TUCAM-API
开发指南

目录

1. 使用前阅读

这份文档和软件示例代码是 TUCSEN 的内部文件和公布内容，以用户能够创建使用 TUCSEN 数字相机中的应用。

本文档和软件示例代码只针对上述目的而公开的，并且不构成所有者的许可、转让或任何其他权力。

使用软件文档的所有风险和结果仍然取决于用户。

本文档可能包括技术错误或印刷错误。并且不能保证这样的错误或文本所产生的任何损害。

TUCSEN 不承诺更新或保持当前的这个文档中所包含的信息。

所有品牌和产品名称都是其各自所有者的商标或注册商标。

2. 简介

本手册详细描述了 TUCAM-API 规范操作使用 TUCSEN 数字相机。TUCAM-API 软件开发工具包被称为“SDK”。TUCAM-API 控制数字相机的部分被成为“模块”。

SDK 包含源代码模块和一个示例应用程序，展示了如何访问 TUCAM-API。SDK 用户免费使用该软件在任何他们喜欢的方式，如部分修改源代码或创建完全独立的项目。

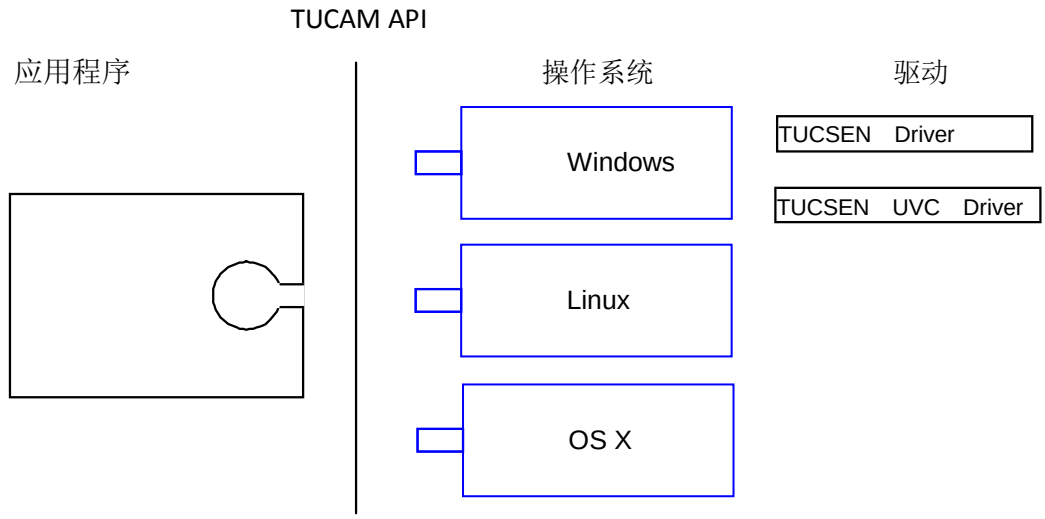
这个 SDK 设计特别容易理解。出于这个原因，函数接口的数量限制到最少，并且函数的调用格式采用 C 语言的写法。

部分扩展的函数是某些特定数字相机可以使用的附加功能。

不同数字相机的数值可能不同，这取决于捕捉图像所使用的数字相机型号。数值应该简单地视为指南，而不是精确值。

3. 概述

3.1 层结构



TUCSEN 数字相机通过 SDK 连接不同的操作系统的数字相机驱动来达到控制数字相机和采集图像数据的作用。

目前的 SDK 只支持 Windows 系统。

3.2 原理

数字相机的具体总线接口和库通过 TUCAM-API 封装。您只需要访问 TUCAM-API 层。模块层提供更高级的 TUCAM-API 集成。模块可以不断更新访问新相机和提供新接口技术，而无需重新编译您的软件。

3.3 接口类型

TUCAM-API 功能可以分为很多类型

起始/终止处理

相机信息采集

性能/属性获取和设置

内存管理

捕获控制

文件控制

扩展控制

TUCAM-API 不包含用于显示图像的程序。因为一些显示图像的方法难以预测，这取决于应用程序，它是不可能支持所有这些通用模块的。当调用显示程序时，检测图像是否更新，并且在更新后绘制图像。对于更详细的信息，请参阅示例源代码。

3.4 术语

3.4.1 捕获模式

相机的捕获模式分为以下 2 类：

序列模式（流模式）：用来捕获连续的图像数据。

触发模式：相机通过外部信号来捕获图像。我们称这个选项为“触发模式”，您可以调用 TUCAM_Cap_SetTrigger()来配置此选项。我们也把外部信号称为“外部触发”。

3.4.2 图像单元

通常情况下是二维的，具有垂直和水平方向。

帧：是一个用于图像数据的单位。对于一帧，一个像素的数据是从左到右和从上到下对齐的。这是一系列的图像数据单位。

3.4.3 触发模式

标准模式（Standard）：当相机接收到电平信号后（由激活边沿决定）开始进行一帧或多帧的图像捕获，捕获帧数由配置参数决定。参考 TUCAM_TRIGGER_ATTR 结构体。

同步模式（Synchronization）：当相机接收到电平信号后（由激活边沿决定）开始进行曝光，当收到相反的电平信号后，结束曝光、并且进行图像数据的捕获。即实现每一帧的曝光与读出，均与外触发信号完全同步。

全局模式（Global）：在相机未触发前进行预触发，当相机接收到电平信号后（由激活边沿决定）或者为软件设定的曝光时间时，结束当前正在进行的重置操作，待曝光结束时捕获图像数据，并重新开始预触发。该方式用于控制卷帘曝光模式的相机实现全局曝光模式。

曝光模式：
曝光时间：接收到触发信号后，由 TUIDP_EXPOSURETM 设置的曝光时间决定
电平宽度：接收到触发信号后，曝光时间由电平的宽度所决定
注：标准模式（Standard）和全局模式（Global）可配置这两个选项。同步模式（Synchronization）只能是电平宽度。

激发电平类型：
上升电平（Rising Edge）：接收到的触发电平处于上升沿时开始曝光
下降电平（Falling Edge）：接收到的触发电平处于下降沿时开始曝光

3.4.4 相机状态

相机的状态决定了能调用那些函数。一些函数会改变相机的状态。下面描述了 4 种相机状态：

不稳定：参数设置和其他函数调用，但它们不在被设置的状态。

稳定的：参数和函数被设置，但是因为没有帧内存被创建，捕获图像不能开始。

准备： 帧内存已经被创建，图像捕获可以开始。

繁忙： 图像捕获正在被执行。

3.5 接口列表

```
// Initialize uninitialize and misc.
TUCAMRET   TUCAM_Api_Init(PTUCAM_INIT pInitParam);
TUCAMRET   TUCAM_Api_Uninit ();

TUCAMRET   TUCAM_Dev_Open (PTUCAM_OPEN pOpenParam);
TUCAMRET   TUCAM_Dev_Close (HDTUCAM hTUCam);

// Get some device information (VID/PID/Version)
TUCAMRET   TUCAM_Dev_GetInfo (HDTUCAM hTUCam, PTUCAM_VALUE_INFO pInfo);

// Capability control   see enumerate TUCAM_IDCAPA
TUCAMRET   TUCAM_Capa_GetAttr (HDTUCAM hTUCam, PTUCAM_CAPA_ATTR pAttr);
TUCAMRET   TUCAM_Capa_GetValue (HDTUCAM hTUCam, INT32 nCapa, INT32 *pnVal);
TUCAMRET   TUCAM_Capa_SetValue (HDTUCAM hTUCam, INT32 nCapa, INT32 nVal);
TUCAMRET   TUCAM_Capa_GetValueText (HDTUCAM hTUCam, PTUCAM_VALUE_TEXT pVal);

// Property control see enumerate PTUCAM_PROP_ATTR
TUCAMRET   TUCAM_Prop_GetAttr (HDTUCAM hTUCam, PTUCAM_PROP_ATTR pAttr);
TUCAMRET   TUCAM_Prop_GetValue (HDTUCAM hTUCam, INT32 nProp, DOUBLE *pdbVal, INT32 nChn);
TUCAMRET   TUCAM_Prop_SetValue (HDTUCAM hTUCam, INT32 nProp, DOUBLE dbVal, INT32 nChn );
TUCAMRET   TUCAM_Prop_GetValueText (HDTUCAM hTUCam, PTUCAM_VALUE_TEXT pVal, INT32
nChn );

// Buffer control
TUCAMRET   TUCAM_Buf_Alloc (HDTUCAM hTUCam, PTUCAM_FRAME pFrame);
TUCAMRET   TUCAM_Buf_Release (HDTUCAM hTUCam);
TUCAMRET   TUCAM_Buf_AbortWait (HDTUCAM hTUCam);
TUCAMRET   TUCAM_Buf_WaitForFrame (HDTUCAM hTUCam, PTUCAM_FRAME pFrame);
```

```
TUCAMRET    TUCAM_Buf_CopyFrame (HDTUCAM hTUCam, PTUCAM_FRAME pFrame);

// Capturing control
// ROI
TUCAMRET    TUCAM_Cap_SetROI (HDTUCAM hTUCam, TUCAM_ROI_ATTR roiAttr);
TUCAMRET    TUCAM_Cap_GetROI (HDTUCAM hTUCam, PTUCAM_ROI_ATTR pRoiAttr);
// Trigger
TUCAMRET    TUCAM_Cap_SetTrigger (HDTUCAM hTUCam, TUCAM_TRIGGER_ATTR tgrAttr);
TUCAMRET    TUCAM_Cap_GetTrigger (HDTUCAM hTUCam, PTUCAM_TRIGGER_ATTR pTgrAttr);
TUCAMRET    TUCAM_Cap_DoSoftwareTrigger(HDTUCAM hTUCam);     // in trigger mode

// OutPutTrigger
TUCAMRET TUCAM_Cap_SetTriggerOut(HDTUCAM hTUCam, TUCAM_TRGOUT_ATTR tgroutAttr);
TUCAMRET TUCAM_Cap_GetTriggerOut(HDTUCAM hTUCam, PTUCAM_TRGOUT_ATTR pTgrOutAttr);

// Capturing
// uiMode see enumerate TUCAM_CAPTURE_MODES
TUCAMRET    TUCAM_Cap_Start(HDTUCAM hTUCam, UINT32 uiMode);
TUCAMRET    TUCAM_Cap_Stop (HDTUCAM hTUCam);

// File control
// Image
TUCAMRET    TUCAM_File_SaveImage (HDTUCAM hTUCam, TUCAM_FILE_SAVE fileSave);
// Video
TUCAMRET    TUCAM_Rec_Start(HDTUCAM hTUCam, TUCAM_REC_SAVE recSave);
TUCAMRET    TUCAM_Rec_AppendFrame(HDTUCAM hTUCam, PTUCAM_FRAME pFrame);
TUCAMRET    TUCAM_Rec_Stop (HDTUCAM hTUCam);

// Extened control
TUCAMRET TUCAM_Reg_Read (HDTUCAM hTUCam, TUCAM_REG_RW regRW);
TUCAMRET TUCAM_Reg_Write(HDTUCAM hTUCam, TUCAM_REG_RW regRW);
```

4. 应用程序调用 TUCAM-API

正在使用 TUCAM-API 控制相机时，函数调用应该按照下列的调用过程：

- 初始化相机
- 设置相机参数
- 开始捕获数据
- 确保拍摄已经完成，并获得数据
- 进行相机终止处理

4.1 初始化和终止程序

4.1.1 接口

```
// Initialize uninitialize and misc.
TUCAMRET    TUCAM_Api_Init(PTUCAM_INIT pInitParam);
TUCAMRET    TUCAM_Api_Uninit ();

TUCAMRET    TUCAM_Dev_Open (PTUCAM_OPEN pOpenParam);
TUCAMRET    TUCAM_Dev_Close (HDTUCAM hTUCam);

// Get some device information (VID/PID/Version)
TUCAMRET    TUCAM_Dev_GetInfo (HDTUCAM hTUCam, PTUCAM_VALUE_INFO pInfo);
```

