_RW 或者	

	ITKFEATURE_VAL_ACCESS_MODE_RO 的相机特征，无法调用该函数获取特征值。
--	--

示例代码

无

ItkFeatureFromString 设置字符串类型或枚举类型特征的值

C++/C	ITKSTATUS ItkFeatureFromString (ITKFEATURE hFeature, const char* value);	
参数说明	phFeature	相机特征句柄
	value	字符串类型或者枚举类型的相机特征的值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说明	设置字符串类型或者枚举类型相机特征的值。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	对于访问模式不是 ITKFEATURE_VAL_ACCESS_MODE_RW 或者 ITKFEATURE_VAL_ACCESS_MODE_WO 的相机特征，无法调用该函数设置特征值。	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevFromString 设置字符串类型或枚举类型特征的值

C++/C	ITKSTATUS ItkDevFromString (ITKDEVICE hDev, const char* featureName, const char* value);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	value	字符串类型或者枚举类型的相机特征的值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说明	设置字符串类型或者枚举类型相机特征的值。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	对于访问模式不是 ITKFEATURE_VAL_ACCESS_MODE_RW 或者 ITKFEATURE_VAL_ACCESS_MODE_WO 的相机特征，无法调用该函数设置	

特征值。

示例代码

无

ItkFeatureGetEnumCount 获取枚举类型特征的枚举数量

C++/C	ITKSTATUS ItkFeatureGetEnumCount (ITKFEATURE hFeature, uint32_t* pCount);	
参数说明	phFeature	相机特征句柄
	pCount	相机特征的枚举项数量
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数	
说明	获取枚举类型相机特征的枚举项数量。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevGetEnumCount 获取枚举类型特征的枚举数量

C++/C	ITKSTATUS ItkDevGetEnumCount (ITKDEVICE hDev, const char* featureName, uint32_t* pCount);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	pCount	相机特征的枚举项数量
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 相机特征不匹配当前输入参数	
说明	获取枚举类型相机特征的枚举项数量。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	

示例代码

无

ItkFeatureGetEnumString 获取枚举类型特征的枚举名

C++/C	ITKSTATUS ItkFeatureGetEnumString (ITKFEATURE hFeature, uint32_t index, char* pBuff, uint32_t* pBuffLen);	
参数说明	phFeature	相机特征句柄
	index	枚举项的索引
	pBuff	枚举名称
	pBuffLen	作为输入，该参数指明 <i>pBuff</i> 的最大有效长度；作为输出，该参数指明 <i>pBuff</i> 的实际有效长度
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_BUFFER_TOO_SMALL : 输入缓冲区太小 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 输入特征不匹配当前执行函数	
说明	获取枚举类型相机特征的枚举名。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevGetEnumString 获取枚举类型特征的枚举名

C++/C	ITKSTATUS ItkDevGetEnumString (ITKDEVICE hDev, const char* featureName, uint32_t index, char* pBuff, uint32_t* pStringSize);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	index	枚举项的索引
	pBuff	枚举名称
	pBuffLen	作为输入，该参数指明 <i>pBuff</i> 的最大有效长度；作为输出，该参数指明 <i>pBuff</i> 的实际有效长度
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_BUFFER_TOO_SMALL : 输入缓冲区太小 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 输入特征不匹配当前执行函数	
说明	获取枚举类型相机特征的枚举名。	

关联文件	IKapC.h
关联库	IKapC.lib
注意事项	无

示例代码

无

ItkFeatureGetEnumIsAvailable 判断枚举名是否存在

C++/C	ITKSTATUS ItkFeatureGetEnumIsAvailable(ITKFEATURE hFeature, const char* enumString, _Bool* bAvailable);	
参数说明	phFeature	相机特征句柄
	enumString	枚举名称
	bAvailable	0：相机不支持该枚举类型名称 1：相机支持该枚举类型名称
返回值	ITKSTATUS_OK ：调用成功 ITKSTATUS_INVALID_ARG ：无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE ：输入特征不匹配当前执行函数	
说明	判断该特征是否支持输入的枚举名称。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevGetEnumIsAvailable 判断枚举名是否存在

C++/C	ITKSTATUS Itk DevGetEnumIsAvailable (ITKDEVICE hDev, const char* featureName, const char* enumString, _Bool* bAvailable);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	enumString	枚举名称
	bAvailable	0：相机不支持该枚举类型名称 1：相机支持该枚举类型名称
返回值	ITKSTATUS_OK ：调用成功 ITKSTATUS_INVALID_ARG ：无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE ：输入特征不匹配当前执行函数	

说 明	判断该特征是否支持输入的枚举名称。
关联文件	IKapC.h
关 联 库	IKapC.lib
注意事项	无

示例代码

无

ItkFeatureExecuteCommand 根据相机特征执行相应命令

C++/C	ITKSTATUS ItkFeatureExecuteCommand(ITKFEATURE hFeature);	
参数说明	phFeature	相机特征句柄
返 回 值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 输入特征不匹配当前执行函数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说 明	根据相机特征执行相应命令。	
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	无	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevExecuteCommand 根据相机特征执行相应命令

C++/C	ITKSTATUS ItkDevExecuteCommand (ITKDEVICE hDev, const char* featureName);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
返 回 值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 输入特征不匹配当前执行函数 ITKSTATUS_NOT_IMPLEMENT : 相机特征读写方法未实现	
说 明	根据相机特征执行相应命令。	
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	无	

示例代码

无

ItkFeatureIsSelector 判断特征是否为选择器

C++/C	ITKSTATUS ItkFeatureIsSelector(ITKFEATURE hFeature, _Bool* bSelector);	
参数说明	hFeature	相机特征句柄
	bSelector	是否为选择器
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 输入特征不匹配当前执行函数	
说明	判断特征是否为选择器。如果相机特征是选择器，在当该特征值改变时应该更新所有被选择特征的值。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevIsSelector 判断特征是否为选择器

C++/C	ITKSTATUS ItkDevIsSelector(ITKDEVICE hDev, const char* featureName, _Bool* bSelector);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	bSelector	是否为选择器
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 输入特征不匹配当前执行函数	
说明	判断特征是否为选择器。如果相机特征是选择器，在当该特征值改变时应该更新所有被选择特征的值。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	

示例代码

无

ItkFeatureGetSelectedFeatureCounts 获取被选择特征的数量

C++/C	ITKSTATUS ItkFeatureGetSelectedFeatureCounts(ITKFEATURE hFeature, uint32_t* pValue);	
参数说明	hFeature	相机特征句柄
	pValue	被选择特征的数量
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 输入特征不匹配当前执行函数	
说明	获取选择器中被选择特征的数量，如果相机特征是选择器，在当该特征值改变时应该更新所有被选择特征的值。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevGetSelectedFeatureCounts 获取被选择特征的数量

C++/C	ITKSTATUS ItkDevGetSelectedFeatureCounts (ITKDEVICE hDev, const char* featureName, uint32_t* pValue);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	pValue	被选择特征的数量
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 输入特征不匹配当前执行函数	
说明	获取选择器中被选择特征的数量，如果相机特征是选择器，在当该特征值改变时应该更新所有被选择特征的值。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	

示例代码

无

ItkFeatureGetSelectedFeatureName 获取被选择特征的名称

C++/C	ITKSTATUS ItkFeatureGetSelectedFeatureName(ITKFEATURE hFeature, uint32_t index, char* pBuf, uint32_t* pBufLen);	
参数说明	hFeature	相机特征句柄
	index	特征索引
	pBuf,	被选择特征的名称
	pBufLen	被选择特征的名称的长度
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 输入特征不匹配当前执行函数	
说明	获取选择器中被选择特征的名称，如果相机特征是选择器，在当该特征值改变时应该更新所有被选择特征的值。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevGetSelectedFeatureName 获取被选择特征的名称

C++/C	ITKSTATUS ItkDevGetSelectedFeatureName (ITKDEVICE hDev, const char* featureName, uint32_t index, char* pBuf, uint32_t* pBufLen);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	index	特征索引
	pBuf,	被选择特征的名称
	pBufLen	被选择特征的名称的长度
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 输入特征不匹配当前执行函数	
说明	获取选择器中被选择特征的名称，如果相机特征是选择器，在当该特征值改变时应该更新所有被选择特征的值。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	

示例代码

无

ItkFeatureGetPollingTime 获取相机特征的轮询周期

C++/C	ITKSTATUS ItkFeatureGetPollingTime (ITKFEATURE hFeature, uint32_t* pPollingTime);	
参数说明	hFeature	相机特征句柄
	pPollingTime	相机特征查询的轮询时间
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 输入特征不匹配当前执行函数	
说明	获取相机特征的轮询周期。对于只读型特征例如“相机温度”，可能需要周期性的更新该特征值来实时监控相机状态， <i>pPollingTime</i> 指明更新周期的最小值。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkDevAllocFeature()	

示例代码

无

ItkDevGetPollingTime 获取相机特征的轮询周期

C++/C	ITKSTATUS ItkDevGetPollingTime (ITKDEVICE hDev, const char* featureName, uint32_t* pPollingTime);	
参数说明	hDev	相机设备句柄
	featureName	相机特征名称
	pPollingTime	相机特征查询的轮询时间
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE : 输入特征不匹配当前执行函数	
说明	获取相机特征的轮询周期。对于只读型特征例如“相机温度”，可能需要周期性的更新该特征值来实时监控相机状态， <i>pPollingTime</i> 指明更新周期的最小值。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	

示例代码

无

ItkStreamAbort 异步结束数据流采集

C++/C	ITKSTATUS ItkStreamAbort (ITKSTREAM hStream);	
参数说明	hStream	数据流句柄
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄	
说明	异步停止数据流采集过程。 当本函数执行完成后，采集过程将会被终止但是无法保证最后一帧图像的完整性。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	

示例代码

无

ItkStreamAddBuffer 向数据流添加缓冲区

C++/C	ITKSTATUS ItkStreamAddBuffer (ITKSTREAM hStream, ITKBUFFER hBuffer);	
参数说明	hStream	数据流句柄
	hBuffer	数据缓冲区
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄	
说明	向数据流缓冲区链表中添加缓冲区。 每个数据采集流都会维护独立的缓冲区链表，采集得到的图像数据将会顺序填入缓冲区中。本函数将会添加一个图像缓冲区到缓冲区链表的尾部。 关于采集过程中图像缓冲区的具体工作流程，参见“ 6.5 缓存管理 ”。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	数据采集流不会多次添加同一个数据缓冲区。	

