



TUCAM-API-GenICam 开发指南



鑫图光电有限公司

版权(c) 2011-2025 Xintu Photonics Co., Ltd.(TUCSEN)

保留所有的权利

目录

1. 使用前阅读	3
2. 简介	3
3. 概述	4
3.1 层结构	4
3.2 原理	4
3.3 规则	5
3.3.1 句柄相关的内容	5
3.3.2 函数相关的内容	5
3.3.3 属性相关的内容	6
4. 编程指引	6
4.1 搭建编程环境	6
4.1.1 VS2013 开发环境配置	6
4.2 快速上手	10
4.2.1 起始和终止	11
4.2.2 属性获取和设置	13
4.2.3 内存管理	16
4.2.4 文件管理	19
4.2.5 其它	23
5. 参考	26
5.1 类型和常量	26
5.1.1 TUCAMRET 错误代码	26
5.1.2 TUCAM_IDINFO 产品信息代码	28
5.1.3 TUELEM_TYPE 属性节点数据类型	28
5.1.4 TUACCESS_MODE 属性节点的访问权限	29
5.1.5 TU_VISIBILITY 节点可见	29
5.1.6 TU_REPRESENTATION 控件的呈现形式	29
5.1.7 TUXML_DEVICE XML 文件对应的设备类型	30
5.1.8 TUIMG_FORMATS 图像格式代码	30
5.1.9 TUFRM_FORMATS 帧格式代码	31

5.1.10 TUDRAW_MODE 图像绘制模式代码	31
5.1.11 TUREC_MODE 录制文件关键帧模式代码	31
5.1.12 TUDATA_RW 数据读写标志符	31
5.2 结构体	31
5.2.1 初始化	32
5.2.2 打开相机	32
5.2.3 打开图像	32
5.2.4 回调函数缓冲区头部信息	33
5.2.5 属性节点信息	34
5.2.6 处理图像属性	36
5.2.7 帧结构	36
5.2.8 文件保存	38
5.2.9 录像保存	38
5.2.10 图像绘制初始化	39
5.2.11 图像绘制参数（仅 Windows）	39
5.3 函数	40
5.3.1 API 初始化 / 反初始化	40
5.3.2 打开、关闭相机	40
5.3.3 GenICam 相关接口	42
5.3.4 内存管理	46
5.3.5 捕获控制	50
5.3.6 文件管理	52
5.3.7 扩展控制	55
6. 常见问题解答（FAQ）	57
6.1 问题排查思路	57
6.2 常见问题解决方法	57

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，以使用户能够创建使用 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。

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。TUCSEN 对文档的版权保留所有权利。在没有 TUCSEN 的事先书面许可下，文档的任何部分不得被复制、传递、转录，存储在检索系统或翻译成任何语言或计算机语言，以任何的形式，或以任何方式，任何的手段如：电子、机械、电磁、光学、化学手动或其他。

2. 简介

TUCAM-API

TUCAM-API 是用于控制 TUCSEN 制造的相机的应用软件程序接口。TUCAM-API 目前支持连接 USB 、 CameraLink 、 CoaXPress 、 Gige 数据接口，在初始化时自动进行匹配工作。另外 API 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。

SDK

TUCAM-API 软件开发套件（以下简称“SDK”）为开发者制作主机软件程序提供资源。 SDK 由头文件、库文件、 TUCAM-API 函数参考文件、系列相机性能属性参考文件和示例代码组成。例如部分修改源代码和创建完全独立的程序，和/或将主机软件或插件模块提供给所有人。

说明

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。具体可以参考系列相机性能、属性文件。

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。

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。

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。

3.3 规则

3.3.1 句柄相关的内容

名称	描述
HDTUCAM	是 API 上指定目标相机的句柄，几乎所有的 API 都需要这个句柄作为参数。 TUCAM_Dev_Open 函数提供这个句柄，利用 TUCAM_Dev_Release 函数释放句柄，终止设备使用。
HDTUIMG	是 API 指定打开图片的句柄。 TUIMG_File_Open 函数提供这个句柄，利用 TUCAM_File_Close 关闭句柄，释放图片资源。