4.1.2 调用顺序

首先，驱动程序初始化。当应用程序安装传输句柄的初始化已经成功完成，获取可以控制的相机数量。

当应用程序启动时，调用相机初始化函数执行初始化操作。当初始化函数调用成功后，其他函数才能正常被调用执行。

相机终止函数用用于程序的关闭。当一个相机被挂起，或资源被释放而不再控制相机时执行的函数。例如，当应用程序退出。当终止函数被调用是，其他的功能函数调用将不被执行，直到初始化函数再次调用之后。

4.1.3 驱动初始化

驱动使用 TUCAM_Api_Init 函数进行初始操作。该函数初始化帧采集和控制相机。

4.1.4 相机初始化

相机初始化使用 TUCAM_Dev_Open 函数。该函数获取必要的相机句柄来做为其他函数的输入参数。

4.1.5 相机产品信息

当调用 TUCAM_Dev_Open 函数打开相机之后，可以通过相机句柄获取相机的产品信息。

```
// enumerate information id
typedef enum
{
    TUIDI_BUS               = 0x01,     // USB 口类型： USB2.0/USB3.0
    TUIDI_VENDOR            = 0x02,     // 厂商 ID
    TUIDI_PRODUCT           = 0x03,     // 产品 ID
    TUIDI_VERSION_API       = 0x04,     // TUCAM- API 版本号
    TUIDI_VERSION_FRMW      = 0x05,     // 固件版本号
    TUIDI_VERSION_FPGA      = 0x06,     // FPGA 版本号（保留）
    TUIDI_VERSION_DRIVER    = 0x07,     // 驱动版本号 （保留）
    TUIDI_TRANSFER_RATE     = 0x08,     // USB 传输速率
    TUIDI_CAMERA_MODEL      = 0x09,     // 相机型号（字符串类型）
    TUIDI_ENDINFO           = 0x0A,     // 产品信息 ID 结束位
}TUCAM_IDINFO;
```

示例

```
    TUIDI_BUS               = 0x300,    // USB3.0
    TUIDI_VENDOR            = 0x5453,   // TUCSEN
    TUIDI_PRODUCT           = 0x6404,   // Dhyana 400D
    TUIDI_VERSION_API       = “1.0.0.1”,    // “1.0.0.1”
```

```
TUIDI_VERSION_FRMW      =-230244288 ,     // “f246c040”
TUIDI_VERSION_FPGA      = 0,              // the FPGA version （保留）
TUIDI_VERSION_DRIVER    = “1.2.3.10”,     // “1.2.3.10”
TUIDI_TRANSFER_RATE     = 292,            // 292MB / Sec
TUIDI_CAMERA_MODEL      = “Dhyana 400D”,  // “Dhyana 400D”
```

4.1.6 终止程序

终止相机程序使用 TUCAM_Dev_Close 函数。调用这个函数释放被用于相机帧获取的端口及资源。这个函数被调用后，相机将不再被控制。

4.1.7 示例代码

```
int main (int argc, char** argv)
{
    TUCAM_INIT  itApi;    // 初始化 SDK 环境参数
    TUCMA_OPEN  opCam;    // 打开相机参数

    itApi.pstrConfigPath = NULL;
    itApi.uiCamCount = 0;
    if (TUCAMRET_SUCCESS != TUCAM_Api_Init(&itApi))
    {
        // 初始化 SDK API 环境失败
        return 0;
    }

    if (0 == itApi.uiCamCount)
    {
        // 没有相机
        return 0;
    }

    opCam.hIdxTUCam = 0;
    opCam.uiIdxOpen = 0;

    if (TUCAMRET_SUCCESS != TUCAM_Dev_Open(&opCam))
    {
        // 打开相机失败
        return 0;
    }

    // 应用程序可以使用 opCam.hIdxTUCam 句柄
```

```
30.
31.     TUCAM_Dev_Close(opCam.hIdxTUCam);       // 关闭相机
32.     TUCAM_Api_Uninit();                     // 反初始化 SDK API 环境
33.
34.     return 0;
35. }
36.
```

4.2 性能获取和设置

4.2.1 接口

```
// Capability control   see enumerate TUCAM_IDCAPA
TUCAMRET   TUCAM_Capa_GetAttr (HDTUCAM hTUCam, PTUCAM_CAPA_ATTR pAttr);
TUCAMRET   TUCAM_Capa_GetValue (HDTUCAM hTUCam, INT32 nCapa, INT32 *pnVal);
TUCAMRET   TUCAM_Capa_SetValue (HDTUCAM hTUCam, INT32 nCapa, INT32 nVal);
TUCAMRET   TUCAM_Capa_GetValueText (HDTUCAM hTUCam, PTUCAM_VALUE_TEXT pVal);
```

4.2.2 调用顺序

设置和获取通常在相机进行捕获之前或者之后已经完成。如果设置函数在数据捕获时被调用，可能会返回错误的代码 TUCAMRET 在某些情况下。

4.2.3 性能索引

```
// enumerate capability id
typedef enum
{
    TUIDC_RESOLUTION        = 0x00,     //分辨率
    TUIDC_PIXELCLOCK        = 0x01,     // 像素时钟
    TUIDC_BITOFDEPTH        = 0x02,     // 数据位宽
    TUIDC_ATEXPOSURE        = 0x03,     // 自动曝光
    TUIDC_HORIZONTAL        = 0x04,     // 水平镜像
    TUIDC_VERTICAL          = 0x05,     // 垂直镜像
    TUIDC_ATWBBALANCE       = 0x06,     // 自动白平衡 （彩色相机）
    TUIDC_FAN_GEAR          = 0x07,     // 风扇档位   （制冷相机）
    TUIDC_IMGMODESELECT     = 0x16,     // 图像模式选择(0x01:CMS 模式 0x02: 11BIT 模式)
    TUIDC_LEDENBALE         = 0x1E,     // Led 灯开关
    TUIDC_ENDCAPABILITY     = 0x1F,     // 性能 ID 结束位
}TUCAM_IDCAPA;
```

注：如果相机不支持该性能 ID，将返回错误代码 TUCAMRET_NOT_SUPPORT.

4.2.3 示例代码

```
// 以分辨率 TUIDC_RESOLUTION 为例
// 获取分辨率范围
void GetResolutionRange()
{
    TUCAM_CAPA_ATTR attrCapa;
    TUCAM_VALUE_TEXT valText;

    char szRes[64] = {0};
    valText.nTextSize = 64;
    valText.pText = &szRes[0];
    attrCapa.idCapa = TUIDC_RESOLUTION;
    if (TUCAMRET_SUCCESS == TUCAM_Capa_GetAttr(opCam.hIdxTUCam, &attrCapa))
    {
        // 获取分辨率个数
        int nCnt = attrCapa.nValMax - attrCapa.nValMin + 1;
        valText.nID = TUIDC_RESOLUTION;

        for (int i=0; i<nCnt; ++i)
        {
            valText.dbValue = i;
            TUCAM_Capa_GetValueText(opCam.hIdxTUCam, &valText);
            szRes = valText.pText;
            // 将分辨率文本添加到下拉菜单
        }
    }
}

// 获取当前分辨率
void GetCurrentResolution()
{
    int nVal = 0;

    if (TUCAMRET_SUCCESS == TUCAM_Capa_GetValue(opCam.hIdxTUCam, \
                                                TUIDC_RESOLUTION, \
                                                &nVal))
    {
        // nVal 返回当前分辨率索引
    }
}

// 设置当前分辨率
```

```
42. void SetCurrentResolution(int nIdxRes)
43. {
44.     TUCAM_Capa_SetValue(opCam.hIdxTUCam, TUIDC_RESOLUTION, nIdxRes);
45. }
46.
```

4.3 属性获取和设置

4.3.1 接口

```
// Property control see enumerate TUCAM_IDPROP
TUCAMRET    TUCAM_Prop_GetAttr (HDTUCAM hTUCam, PTUCAM_PROP_ATTR pAttr);
TUCAMRET    TUCAM_Prop_GetValue (HDTUCAM hTUCam, INT32 nProp, DOUBLE *pdbVal, INT32 nChn);
TUCAMRET    TUCAM_Prop_SetValue       (HDTUCAM hTUCam, INT32 nProp, DOUBLE dbVal, INT32
nChn);
TUCAMRET    TUCAM_Prop_GetValueText (HDTUCAM hTUCam, PTUCAM_VALUE_TEXT pVal, INT32
nChn );
```

4.3.2 调用顺序

设置和获取通常在相机进行捕获之前或者之后已经完成。如果设置函数在数据捕获时被调用，可能会返回错误的代码 TUCAMRET 在某些情况下。

4.3.3 属性索引

```
// enumerate property id
typedef enum
{
    TUIDP_GLOBALGAIN        = 0x00,     // 全局增益
    TUIDP_EXPOSURETM        = 0x01,     // 曝光时间
    TUIDP_BRIGHTNESS        = 0x02,     // 亮度 (自动曝光状态有效)
    TUIDP_BLACKLEVEL        = 0x03,     // 黑电平
    TUIDP_TEMPERATURE       = 0x04,     // 温度
    TUIDP_SHARPNESS         = 0x05,     // 锐化
    TUIDP_NOISELEVEL        = 0x06,     // 降噪等级
    TUIDP_HDR_KVALUE        = 0x07,     // HDR 的 K 值（sCMOS 相机支持）

    // image process property
    TUIDP_GAMMA             = 0x08,     // 伽玛
```

```
    TUIDP_CONTRAST      = 0x09,     // 对比度
    TUIDP_LFTLEVELS     = 0x0A,     // 左色阶
    TUIDP_RGTLEVELS     = 0x0B,     // 右色阶
    TUIDP_CHNLGAIN      = 0x0C,     // 通道增益 （彩色相机支持）
    TUIDP_SATURATION    = 0x0D,     // 饱和度   （彩色相机支持）
    TUIDP_ENDPROPERTY   = 0x0E,     // 属性 ID 结束位
}TUCAM_IDPROP;
```

注：如果相机不支持该属性 ID，将返回错误代码 TUCAMRET_NOT_SUPPORT。

4.3.4 示例代码

```
// 以曝光时间为例
// 获取曝光时间范围
void GetExposureTimeRange()
{
    TUCAM_PROP_ATTR attrProp;

    attrProp.nIdxChn = 0;    // 当前通道
    attrProp.idProp = TUIDP_EXPOSURETM;

    if (TUCAMRET_SUCCESS == TUCAM_Prop_GetAttr(opCam.hIdxTUCam, &attrProp))
    {
        // 曝光时间范围
        attrProp.dbValMin;    // 最小曝光时间
        attrProp.dbValMax;    // 最大曝光时间
        attrProp.dbValDft;    // 默认曝光时间
        attrProp.dbValStep;   // 曝光时间步长
    }
}

// 获取当前曝光时间
void GetCurrentExposureTime()
{
    double dbVal = 1.0f;

    if (TUCAMRET_SUCCESS == TUCAM_Prop_GetValue(opCam.hIdxTUCam, \
                                                TUIDP_EXPOSURETM, \
                                                &dbVal))
    {
        // dbVal 返回当前曝光时间，单位 ms
    }
}

// 设置当前曝光时间
```

```
34. void SetCurrentExposureTime(double dbVal)
35. {
36.     TUCAM_Prop_SetValue(opCam.hIdxTUCam, TUIDP_EXPOSURETM, dbVal);
37. }
38.
```