示例代码

```
ITKSTATUS res = ItkManInitilize();
if (res == ITKSTATUS_OK)
{
    uint32_t count;
    ItkManGetDeviceCount(&count);
}
```

```

if (count > 0)
{
    ITKDEVICE hDev;
    ItkDevOpen(0, ITKDEV_VAL_ACCESS_MODE_CONTROL, &hDev);
    // 获取数据流数量
    uint32_t streamCount;
    ItkDevGetStreamCount(hDev, &streamCount);
    // 创建缓冲区
    ITKBUFFER hBuffer;
    ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hBuffer);
    // 申请数据流资源
    ITKSTREAM hStream;
    ItkDevAllocStream(hDev, 0, hBuffer, &hStream);
    /* 处理数据流 */
    for (int i=0; i < 10; ++i)
    {
        ITKBUFFER hBuf;
        ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hBuf);
        ItkStreamAddBuffer(hStream, hBuf);
    }
    // 释放数据流资源
    ItkDevFreeStream(hStream);
    // 释放缓冲区
    ItkBufferFree(hBuffer);
    ItkDevClose(hDev);
}
ItkManTerminate();
}

```

ItkStreamRemoveBuffer 从数据流移除缓冲区

C++/C	ITKSTATUS ItkStreamRemoveBuffer(ITKSTREAM hStream, ITKBUFFER hBuffer);	
参数说明	hStream	数据流句柄
	hBuffer	数据缓冲区
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄	
说明	从数据流缓冲区链表中删除缓冲区。每个数据采集流都会维护独立的缓冲区链表，采集得到的图像数据将会顺序填入缓冲区中。本函数将会从数据采集流缓冲区链表中删除一个指定的缓冲区。 关于采集过程中图像缓冲区的具体工作流程，参见“ 6.5 缓存管理 ”。	
关联文件	IKapC.h	
关联库	IKapC.lib	

注意事项	为了确保数据采集过程的正常运行，数据采集流的缓冲区链表中至少要一个可用缓冲区。
------	---

示例代码
无

ItkStreamClearBuffer 清空数据流缓冲区

C++/C	ITKSTATUS ItkStreamClearBuffer(ITKSTREAM hStream, ITKBUFFER hBuffer);	
参数说明	hStream	数据流句柄
	hBuffer	数据缓冲区
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄	
说明	清除数据流缓冲区的状态，适用于流自动清空机制禁用的情况。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	为了确保数据采集过程的正常运行，数据采集流的缓冲区链表中至少要一个可用缓冲区。	

示例代码
无

ItkStreamGetPrm 获取数据流参数

C++/C	ITKSTATUS ItkStreamGetPrm(ITKSTREAM hStream, uint32_t prm, void* pValue);	
参数说明	hStream	数据流句柄
	prm	参数 ID
	pValue	参数值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄 ITKSTATUS_INVALID_ARG : 无效的参数	
说明	获取数据采集流的参数。 关于数据采集流的参数列表和详细信息参见“ 8.2 数据流参数 ”。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkStreamSetPrm()	

示例代码

无

ItkStreamSetPrm 设置数据流参数

C++/C	ITKSTATUS ItkStreamSetPrm(ITKSTREAM hStream, uint32_t prm, const void* pValue);	
参数说明	hStream	数据流句柄
	prm	参数 ID
	pValue	参数值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄 ITKSTATUS_INVALID_ARG : 无效的参数	
说明	设置数据采集流的参数。 关于数据采集流的参数列表和详细信息参见“ 8.2 数据流参数 ”。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkStreamGetPrm()	

示例代码

无

ItkStreamStart 开始数据流采集工作

C++/C	ITKSTATUS ItkStreamStart(ITKSTREAM hStream, uint32_t count);	
参数说明	hStream	数据流句柄
	count	ITKSTREAM_CONTINUOUS : 连续采集 1: 单帧采集 2-UINT32_MAX-1: 采集图像序列
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄 ITKSTATUS_INVALID_ARG : 无效的参数 ITKSTATUS_STREAM_IN_PROCESS : 当前存在采集过程 ITKSTATUS_TIMEOUT : 采集超时	
说明	开始数据流采集过程。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	如果数据流正在进行采集，则本函数不会执行并返回 ITKSTATUS_STREAM_IN_PROCESS 。 如果参数 ITKSTREAM_PRM_START_MODE 被设置为 ITKSTREAM_VAL_START_MODE_NON_NON_BLOCK ，则本函数不等待采	

	<p>集过程结束而直接返回。</p> <p>如果参数 ITKSTREAM_PRM_START_MODE 被设置为 ITKSTREAM_VAL_START_MODE_BLOCK，则本函数会等待所有图像数据采集完毕或者采集图像超时才会返回。</p> <p>每个采集数据流维护独立的缓冲区链表，数据采集流将会顺序写入缓冲区链表中。</p> <p>缓冲区链表长度为 4，则采集得到的图像将按照如下顺序排列</p> <p>单次采集(<i>count</i> = 1) : {1→1→1→1→...}</p> <p>序列采集(<i>count</i> = 2) : {1→2→1→2→...}</p> <p>序列采集(<i>count</i> = 6) : {(1→2→3→4→1→2)→(1→2→3→4→1→2)...}</p> <p>连续采集</p> <p>(<i>count</i> = ITKSTREAM_CONTINUOUS): {(1→2→3→4)→(1→2→3→4)...}</p>
相关函数	ItkStreamStop(); ItkStreamAbort()

示例代码

无

ItkStreamStop 同步结束数据流采集工作

C++/C	ITKSTATUS ItkStreamStop(ITKSTREAM hStream);	
参数说明	hStream	数据流句柄
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄	
说明	同步停止数据流采集过程。 当本函数执行后，直到当前正在采集的图像采集完毕才会结束采集过程，因此可以保证最后一帧图像的完整性。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	如果需要立即结束采集过程，调用 ItkStreamAbort() 异步结束采集过程。 本函数将会等待当前帧采集完毕才会返回。	

示例代码

无

ItkStreamWait 等待数据流采集结束

C++/C	ITKSTATUS ItkStreamWait(ITKSTREAM hStream);	
参数说明	hStream	数据流句柄
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_TIMEOUT : 采集超时 ITKSTATUS_INVALID_HANDLE : 无效的句柄	

说 明	当异步采集单帧或者多帧图像数据时，可以调用该函数等待图像采集操作完成。
关联文件	IKapC.h
关 联 库	IKapC.lib
注意事项	本函数对于连续图像采集无效。

示例代码

```
// 异步采集 10 帧图片
res =ItkStreamStart(hStream, 10);
// 等待采集结束
res = ItkStreamWait(hStream);
```

ItkStreamRegisterCallback 注册数据流回调函数

C++/C	ITKSTATUS ItkStreamRegisterCallback(ITKSTREAM hStream, uint32_t eventType, PITKSTREAMCALLBACK callback, void* context);	
参数说明	hStream	数据流句柄
	eventType	注册回调事件的事件类型。当事件触发后回调函数将会被执行。具体信息参见 ITKSTREAM_PRM_SUPPORT_EVENT_TYPE
	callback	用户回调函数，定义如下： void IKAPC_CC (uint32_t eventType, uint32_t eventCount, void* pContext) <i>eventType</i> 指明回调函数的类型 <i>eventCount</i> 指明该类型事件触发的次数，从 1 开始计数 当回调函数被调用时， <i>pContext</i> 将会包含注册回调函数过程中用户输入的上下文指针
	context	用户上下文指针，将会在回调函数调用时传入
返 回 值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄 ITKSTATUS_INVALID_ARG : 无效的输入参数	
说 明	如果用户在回调函数中进行过于费时的操作或者事件触发频率过快，可能会导致触发事件丢失。	
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	为相同事件注册多次回调函数，仅会保证最近一次设置的回调函数会被调用。当为 ITKSTREAM_VAL_EVENT_TYPE_END_OF_LINE 注册回调函数时，必须指明监听的行数 <i>n</i> ，其回调事件类型为 ITKSTREAM_VAL_EVENT_TYPE_END_OF_LINE + n 。该类型仅适用于线扫描相机。	
相关函数	ItkStreamUnregisterCallback()	

示例代码

无

ItkStreamUnregisterCallback 清空数据流回调函数

C++/C	ITKSTATUS ItkStreamUnregisterCallback(ITKSTREAM hStream, uint32_t eventType);	
参数说明	hStream	数据流句柄
	eventType	回调事件类型，事件触发后回调函数将会被执行。具体信息参见 ITKSTREAM_PRM_SUPPORT_EVENT_TYPE
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄 ITKSTATUS_INVALID_ARG : 无效的输入参数	
说明	清除数据流传输过程中的回调函数。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkStreamRegisterCallback()	

示例代码

无

ItkBufferNew 新建缓冲区

C++/C	ITKSTATUS ItkBufferNew(int64_t width, int64_t height, uint32_t format, ITKBUFFER* phBuffer);	
参数说明	width	图像缓冲区的宽度（单位：像素）
	height	图像缓冲区的高度（单位：像素）
	format	图像缓冲区的像素格式，具体信息参见 ITKBUFFER_PRM_FORMAT
	hBuffer	数据缓冲区句柄
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_NO_MEMORY : 无法申请资源 ITKSTATUS_INVALID_ARG : 无效的输入参数	
说明	创建数据缓冲区。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkBufferFree()	

示例代码


```
// 创建宽 2048，高 2048，数据格式为 ITKBUFFER_VAL_FORMAT_MONO8 的图像缓冲区
ITKBUFFER hBuffer;
ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hBuffer);
/* 处理缓冲区 */
ItkBufferFree(hBuffer);
```

ItkBufferFree 销毁缓冲区

C++/C	ITKSTATUS ItkBufferFree (ITKBUFFER hBuffer);	
参数列表	hBuffer	数据缓冲区句柄
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_IVNALID_HANDLE : 无效的句柄	
说明	释放数据缓冲区。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkBufferNew()	

示例代码

参见 “[ItkBufferNew](#)”

ItkBufferGetPrm 获取缓冲区参数

C++/C	ITKSTATUS ItkBufferGetPrm(ITKBUFFER hBuffer, uint32_t prm, void* pValue);	
参数列表	hBuffer	数据缓冲区句柄
	prm	参数索引
	pValue	参数值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_IVNALID_HANDLE : 无效的句柄 ITKSTATUS_ARG_OUT_OF_RANGE : 参数越界	
说明	获取数据缓冲区的参数。 关于数据缓冲区的参数列表和详细信息参见 “ 8.3 缓冲区参数 ”。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkBufferSetPrm()	

示例代码

无

ItkBufferSetPrm 设置缓冲区参数

C++/C	ITKSTATUS ItkBufferSetPrm(ITKBUFFER hBuffer, uint32_t prm, const void* pValue);	
参数列表	hBuffer	数据缓冲区句柄
	prm	参数索引
	pValue	参数值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_IVNALID_HANDLE : 无效的句柄 ITKSTATUS_ARG_OUT_OF_RANGE : 参数越界	
说明	设置数据缓冲区的参数。 关于数据缓冲区的参数列表和详细信息参见“ 8.3 缓冲区参数 ”。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkBufferGetPrm()	