3.3.2 函数相关的内容

1) 函数名称

所有 TUCAM-API 函数都以大写字母“TUCAM_”作为前缀。TUCAM-API 所有函数按照性能被归类，同组的函数有相同前缀。以下是函数前缀列表：

函数前缀	函数名称	说明
TUCAM_Api_	Init, Uninit	初始化、反初始化 API
TUCAM_Dev_	Open, Close, GetInfo	打开、关闭相机，获取部分信息
TUCAM_GenICam_	BuffDataCallBack, GetBuffData, ElementAttr, ElementAttrNext, SetElementValue, GetElementValue, SetRegisterValue, GetRegisterValue	GenICam 协议相关接口
TUCAM_Buf_	Alloc, Release, AbortWait, WaitForFrame, CopyFrame	内存管理相关控制
TUCAM_File_	SaveImage, LoadProfiles, SaveProfiles, Open, Close	图片文件、配置文件相关操作
TUCAM_Rec_	Start, AppendFrame, Stop	视频文件保存相关接口
TUCAM_Draw_	Init, Frame, UnInit	绘制图像接口（Windows 系统）

2) 函数参数

除了 TUCAM_Api_Init 初始化、TUCAM_Api_Uninit 反初始化、TUCAM_Dev_Open 打开相机等函数，几乎所有的 TUCAM-API 都需要 HDTUCAM 句柄作为第一个参数。一些函数还需要一个指向结构体的指针作为参数，在结构体参数中有 [in] 标识变量，意味着应用程序必须在调用之前设置值，有 [out] 标识变量，意味着函数返回时填充一个值。

3) 函数返回

TUCAM-API 所有的函数执行后都有返回一个 TUCAMRET 值。所有的返回结果都被定义成 TUCAMRET_ 开头的。如果函数执行成功返回 TUCAMRET_SUCCESS，在开发中建议您在执行完函数后，判断返回结果值为 TUCAMRET_SUCCESS 再进行下一步操作，否则给出错误提醒，方便问题的查找。

3.3.3 属性相关的内容

在 TUCAM 中什么是属性？

“属性”是指设备参数。TUCAM-API 读取相机中定义配置。用户通过 TUCAM_GenICam_ElementAttr、TUCAM_GenICam_ElementAttrNext、

TUCAM_GenICam_SetElementValue、TUCAM_GenICam_GetElementValue 函数配置中的常量获取、设置、查询设备参数。每个属性值都有类型，每个属性都有自己 ID、最小值、最大值、默认值。具体参考结构体 TUCAM_ELEMENT。