4.4 内存管理

4.4.1 接口

```
// Buffer control    see structure TUCAM_FRAME
TUCAMRET    TUCAM_Buf_Alloc (HDTUCAM hTUCam, PTUCAM_FRAME pFrame);
TUCAMRET    TUCAM_Buf_Release (HDTUCAM hTUCam);
TUCAMRET    TUCAM_Buf_AbortWait (HDTUCAM hTUCam);
TUCAMRET    TUCAM_Buf_WaitForFrame (HDTUCAM hTUCam, PTUCAM_FRAME pFrame);
TUCAMRET    TUCAM_Buf_CopyFrame (HDTUCAM hTUCam, PTUCAM_FRAME pFrame);
```

4.4.2 调用顺序

内存的分配和释放，内存的分配 TUCAM_Buf_Alloc 必须在 TUCAM_Cap_Start 数据开始捕获之前调用；而内存的释放 TUCAM_Buf_Release 必须在 TUCAM_Cap_Stop 停止数据捕获之后调用。

数据的获取，必须在 TUCAM_Cap_Start 数据开始捕获之后调用 TUCAM_Buf_WaitForFrame

等待数据捕获的完成，并且可以通过 TUCAM_Buf_CopyFrame 拷贝不同格式的数据。

如有进行数据等待和数据拷贝的调用，在停止数据捕获之前调用 TUCAM_Buf_AbortWait 结束数据等待，之后再调用 TUCAM_Cap_Stop 停止数据捕获。

4.4.3 帧结构体

```
// the camera frame structure
typedef struct    _tagTUCAM_FRAME
{
    // TUCAM_Buf_WaitForFrame 使用这个结构体，一些成员变量有不同的调用方向。
    // "input" 表示应用程序在设置之前调用
    // "output"表示程序调用接口后返回所需值

    CHAR szSignature[8];        //   [out] 版权信息

    //   The based information
    USHORT usHeader;            //   [out] 帧的头部大小
    USHORT usOffset;            //   [out] 帧数据的头部偏移量（一般与头部大小相同）
    USHORT usWidth;             //   [out] 帧的宽度
    USHORT usHeight;            //   [out] 帧的高度
    UINT32 uiWidthStep;         //   [out] 帧的宽度步长

    UCHAR    ucDepth;           //   [out] 帧的数据位深
    UCHAR    ucFormat;          //   [out] 帧的数据格式
    UCHAR    ucChannels;        //   [out] 帧的数据通道数
    UCHAR    ucElemBytes;       //   [out] 帧的像素占字节数
    UCHAR    ucFormatGet;       //   [in]   获取帧的格式          （参考 TUFRM_FORMATS）

    UINT32 uiIndex;             //   [in/out] 帧的当前序号
    UINT32 uiImgSize;           //   [out] 帧的大小
    UINT32 uiRsdSize;           //   [in]   帧保留的数量          （需要多少帧，触发使用）
    UINT32 uiHstSize;           //   [out] 帧直方图统计大小       （保留位）

    PUCHAR pBuffer;             //   [in/out] 帧缓冲区
} TUCAM_FRAME, *PTUCAM_FRAME;

// Define the struct of image header
typedef struct _tagTUCAM_IMG_HEADER
{
    CHAR szSignature[8];                    //   [out] 版权信息

    //   The based information
    USHORT usHeader;                        //   [out] 帧头部大小
    USHORT usOffset;                        //   [out] 帧数据偏移大小
    USHORT usWidth;                         //   [out] 帧图像的宽度
    USHORT usHeight;                        //   [out] 帧图像的高度
    UINT32 uiWidthStep;                     //   [out] 帧图像的宽度步长
```

```
    UCHAR    ucDepth;                    //  [out] 帧图像的数据位深
    UCHAR    ucFormat;                   //  [out] 帧图像的数据格式
    UCHAR    ucChannels;                 //  [out] 帧图像的通道数
    UCHAR    ucElemBytes;                //  [out] 帧图像的数据字节数
    UCHAR    ucFormatGet;                //  [in/out] 需要获取的图像格式 TUFRM_FORMATS

    UINT32 uiIndex;                      //  [out] 帧图像的序列号（保留）
    UINT32 uiImgSize;                    //  [out] 帧图像数据的大小
    UINT32 uiRsdSize;                    //  [in]  需要获取的帧数
    UINT32 uiHstSize;                    //  [out] 帧图像的保留字段
    //  The data
    PUCHAR pImgData;                     //  [in/out] 指向帧数据的缓冲区
    UINT32 *pImgHist;                    //  [in/out] 指向直方图数据的缓冲区

    USHORT usLLevels;                    //  [out] The image left levels value
    USHORT usRLevels;                    //  [out] The image right levels value

    CHAR ucRsd1[64];                     //  The reserved

    DOUBLE dblExposure;                  //  [in/out] 当前曝光时间
    CHAR     ucRsd2[170];                //  The reserved
    DOUBLE dblTimeStamp;                 //  [in/out] 当前时间戳
    DOUBLE dblTimeLast;                  //  [in/out] 当前持续时间
#ifdef  TUCAM_TARGETOS_IS_WIN32
#ifndef  _WIN64
    CHAR     ucRsd3[697];       // The reserved
#else
    CHAR     ucRsd3[681];       // The reserved
#endif

#else
    CHAR     ucRsd3[681];       // The reserved
#endif
}TUCAM_IMG_HEADER, *PTUCAM_IMG_HEADER;

// enumerate frame format
typedef enum
{
    TUFRM_FMT_RAW           = 0x10,      // Raw 格式数据
    TUFRM_FMT_USUAl         = 0x11,      // 一般的数据（8bit/16bit、黑白、彩色）
    TUFRM_FMT_RGB888        = 0x12,      // RGB888 的数据用于显示
}TUFRM_FORMATS;
```

4.4.4 示例代码

```
TUCAM_FRAME m_frame;      // 帧对象
HANDLE m_hThdGrab;        // 取图线程事件句柄
BOOL m_bLiving;           // 是否取图

BOOL CDlgTUCam::StartCapture()
{
    m_frame.pBuffer     = NULL;
    m_frame.ucFormatGet = TUFRM_FMT_RGB888;  // 帧数据格式（RGB888）
    m_frame.uiRsdSize   = 1;    // 一次捕获帧数（TUCCM_TRIGGER_STANDARD 可大于 1）

    if（TUCAMRET_SUCCESS != TUCAM_Buf_Alloc(m_opCam.hIdxTUCam， &m_frame))
    {
        return FALSE;
    }

    if (TUCAMRET_SUCCESS != TUCAM_Cap_Start(m_opCam.hIdxCam, TUCCM_SEQUENCE))
    {
        TUCAM_Buf_Release(m_opCam.hIdxTUCam);
        return FALSE;
    }

    m_bLiving  = TRUE;
    m_hThdGrab = CreateEvent(NULL, TRUE, FALSE, NULL);
    _beginthread(GrabThread, 0, this);

    return TRUE;
}

Void __cdecl CDlgTUCam::GrabThread(LPVOID lParam)
{
    CDlgTUCam *pTuCam = (CDlgTUCam *)lParam;

    While (pTUCam->m_bLiving)
    {
        pTUCam->m_frame.ucFormatGet = TUFRM_FMT_RGB888;
        if(TUCAMRET_SUCCESS == TUCAM_Buf_WaitForFrame(pTUCam->m_opCam.hIdxTUCam,\
                                                      &pTUCam->m_frame))
        {
            // pTUCam->m_frame.pBuffer 返回捕获的图像数据，格式为 TUFRM_FMT_RGB88
            // 该数据可以用于显示

            TUCAM_IMG_HEADER     frmhead;
```

```
43.             memcpy(&frmhead, pIn->m_frame.pBuffer, sizeof(TUCAM_IMG_HEADER));
44.             // 该数据可获取头部信息
45.
46.             // 可获取其他格式数据
47.             pTUCam->m_frame.ucFormatGet = TUFRM_FMT_USUAL;
48.             if(TUCAMRET_SUCCESS==TUCAM_Buf_CopyFrame(pTUCam->m_opCam.hIdxTUCam,\
49.                                                      &pTUCam->m_frame))
50.             {
51.                 // pTUCam->m_frame.pBuffer 返回捕获的图像数据
52.             }
53.         }
54.     }
55.
56.     SetEvent(pTUCam->m_hThdGrab);
57.     _endthread();
58. }
59.
60. void CDlgTUCam::StopCapture()
61. {
62.     m_bLiving = FALSE;
63.     TUCAM_BUF_AbortWait();    // 如果调用 TUCAM_Buf_WaitForFrame 接口
64.
65.     WaitForSingleObject(m_hThdGrab, INFINITE);    // 等待取图线程退出
66.     CloseHandle(m_hThdGrab);
67.     m_hThdGrab = NULL;
68.
69.     TUCAM_Cap_Stop(m_opCam.hIdxTUCam);        // 停止数据捕获
70.     TUCAM_Buf_Release(m_opCam.hIdxTUCam);     // 释放分配的内存
71. }
72.
```

4.5 捕获控制

4.5.1 接口

```
// Capturing control
// ROI
TUCAMRET    TUCAM_Cap_SetROI (HDTUCAM hTUCam, TUCAM_ROI_ATTR roiAttr);
TUCAMRET    TUCAM_Cap_GetROI (HDTUCAM hTUCam, PTUCAM_ROI_ATTR pRoiAttr);
// Trigger
TUCAMRET    TUCAM_Cap_SetTrigger (HDTUCAM hTUCam, TUCAM_TRIGGER_ATTR tgrAttr);
```

```
TUCAMRET    TUCAM_Cap_GetTrigger (HDTUCAM hTUCam, PTUCAM_TRIGGER_ATTR pTgrAttr);
TUCAMRET    TUCAM_Cap_DoSoftwareTrigger(HDTUCAM hTUCam);     // in trigger mode

// Capturing
// uiMode see enumerate TUCAM_CAPTURE_MODES
TUCAMRET    TUCAM_Cap_Start(HDTUCAM hTUCam, UINT32 uiMode);
TUCAMRET    TUCAM_Cap_Stop (HDTUCAM hTUCam);
```

4.5.2 调用顺序

设置 ROI 属性 TUCAM_Cap_SetROI 和触发属性 TUCAM_Cap_SetTrigger，需要在开始捕获数据之前调用，如果在数据捕获时调用有可能返回 TUCAMRET 错误代码。

4.5.3 捕获模式索引

```
// enumerate the capture mode
typedef enum
{
    TUCCM_SEQUENCE                  = 0x00,        //  数据捕获采用序列模式
    TUCCM_TRIGGER_STANDARD          = 0x01,        //  数据捕获采用标准触发模式
    TUCCM_TRIGGER_SYNCHRONOUS  = 0x02,        //  数据捕获采用同步触发模式
    TUCCM_TRIGGER_GLOBAL            = 0x03,        //  数据捕获采用全局触发模式
    TUCCM_TRIGGER_SOFTWARE          = 0x04,        //  数据捕获采用软件触发模式
}TUCAM_CAPTURE_MODES;
```