示例代码

无

ItkBufferRead 从缓冲区中读取数据

C++/C	ITKSTATUS ItkBufferRead(ITKBUFFER hBuffer, uint32_t offset, void* data, uint32_t size);	
参数列表	hBuffer	数据缓冲区句柄
	offset	数据缓冲区的偏移量（单位：字节）
	data	用户缓冲区
	size	用户缓冲区的大小（单位：字节）
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_IVNALID_HANDLE : 无效的句柄 ITKSTATUS_ARG_OUT_OF_RANGE : 参数越界	
说明	从数据缓冲区中读取一段连续的内存数据到用户缓冲区中。用户指明数据缓冲区的起始偏移量，并且读取大小为 <i>size</i> 的数据到用户缓冲区中。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	数据缓冲区的偏移量 <i>offset</i> 和用户缓冲区大小 <i>size</i> 的和不能超过数据缓冲区的大	

	小 ITKBUFFER_PRM_SIZE ，即： $offset + size \leq \text{ITKBUFFER_PRM_SIZE}$
相关函数	ItkBufferWrite()

示例代码

```
// 创建宽 2048，高 2048，数据格式为 ITKBUFFER_VAL_FORMAT_MONO8 的图像缓冲区
ITKBUFFER hBuffer;
ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hBuffer);
/* 处理缓冲区 */
int64_t size = 0;
ItkBufferGetPrm(hBuffer, ITKBUFFER_PRM_SIZE, &size);
char* pData = malloc(size);
// 读取数据
ItkBufferRead(hBuffer, 0, pData, size);
// 释放资源
free(pData);
ItkBufferFree(hBuffer);
```

ItkBufferWrite 向缓冲区写入数据

C++/C	ITKSTATUS ItkBufferWrite(ITKBUFFER hBuffer, uint32_t offset, const void* data, uint32_t size);	
参数列表	hBuffer	数据缓冲区句柄
	offset	数据缓冲区的偏移量（单位：字节）
	data	用户缓冲区
	size	用户缓冲区的大小（单位：字节）
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INVALID_HANDLE : 无效的句柄 ITKSTATUS_ARG_OUT_OF_RANGE : 输入参数越界	
说明	向数据缓冲区中写入一段连续的内存数据。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	数据缓冲区的偏移量 <i>offset</i> 和用户缓冲区大小 <i>size</i> 的和不能超过数据缓冲区的大小 ITKBUFFER_PRM_SIZE ，即： $offset + size \leq \text{ITKBUFFER_PRM_SIZE}$	
相关函数	ItkBufferWrite()	

示例代码

无

ItkBufferBayerConvert 将 Bayer 图像转换成彩色图像

C++/C	ITKSTATUS ItkBufferBayerConvert(ITKBUFFER hSrc, ITKBUFFER hDst, uint32_t options);	
参数列表	hSrc	Bayer 编码的源图像，其数据格式必须是下面取值之一： ITKBUFFER_VAL_FORMAT_MONO8 ITKBUFFER_VAL_FORMAT_MONO10 ITKBUFFER_VAL_FORMAT_MONO12 ITKBUFFER_VAL_FORMAT_MONO14 ITKBUFFER_VAL_FORMAT_MONO16
	hDst	转换后的彩色图像，其数据格式必须是下面取值之一： ITKBUFFER_VAL_FORMAT_RGB888 ITKBUFFER_VAL_FORMAT_RGB101010 ITKBUFFER_VAL_FORMAT_RGB121212 ITKBUFFER_VAL_FORMAT_RGB141414 ITKBUFFER_VAL_FORMAT_RGB161616 ITKBUFFER_VAL_FORMAT_RGB888 ITKBUFFER_VAL_FORMAT_BGR101010 ITKBUFFER_VAL_FORMAT_BGR121212 ITKBUFFER_VAL_FORMAT_BGR141414 ITKBUFFER_VAL_FORMAT_BGR161616
	options	Bayer 图像的编码方式，必须是下面取值之一：
		ITKBUFFER_VAL_BAYER_BGGR 
		ITKBUFFER_VAL_BAYER_RGGB 
		ITKBUFFER_VAL_BAYER_GBRG 
		ITKBUFFER_VAL_BAYER_GRBG 
返回值	ITKSTATUS_OK: 调用成功 ITKSTATUS_INVALID_ARG: 无效的输入参数 ITKSTATUS_IVNALID_HANDLE: 无效的句柄	
说明	将 Bayer 格式编码的源图像转换成彩色图像。在 Bayer 格式的图像中，每个像素的灰度值都对应了彩色图像的某个像素通道值，通过最邻近差值方法可以从 Bayer 图像中重构出彩色图像。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	源图像和转换后的图像必须具有相同的宽度、高度和像素深度，否则无法进行 Bayer 转换。	

相关函数 ItkBufferNew()

示例代码

```

ItkBuffer hSrc;
ItkBuffer hDst;
ITKSTATUS res;
res = ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hSrc);
Check(res);
res = ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_RGB888, &hDst);
Check(res);
res = ItkBufferBayerConvert(hSrc, hDst, ITKBUFFER_VAL_BAYER_BGGR);
Check(res);
    
```

ItkBufferClear 清除缓冲区中的数据

C++/C	ITKSTATUS ItkBufferClear(ITKBUFFER hBuffer, const void* pValue, uint32_t size);		
参数列表	hBuffer	数据缓冲区句柄	
	pValue	缓冲区中像素的设定值	
	size	pValue 的长度（单位：字节）	
返 回 值	ITKSTATUS_OK：调用成功 ITKSTATUS_INVALID_ARG：无效的输入参数 ITKSTATUS_IVNALID_HANDLE：无效的句柄		
说 明	将缓冲区中所有像素都设置成同一个值。若设定值的长度与缓冲区中像素的有效长度不等，IKapC 返回输入参数无效。		
关联文件	IKapC.h		
关 联 库	IKapC.lib		
注意事项	缓冲区的数据格式决定了像素占据的内存字节数，其对应信息如下：		
	数据类型	说明	有效字节数
	ITKBUFFER_VAL_FORMAT_MONO8	灰度 8bit 图像数据	1
	ITKBUFFER_VAL_FORMAT_MONO10	灰度 10bit 图像数据	2
	ITKBUFFER_VAL_FORMAT_MONO12	灰度 12bit 图像数据	2
	ITKBUFFER_VAL_FORMAT_MONO14	灰度 14bit 图像数据	2
	ITKBUFFER_VAL_FORMAT_MONO16	灰度 16bit 图像数据	2
	ITKBUFFER_VAL_FORMAT_RGB888	彩色 8bit 图像，内存排列方式为 RGB	3

	ITKBUFFER_VAL_FORMAT_RGB101010	彩色 10bit, 图像内存排列方式为 RGB	6
	ITKBUFFER_VAL_FORMAT_RGB121212	彩色 12bit 图像, 内存排列方式为 RGB	6
	ITKBUFFER_VAL_FORMAT_RGB141414	彩色 14bit 图像, 内存排列方式为 RGB	6
	ITKBUFFER_VAL_FORMAT_RGB161616	彩色 16bit 图像, 内存排列方式为 RGB	6
相关函数	ItkBufferClearBlack()		

示例代码

```
ITKBUFFER hBuffer;
ITKSTATUS res;
res = ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hBuffer);
Check(res);
char data = 0xff;
res = ItkBufferClear(hBuffer, &data, 1);
ItkBufferFree(hBuffer);
```

ItkBufferClearBlack 将缓冲区中所有像素设为黑色

C++/C	ITKSTATUS ItkBufferClearBlack(ITKBUFFER hBuffer);	
参数列表	hBuffer	数据缓冲区句柄
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄	
说明	将缓冲区中所有像素都设置成黑色。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkBufferClear()	

示例代码

```
ITKBUFFER hBuffer;
ITKSTATUS res;
res = ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hBuffer);
Check(res);
res = ItkBufferClearBlack(hBuffer);
```

```
ItkBufferFree(hBuffer);
```

ItkBufferCopy 拷贝源缓冲区中的数据到目的缓冲区中

C++/C	ITKSTATUS ItkBufferCopy(ITKBUFFER hSrc, uint32_t dstOffsetX, uint32_t dstOffsetY, ITKBUFFER hDst);	
参数列表	hSrc	源缓冲区句柄
	dstOffsetX	目的缓冲区的水平位置
	dstOffsetY	目的缓冲区的垂直位置
	hDst	目的缓冲区句柄
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄 ITKSTATUS_ARG_OUT_OF_RANGE : 输入参数越界	
说明	拷贝源缓冲区中的数据到目的缓冲区中，目的缓冲区的起始位置由(<i>dstOffsetX</i> , <i>dstOffsetY</i>)决定。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	若源缓冲区大于目的缓冲区，则只有在目的缓冲区有效范围内的数据才被拷贝。 若源缓冲区为三通道图像数据，目的缓冲区为单通道图像数据，则拷贝第一个通道的数据到目的缓冲区中（如 RGB 图像拷贝 R 通道数值）。 若源缓冲区为单通道图像数据，目的缓冲区为三通道图像数据，则将数据拷贝到第一个通道中。 若源缓冲区图像数据深度大于目的缓冲区，图像的高位信息将会丢失。	
相关函数	ItkBufferCopyRect()	

示例代码

```
ItkBuffer hSrc;
ItkBuffer hDst;
ITKSTATUS res;
res = ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hSrc);
Check(res);
res = ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_RGB888, &hDst);
Check(res);
res = ItkBufferCopy(hSrc, 4, 4, hDst);
Check(res);
```

ItkBufferCopyRect 拷贝源缓冲区矩形区域的数据到目的缓冲区中

C++/C	ITKSTATUS ItkBufferCopyRect(ITKBUFFER hSrc, uint32_t srcOffsetX, uint32_t srcOffsetY, uint32_t width, uint32_t height, ITKBUFFER hDst, uint32_t dstOffsetX, uint32_t dstOffsetY);
-------	---

参数列表	hSrc	源缓冲区句柄
	srcOffsetX	源缓冲区矩形区域的水平起始位置
	srcOffsetY	源缓冲区矩形区域的垂直起始位置
	Width	源缓冲区矩形区域的宽度
	height	源缓冲区矩形区域的高度
	dstOffsetX	目的缓冲区中的水平起始位置
	dstOffsetY	目的缓冲区中的垂直起始位置
	hDst	目的缓冲区句柄
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄 ITKSTATUS_ARG_OUT_OF_RANGE : 输入参数越界	
说明	拷贝源缓冲区矩形区域中的数据到目的缓冲区中	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkBufferCopy()	

示例代码

```

ItkBuffer hSrc;
ItkBuffer hDst;
ITKSTATUS res;
res = ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hSrc);
Check(res);
res = ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_RGB888, &hDst);
Check(res);
res = ItkBufferCopyRect(hSrc, 4, 4, 1024, 1024, hDst, 400, 400);
Check(res);