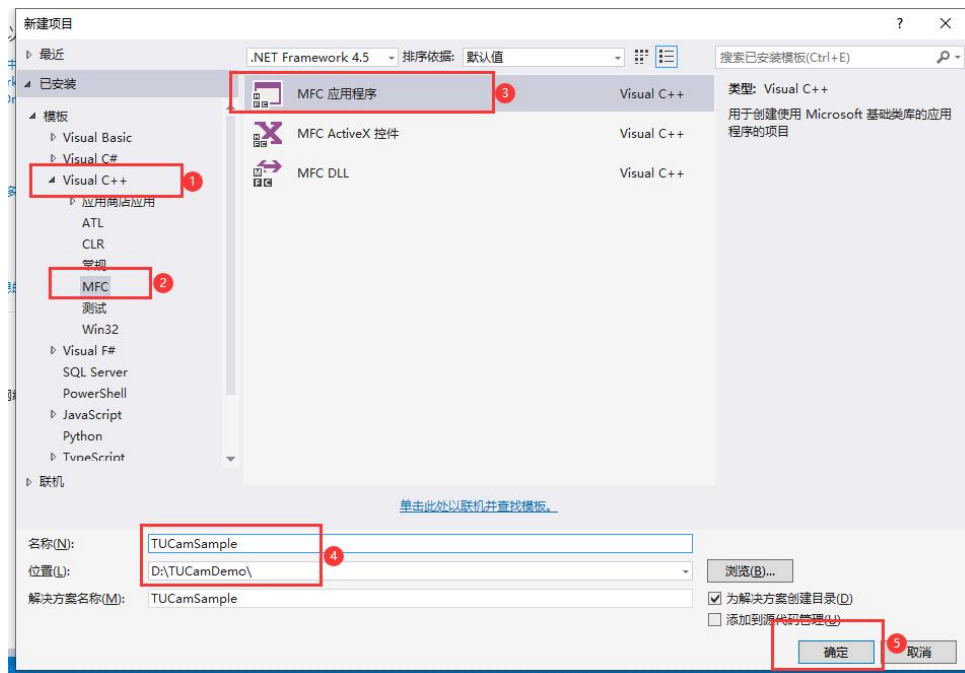
4. 编程指引

4.1 搭建编程环境

4.1.1 VS2013 开发环境配置

1) 新建工程

新建工程选择 MFC，点击“确定”后，对话框中点击“下一步”：



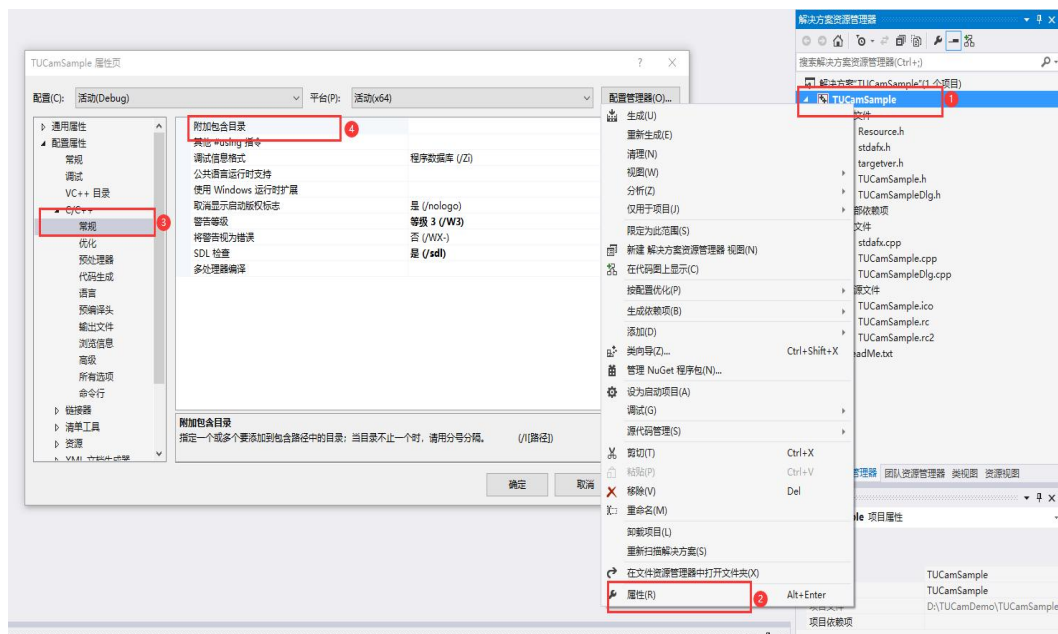
2) 配置引用

在弹出 MFC 应用程序向导对话框中选择“基于对话框”，点击完成。就在指定的目录中创建了 MFC 程序工程。



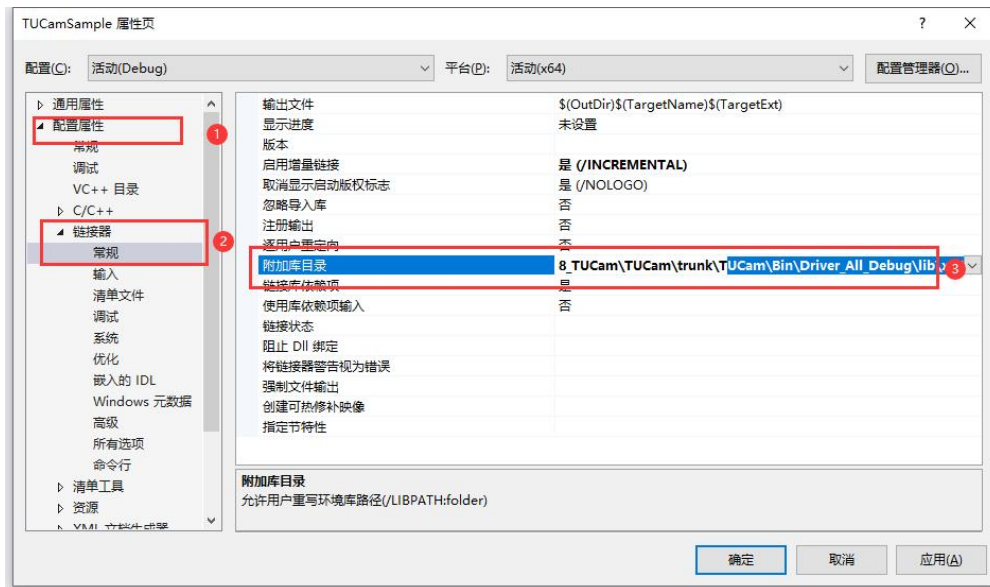
3) 配置引用头文件

在解决方案资源管理器窗口选中刚创建的工程，右键选中菜单中的“属性”选项，弹出属性窗口。选择 配置属性->C/C++->常规 在“附加包含目录”中填写 TUCamApi.h 所在目录路径地址（依用户安装目录为准），如图所示：