4.5.4 示例代码

```
1.  // 设置 ROI 模式
2.  void SetROIMode()
3.  {
4.      TUCAM_ROI_ATTR roiAttr;
5.      roiAttr.bEnable = TRUE;
6.      roiAttr.nVOffset= 100;
7.      roiAttr.nHOffset = 100;
8.      roiAttr.nWidth   = 800;
9.      roiAttr.nHeight  = 600;
10.
11.     TUCAM_Cap_SetROI(m_opCam.hIdxTUCam, roiAttr);
12.     TUCAM_Cap_Start(m_opCam.hIdxCam, TUCCM_SEQUENCE);     // 序列模式（即流模式）
13.
14.     // 数据获取参考内存管理示例代码
```

```
15. }
16.
17. // 设置触发模式
18. void SetTriggerMode()
19. {
20.     TUCAM_TRIGGER_ATTR tgrAttr;
21.
22.     tgrAttr.nTgrMode = TUCCM_TRIGGER_STANDARD;    // 标准触发模式
23.     tgrAttr.nExpMode = TUCTE_EXPTM;               // 曝光模式
24.     tgrAttr.nEdgeMode= TUCTE_RISING;              // 激发上升沿
25.     tgrAttr.nFrames  = 1;                         // 触发 1 帧
26.     tgrAttr.nDelayTm = 0;                         // 延时 0 ms
27.
28.     TUCAM_Cap_SetTrigger(m_opCam.hIdxTUCam, tgrAttr);
29.     TUCAM_Cap_Start(m_opCam.hIdxCam, TUCCM_STANDARD);     // 标准触发模式
30.
31.     // 数据获取参考内存管理示例代码
32. }
33.
```

4.6 文件控制

4.6.1 接口

```
// File control
// Image
TUCAMRET   TUCAM_File_SaveImage (HDTUCAM hTUCam, TUCAM_FILE_SAVE fileSave);
// Video
TUCAMRET   TUCAM_Rec_Start(HDTUCAM hTUCam, TUCAM_REC_SAVE recSave);
TUCAMRET   TUCAM_Rec_AppendFrame(HDTUCAM hTUCam, PTUCAM_FRAME pFrame);
TUCAMRET   TUCAM_Rec_Stop (HDTUCAM hTUCam);
```

4.6.2 调用顺序

录像开始 TUCAM_Rec_Start 需要在 TUCAM_Cap_Start 开始捕获数据之后调用，并且设置的捕获方式必须是 TUCCM_SEQUENCE 模式。录像过程中通过调用 TUCAM_Rec_AppendFrame 要将图像数据写入文件中，录像结束调用 TUCAM_Rec_Stop 来结束录像过程。

4.6.3 文件结构体

```
// the file save structure
typedef struct _tagTUCAM_FILE_SAVE
{
    INT32       nSaveFmt;          //  [in]  保存文件的格式   （参考 TUIMG_FORMATS）
    PCHAR     pstrSavePath;        //  [in]  保存文件的路径   （包含文件名，但不包括扩展名）
    PTUCAM_FRAME pFrame;           //  [in]  帧结构体的指针
} TUCAM_FILE_SAVE, *PTUCAM_FILE_SAVE;

// the record save structure
typedef struct _tagTUCAM_REC_SAVE
{
    INT32       nCodec;            //  [in]  编码解码器的类型
    PCHAR      pstrSavePath;       //  [in]  保存文件的路径   （包含文件名，但不包括扩展名）
    Float       fFps;              //  [in]  当前的帧率       （录像的帧率）
} TUCAM_REC_SAVE, *PTUCAM_REC_SAVE;
```

4.6.4 示例代码

```
1.  // 保存图片文件
2.  void SaveImage()
3.  {
4.      m_frame.ucFormatGet = TUFRM_FMT_USUAL;
5.      if(TUCAMRET_SUCCESS==TUCAM_Buf_WaitForFrame(m_opCam.hIdxTUCam, &m_frame))
6.      {
7.          TUCAM_FILE_SAVE fileSave;
8.          fileSave.nSaveFmt      = TUFMT_TIF;      // 保存 Tiff 格式
9.          fileSave.pFrame        = &m_frame;       // 需要保存的帧指针
10.         fileSave.pstrSavePath = "C:\\image";     // 路径包含文件名（不包含扩展名）
11.
12.         if (TUCAMRET_SUCCESS == TUCAM_File_SaveImage(m_opCam.hIdxTUCam, fileSave))
13.         {
14.             // 保存图像文件成功
15.         }
16.     }
17. }
18.
19. // 保存录像文件
20. void StartRecording()
21. {
22.     TUCAM_REC_SAVE  recSave;
23.     recSave.fFps         = 15.0f;              // 需要保存的帧率
```

```
24.     recSave.nCodec       = m_dwFccHandler;
25.     recSave.pstrSavePath = “C:\\TUVideo.avi”  // 全路径
26.
27.     if (TUCAMRET_SUCCESS == TUCAM_Rec_Start(m_opCam.hIdxTUcam, recSave))
28.     {
29.         // 开始录像。。。
30.     }
31. }
32.
33. Void AppendFrame()
34. {
35.     m_frame.ucFormatGet = TUFRM_FMT_RGB888;
36.     if(TUCAMRET_SUCCESS==TUCAM_Buf_WaitForFrame(m_opCam.hIdxTUCam, &m_frame))
37.     {
38.         TUCAM_Rec_AppendFrame(m_opCam.hIdxTUCam, &m_frame);
39.     }
40. }
41.
42. void StopRecording()
43. {
44.     TUCAM_Rec_Stop(m_opCam.hIdxTUCam);
45. }
46.
```

4.7 扩展控制

4.7.1 接口

// Extened control

TUCAMRET TUCAM_Reg_Read (HDTUCAM hTUCam, TUCAM_REG_RW regRW);

TUCAMRET TUCAM_Reg_Write(HDTUCAM hTUCam, TUCAM_REG_RW regRW);

4.7.2 调用顺序

读写寄存器 TUCAM_Reg_Read / TUCAM_Reg_Write 的调用必须在 TUCAM_Dev_Open 打开相机后调用，如果 TUCAM_Dev_Close 关闭相机后将无法读写寄存器，必须重新打开相机。

4.7.3 文件结构体

```
// the register read/write struct
typedef struct _tagTUCAM_REG_RW
{
    INT32       nRegFmt;                //  [in] 读写寄存器的类型      （参考 TUREG_TYPE）
    PCHAR       pBuf;                   //  [in/out] 指向缓冲区的指针
    INT32       nBufSize;               //  [in] 缓冲区的大小
} TUCAM_REG_RW, *PTUCAM_REG_RW;
```

4.7.4 示例代码

```
1.  // 读取寄存器数据
2.  void ReadRegisterData()
3.  {
4.      char cSN[TUSN_SIZE] = {0};
5.      TUCAM_REG_RW regRW;
6.
7.      regRW.nRegType  = TUREG_SN;
8.      regRW.pBuf      = &cSN[0];
9.      regRW.nBufSize  = TUSN_SIZE;
10.
11.     if (TUCAMRET_SUCCESS == TUCAM_Reg_Read(m_opCam.hIdxTUcam, regRW))
12.     {
13.         // 获取 SN 数据
14.     }
15. }
16.
17. // 写入寄存器数据
18. void WriteRegisterData()
19. {
20.     char cSN[TUSN_SIZE] = {‘S’, ‘N’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’};// “SN123456”
21.     TUCAM_REG_RW regRW;
22.
23.     regRW.nRegType  = TUREG_SN;
24.     regRW.pBuf      = &cSN[0];
25.     regRW.nBufSize  = TUSN_SIZE;
26.
27.     if (TUCAMRET_SUCCESS == TUCAM_Reg_Write(m_opCam.hIdxTUcam, regRW))
28.     {
29.         // 将 SN 写入寄存器成功
30.     }
31. }
```

5. 参考

5.1 类型和常量

5.1.1 TUCAMRET 错误代码

```
TUCAMRET_SUCCESS            = 0x00000001,    // 没有错误，一般成功的代码
TUCAMRET_FAILURE            = 0x80000000,    // 错误

//  initialization error
TUCAMRET_NO_MEMORY          = 0x80000101,    // 没有足够的内存
TUCAMRET_NO_RESOURCE        = 0x80000102,    // 没有足够的资源（不包括内存）
TUCAMRET_NO_MODULE          = 0x80000103,    // 没有子模块
TUCAMRET_NO_DRIVER          = 0x80000104,    // 没有驱动
TUCAMRET_NO_CAMERA          = 0x80000105,    // 没有相机
TUCAMRET_NO_GRABBER         = 0x80000106,    // 没有取图
TUCAMRET_NO_PROPERTY        = 0x80000107,    // 没有代替的属性 ID

TUCAMRET_FAILOPEN_CAMERA    = 0x80000110,    // 打开相机失败
TUCAMRET_FAILOPEN_BULKIN    = 0x80000111,    // 打开批传输输入端点失败
TUCAMRET_FAILOPEN_BULKOUT   = 0x80000112,    // 打开批传输输出端点失败
TUCAMRET_FAILOPEN_CONTROL   = 0x80000113,    // 打开控制端点失败
TUCAMRET_FAILCLOSE_CAMERA   = 0x80000114,    // 关闭相机失败
TUCAMRET_FAILOPEN_FILE      = 0x80000115,    // 打开文件失败

//  status error
TUCAMRET_INIT               = 0x80000201,    //  API 需要初始化状态.
TUCAMRET_BUSY               = 0x80000202,    //  API 处于繁忙状态
TUCAMRET_NOT_INIT           = 0x80000203,    //  API 未初始化
TUCAMRET_EXCLUDED           = 0x80000204,    //  一些资源被独占使用
TUCAMRET_NOT_BUSY           = 0x80000205,    //  API 未处于繁忙状态
TUCAMRET_NOT_READY          = 0x80000206,    //  API 未处于就绪状态

//  wait error
TUCAMRET_ABORT              = 0x80000207,    // 终止处理
TUCAMRET_TIMEOUT            = 0x80000208,    // 超时
TUCAMRET_LOSTFRAME          = 0x80000209,    // 帧丢失
TUCAMRET_MISSFRAME          = 0x8000020A,    // 帧丢失但是是底层驱动问题

// calling error
TUCAMRET_INVALID_CAMERA     = 0x80000301,    // 无效相机
TUCAMRET_INVALID_HANDLE     = 0x80000302,    // 无效相机句柄
```

```
TUCAMRET_INVALID_OPTION     = 0x80000303,    // 无效配置的值
TUCAMRET_INVALID_IDPROP     = 0x80000304,    // 无效属性 ID
TUCAMRET_INVALID_IDCAPA     = 0x80000305,    // 无效性能 ID
TUCAMRET_INVALID_IDPARAM    = 0x80000306,    // 无效参数 ID
TUCAMRET_INVALID_PARAM      = 0x80000307,    // 无效参数
TUCAMRET_INVALID_FRAMEIDX   = 0x80000308,    // 无效帧序号
TUCAMRET_INVALID_VALUE      = 0x80000309,    // 无效值
TUCAMRET_INVALID_EQUAL      = 0x8000030A,    // 值相等，参数无效
TUCAMRET_INVALID_CHANNEL    = 0x8000030B,    // 属性 ID 指定通道，但是通道是无效的
TUCAMRET_INVALID_SUBARRAY   = 0x8000030C,    // 子数组的值是无效的

TUCAMRET_INVALID_VIEW       = 0x8000030D,    // 无效的显示窗口句柄
TUCAMRET_INVALID_PATH       = 0x8000030E,    // 无效的文件路径

TUCAMRET_NO_VALUETEXT       = 0x80000310,    // 属性没有值的文本
TUCAMRET_OUT_OF_RANGE       = 0x80000311,    // 值超出范围

TUCAMRET_NOT_SUPPORT        = 0x80000312,    // 不支持的功能或属性
TUCAMRET_NOT_WRITABLE       = 0x80000313,    // 属性不可写
TUCAMRET_NOT_READABLE       = 0x80000314,    // 属性不可读


TUCAMRET_WRONG_HANDSHAKE    = 0x80000410,    // 错误发生在获取错误代码时
TUCAMRET_NEWAPI_REQUIRED    = 0x80000411,    // 旧 API 不支持，只有新的 API 支持
TUCAMRET_ACCESSDENY         = 0x80000412,    // 当相机处于某种状态属性无法访问

TUCAMRET_NO_CORRECTIONDATA  = 0x80000501,    // 没有彩点校正数据.

//    camera or bus trouble
TUCAMRET_FAIL_READ_CAMERA   = 0x83001001,    // 从相机读取失败
TUCAMRET_FAIL_WRITE_CAMERA  = 0x83001002,    // 写入相机失败
TUCAMRET_OPTICS_UNPLUGGED   = 0x83001003,    // 为插入
```