```

ItkBufferLoad 从文件中载入数据到缓冲区

C++/C	ITKSTATUS ItkBufferLoad(ITKBUFFER hBuffer, const char* filename, uint32_t options);		
参数列表	hBuffer	数据缓冲区句柄	
	filename	文件名	
	options	ITKBUFFER_VAL_BMP	Bmp 数据格式
		ITKBUFFER_VAL_TIFF	Tiff 数据格式
		ITKBUFFER_VAL_RAW	原始图像数据
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INVALID_HANDLE : 无效的句柄		
说明	将指定格式的文件数据载入到缓冲区中。		

关联文件	IKapC.h
关 联 库	IKapC.lib
注意事项	图像缓冲区的数据位数和文件的数据位数必须一致。
相关函数	ItkBufferSave()

示例代码

```
ITKBUFFER hBuffer;
ITKSTATUS res;
char myFilename[MAX_PATH] = { 0 };
res = ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hBuffer);
Check(res);
strcpy(myFilename, "C:\\myFile\\myImg.bmp");
res = ItkBufferSave(hBuffer, myFilename, ITKBUFFER_VAL_BMP);
Check(res);
res = ItkBufferLoad(hBuffer, myFilename, ITKBUFFER_VAL_BMP);
Check(res);
```

ItkBufferSave 将缓冲区中的数据保存到文件

C++/C	ITKSTATUS ItkBufferSave(ITKBUFFER hBuffer, const char* filename, uint32_t options);		
参数列表	hBuffer	数据缓冲区句柄	
	filename	文件名	
	options	ITKBUFFER_VAL_BMP	Bmp 数据格式
		ITKBUFFER_VAL_TIFF	Tiff 数据格式
		ITKBUFFER_VAL_RAW	原始图像数据
返 回 值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_IVNALID_HANDLE : 无效的句柄		
说 明	将缓冲区中的数据保存到指定格式的文件中。		
关联文件	IKapC.h		
关 联 库	IKapC.lib		
注意事项	无		
相关函数	ItkBufferLoad()		

示例代码

参见 “[ItkBufferLoad\(\)](#)”

ItkBufferReadElement 读取缓冲区中的一个像素值

C++/C	ITKSTATUS ItkBufferReadElement(ITKBUFFER hBuffer, uint32_t posX, uint32_t posY, void* element, uint32_t size);	
参数列表	hBuffer	数据缓冲区句柄
	posX	读取像素的水平位置
	posY	读取像素的垂直位置
	element	读取像素的值
	size	读取数据的大小（单位：字节）
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INVALID_HANDLE : 无效的句柄 ITKSTATUS_ARG_OUT_OF_RANGE : 输入参数越界	
说明	读取缓冲区中单个像素。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	原点位置在缓冲区左上角，缓冲区像素大小参见 ITKBUFFER_PRM_FORMAT	
相关函数	ItkBufferWriteElement()	

示例代码

```
ITKBUFFER hBuffer;
ITKSTATUS res;
uint32_t size;
res = ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hBuffer);
Check(res);
char data = 0x00;
res = ItkBufferReadElement(hBuffer, 1, 1, &data, sizeof(char));
Check(res);
res = ItkBufferWriteElement(hBuffer, 2, 2, &data, sizeof(char));
Check(res);
```

ItkBufferReadLine 读取缓冲区中直线上的像素值

C++/C	ITKSTATUS ItkBufferReadLine(ITKBUFFER hBuffer, uint32_t startX, uint32_t startY, uint32_t endX, uint32_t endY, uint32_t* uElements, void* array, uint32_t size);	
参数列表	hBuffer	数据缓冲区句柄
	startX	缓冲区起点的水平位置
	startY	缓冲区起点的垂直位置
	endX	缓冲区终点的水平位置
	endY	缓冲区终点的垂直位置

	array	存放读取像素的数组
	uElements	读取像素的个数
	size	数组的大小（单位：字节，不小于 $uElements * ITKBUFFER_PRM_DATA_BIT/8$ ）
返 回 值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INVALID_HANDLE : 无效的句柄 ITKSTATUS_ARG_OUT_OF_RANGE : 输入参数越界	
说 明	读取缓冲区中一条直线上的像素值。使用 Bresenham 算法确定(<i>startX</i> , <i>startY</i>)和(<i>endX</i> , <i>endY</i>)两点间的理论直线。	
关联文件	IKapC.h	
关 联 库	IKapC.lib	
注意事项	原点位置在缓冲区左上角，缓冲区像素大小参见 ITKBUFFER_PRM_FORMAT	
相关函数	ItkBufferWriteLine()	

示例代码

```

ITKBUFFER hBuffer;
ITKSTATUS res;
uint32_t size, uElements;
res = ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hBuffer);
Check(res);
char* pData = malloc(51);
res = ItkBufferReadLine(hBuffer, 0, 0, 0, 50, &uElements, pData, 60);
Check(res);
res = ItkBufferWriteLine(hBuffer, 0, 0, 0, 50, &uElements, pData, 60);
Check(res);
free(pData);

```

ItkBufferReadRect 读取缓冲区中矩形区域的像素值

C++/C	ITKSTATUS ItkBufferReadRect(ITKBUFFER hBuffer, uint32_t offsetX, uint32_t offsetY, uint32_t width, uint32_t height, void* array, uint32_t size);	
参数列表	hBuffer	数据缓冲区句柄
	offsetX	缓冲区的水平起始位置
	offsetY	缓冲区的垂直起始位置
	width	缓冲区矩形区域的宽度
	height	缓冲区矩形区域的高度
	array	存放读取数据的数组
	size	数组的大小（单位：字节 $width * height * ITKBUFFER_PRM_DATA_BIT / 8$ ）

返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INVALID_HANDLE : 无效的句柄 ITKSTATUS_ARG_OUT_OF_RANGE : 输入参数越界
说明	读取缓冲区矩形区域中的数据。
关联文件	IKapC.h
关联库	IKapC.lib
注意事项	原点位置在缓冲区左上角，缓冲区像素大小参见 ITKBUFFER_PRM_FORMAT
相关函数	ItkBufferWriteRect()

示例代码

```

ITKBUFFER hBuffer;
ITKSTATUS res;
uint32_t dataBit, size;
res = ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hBuffer);
Check(res);
res = ItkBufferGetPrm(hBuffer, ITKBUFFER_PRM_DATA_BIT, &dataBit);
Check(res);
size = 100*100*dataBit/8;
char* pData = malloc(size);
res = ItkBufferReadRect(hBuffer, 0, 0, 100, 100, pData, size);
Check(res);
free(pData);

```

ItkBufferWriteElement 向缓冲区写入单个像素数据

C++/C	ITKSTATUS ItkBufferWriteElement(ITKBUFFER hBuffer, uint32_t posX, uint32_t posY, const void* element, uint32_t size);	
参数列表	hBuffer	数据缓冲区句柄
	posX	写入像素的水平位置
	posY	写入像素的垂直位置
	element	写入像素的值
	size	写入数据的大小（单位：字节）
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INVALID_HANDLE : 无效的句柄 ITKSTATUS_ARG_OUT_OF_RANGE : 输入参数越界	
说明	向缓冲区写入单个像素数据。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	原点位置在缓冲区左上角，缓冲区像素大小参见	

	ITKBUFFER_PRM_FORMAT
相关函数	ItkBufferReadElement()

示例代码

参见 “[ItkBufferReadElement\(\)](#)”

ItkBufferWriteLine 向缓冲区写入一行像素数据

C++/C	ITKSTATUS ItkBufferWriteLine(ITKBUFFER hBuffer, uint32_t startX, uint32_t startY, uint32_t endX, uint32_t endY, uint32_t* uElements, const void* array, uint32_t size);	
参数列表	hBuffer	数据缓冲区句柄
	startX	缓冲区起点的水平位置
	startY	缓冲区起点的垂直位置
	endX	缓冲区终点的水平位置
	endY	缓冲区终点的垂直位置
	array	存放写入像素的数组
	uElements	写入像素的个数
	size	写入像素数组的大小（单位：字节，不小于 <i>uElements</i> * ITKBUFFER_PRM_DATA_BIT / 8）
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INVALID_HANDLE : 无效的句柄 ITKSTATUS_ARG_OUT_OF_RANGE : 输入参数越界	
说明	向缓冲区中写入一条直线的像素值。使用 Bresenham 算法确定 (<i>startX</i> , <i>startY</i>) 和 (<i>endX</i> , <i>endY</i>) 两点间的理论直线。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	原点位置在缓冲区左上角，缓冲区像素大小参见 ITKBUFFER_PRM_FORMAT	
相关函数	ItkBufferReadLine()	

示例代码

参见 “[ItkBufferReadLine\(\)](#)”

ItkBufferWriteRect 向缓冲区的矩形区域写入数据

C++/C	ITKSTATUS ItkBufferWriteRect(ITKBUFFER hBuffer, uint32_t offsetX, uint32_t offsetY, uint32_t width, uint32_t height, const void* array, uint32_t size);	
参数列表	hBuffer	数据缓冲区句柄

	offsetX	缓冲区的水平起始位置
	offsetY	缓冲区的垂直起始位置
	width	缓冲区矩形区域的宽度
	height	缓冲区矩形区域的高度
	array	包含写入像素值的数组
	size	写入的数据的大小（单位：字节 $width * height * \text{ITKBUFFER_PRM_DATA_BIT} / 8$ ）
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INVALID_HANDLE : 无效的句柄 ITKSTATUS_ARG_OUT_OF_RANGE : 输入参数越界	
说明	向缓冲区的矩形区域中写入数据	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	原点位置在缓冲区左上角，缓冲区像素大小参见 ITKBUFFER_PRM_FORMAT	
相关函数	ItkBufferReadRect()	

示例代码

无

ItkBufferDenoise 阈值影响降噪效果

C++/C	ITKSTATUS ItkBufferDenoise(ITKBUFFER hSrc, ITKBUFFER hDst, int threshold);	
参数列表	hSrc	源缓冲区句柄
	hDst	目的缓冲区句柄
	threshold	阈值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INVALID_HANDLE : 无效的句柄	
说明	阈值影响降噪效果。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	

示例代码

无

ItkViewNew 新建图像显示窗口

C++/C	ITKSTATUS ItkViewNew(ITKBUFFER hBuffer, const void* hParentWnd, ITKVIEW* hView);	
参数说明	hBuffer	数据缓冲区句柄
	hParentWnd	父窗口句柄
	hView	显示图像窗口句柄
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_NO_MEMORY : 无法申请资源 ITKSTATUS_INVALID_ARG : 无效的输入参数 ITKSTATUS_INVALID_HANDLE : 无效的数据缓冲区句柄	
说明	创建显示图像窗口。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	<i>hParentWnd</i> 指向父窗口句柄，当父窗口句柄不为空时，新建窗口共享父窗口消息循环队列；当父窗口句柄为空时，新建窗口会新建消息循环队列。	
相关函数	ItkViewFree()	