5.1.2 TUCAM_IDINFO 产品信息代码

```
TUIDI_BUS                   = 0x01,    // USB 口类型： USB2.0/USB3.0
TUIDI_VENDOR                = 0x02,    // 厂商 ID
TUIDI_PRODUCT               = 0x03,    // 产品 ID
TUIDI_VERSION_API           = 0x04,    // TUCAM- API 版本号
TUIDI_VERSION_FRMW          = 0x05,    // 固件版本号
TUIDI_VERSION_FPGA          = 0x06,    // FPGA 版本号 （保留）
TUIDI_VERSION_DRIVER        = 0x07,    // 驱动版本号 （保留）
TUIDI_TRANSFER_RATE         = 0x08,    // USB 传输速率
TUIDI_CAMERA_MODEL          = 0x09,    // 相机型号（字符串类型）
```

```
TUIDI_ENDINFO          = 0x0A,        // 产品信息 ID 结束位
```

5.1.3 TUCAM_IDCAPA 性能代码

```
TUIDC_RESOLUTION       = 0x00,        // 分辨率
TUIDC_PIXELCLOCK       = 0x01,        // 像素时钟
TUIDC_BITOFDEPTH       = 0x02,        // 数据位宽
TUIDC_ATEXPOSURE       = 0x03,        // 自动曝光
TUIDC_HORIZONTAL       = 0x04,        // 水平镜像
TUIDC_VERTICAL         = 0x05,        // 垂直镜像
TUIDC_ATWBBALANCE      = 0x06,        // 自动白平衡 （彩色相机）
TUIDC_FAN_GEAR         = 0x07,        // 风扇档位 （制冷相机）
TUIDC_IMGMODESELECT    = 0x16,        // 图像模式选择(0x01 :CMS 0x02: 11BIT)
TUIDC_LEDENBALE        = 0x1E,        // LED 控制开关(0x01 :开 0x00 :关)
TUIDC_ENDCAPABILITY    = 0x1F,        // 性能 ID 结束位
```

5.1.4 TUCAM_IDPROP 属性代码

```
TUIDP_GLOBALGAIN       = 0x00,        // 全局增益
TUIDP_EXPOSURETM       = 0x01,        // 曝光时间
TUIDP_BRIGHTNESS       = 0x02,        // 亮度 (自动曝光状态有效）
TUIDP_BLACKLEVEL       = 0x03,        // 黑电平
TUIDP_TEMPERATURE      = 0x04,        // 温度
TUIDP_SHARPNESS        = 0x05,        // 锐化
TUIDP_NOISELEVEL       = 0x06,        // 降噪等级
TUIDP_HDR_KVALUE       = 0x07,        // HDR 的 K 值（sCMOS 相机支持）

// image process property
TUIDP_GAMMA            = 0x08,        // 伽玛
TUIDP_CONTRAST         = 0x09,        // 对比度
TUIDP_LFTLEVELS        = 0x0A,        // 左色阶
TUIDP_RGTLEVELS        = 0x0B,        // 右色阶
TUIDP_CHNLGAIN         = 0x0C,        // 通道增益 （彩色相机支持）
TUIDP_SATURATION       = 0x0D,        // 饱和度 （彩色相机支持）
TUIDP_ENDPROPERTY      = 0x0E,        // 属性 ID 结束位
```

5.1.5 TUCAM_CAPTURE_MODES 捕获模式代码

```
TUCCM_SEQUENCE         = 0x00,        // 采用序列模式（流模式）
TUCCM_TRIGGER_STANDARD = 0x01,        // 采用标准触发模式
TUCCM_TRIGGER_SYNCHRONOUS = 0x02,     // 采用同步触发模式
TUCCM_TRIGGER_GLOBAL   = 0x03,        // 采用全局触发模式
TUCCM_TRIGGER_SOFTWARE = 0x04,        // 采用软件触发模式
```

5.1.6 TUIMG_FORMATS 图像格式代码

```
TUFMT_RAW       = 0x01,    //  RAW 格式
TUFMT_TIF       = 0x02,    //  TIFF 格式
TUFMT_PNG       = 0x04,    //  PNG 格式
TUFMT_JPG       = 0x08,    //  JPEG 格式
TUFMT_BMP       = 0x10,    //  BMP 格式
```

5.1.7 TUREG_TYPE 寄存器类型代码

```
TUREG_SN        = 0x01,    // 读写 SN 码寄存器
TUREG_DATA      = 0x02,    // 读写 DATA 寄存器（预留）
```

5.1.8 TUCAM_TRIGGER_EXP 触发曝光模式代码

```
TUCTE_EXPTM     = 0x00,    // 触发使用曝光时间模式
TUCTE_WIDTH     = 0x01,    // 触发使用电平宽度模式
```

5.1.9 TUCAM_TRIGGER_EDGE 触发激发边沿代码

```
TUCTD_RISING    = 0x01,    // 激发上升沿
TUCTD_FAILING   = 0x00,    // 激发下降沿
```

5.1.10 TUFRM_FORMATS 帧格式代码

```
TUFRM_FMT_RAW       = 0x10,    // RAW 格式数据
TUFRM_FMT_USUAl     = 0x11,    // 通常格式数据（8bit/16bit、黑白/彩色）
TUFRM_FMT_RGB888    = 0x12,    // RGB888 格式数据（可用与显示）
```

5.2 结构体

5.2.1 初始化

```
// the camera initialize structure
typedef struct _tagTUCAM_INIT
{
    UINT32    uiCamCount;                    //  [out] 返回当前连接的相机个数
    PCHAR     pstrConfigPath;                //  [in] 输入保存相机参数的路径
}TUCAM_INIT, *PTUCAM_INIT;
```

5.2.2 打开相机

```
//   the camera open structure
typedef struct _tagTUCAM_OPEN
{
    UINT32    uiIdxOpen;                     //  [in]  输入需要打开相机的序列号
    HDTUCAM hIdxTUCam;                       //  [out] 输出打开相机的句柄
}TUCAM_OPEN, *PTUCAM_OPEN;
```

5.2.3 相机信息

```
// the camera value text structure
typedef struct _tagTUCAM_VALUE_INFO
{
    INT32     nID;                           //  [in] 信息 ID TUCAM_IDINFO
    INT32     nValue;                        //  [in] 信息的值
    PCHAR     pText;                         //  [in/out] 指向文本数据的指针
    INT32     nTextSize;                     //  [in] 文本缓存大小
}TUCAM_VALUE_INFO, *PTUCAM_VALUE_INFO;
```

5.2.4 性能/属性值文本

```
// the camera value text structure
typedef struct _tagTUCAM_VALUE_TEXT
{
    INT32       nID;                         //  [in]  ID TUCAM_IDPROP / TUCAM_IDCAPA
    DOUBLE    dbValue;                       //  [in] 性能/属性的值
    PCHAR       pText;                       //  [in/out] 指向文本数据的指针
    INT32       nTextSize;                   //  [in] 文本缓存大小
}TUCAM_VALUE_TEXT, *PTUCAM_VALUE_TEXT;
```

5.2.5 性能的属性

```
// the camera capability attribute
typedef struct _tagTUCAM_CAPA_ATTR
{
    INT32    idCapa;                    //  [in]   ID TUCAM_IDCAPA

    INT32    nValMin;                   //  [out] 最小值
    INT32    nValMax;                   //  [out] 最大值
    INT32    nValDft;                   //  [out] 默认值
    INT32    nValStep;                  //  [out] 步长
}TUCAM_CAPA_ATTR, *PTUCAM_CAPA_ATTR;
```

5.2.6 属性的属性

```
// the camera property attribute
typedef struct _tagTUCAM_PROP_ATTR
{
    INT32    idProp;                    //  [in]   ID TUCAM_IDPROP
    INT32    nIdxChn;                   //  [in/out] 当前通道索引号
    DOUBLE   dbValMin;                  //  [out] 最小值
    DOUBLE   dbValMax;                  //  [out] 最大值
    DOUBLE   dbValDft;                  //  [out] 默认值
    DOUBLE   dbValStep;                 //  [out] 步长
}TUCAM_PROP_ATTR, *PTUCAM_PROP_ATTR;
```

5.2.7 ROI 的属性

```
// the camera ROI attribute
typedef struct _tagTUCAM_ROI_ATTR
{
    BOOL     bEnable;                   //  [in/out]   ROI 使能

    INT32    nHOffset;                  //  [in/out]   水平偏移量
    INT32    nVOffset;                  //  [in/out]   垂直偏移量
    INT32    nWidth;                    //  [in/out]   ROI 的宽度
    INT32    nHeight;                   //  [in/out]   ROI 的高度
}TUCAM_ROI_ATTR, *PTUCAM_ROI_ATTR;
```

5.2.8 触发的属性

```
// the camera trigger attribute
typedef struct _tagTUCAM_TRIGGER_ATTR
```

```
{
    INT32    nTgrMode;                    //  [in/out]  触发模式
    INT32    nExpMode;                    //  [in/out]  曝光模式值[0,1] 0:曝光时间  1:电平宽度
    INT32    nEdgeMode;                   //  [in/out]  边沿激发模式[0, 1] 0:下降沿 1:上升沿
    INT32    nDelayTm;                    //  [in/out]  触发延迟时间 ms
    INT32    nFrames;                     //  [in/out]  一次触发输出帧数
}TUCAM_TRIGGER_ATTR, *PTUCAM_TRIGGER_ATTR;
```

5.2.9 触发输出的属性

```
// the camera trigger attribute
typedef struct _tagTUCAM_TRGOUT_ATTR
{
    INT32    nTgrOutPort;                 //  [in/out]  触发输出口0:Port1    1:Port2    2:Port3
    INT32    nTgrOutMode;                 //  [in/out]  触发输出模式0:低电平  1:高电平  2:触
发输入  3:曝光开始参考点  4:全局曝光  5:读出结束
    INT32    nEdgeMode;                   //  [in/out]  边沿激发模式[0, 1] 1:下降沿 2:上升沿
    INT32    nDelayTm;                    //  [in/out]  触发输出延迟时间
    INT32    nWidth;                      //  [in/out]  触发输出脉宽
}TUCAM_TRGOUT_ATTR, *PTUCAM_TRGOUT_ATTR;
```

5.2.10 帧结构

```
// the camera frame structure
typedef struct _tagTUCAM_FRAME
{
    CHAR szSignature[8];                  //  [out]  版权信息

    //   The based information
    USHORT usHeader;                      //  [out]  帧头部大小
    USHORT usOffset;                      //  [out]  帧数据偏移大小
    USHORT usWidth;                       //  [out]  帧图像的宽度
    USHORT usHeight;                      //  [out]  帧图像的高度
    UINT32 uiWidthStep;                   //  [out]  帧图像的宽度步长

    UCHAR    ucDepth;                     //  [out]  帧图像的数据位深
    UCHAR    ucFormat;                    //  [out]  帧图像的数据格式
    UCHAR    ucChannels;                  //  [out]  帧图像的通道数
    UCHAR    ucElemBytes;                 //  [out]  帧图像的数据字节数
    UCHAR    ucFormatGet;                 //  [in/out]  需要获取的图像格式 TUFRM_FORMATS

    UINT32 uiIndex;                       //  [out]  帧图像的序列号（保留）
    UINT32 uiImgSize;                     //  [out]  帧图像数据的大小
    UINT32 uiRsdSize;                     //  [in]   需要获取的帧数
```

```
    UINT32 uiHstSize;                    //  [out] 帧图像的保留字段

    PUCHAR pBuffer;                      //  [in/out] 指向帧数据的缓冲区
} TUCAM_FRAME, *PTUCAM_FRAME;
```