示例代码

```
// 创建宽 2048，高 2048，数据格式为 ITKBUFFER_VAL_FORMAT_MONO8 的图像缓冲区
ITKBUFFER hBuffer;
ITKVIEW hView;
ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hBuffer);
// 创建图像显示窗口
ItkViewNew(hBuffer, NULL, &hView);
/* 显示图像 */
// 销毁图像窗口
ItkViewFree(hView);
```

ItkViewFree 释放图像显示窗口

C++/C	ITKSTATUS ItkViewFree(ITKVIEW hView);	
参数说明	hView	显示图像窗口句柄
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的数据缓冲区句柄	
说明	销毁显示图像窗口。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	当用户点击“关闭”按钮关闭显示窗口后，仍然需要显示的调用本函数释放在建立图像窗口中申请的资源。	
相关函数	ItkViewNew()	

示例代码

无

ItkViewAddBuffer 图像显示窗口添加缓冲区

C++/C	ITKSTATUS ItkViewAddBuffer(ITKVIEW hView, ITKBUFFER hBuffer);	
参数说明	hView	图像显示窗口句柄
	hBuffer	数据缓冲区
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄	
说明	向图像显示窗口中添加缓冲区。 每个图像显示窗口中都会维护独立的缓冲区链表，每次调用 ItkViewShow() 或者 ItkViewShowNext() 时，图像显示窗口都会从缓冲区链表取出一个或者多个图像缓冲区进行显示。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	图像显示窗口不会多次添加同一个数据缓冲区。	

示例代码

```
ITKSTATUS res = ItkManInitilize();
if (res == ITKSTATUS_OK)
{
    // 创建缓冲区
    ITKVIEW hView;
    ITKBUFFER hBuffer;
    ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hBuffer);
    // 创建窗口
    ItkViewNew(hBuffer, NULL, &hView);
    for (int i=0; i < 10; ++i)
    {
        ITKBUFFER hBuf;
        ItkBufferNew(2048, 2048, ITKBUFFER_VAL_FORMAT_MONO8, &hBuf);
        ItkViewAddBuffer(hView, hBuf);
    }
    // 释放窗口
    ItkViewFree(hView);
    // 释放缓冲区
    ItkBufferFree(hBuffer);
    ItkDevClose(hDev);
}
ItkManTerminate();
```



```
}

```

ItkViewRemoveBuffer 图像显示窗口移除缓冲区

C++/C	ITKSTATUS ItkViewRemoveBuffer(ITKVIEW hView, ITKBUFFER hBuffer);	
参数说明	hView	图像显示窗口句柄
	hBuffer	数据缓冲区
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄	
说明	从图像显示窗口中移除缓冲区。 每个图像显示窗口中都会维护独立的缓冲区链表，每次调用 ItkViewShow() 或者 ItkViewShowNext() 时，图像显示窗口都会从缓冲区链表取出一个或者多个图像缓冲区进行显示。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	图像显示窗口应该至少保留一个图像缓冲区，否则无法显示任何图像数据。	

示例代码

无

ItkViewRemoveAllBuffer 图像显示窗口移除所有缓冲区

C++/C	ITKSTATUS ItkViewRemoveAllBuffer(ITKVIEW hView);	
参数说明	hView	图像显示窗口句柄
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄	
说明	从图像显示窗口中移除所有图像缓冲区。 每个图像显示窗口中都会维护独立的缓冲区链表，每次调用 ItkViewShow() 或者 ItkViewShowNext() 时，图像显示窗口都会从缓冲区链表取出一个或者多个图像缓冲区进行显示。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	图像显示窗口应该至少保留一个图像缓冲区，否则无法显示任何图像数据。	

示例代码

无

ItkViewGetPrm 获取图像显示窗口的参数

C++/C	ITKSTATUS ItkViewGetPrm(ITKVIEW hView, uint32_t prm, void* pValue);
-------	---

参数说明	hView	图像显示窗口句柄
	prm	参数 ID
	pValue	参数值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄 ITKSTATUS_INVALID_ARG : 无效的参数	
说明	获取图像显示窗口的参数。 关于图像显示窗口的参数列表和详细信息参见“ 8.4 图像显示窗口参数 ”。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkViewSetPrm()	

示例代码

无

ItkViewSetPrm 设置图像显示窗口的参数

C++/C	ITKSTATUS ItkViewSetPrm(ITKVIEW hView, uint32_t prm, const void* pValue);	
参数说明	hView	图像显示窗口句柄
	prm	参数 ID
	pValue	参数值
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄 ITKSTATUS_INVALID_ARG : 无效的参数	
说明	设置图像显示窗口的参数。 关于图像显示窗口的参数列表和详细信息参见“ 8.4 图像显示窗口参数 ”。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	
相关函数	ItkViewGetPrm()	

示例代码

无

ItkViewHide 隐藏图像显示窗口

C++/C	ITKSTATUS ItkViewHide(ITKVIEW hView);	
参数说明	hView	图像显示窗口句柄

返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄
说明	隐藏图像显示窗口的参数。 可以调用 ItkViewShow() 或者 ItkViewShowNext() 显示窗口。
关联文件	IKapC.h
关联库	IKapC.lib
注意事项	无
相关函数	ItkViewShow() ; ItkViewShowNext()

示例代码

无

ItkViewShow 显示所有采集完毕的图像缓冲区数据

C++/C	ITKSTATUS ItkViewShow(ITKVIEW hView);	
参数说明	hView	图像显示窗口句柄
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄	
说明	每个图像显示窗口都会维护单独的缓冲区链表，当调用该函数时，显示窗口会显示所有已经采集完毕的图像缓冲区数据。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	本函数在不影响图像数据传输的情况下显示尽可能多的图像数据，这意味着如果显示图像的操作过于耗时，可能会跳过部分数据图像的显示过程。 本函数仅当图像缓冲区为满时才会显示。	
相关函数	ItkViewShowNext()	

示例代码

无

ItkViewShowNext 显示下一帧图像缓冲区数据

C++/C	ITKSTATUS ItkViewShow(ITKVIEW hView);	
参数说明	hView	图像显示窗口句柄
返回值	ITKSTATUS_OK : 调用成功 ITKSTATUS_INVALID_HANDLE : 无效的句柄	
说明	每个图像显示窗口都会维护单独的缓冲区链表，当调用该函数时，显示窗口将会显示下一帧图像缓冲区数据。	
关联文件	IKapC.h	
关联库	IKapC.lib	
注意事项	无	

相关函数

ItkViewShowNext()

示例代码

无

8 参数列表

8.1 全局参数

ITKMANAGER_COAXPRESS_12_SUPPORT 是否使用 CXP12 协议

说明	是否使用 CXP12 协议	
数据类型	uint32_t	
是否可读	是	
是否可写	是	
值	0	不使用 CXP12 协议
	1	使用 CXP12 协议
注意事项	该参数默认支持 CXP12 协议。对于不支持 CXP12 协议的相机，可以将参数设置为 0。	

8.2 采集卡参数

ITKBOARD_PRM_WIDTH 采集卡图像缓冲区的宽度

说明	采集卡图像缓冲区的宽度	
数据类型	int64_t	
是否可读	是	
是否可写	是	
值	无	
注意事项	无	

ITKBOARD_PRM_HEIGHT 采集卡图像缓冲区的高度

说明	采集卡图像缓冲区的高度	
数据类型	int64_t	
是否可读	是	
是否可写	是	
值	无	
注意事项	无	

ITKBOARD_PRM_OFFSET_X 水平方向偏移量

说明	水平方向偏移量
数据类型	int64_t
是否可读	是
是否可写	是
值	无
注意事项	无

ITKBOARD_PRM_OFFSET_Y 垂直方向偏移量

说明	垂直方向偏移量
数据类型	int64_t
是否可读	是
是否可写	是
值	无
注意事项	无

8.3 设备参数

ITKDEV_PRM_HEARTBEAT_TIMEOUT 设备心跳包超时时间

说明	设备心跳包超时时间
数据类型	uint32_t
是否可读	是
是否可写	是
值	100 ~ 0xFFFFFFFF-1
注意事项	读写设备心跳包超时时间，单位 ms，当用户调试千兆网相机设备时可以增大该参数保证相机不会因为未及时接受心跳包而停止工作。

ITKDEV_PRM_INTERFACE_TYPE 设备接口类型

说明	设备接口类型
数据类型	uint32_t
是否可读	是
是否可写	是

值	无
注意事项	无

8.4 数据流参数

ITKSTREAM_PRM_STATUS 数据流传输状态

说明	数据流的传输状态	
数据类型	uint32_t	
是否可读	是	
是否可写	否	
值	ITKSTREAM_VAL_STATUS_STOPPED	数据流停止采集状态
	ITKSTREAM_VAL_STATUS_ACTIVE	数据流采集状态
	ITKSTREAM_VAL_STATUS_PENDING	数据流等待状态
	ITKSTREAM_VAL_STATUS_ABORTED	数据流异步停止采集状态
	ITKSTREAM_VAL_STATUS_TIMEOUT	数据流超时状态
注意事项	无	

ITKSTREAM_PRM_SUPPORT_EVENT_TYPE 数据流支持回调函数种类

说明	数据流支持回调函数种类	
数据类型	uint32_t	
是否可读	是	
是否可写	否	
值	ITKSTREAM_VAL_EVENT_TYPE_START_OF_STREAM	回调函数在数据流采集开始时被调用
	ITKSTREAM_VAL_EVENT_TYPE_END_OF_STREAM	回调函数在数据流采集停止时被调用
	ITKSTREAM_VAL_EVENT_TYPE_START_OF_FRAME	回调函数在一帧图像开始采集时候被调用
	ITKSTREAM_VAL_EVENT_TYPE_END_OF_FRAME	回调函数在一帧图像采集完成时候被调用
	ITKSTREAM_VAL_EVENT_TYPE_END_OF_LINE + n	回调函数在第n行图像被采集完成时调用
	ITKSTREAM_VAL_EVENT_TYPE_TIMEOUT	回调函数在采集超时被调用
	ITKSTREAM_VAL_EVENT_TYPE_FRAME_LOST	采集丢帧时调用

注意事项	ItkStreamGetPrm() 获取的值可能是上述值的异或。
------	---

ITKSTREAM_PRM_START_MODE 数据流采集启动模式

说明	数据流采集启动模式	
数据类型	uint32_t	
是否可读	是	
是否可写	是	
值	ITKSTREAM_VAL_START_MODE_NON_BLOCK	非阻塞式启动图像采集， ItkStreamStart() 将会立刻返回。
	ITKSTREAM_VAL_START_MODE_BLOCK	阻塞式启动图像采集， ItkStreamStart() 将会等待所有图像传输完毕才返回。
注意事项	该参数不会影响连续采集的执行，对于连续采集过程始终采用非阻塞式启动方式。	