5.2.11 文件保存

```
// the file save structure
typedef struct _tagTUCAM_FILE_SAVE
{
    INT32    nSaveFmt;                   //  [in] 保存图片的格式 参考 TUIMG_FORMATS
    PCHAR    pstrSavePath;               //  [in] 保存的路径（不包含扩展名）

    PTUCAM_FRAME pFrame;                 //  [in] 指向帧的结构体
} TUCAM_FILE_SAVE, *PTUCAM_FILE_SAVE;
```

5.2.12 录像保存

```
// the record save structure
typedef struct _tagTUCAM_REC_SAVE
{
    INT32     nCodec;                    //  [in] 编码解码类型
    PCHAR     pstrSavePath;              //  [in] 保存的路径包含文件名
    float     fFps;                      //  [in]  需要保存的帧率
} TUCAM_REC_SAVE, *PTUCAM_REC_SAVE;
```

5.2.13 寄存器读写

```
// the register read/write structure
typedef struct _tagTUCAM_REG_RW
{
    INT32    nRegType;                   //  [in] 读写寄存器类型    参考 TUREG_TYPE

    PCHAR    pBuf;                       //  [in/out] 指向读写内容的缓冲区
    INT32nBufSize;                       //  [in]  缓冲区的大小
} TUCAM_REG_RW, *PTUCAM_REG_RW;
```

5.3 函数

TUCAM_Api_Init
TUCAM_Api_Uninit

描述

卸载 TUCAM-API 库，包括释放驱动绑定和一些内部资源。整个程序结束时调用一次。

声明

TUCAMRET TUCAM_Api_Uninit ();

参数

无输入参数

错误代码

TUCAMRET_NOT_INIT	TUCAM-API 未初始化

相关接口

TUCAM_Api_Uninit

TUCAM_Dev_Open

描述

打开相机，调用后相机处于工作模式，可以响应其他接口的调用。在此之前要保证有相机存在，即要在调用 TUCAM_Api_Init 初始化之后。

声明

TUCAMRET TUCAM_Dev_Open (PTUCAM_OPEN pOpenParam);

参数

PTUCAM_OPEN pOpenParam 打开相机结构体指针,参考结构体 TUCAM_OPEN

错误代码

TUCAMRET_NOT_INIT	TUCAM-API 未初始化
TUCAMRET_INVALID_PARAM	无效参数，当 pOpenParam 指针为空时
TUCAMRET_OUT_OF_RANGE	超出范围，当需要打开的相机索引号超出连接相机的范围时
TUCAMRET_FAILOPEN_CAMERA	打开相机失败
TUCAMRET_INVALID_CAMERA	无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Close

TUCAM_Dev_Close

描述

关闭相机，调用后相机处于待机状态，不响应其他接口的调用。

声明

```
TUCAMRET   TUCAM_Dev_Close ();
```

参数

无输入参数

错误代码

TUCAMRET_NOT_INIT	TUCAM-API 未初始化

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open

TUCAM_Dev_GetInfo

描述

获取相机的相关信息，如 USB 口类型，相机产品号，API 版本号、固件版本号、相机类型等。在此之前要保证有相机存在，并且保证相机打开，即要在调用 TUCAM_Api_Init 初始化和调用 TUCAM_Api_Open 之后。

声明

TUCAMRET TUCAM_Dev_GetInfo (HDTUCAM hTUCam, PTUCAM_VALUE_INFO pInfo);

参数

HDTUCAM hTUCam 相机的句柄

PTUCAM_VALUE_INFO pInfo 相机信息值的结构体指针，参考 TUCAM_VALUE_INFO

错误代码

TUCAMRET_NOT_INIT TUCAM-API 未初始化

TUCAMRET_INVALID_PARAM 无效参数，当产品信息代码不存在时，参考 TUCAM_IDINFO

TUCAMRET_INVALID_CAMERA 无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Capa_GetAttr

描述

获取性能参数的属性值。获取的属性包含该参数的最小值、最大值、默认值和步长。具体支

持的性能类型参考 TUCAM_IDCAPA。

声明

TUCAMRET TUCAM_Capa_GetAttr (HDTUCAM hTUCam, PTUCAM_CAPA_ATTR pAttr);

参数

HDTUCAM hTUCam 相机的句柄

PTUCAM_CAPA_ATTR pAttr 相机性能属性结构体指针，参考 TUCAM_CAPA_ATTR

错误代码

TUCAMRET_NOT_INIT TUCAM-API 未初始化

TUCAMRET_INVALID_IDCAPA 无效的性能代码，当性能代码不存在时，参考 TUCAM_IDCAPA

TUCAMRET_INVALID_VALUE 无效的值，当 pAttr 指针为空时

TUCAMRET_NOT_SUPPORT 当底层请求不支持时

TUCAMRET_INVALID_CAMERA 无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Capa_GetValue、TUCAM_Capa_SetValue、TUCAM_Capa_GetValueText

TUCAM_Capa_GetValue

描述

获取性能参数的当前属性值。具体支持的性能类型参考 TUCAM_IDCAPA。

声明

TUCAMRET TUCAM_Capa_GetValue (HDTUCAM hTUCam, INT32 nCapa, INT32

*pnVal);

参数

HDTUCAM hTUCam 相机的句柄

INT32 nCapa 相机性能属性 ID，参考 TUCAM_IDCAPA

INT32 *pnVal 返回当前值

错误代码

TUCAMRET_NOT_INIT TUCAM-API 未初始化

TUCAMRET_INVALID_IDCAPA 无效的性能代码，当性能代码不存在时，参考 TUCAM_IDCAPA

TUCAMRET_INVALID_VALUE 无效的值，当 pAttr 指针为空时

TUCAMRET_NOT_SUPPORT 当底层请求不支持时

TUCAMRET_INVALID_CAMERA 无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Capa_GetAttr、TUCAM_Capa_SetValue、TUCAM_Capa_GetValueText

TUCAM_Capa_SetValue

描述

设置性能参数的当前属性值。具体支持的性能类型参考 TUCAM_IDCAPA。

声明

TUCAMRET TUCAM_Capa_SetValue (HDTUCAM hTUCam, INT32 nCapa, INT32 nVal);

参数

HDTUCAM hTUCam 相机的句柄

INT32 nCapa 相机性能属性 ID，参考 TUCAM_IDCAPA

INT32 nVal 需要设置的当前值

错误代码

TUCAMRET_NOT_INIT TUCAM-API 未初始化

TUCAMRET_INVALID_IDCAPA 无效的性能代码，当性能代码不存在时，参考 TUCAM_IDCAPA

TUCAMRET_INVALID_VALUE 无效的值，当 pAttr 指针为空时

TUCAMRET_NOT_SUPPORT 当底层请求不支持时

TUCAMRET_INVALID_CAMERA 无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Capa_GetAttr、TUCAM_Capa_GetValue、TUCAM_Capa_GetValueText

TUCAM_Capa_GetValueText

描述

获取性能参数的当前属性值的文本信息。具体支持的性能类型参考 TUCAM_IDCAPA。

声明

TUCAMRET TUCAM_Capa_GetValueText (HDTUCAM hTUCam,

PTUCAM_VALUE_TEXT pVal);

参数

HDTUCAM hTUCam 相机的句柄

PTUCAM_VALUE_TEXT pVal 获取性能参数的文本信息结构体指针，TUCAM_VALUE_TEXT

错误代码

TUCAMRET_NOT_INIT TUCAM-API 未初始化

TUCAMRET_FAILURE 文本缓冲区大小为 0 或 pText 指针为空时

TUCAMRET_INVALID_CAMERA 无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Capa_GetAttr、TUCAM_Capa_SetValue、TUCAM_Capa_GetValue

TUCAM_Prop_GetAttr

描述

获取属性参数的属性值。获取的属性包含该参数的最小值、最大值、默认值和步长。具体支持的性能类型参考 TUCAM_IDPROP。

声明

TUCAMRET TUCAM_Prop_GetAttr (HDTUCAM hTUCam, PTUCAM_PROP_ATTR pAttr);

参数

HDTUCAM hTUCam 相机的句柄

PTUCAM_PROP_ATTR pAttr 相机性能属性结构体指针，参考 TUCAM_PROP_ATTR

错误代码

TUCAMRET_NOT_INIT TUCAM-API 未初始化

TUCAMRET_INVALID_IDPROP 无效的性能代码，当性能代码不存在时，参考 TUCAM_IDPROP

TUCAMRET_INVALID_VALUE 无效的值，当 pAttr 指针为空时

TUCAMRET_NOT_SUPPORT 当底层请求不支持时

TUCAMRET_OUT_OF_RANGE 需要获取的通道超出范围时

TUCAMRET_INVALID_CAMERA 无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Prop_GetValue、TUCAM_Prop_SetValue、TUCAM_Prop_GetValueText

TUCAM_Prop_GetValue

描述

获取属性参数的当前属性值。具体支持的性能类型参考 TUCAM_IDPROP。

声明

```
TUCAMRET  TUCAM_Prop_GetValue (HDTUCAM hTUCam, INT32 nProp, DOUBLE *pdbVal, INT32 nChn = 0);
```

参数

HDTUCAM hTUCam 相机的句柄

INT32 nProp 相机属性的 ID，参考 TUCAM_IDPROP

DOUBLE *pdbVal 返回当前值

INT32 nChn 需要获取的当前通道（默认为 0，黑白相机为 0）

错误代码

TUCAMRET_NOT_INIT TUCAM-API 未初始化

TUCAMRET_INVALID_IDPROP 无效的性能代码，当性能代码不存在时，参考 TUCAM_IDPROP

TUCAMRET_INVALID_VALUE 无效的值，当 pAttr 指针为空时

TUCAMRET_NOT_SUPPORT 当底层请求不支持时

TUCAMRET_OUT_OF_RANGE 需要获取的通道超出范围时

TUCAMRET_INVALID_CAMERA 无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Prop_GetAttr、TUCAM_Prop_SetValue、TUCAM_Prop_GetValueText

TUCAM_Prop_SetValue

描述

设置属性参数的当前属性值。具体支持的性能类型参考 TUCAM_IDPROP。

声明

```
TUCAMRET  TUCAM_Prop_SetValue (HDTUCAM hTUCam, INT32 nProp, DOUBLE dbVal, INT32 nChn = 0);
```

参数

HDTUCAM hTUCam 相机的句柄

INT32 nProp 相机属性的 ID，参考 TUCAM_IDPROP

DOUBLE dbVal 需要设置的当前值

INT32 nChn 需要获取的当前通道（默认为 0，黑白相机为 0）

错误代码

TUCAMRET_NOT_INIT TUCAM-API 未初始化

TUCAMRET_INVALID_IDPROP 无效的性能代码，当性能代码不存在时，参考 TUCAM_IDPROP

TUCAMRET_INVALID_VALUE 无效的值，当 pAttr 指针为空时

TUCAMRET_NOT_SUPPORT 当底层请求不支持时

TUCAMRET_OUT_OF_RANGE 需要获取的通道超出范围时

TUCAMRET_INVALID_CAMERA 无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Prop_GetAttr、TUCAM_Prop_GetValue、TUCAM_Prop_GetValueText

TUCAM_Prop_GetValueText

描述

获取属性参数的当前属性值的文本信息。具体支持的性能类型参考 TUCAM_IDPROP。

声明

```
TUCAMRET  TUCAM_Prop_GetValueText (HDTUCAM hTUCam,
PTUCAM_VALUE_TEXT pVal);
```

参数

HDTUCAM hTUCam 相机的句柄

PTUCAM_VALUE_TEXT pVal 获取性能参数的文本信息结构体指针，TUCAM_VALUE_TEXT

错误代码

TUCAMRET_NOT_INIT	TUCAM-API 未初始化
TUCAMRET_FAILURE	文本缓冲区大小为 0 或 pText 指针为空时
TUCAMRET_INVALID_CAMERA	无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Prop_GetAttr、TUCAM_Prop_SetValue、TUCAM_Prop_GetValue

TUCAM_Buf_Alloc

描述

分配内存空间用于数据捕获。当应用程序调用这个接口时，SDK 分配必要的内部缓冲区来缓冲图像采集。捕获不从这一时刻开始。开始采集，应用程序必须调用 TUCAM_Cap_Start 接口。如果缓冲区不再是必要的，应用程序应该调用 TUCAM_Buf_Release 接口来释放内部缓冲区。

声明

TUCAMRET TUCAM_Buf_Alloc (HDTUCAM hTUCam, PTUCAM_FRAME pFrame);

参数

HDTUCAM hTUCam	相机的句柄
PTUCAM_FRAME pFrame	图像时间帧结构的指针，参考 TUCAM_FRAME

错误代码

TUCAMRET_NOT_INIT	TUCAM-API 未初始化
TUCAMRET_INVALID_PARAM	当 pFrame 指针为空时
TUCAMRET_EXCLUDED	当 TUCAM_Buf_Alloc 被调用，且未释放时
TUCAMRET_OUT_OF_RANGE	当需要获取的帧数超出范围时

TUCAMRET_NO_MEMORY 当内存不足时

TUCAMRET_INVALID_CAMERA 无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Buf_Release

TUCAM_Buf_AbortWait、TUCAM_Buf_WaitForFrame、TUCAM_Buf_CopyFrame

TUCAM_Cap_Start、TUCAM_Cap_Stop

TUCAM_Buf_Release

描述

释放用于数据捕获的内存空间。如果在捕获过程中调用该接口，这个接口将返回相机处于繁忙的状态。

声明

TUCAMRET TUCAM_Buf_Release (HDTUCAM hTUCam);

参数

HDTUCAM hTUCam 相机的句柄

错误代码

TUCAMRET_NOT_INIT TUCAM-API 未初始化

TUCAMRET_BUSY 相机处于繁忙状态

TUCAMRET_INVALID_CAMERA 无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Buf_Alloc

TUCAM_Buf_AbortWait、TUCAM_Buf_WaitForFrame、TUCAM_Buf_CopyFrame

TUCAM_Cap_Start、TUCAM_Cap_Stop

TUCAM_Buf_AbortWait

描述

用于停止数据捕获时的等待。调用 TUCAM_Buf_WaitForFrame 进行数据捕获等待之后，使用该接口终止等待。

声明

TUCAMRET TUCAM_Buf_AbortWait (HDTUCAM hTUCam);

参数

HDTUCAM hTUCam	相机的句柄

错误代码

TUCAMRET_NOT_INIT	TUCAM-API 未初始化
TUCAMRET_INVALID_CAMERA	无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Buf_Alloc、TUCAM_Buf_Release

TUCAM_Buf_WaitForFrame、TUCAM_Buf_CopyFrame

TUCAM_Cap_Start、TUCAM_Cap_Stop

TUCAM_Buf_WaitForFrame

描述

用于等待数据捕获完成。通过调用 TUCAM_Buf_Alloc 分配来的空间，来获取捕获到的帧数据。必须在调用 TUCAM_Cap_Start 开始捕获之后使用，否则会返回未准备的状态。

该函数属于阻塞函数，直至数据捕获完成或调用 TUCAM_Buf_AbortWait 终止。

帧结构体中 uiRsdSize 设置需要捕获的帧数，针对触发模式有效。例如：当触发一次需要返回 5 帧时，该函数等待 5 帧数据都捕获结束之后才返回。

注：返回的帧结构 pBuffer 的数据排列，是帧头部(usHeader)+图像数据(uiImgSize) +保留位(uiHstSize)。如果是多帧返回则按此顺序往后排列。

声明

TUCAMRET TUCAM_Buf_WaitForFrame (HDTUCAM hTUCam, PTUCAM_FRAME pFrame);

参数

HDTUCAM hTUCam	相机的句柄
PTUCAM_FRAME pFrame	帧结构体的指针

错误代码

TUCAMRET_NOT_INIT	TUCAM-API 未初始化
TUCAMRET_NOT_READY	当未调用 TUCAM_Cap_Start 开始捕获时
TUCAMRET_NO_MEMORY	当未调用 TUCAM_Buf_Alloc 创建内存空间时
TUCAMRET_NO_RESOURCE	当 pFrame 指针为空时
TUCAMRET_OUT_OF_RANGE	当需要获取的帧数大于 1 时且获取的格式和 TUCAM_Buf_Alloc 不同
TUCAMRET_INVALID_CAMERA	无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Buf_Alloc、TUCAM_Buf_Release

TUCAM_Buf_AbortWait、TUCAM_Buf_CopyFrame

TUCAM_Cap_Start、TUCAM_Cap_Stop

TUCAM_Buf_CopyFrame

描述

用于等待数据捕获完成之后拷贝不同于 TUCAM_Buf_Alloc 分配的图像格式数据。必须在 TUCAM_Buf_WaitForFrame 返回之后调用，否则无法获取正确的图像数据。

例如：分配的图像格式为 TUFRM_FMT_RGB888，通过该函数可以拷贝其他格式的数据（如 TUFRM_FMT_RAW），此接口无法拷贝大于 1 帧的数据即帧结构中的 uiRsdSize 不能大于 1。

注：返回的帧结构 pBuffer 的数据排列，是帧头部(usHeader)+图像数据(uiImgSize) +保留位(uiHstSize)。不支持多帧数据返回。

声明

TUCAMRET TUCAM_Buf_CopyFrame (HDTUCAM hTUCam, PTUCAM_FRAME pFrame);

参数

HDTUCAM hTUCam	相机的句柄
PTUCAM_FRAME pFrame	帧结构体的指针

错误代码

TUCAMRET_NOT_INIT	TUCAM-API 未初始化
TUCAMRET_NOT_READY	当未调用 TUCAM_Cap_Start 开始捕获时
TUCAMRET_NO_MEMORY	当未调用 TUCAM_Buf_Alloc 创建内存空间时
TUCAMRET_NO_RESOURCE	当 pFrame 指针为空时
TUCAMRET_OUT_OF_RANGE	当需要获取的帧数大于 1 时且获取的格式和

TUCAM_Buf_Alloc 不同

TUCAMRET_INVALID_CAMERA 无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Buf_Alloc、TUCAM_Buf_Release

TUCAM_Buf_AbortWait、TUCAM_Buf_WaitForFrame

TUCAM_Cap_Start、TUCAM_Cap_Stop

TUCAM_Cap_SetROI

描述

用于设定图像的感兴趣区域，以左上角为坐标原点。设置的水平偏移量、垂直偏移量、宽度和高度必须为 4 的倍数。

声明

TUCAMRET TUCAM_CAP_SETROI (HDTUCAM HTUCAM, TUCAM_ROI_ATTR ROIATTR);

参数

HDTUCAM HTUCAM 相机的句柄

TUCAM_ROI_ATTR ROIATTR ROI 属性结构体的对象

错误代码

TUCAMRET_NOT_INIT TUCAM-API 未初始化

TUCAMRET_NOT_SUPPORT 不支持 ROI 设置

TUCAMRET_INVALID_CAMERA 无效的相机，相机句柄不存在时

相关接口

TUCAM_API_INIT、TUCAM_API_UNINIT

TUCAM_DEV_OPEN、TUCAM_DEV_CLOSE

TUCAM_BUF_ALLOC、TUCAM_BUF_RELEASE

TUCAM_BUF_ABORTWAIT、TUCAM_BUF_WAITFORFRAME

TUCAM_CAP_START、TUCAM_CAP_STOP

TUCAM_CAP_GETROI

TUCAM_Cap_GetROI

描述

用于设定图像的感兴趣区域，以左上角为坐标原点。设置的水平偏移量、垂直偏移量、宽度和高度必须为 4 的倍数。

捕获不从这一时刻开始。开始采集调用 TUCAM_Cap_Start 接口之后。

声明

TUCAMRET TUCAM_Cap_GetROI (HDTUCAM hTUCam, PTUCAM_ROI_ATTR pRoiAttr);

参数

HDTUCAM hTUCam	相机的句柄
PTUCAM_ROI_ATTR pRoiAttr	ROI 属性结构体的指针

错误代码

TUCAMRET_NOT_INIT	TUCAM-API 未初始化
TUCAMRET_NOT_SUPPORT	不支持 ROI 设置

TUCAMRET_INVALID_CAMERA 无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Buf_Alloc、TUCAM_Buf_Release

TUCAM_Buf_AbortWait、TUCAM_Buf_WaitForFrame

TUCAM_Cap_Start、TUCAM_Cap_Stop

TUCAM_Cap_SetROI

TUCAM_Cap_SetTrigger

描述

用于设定触发的属性，捕获不从这一时刻开始。开始采集调用 TUCAM_Cap_Start 接口之后。

曝光模式：

TUCTE_EXPTM 表示曝光时间由软件设定

TUCTE_WIDTH 示曝光时间由输入电平宽度设定

激发边沿模式：

TUCTD_RISING 表示触发信号为上升沿有效

TUCTD_FAILING 表示触发信号为下降沿有效

帧数：表示接收一个触发信号后，拍摄多少张图像，每张图像的曝光时间是相同的，取决于软件设定。（选择电平宽度时，该功能无效。）

延迟：表示接收到一个触发信号后，可以设置多长的延迟时间才使相机进行触发曝光。

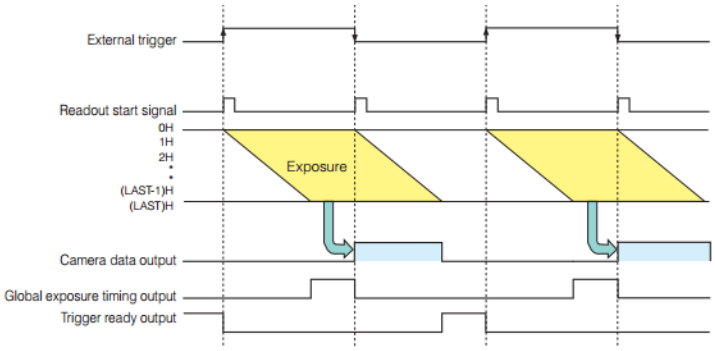


Fig. 19: Normal level trigger mode (rising edge)

触发模式支持的参数：

模式	TUCAM_TRIGGER_EXP	TUCAM_TRIGGER_EDGE	延时	帧数
标准触发	支持	支持	支持	支持
同步触发	支持	支持	不支持	不支持
全局触发	不支持	支持	不支持	不支持
软件触发	不支持	不支持	不支持	不支持