ITKSTREAM_PRM_TRANSFER_MODE 数据流数据传输模式

说明	数据流的传输模式，决定了数据传输过程中下一块传输缓冲区的索引	
数据类型	uint32_t	
是否可读	是	
是否可写	是	
值	ITKSTREAM_VAL_TRANSFER_MODE_SYNCHRONOUS_WITH_PROTECT	同步保护模式，如果下一块缓冲区为空则传输到下一块缓冲区中，否则丢弃图像数据
注意事项	关于数据传输模式的具体使用方法参见“ 6.5 缓存管理 ”。	

ITKSTREAM_PRM_AUTO_CLEAR 数据流自动清空标志

说明	数据流自动清空标志	
数据类型	uint32_t	
是否可读	是	
是否可写	是	
值	ITKSTREAM_VAL_AUTO_CLEAR_DISABLE	使能数据流自动清空机制。
	ITKSTREAM_VAL_AUTO_CLEAR_ENABLE	禁用数据流自动清空机制。
注意事项	关于数据流自动清空标志的使用方法参见“ 6.5 缓存管理 ”。	

ITKSTREAM_PRM_TIME_OUT 数据流采集超时时间

说明	数据流采集超时时间
数据类型	uint32_t
是否可读	是
是否可写	是
值	0-INFINITE
注意事项	数据采集过程中，当连续两帧的时间间隔超过设定值，则会终止数据采集过冲。对于较大图像，可能需要较长时间完成采集，通过增加超时时间确保图像可以采集得到。

ITKSTREAM_PRM_FRAME_RATE 数据流实时采集帧率

说明	数据流实时采集帧率
数据类型	uint32_t
是否可读	是
是否可写	否
值	0-INFINITE
注意事项	本参数指示相机图像采集过程中的实时帧率。

ITKSTREAM_PRN_GV_PACKET_MAX_RESEND_COUNT 千兆网单个数据包最大重发次数

说明	千兆网单个数据包最大重发次数
数据类型	uint32_t
是否可读	是
是否可写	是
值	0-INFINITE
注意事项	本参数仅对千兆网相机有效，当 PC 端丢失千兆网数据包后会重新请求丢失的数据包，单个数据包的最大请求次数不超过本参数。

ITKSTREAM_PRM_GV_PACKET_RESEND_TIMEOUT 千兆网单个数据包超时时间

说明	千兆网单个数据包超时时间
----	--------------

数据类型	uint32_t
是否可读	是
是否可写	是
值	0-INFINITE
注意事项	本参数仅对千兆网相机有效，当 PC 端丢失千兆网数据包后会重新请求丢失的数据包，单个数据包请求的超时时间不超过本参数。

ITKSTREAM_PRN_GV_BLOCK_MAX_RESEND_PACKET_COUNT 千兆网数据块最大重传包数

说明	千兆网数据块最大重传包数
数据类型	uint32_t
是否可读	是
是否可写	是
值	0-INFINITE
注意事项	数据块最大重传包数，一帧图像允许的重传包数最大值，如果某帧图像统计发送重传包太多而超过此值，则不会再发送重传命令，而是直接判定残帧。此值为 0 表示关闭此功能。

ITKSTREAM_PRM_GV_BLOCK_MAX_WAIT_PACKET_COUNT 千兆网数据块最大等待包数

说明	千兆网数据块最大等待包数
数据类型	uint32_t
是否可读	是
是否可写	是
值	0-INFINITE
注意事项	最大等待包数，如果某帧图像还没接收完，但是下一帧图像已经到来，如果下一帧图像接收了 N 个包，上一帧还没接收完，则强制结束上一帧，将其判断为残帧。此值为 0 表示关闭此功能。

ITKSTREAM_PRM_GV_BLOCK_RESEND_WINODW_SIZE 千兆网数据块重传窗口大小

说明	千兆网数据块重传窗口大小
数据类型	uint32_t

是否可读	是
是否可写	是
值	0-INFINITE
注意事项	无

ITKSTREAM_PRM_GV_PACKET_INTER_TIMEOUT 千兆网单个数据包超时时间

说明	千兆网单个数据包超时时间
数据类型	uint32_t
是否可读	是
是否可写	是
值	0-INFINITE
注意事项	无

ITKSTREAM_PRM_GV_PACKET_POLLING_TIME 千兆网单个数据包轮询时间

说明	千兆网单个数据包轮询时间
数据类型	uint32_t
是否可读	是
是否可写	是
值	0-INFINITE
注意事项	无

ITKSTREAM_PRM_GV_KERNEL_BUFFER_COUNT 千兆网内核缓冲区个数

说明	千兆网内核缓冲区个数
数据类型	uint32_t
是否可读	是
是否可写	否
值	0-INFINITE
注意事项	无

8.5 缓冲区参数

ITKBUFFER_PRM_FORMAT 缓冲区数据格式

说明	缓冲区数据格式	
数据类型	uint32_t	
是否可读	是	
是否可写	是	
值	ITKBUFFER_VAL_FORMAT_MONO8	灰度 8bit 图像数据，每个像素占据 1 个字节
	ITKBUFFER_VAL_FORMAT_MONO10	灰度 10bit 图像数据，每个像素占据 2 个字节
	ITKBUFFER_VAL_FORMAT_MONO10PACKED	灰度 10bit 压缩图像数据
	ITKBUFFER_VAL_FORMAT_MONO12	灰度 12bit 图像数据，每个像素占据 2 个字节
	ITKBUFFER_VAL_FORMAT_MONO12PACKED	灰度 12bit 压缩图像数据
	ITKBUFFER_VAL_FORMAT_MONO14	灰度 14bit 图像数据，每个像素占据 2 个字节
	ITKBUFFER_VAL_FORMAT_MONO16	灰度 16bit 图像数据，每个像素占据 2 个字节
	ITKBUFFER_VAL_FORMAT_RGB888	彩色 8bit 图像，像素值按照 RGB 顺序在内存中排列，每个像素占据 3 个字节
	ITKBUFFER_VAL_FORMAT_RGB101010	彩色 10bit 图像，像素值按照 RGB 顺序在内存中排列，每个像素占据 6 个字节
	ITKBUFFER_VAL_FORMAT_RGB121212	彩色 12bit 图像，像素值按照 RGB 顺序在内存中排列，每个像素占据 6 个字节
	ITKBUFFER_VAL_FORMAT_RGB141414	彩色 14bit 图像，像素值按照 RGB 顺序在内存中排列，每个像素占据 6 个字节
	ITKBUFFER_VAL_FORMAT_RGB161616	彩色 16bit 图像，像素

	值按照 RGB 顺序在内存中排列，每个像素占据 6 个字节
ITKBUFFER_VAL_FORMAT_BGR888	彩色 8bit 图像，像素值按照 BGR 顺序在内存中排列，每个像素占据 3 个字节
ITKBUFFER_VAL_FORMAT_BGR101010	彩色 10bit 图像，像素值按照 BGR 顺序在内存中排列，每个像素占据 6 个字节
ITKBUFFER_VAL_FORMAT_BGR121212	彩色 12bit 图像，像素值按照 BGR 顺序在内存中排列，每个像素占据 6 个字节
ITKBUFFER_VAL_FORMAT_BGR141414	彩色 14bit 图像，像素值按照 BGR 顺序在内存中排列，每个像素占据 6 个字节
ITKBUFFER_VAL_FORMAT_BGR161616	彩色 16bit 图像，像素值按照 BGR 顺序在内存中排列，每个像素占据 6 个字节
ITKBUFFER_VAL_FORMAT_BAYER_GR8	Bayer 格式 8bit 图像，像素值按照 GR 顺序在内存中排列
ITKBUFFER_VAL_FORMAT_BAYER_RG8	Bayer 格式 8bit 图像，像素值按照 RG 顺序在内存中排列
ITKBUFFER_VAL_FORMAT_BAYER_GB8	Bayer 格式 8bit 图像，像素值按照 GB 顺序在内存中排列
ITKBUFFER_VAL_FORMAT_BAYER_BG8	Bayer 格式 8bit 图像，像素值按照 BG 顺序在内存中排列
ITKBUFFER_VAL_FORMAT_BAYER_GR10	Bayer 格式 10bit 图像，像素值按照 GR 顺序在内存中排列
ITKBUFFER_VAL_FORMAT_BAYER_RG10	Bayer 格式 10bit 图像，像素值按照 RG 顺序在内存中排列
ITKBUFFER_VAL_FORMAT_BAYER_GB10	Bayer 格式 10bit 图像，像素值按照 GB 顺序在内存中排列

		顺序在内存中排列
	ITKBUFFER_VAL_FORMAT_BAYER_BG10	Bayer 格式 10bit 图像, 像素值按照 BG 顺序在内存中排列
	ITKBUFFER_VAL_FORMAT_BAYER_GR10PACKED	Bayer 格式 10bit 压缩图像, 像素值按照 GR 顺序在内存中排列
	ITKBUFFER_VAL_FORMAT_BAYER_RG10PACKED	Bayer 格式 10bit 压缩图像, 像素值按照 RG 顺序在内存中排列
	ITKBUFFER_VAL_FORMAT_BAYER_GB10PACKED	Bayer 格式 10bit 压缩图像, 像素值按照 GB 顺序在内存中排列
	ITKBUFFER_VAL_FORMAT_BAYER_BG10PACKED	Bayer 格式 10bit 压缩图像, 像素值按照 BG 顺序在内存中排列
	ITKBUFFER_VAL_FORMAT_BAYER_GR12	Bayer 格式 12bit 图像, 像素值按照 GR 顺序在内存中排列
	ITKBUFFER_VAL_FORMAT_BAYER_RG12	Bayer 格式 12bit 图像, 像素值按照 RG 顺序在内存中排列
	ITKBUFFER_VAL_FORMAT_BAYER_GB12	Bayer 格式 12bit 图像, 像素值按照 GB 顺序在内存中排列
	ITKBUFFER_VAL_FORMAT_BAYER_BG12	Bayer 格式 12bit 图像, 像素值按照 BG 顺序在内存中排列
	ITKBUFFER_VAL_FORMAT_YUV422_8_YUYV	YUV 4:2:2 格式图像 (YUYV)
	ITKBUFFER_VAL_FORMAT_YUV422_8_UYUV	YUV 4:2:2 格式图像 (UYUV)
注意事项	无	

ITKBUFFER_PRM_DATA_BIT 数据缓冲区单个像素大小

说明	数据缓冲区单个像素的大小, 单位 bit
数据类型	uint32_t
是否可读	是
是否可写	否
值	该值由缓冲区的像素格式 ITKBUFFER_PRM_FORMAT 决定。

注意事项	无
------	---

ITKBUFFER_PRM_PIXEL_DEPTH 数据缓冲区像素深度

说明	数据缓冲区像素深度
数据类型	uint32_t
是否可读	是
是否可写	否
值	该值由缓冲区的像素格式 ITKBUFFER_PRM_FORMAT 决定。
注意事项	无

ITKBUFFER_PRM_WIDTH 数据缓冲区的宽度

说明	数据缓冲区的宽度
数据类型	int64_t
是否可读	是
是否可写	是
值	无
注意事项	无

ITKBUFFER_PRM_HEIGHT 数据缓冲区的高度

说明	数据缓冲区的高度
数据类型	int64_t
是否可读	是
是否可写	是
值	无
注意事项	无

ITKBUFFER_PRM_ADDRESS 数据缓冲区的地址

说明	数据缓冲区的地址
数据类型	32 位系统: int32_t 64 位系统: int64_t
是否可读	是

是否可写	是
值	无
注意事项	该值是数据缓冲区的首地址，用户可以通过该地址直接访问缓冲区的数据。

ITKBUFFER_PRM_HOST_COUNTER_STAMP 数据缓冲区填满时候的时间戳

说明	数据缓冲区填满时候的系统时间戳，该值受系统 CPU 的时钟频率影响
数据类型	uint64_t
是否可读	是
是否可写	否
值	无
注意事项	无

ITKBUFFER_PRM_STATE 数据缓冲区的状态

说明	数据缓冲区状态定义了缓冲区为空或者满 当缓冲区为空时，数据流采集得到的图像数据可以存入该缓冲中 当缓冲区为满时，如果数据流继续采集数据并放入缓冲区中，会导致缓冲区被覆盖	
数据类型	uint32_t	
是否可读	是	
是否可写	是	
值	ITKBUFFER_VAL_STATE_EMPTY	缓冲区为空
	ITKBUFFER_VAL_STATE_FULL	缓冲区为满
	ITKBUFFER_VAL_STATE_OVERFLOW	缓冲区被覆盖
	ITKBUFFER_VAL_STATE_UNCOMPLETED	缓冲区非满
注意事项	无	

ITKBUFFER_PRM_SIGNED 数据缓冲区像素值有无符号

说明	数据缓冲区像素值有无符号
数据类型	uint32_t
是否可读	是
是否可写	否
值	该值由缓冲区的像素格式 ITKBUFFER_PRM_FORMAT 决定。
注意事项	无

ITKBUFFER_PRM_SIZE 数据缓冲区的大小

说明	数据缓冲区的大小
数据类型	int64_t
是否可读	是
是否可写	否
值	该值由如下公式获取得到。 $\text{ITKBUFFER_PRM_SIZE} = \text{ITKBUFFER_PRM_WIDTH} * \text{ITKBUFFER_PRM_HEIGHT} * \text{ITKBUFFER_PRM_DATA_BIT} / 8$
注意事项	无

ITKBUFFER_PRM_BLOCK_ID 数据缓冲区索引

说明	数据缓冲区索引
数据类型	uint64_t
是否可读	是
是否可写	否
值	数据缓冲区索引，对于千兆网相机，该索引的有效范围是[1, 65535]，并且随着采集帧数的增加而顺序递增。
注意事项	无

ITKBUFFER_PRM_READY_LINES 数据缓冲区当前采集完成的行数

说明	数据缓冲区当前采集完成的行数
数据类型	int64_t
是否可读	是
是否可写	否
值	无
注意事项	无

ITKBUFFER_PRM_ERROR_REASON 数据缓冲区错误原因

说明	数据缓冲区错误原因
数据类型	int64_t

是否可读	是
是否可写	否
值	无
注意事项	无

8.6 图像显示窗口参数

ITKVIEW_PRM_FLIP_X 显示图像水平方向翻转

说明	显示图像水平方向翻转	
数据类型	int32_t	
是否可读	是	
是否可写	是	
值	ITKVIEW_VAL_FLIP_X_DISABLE	显示图像水平方向翻转不使能
	ITKVIEW_VAL_FLIP_X_ENABLE	显示图像水平方向翻转使能
注意事项	无	

ITKVIEW_PRM_FLIP_Y 显示图像垂直方向翻转

说明	显示图像水平方向翻转	
数据类型	int32_t	
是否可读	是	
是否可写	是	
值	ITKVIEW_VAL_FLIP_Y_DISABLE	显示图像垂直方向翻转不使能
	ITKVIEW_VAL_FLIP_Y_ENABLE	显示图像垂直方向翻转使能
注意事项	无	

ITKVIEW_PRM_LSB 原图像最低 bit 位

说明	原图像最低 bit 位
数据类型	int32_t
是否可读	是
是否可写	否
值	无
注意事项	无

ITKVIEW_PRM_MSB 原图像最高 bit 位

说明	原图像最高 bit 位
数据类型	int32_t
是否可读	是
是否可写	否
值	无
注意事项	无

ITKVIEW_PRM_BUFFER_ROI_HEIGHT 显示图像感兴趣区域的高度

说明	显示图像感兴趣区域的高度
数据类型	int32_t
是否可读	是
是否可写	是
值	1 ~ ITKBUFFER_PRM_HEIGHT ITKBUFFER_PRM_HEIGHT 是用户输入图像缓冲区的高度
注意事项	用户在创建显示窗口的 ROI 区域时候，应该先设置 ROI 区域的高度和宽度，再设置 ROI 区域的上边缘坐标和左边缘坐标。

ITKVIEW_PRM_BUFFER_ROI_WIDTH 显示图像感兴趣区域的宽度

说明	显示图像感兴趣区域的宽度
数据类型	int32_t
是否可读	是
是否可写	是
值	1 ~ ITKBUFFER_PRM_WIDTH ITKBUFFER_PRM_WIDTH 是用户输入图像缓冲区的宽度
注意事项	用户在创建显示窗口的 ROI 区域时候，应该先设置 ROI 区域的高度和宽度，再设置 ROI 区域的上边缘坐标和左边缘坐标。

ITKVIEW_PRM_BUFFER_ROI_LEFT 显示图像感兴趣区域的左边缘坐标

说明	显示图像感兴趣区域的左边缘坐标
数据类型	int32_t
是否可读	是

是否可写	是
值	0 ~ ITKBUFFER_PRM_WIDTH -1 ITKBUFFER_PRM_WIDTH 是用户输入图像缓冲区的宽度
注意事项	用户在创建显示窗口的 ROI 区域时候, 应该先设置 ROI 区域的高度和宽度, 再设置 ROI 区域的上边缘坐标和左边缘坐标。

ITKVIEW_PRM_BUFFER_ROI_TOP 显示图像感兴趣区域的上边缘坐标

说明	显示图像感兴趣区域的上边缘坐标
数据类型	int32_t
是否可读	是
是否可写	是
值	0 ~ ITKBUFFER_PRM_HEIGHT - 1 ITKBUFFER_PRM_HEIGHT 是用户输入图像缓冲区的高度
注意事项	用户在创建显示窗口的 ROI 区域时候, 应该先设置 ROI 区域的高度和宽度, 再设置 ROI 区域的上边缘坐标和左边缘坐标。

ITKVIEW_PRM_ZOOM_METHOD 图像显示的缩放方法

说明	图像显示的缩放方法	
数据类型	int32_t	
是否可读	是	
是否可写	是	
值	ITKVIEW_VAL_ZOOM_NN	最邻近插值算法
	ITKVIEW_VAL_ZOOM_LINEAR	线性插值算法
	ITKVIEW_VAL_ZOOM_CUBIC	立方插值算法
	ITKVIEW_VAL_ZOOM_AREA	像素关系重采样算法
注意事项	IKapC 在显示图像时默认采用 ITKVIEW_VAL_ZOOM_AREA 算法。	

ITKVIEW_PRM_ZOOM_MAX_RATIO 图像最大放大倍数

说明	图像最大放大倍数
数据类型	double
是否可读	是
是否可写	否
值	128.0

注意事项	无
------	---

ITKVIEW_PRM_ZOOM_MIN_RATIO 图像最小缩小倍数

说明	图像最小缩小倍数
数据类型	double
是否可读	是
是否可写	否
值	1/128.0
注意事项	无

ITKVIEW_PRM_HWND 显示图像窗口句柄

说明	显示图像窗口句柄
数据类型	32 位系统: int32_t 64 位系统: int64_t
是否可读	是
是否可写	否
值	无
注意事项	无

ITKVIEW_PRM_HWND_TITLE 显示图像窗口标题

说明	显示图像窗口标题
数据类型	char[128]
是否可读	是
是否可写	是
值	无
注意事项	以空字符(NULL)结尾的字符串，字符串最大长度是 128。

ITKVIEW_PRM_BAYER_MODE 在窗口中显示图像 Bayer 格式

说明	在窗口中显示图像 Bayer 格式
数据类型	int32_t
是否可读	是

是否可写	是	
值	ITKVIEW_VAL_BAYER_MODE_NIL	Bayer 格式 NIL 模式
	ITKVIEW_VAL_BAYER_MODE_BGGR	Bayer 格式 BGGR 模式
	ITKVIEW_VAL_BAYER_MODE_RGGB	Bayer 格式 RGGB 模式
	ITKVIEW_VAL_BAYER_MODE_GBRG	Bayer 格式 GBRG 模式
	ITKVIEW_VAL_BAYER_MODE_GRBG	Bayer 格式 GRBG 模式
注意事项	无	

ITKVIEW_PRM_BUFFER_CURRENT_INDEX 当前缓冲区索引

说明	当前缓冲区索引
数据类型	uint32_t
是否可读	是
是否可写	否
值	无
注意事项	无

8.7 事件信息参数

ITKEVENTINFO_PRM_TYPE 事件类型

说明	事件类型
数据类型	uint32_t
是否可读	是
是否可写	否
值	0x00010000：设备移除回调事件 0x00020000：特征值改变回调事件
注意事项	设备回调事件触发类型。

ITKEVENTINFO_PRM_FEATURE_NAME 特征名称

说明	特征名称
数据类型	char[128]
是否可读	是
是否可写	否
值	无

注意事项	设备特征值改变时触发相应回调函数，该参数指明改变特征值的名称，最大不超过 128 字节。
------	--

ITKEVENTINFO_PRM_HOST_TIME_STAMP 事件发生时间戳

说明	事件发生时间戳
数据类型	uint64_t
是否可读	是
是否可写	否
值	无
注意事项	设备回调函数触发时间戳。

9 错误处理

IKapC 所有函数都会返回一个长度为 32 位的 **ITKSTATUS**，其字段含义如下：

Bits	31-24	23-20	19-16	15-0
字段	保留	模块 ID	错误类别 ID	错误码 ID

字段	详细描述
模块 ID	4bit: 以 ITKSTATUS_MODULE_XXX 开头的常量
错误类别 ID	4bit: 以 ITKSTATUS_LEVEL_XXX 开头的常量
错误码 ID	16bit: 以 ITKSTATUS_XXX 开头的常量