同步触发：即同步取图，第一次触发启动，第二次触发输出同步图像。

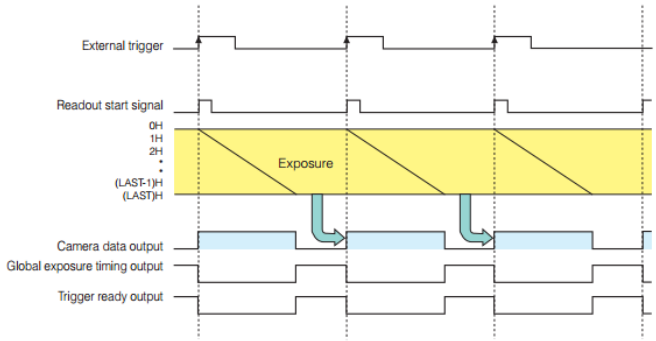


Fig. 21: Normal synchronous readout trigger mode (rising edge)

全局触发：一般用于光源可控的场景。

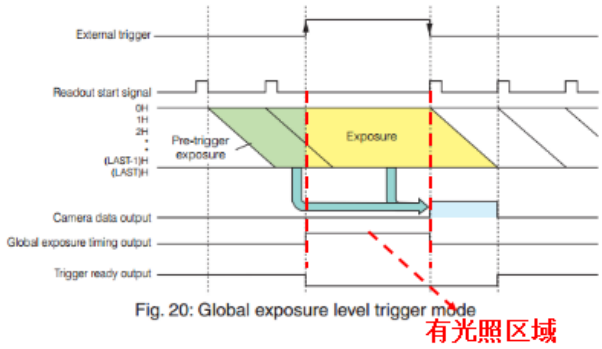


Fig. 20: Global exposure level trigger mode

有光照区域

软件触发：通过软件下发命令模拟触发信号。

声明

TUCAMRET TUCAM_Cap_SetTrigger (HDTUCAM hTUCam,

TUCAM_TRIGGER_ATTR tgrAttr);

参数

HDTUCAM hTUCam 相机的句柄

TUCAM_TRIGGER_ATTR tgrAttr 触发属性结构体的对象

错误代码

TUCAMRET_NOT_INIT TUCAM-API 未初始化

TUCAMRET_INVALID_CAMERA 无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Buf_Alloc、TUCAM_Buf_Release

TUCAM_Buf_AbortWait、TUCAM_Buf_WaitForFrame

TUCAM_Cap_Start、TUCAM_Cap_Stop

TUCAM_Cap_GetTrigger、TUCAM_Cap_DoSoftwareTrigger

TUCAM_Cap_GetTrigger

描述

用于获取触发的属性。

声明

TUCAMRET TUCAM_Cap_GetTrigger (HDTUCAM hTUCam,

PTUCAM_TRIGGER_ATTR pTgrAttr);

参数

HDTUCAM hTUCam 相机的句柄

PTUCAM_TRIGGER_ATTR pTgrAttr 触发属性结构体的指针

错误代码

TUCAMRET_NOT_INIT TUCAM-API 未初始化

TUCAMRET_INVALID_CAMERA 无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Buf_Alloc、TUCAM_Buf_Release

TUCAM_Buf_AbortWait、TUCAM_Buf_WaitForFrame

TUCAM_Cap_Start、TUCAM_Cap_Stop

TUCAM_Cap_SetTrigger

TUCAM_Cap_DoSoftwareTrigger

描述

执行软件触发命令。

声明

TUCAMRET TUCAM_Cap_DoSoftwareTrigger(HDTUCAM hTUCam);

参数

HDTUCAM hTUCam 相机的句柄

错误代码

TUCAMRET_NOT_INIT TUCAM-API 未初始化

TUCAMRET_FAILURE 执行触发命令失败

TUCAMRET_INVALID_CAMERA 无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Buf_Alloc、TUCAM_Buf_Release

TUCAM_Buf_AbortWait、TUCAM_Buf_WaitForFrame

TUCAM_Cap_Start、TUCAM_Cap_Stop

TUCAM_Cap_SetTrigger

TUCAM_Cap_SetTriggerOut

描述

用于设定触发输出的属性。

声明

TUCAMRET TUCAM_Cap_SetTriggerOut (HDTUCAM hTUCam,

TUCAM_TRGOUT_ATTR tgroutAttr);

参数

HDTUCAM hTUCam 相机的句柄

TUCAM_TRGOUT_ATTR tgroutAttr 触发输出属性结构体的对象

错误代码

TUCAMRET_NOT_INIT	TUCAM-API 未初始化
TUCAMRET_NOT_SUPPORT	不支持设置
TUCAMRET_INVALID_CAMERA	无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Buf_Alloc、TUCAM_Buf_Release

TUCAM_Cap_GetTriggerOut

描述

用于获取触发输出的属性。

声明

```
TUCAMRET  TUCAM_Cap_GetTriggerOut  (HDTUCAM hTUCam,
PTUCAM_TRGOUT_ATTR pTgrOutAttr);
```

参数

HDTUCAM hTUCam	相机的句柄
PTUCAM_TRGOUT_ATTR pTgrOutAttr	触发输出属性结构体指针

错误代码

TUCAMRET_NOT_INIT	TUCAM-API 未初始化
TUCAMRET_NOT_SUPPORT	不支持设置
TUCAMRET_INVALID_CAMERA	无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Buf_Alloc、TUCAM_Buf_Release

TUCAM_Cap_Start

描述

开始进行数据捕获。在开始捕获之前，配置好感兴趣区域和触发模式。

声明

TUCAMRET TUCAM_Cap_Start(HDTUCAM hTUCam, UINT32 uiMode);

参数

HDTUCAM hTUCam	相机的句柄
UINT32 uiMode	相机捕获的模式

错误代码

TUCAMRET_NOT_INIT	TUCAM-API 未初始化
TUCAMRET_FAILOPEN_BULKIN	打开相机捕获失败
TUCAMRET_INVALID_CAMERA	无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Buf_Alloc、TUCAM_Buf_Release

TUCAM_Buf_AbortWait、TUCAM_Buf_WaitForFrame

TUCAM_Cap_Start

TUCAM_Cap_SetTrigger、TUCAM_SetROI

TUCAM_Cap_Stop

描述

停止进行数据捕获。

声明

TUCAMRET TUCAM_Cap_Stop (HDTUCAM hTUCam);

参数

HDTUCAM hTUCam 相机的句柄

错误代码

TUCAMRET_NOT_INIT TUCAM-API 未初始化

TUCAMRET_FAILOPEN_BULKIN 打开相机捕获失败

TUCAMRET_INVALID_CAMERA 无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Buf_Alloc、TUCAM_Buf_Release

TUCAM_Buf_AbortWait、TUCAM_Buf_WaitForFrame

TUCAM_Cap_Stop

TUCAM_File_SaveImage

描述

对帧数据进行保存。

声明

TUCAMRET TUCAM_File_SaveImage (HDTUCAM hTUCam, TUCAM_FILE_SAVE

fileSave);

参数

HDTUCAM hTUCam 相机的句柄

TUCAM_FILE_SAVE fileSave 文件保存结构体

错误代码

TUCAMRET_NOT_INIT TUCAM-API 未初始化

TUCAMRET_INVALID_PARAM 输入的参数无效

TUCAMRET_INVALID_PATH 输入的路径不存在

TUCAMRET_FAILURE 文件保存失败

TUCAMRET_INVALID_CAMERA 无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Buf_Alloc、TUCAM_Buf_Release

TUCAM_Buf_WaitForFrame、TUCAM_Buf_CopyFrame

TUCAM_Rec_Start

描述

打开录像文件，对帧数据进行录像保存，此时并未写入数据。设置的帧率需要大于 1fps，不足 1fps 的将按 1fps 来创建录像文件。

声明

TUCAMRET TUCAM_Rec_Start(HDTUCAM hTUCam, TUCAM_REC_SAVE recSave);

参数

HDTUCAM hTUCam 相机的句柄

TUCAM_REC_SAVE recSave 录像文件保存结构体

错误代码

TUCAMRET_NOT_INIT TUCAM-API 未初始化

TUCAMRET_INVALID_PARAM 输入的参数无效

TUCAMRET_INVALID_PATH 输入的路径不存在

TUCAMRET_FAILOPEN_FILE 文件打开失败

TUCAMRET_INVALID_CAMERA 无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Buf_Alloc、TUCAM_Buf_Release

TUCAM_Buf_WaitForFrame

TUCAM_Cap_Start、TUCAM_Cap_Stop

TUCAM_Rec_Stop、TUCAM_Rec_AppendFrame

TUCAM_Rec_AppendFrame

描述

将图像数据写入文件，在 TUCAM_Buf_WaitForFrame 调用该接口。

声明

TUCAMRET TUCAM_Rec_AppendFrame(HDTUCAM hTUCam, PTUCAM_FRAME pFrame);

参数

HDTUCAM hTUCam 相机的句柄

PTUCAM_FRAME pFrame 帧结构体的指针

错误代码

TUCAMRET_NOT_INIT TUCAM-API 未初始化

TUCAMRET_NOT_READY 未调用 TUCAM_Rec_Start 接口

TUCAMRET_OUT_OF_RANGE 写入的图像宽度和高度与创建时不一致

TUCAMRET_INVALID_CAMERA 无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Buf_Alloc、TUCAM_Buf_Release

TUCAM_Buf_WaitForFrame

TUCAM_Cap_Start、TUCAM_Cap_Stop

TUCAM_Rec_Start、TUCAM_Rec_Stop

TUCAM_Rec_Stop

描述

关闭录像文件，此时调用 TUCAM_Rec_AppendFrame 将无法写入数据。

声明

```
TUCAMRET  TUCAM_Rec_Stop (HDTUCAM hTUCam);
```

参数

HDTUCAM hTUCam 相机的句柄

错误代码

TUCAMRET_NOT_INIT TUCAM-API 未初始化

TUCAMRET_INVALID_CAMERA 无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Buf_Alloc、TUCAM_Buf_Release

TUCAM_Buf_WaitForFrame

TUCAM_Cap_Start、TUCAM_Cap_Stop

TUCAM_Rec_Start、TUCAM_Rec_AppendFrame

TUCAM_Reg_Read

描述

读取寄存器的内容。读取的类型参考 TUREG_TYPE。

声明

TUCAMRET TUCAM_Reg_Read (HDTUCAM hTUCam, TUCAM_REG_RW regRW);

参数

HDTUCAM hTUCam 相机的句柄

TUCAM_REG_RW regRW 寄存器读写结构体

错误代码

TUCAMRET_NOT_INIT TUCAM-API 未初始化

TUCAMRET_NO_MEMORY 传入的缓冲区未分配内存空间

TUCAMRET_NOT_SUPPORT 不支持该类型读取

TUCAMRET_INVALID_IDPARAM 无效的类型，参考 TUREG_TYPE

TUCAMRET_INVALID_CAMERA 无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Reg_Write

TUCAM_Reg_Write

描述

写入寄存器的内容。写入的类型参考 TUREG_TYPE。

声明

TUCAMRET TUCAM_Reg_Write(HDTUCAM hTUCam, TUCAM_REG_RW regRW);

参数

HDTUCAM hTUCam 相机的句柄

TUCAM_REG_RW regRW 寄存器读写结构体

错误代码

TUCAMRET_NOT_INIT TUCAM-API 未初始化

TUCAMRET_NO_MEMORY 传入的缓冲区未分配内存空间

TUCAMRET_NOT_SUPPORT 不支持该类型读取

TUCAMRET_INVALID_IDPARAM 无效的类型，参考 TUREG_TYPE

TUCAMRET_INVALID_CAMERA 无效的相机，相机句柄不存在时

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit

TUCAM_Dev_Open、TUCAM_Dev_Close

TUCAM_Reg_Read