9.1 模块 ID

ID	值	模块名称
0x01	ITKSTATUS_MODULE_DEVICE	设备模块
0x02	ITKSTATUS_MODULE_BUFFER	缓冲区模块
0x03	ITKSTATUS_MODULE_LOG	日志模块
0x04	ITKSTATUS_MODULE_MANAGER	管理模块
0x05	ITKSTATUS_MODULE_STREAM	数据流传输模块
0x06	ITKSTATUS_MODULE_PARAM	参数模块
0x07	ITKSTATUS_MODULE_SERIAL	串口模块
0x08	ITKSTATUS_MODULE_EVENTINFO	事件信息模块
0x09	ITKSTATUS_MODULE_FEATURE	相机特征模块
0x0A	ITKSTATUS_MODULE_VIEW	视图模块
0x0B	ITKSTATUS_MODULE_BOARD	采集卡模块
0x0C	ITKSTATUS_MODULE_FILE	文件模块

9.2 错误类别 ID

ID	值	描述
0	ITKSTATUS_LEVEL_FAT	致命错误，可能导致程序异常中断
1	ITKSTATUS_LEVEL_ERR	普通错误
2	ITKSTATUS_LEVEL_WRN	警告
3	ITKSTATUS_LEVEL_INF	信息

9.3 错误码 ID

ID	值	描述
0x0000	ITKSTATUS_OK	成功
0x0001	ITKSTATUS_INVALID_HANDLE	无效的句柄
0x0002	ITKSTATUS_INSUFFICIENT_RESOURCES	系统资源不足

0x0003	ITKSTATUS_BUFFER_TOO_SMALL	输入缓冲区太小
0x0004	ITKSTATUS_MISSING_RESOURCE	丢失系统资源
0x0005	ITKSTATUS_UNINITIALIZE	IKapC 没有初始化运行环境
0x0006	ITKSTATUS_DEVICE_ID_OUTOF_RANGE	设备索引越界
0x0007	ITKSTATUS_SERAIL_PORT_NOT_AVALIABLE	设备串口不可用
0x0008	ITKSTATUS_XML_NOT_FOUND	设备 XML 描述文件丢失
0x0009	ITKSTATUS_DEVICE_NOT_ACCESSABLE	设备无法访问
0x000A	ITKSTATUS_DEVICE_PERMISSION_DENY	设备访问被拒绝
0x000B	ITKSTATUS_REGISTRY_NOT_FOUND	注册表丢失
0x000C	ITKSTATUS_XML_PARSE_ERROR	XML 文件无法解析
0x000D	ITKSTATUS_INVALID_ARG	无效的输入参数
0x000E	ITKSTATUS_INVALID_NAME	无效的特征名称
0x000F	ITKSTATUS_INCOMPATIBLE_FEATURE_TYPE	特征类型不匹配函数类型
0x0010	ITKSTATUS_TIME_OUT	操作超时
0x0011	ITKSTATUS_COMMAND_CRASH	相机命令冲突
0x0012	ITKSTATUS_COMMAND_PARAM_OUT_OF_RANGE	相机参数越界
0x0013	ITKSTATUS_COMMAND_NOT_ALLOW	当前状态下不允许执行该命令
0x0014	ITKSTATUS_COMMAND_NOT_PRASE	相机命令无法解析
0x0015	ITKSTATUS_COMMAND_PENDING	执行命令被挂起
0x0016	ITKSTATUS_ARG_OUT_OF_RANGE	输入参数越界
0x0017	ITKSTATUS_NOT_IMPLEMENT	参数未实现
0x0018	ITKSTATUS_NO_MEMORY	内存不足
0x0019	ITKSTATUS_INCOMPATIBLE_ARG_TYPE	参数不匹配
0x001A	ITKSTATUS_STREAM_IN_PROCESS	数据流正在传输
0x001B	ITKSTATUS_PRM_READ_ONLY	参数只读
0x001C	ITKSTATUS_STREAM_IS_OPENED	数据流已经被打开
0x001D	ITKSTATUS_SYSTEM_ERROR	Windows 系统函数调用错误
0x001E	ITKSTATUS_INVALID_ADDRESS	无效的地址
0x001F	ITKSTATUS_BAD_ALIGNMENT	参数边界不适配
0x0020	ITKSTATUS_DEVICE_BUSY	设备正忙, 无法响应命令
0x0021	ITKSTATUS_DEVICE_IS_REMOVED	设备已经被移除
0x0022	ITKSTATUS_DEVICE_NOT_FOUND	未找到设备
0x0023	ITKSATTUS_BOARD_IS_OPENED	设备已经打开, 不允许重复打开相同的设备
0x0024	ITKSTATUS_BOARD_NO_OPENED	设备未打开
0x0025	ITKSTATUS_PRM_WRITE_ONLY	参数只支持写操作
0x0026	ITKSTATUS_BOARD_CONNECTION_FAIL	采集卡和相机连接断开
0x0027	ITKSTATUS_RUNTIME_ERROR	程序运行时错误

0x0028	ITKSTATUS_IO_ERROR	IO 错误
0x0029	ITKSTATUS_BUFFER_OVERFLOW	缓冲区溢出
0x0030	ITKSTATUS_COMMUNICATION_ERROR	通信异常

10 数据类型定义

10.1 宏定义

ID	值	说明
ITKDEV_INFO_ENTRY_MAX_LENGTH	64	ITKDEV_INFO 结构体中字段的固定长度
ITKGIGEDEV_INFO_LENGTH_32_BYTE	32	ITKGIGEDEV_INFO 结构体中字段长度
ITKGIGEDEV_INFO_LENGTH_64_BYTE	64	ITKGIGEDEV_INFO 结构体中字段长度
ITKGIGEDEV_INFO_LENGTH_128_BYTE	128	ITKGIGEDEV_INFO 结构体中字段长度
ITKGIGEDEV_INFO_LENGTH_256_BYTE	256	ITKGIGEDEV_INFO 结构体中字段长度

10.2 结构体定义

- ITKDEV_INFO
相机设备基本信息结构体

字段	含义
char FullName[ITKDEV_INFO_ENTRY_MAX_LENGTH]	设备全名
char FriendlyName[ITKDEV_INFO_ENTRY_MAX_LENGTH]	用户界面显示名称
char VendorName[ITKDEV_INFO_ENTRY_MAX_LENGTH]	设备生产厂商
char ModelName[ITKDEV_INFO_ENTRY_MAX_LENGTH]	设备模型名称
char SerialNumber[ITKDEV_INFO_ENTRY_MAX_LENGTH]	设备序列号
char DeviceClass[ITKDEV_INFO_ENTRY_MAX_LENGTH]	设备类型名称
char DeviceVersion[ITKDEV_INFO_ENTRY_MAX_LENGTH]	设备版本号
char UserDefinedName[ITKDEV_INFO_ENTRY_MAX_LENGTH]	设备自定义名称

- ITKGIGEDEV_INFO
千兆网相机设备专有信息结构体

字段	含义
char MAC[ITKGIGEDEV_INFO_LENGTH_32_BYTE]	设备 MAC 地址
char Ip[ITKGIGEDEV_INFO_LENGTH_32_BYTE]	设备 IP 地址
char SubNetMask[ITKGIGEDEV_INFO_LENGTH_32_BYTE]	设备子网掩码
char GateWay[ITKGIGEDEV_INFO_LENGTH_32_BYTE]	设备默认网关
char NICMac[ITKGIGEDEV_INFO_LENGTH_32_BYTE]	通讯主机 MAC 地址
char NICIp[ITKGIGEDEV_INFO_LENGTH_32_BYTE]	通讯主机 IP 地址
char NICSubNetMask[ITKGIGEDEV_INFO_LENGTH_32_BYTE]	通讯主机子网掩码
char NICGateWay[ITKGIGEDEV_INFO_LENGTH_32_BYTE]	通讯主机默认网关
char NICAdapterName[ITKGIGEDEV_INFO_LENGTH_256_BYTE + 4]	通讯主机网卡名称
char NICDescription[ITKGIGEDEV_INFO_LENGTH_128_BYTE + 4]	通讯主机网卡详细描述信息
char reserved[128]	预留字段

- ITK_CL_DEV_INFO
Camera Link 相机设备专有信息结构体

字段	含义
uint32_t HostInterface	主接口类型
uint32_t BoardIndex	板卡索引
char Reserved[256]	预留字段

- ITK_CXP_DEV_INFO
CoaXPress 相机设备专有信息结构体

字段	含义
uint32_t BoardIndex	板卡索引
uint32_t MasterPort	主端口号
uint32_t SlaveCount	从端口数量
uint32_t SlavePort[7]	从端口号
uint32_t CameraId	相机 ID
char Reserved[256]	预留字段

- ITK_U3V_DEV_INFO
USB30 相机设备专有信息结构体

字段	含义
uint32_t VID	相机 VID
uint32_t PID	相机 PID
uint32_t USBVersion	USB 连接版本
char Reserved[256]	预留字段

- ITKBOARD_INFO
采集卡设备专有信息结构体

字段	含义
char Name[ITKDEV_INFO_ENTRY_MAX_LENGTH]	用于标识的设备名称
char Reserved[256]	预留字段

10.3 句柄定义

定义	说明
IKAPC_DECLARE_HANDLE(ITKDEVICE)	相机设备句柄
IKAPC_DECLARE_HANDLE(ITKBOARD)	采集卡设备句柄
IKAPC_DECLARE_HANDLE(ITKFEATURE)	相机特征句柄
IKAPC_DECLARE_HANDLE(ITKEVENTINFO)	事件信息句柄
IKAPC_DECLARE_HANDLE(ITKSTREAM)	数据流句柄
IKAPC_DECLARE_HANDLE(ITKBUFFER)	缓冲区句柄
IKAPC_DECLARE_HANDLE(ITKVIEW)	视图句柄

10.4 类型定义

定义	说明
<code>typedef __int8 int8_t</code>	8bit 有符号整数类型
<code>typedef __int16 int16_t</code>	16bit 有符号整数类型
<code>typedef __int32 int32_t</code>	32bit 有符号整数类型
<code>typedef __int64 int64_t</code>	64bit 有符号整数类型
<code>typedef unsigned __int8 uint8_t</code>	8bit 无符号整数类型
<code>typedef unsigned __int16 uint16_t</code>	16bit 无符号整数类型
<code>typedef unsigned __int32 uint32_t</code>	32bit 无符号整数类型
<code>typedef unsigned __int64 uint64_t</code>	64bit 无符号整数类型
<code>typedef bool _Bool</code>	C99 标准内置布尔类型

10.5 回调函数定义

定义	说明
<pre>typedef void (IKAPC_CC *PITKEVENTINFOCALLBACK)(void* pContext, ITKEVENTINFO hEventInfo)</pre>	设备事件处理回调函数
<pre>typedef void (IKAPC_CC *PITKSTREAMCALLBACK)(uint32_t eventType, void* pContext)</pre>	数据流事件处理回调函数