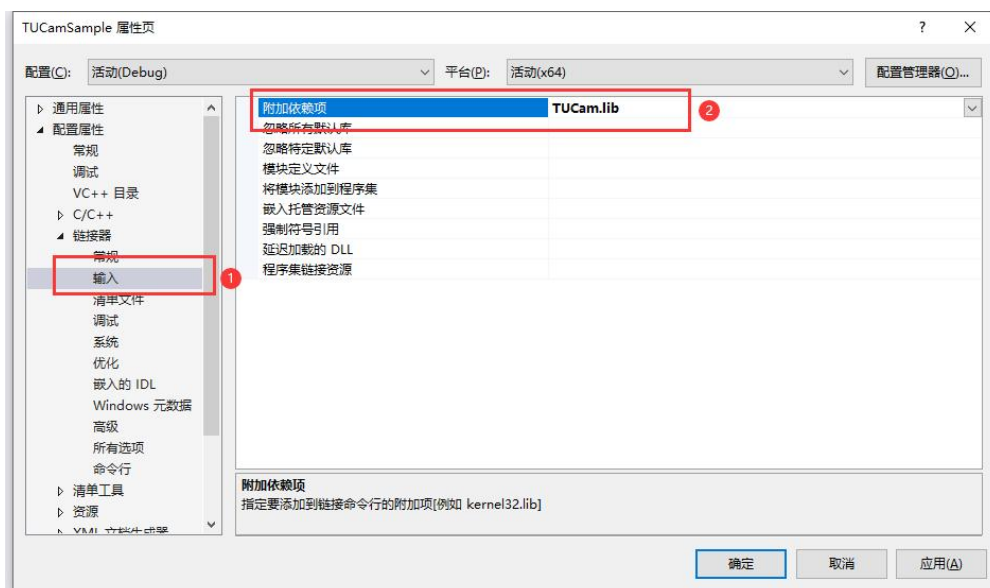


4) 配置 lib 文件

选择配置属性->链接器->常规，在 Additional Library Directories 中 填写 TUCam.lib 所在目录路径地址（依用户安装目录为准），如图所示：

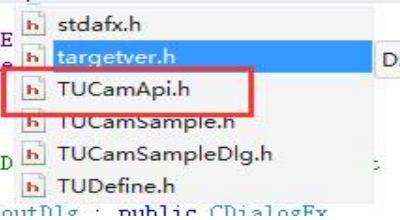


然后选择配置属性->链接器->输入在“附加依赖项”中填写 TUCam.lib，如图所示：



引用头部文件，有如下图提示：

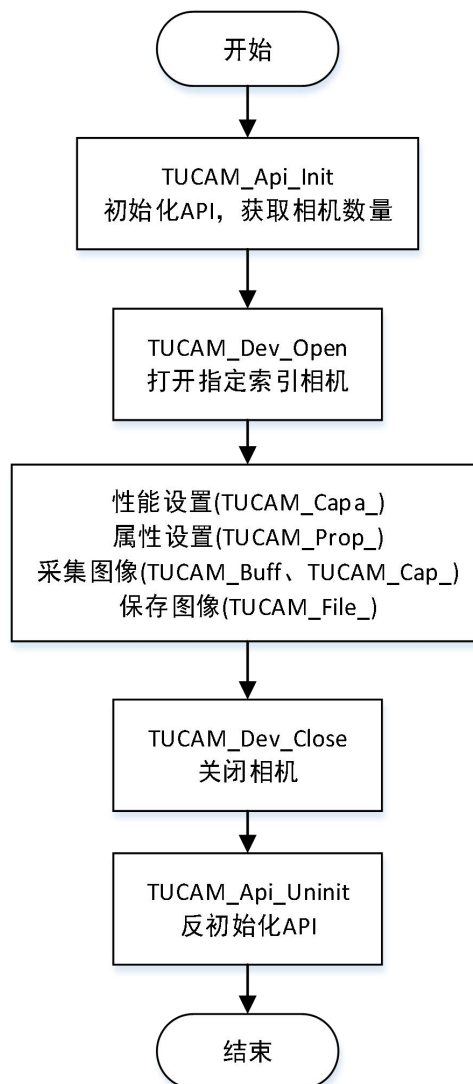
```
3 //
4
5 #include "stdafx.h"
6 #include "TUCamSample.h"
7 #include "TUCamSampleDlg.h"
8 #include "afxdialogex.h"
9 #include "t"
10
11 #ifdef _DE
12 #define ne
13 #endif
14
15 // CAboutD
16
17
18 class CAboutDlg : public CDialogEx
```



注意：根据用户应用选择适合（x86、x64）引用。

4.2 快速上手

TUCAM-API 工作调用流程：



4.2.1 起始和终止

4.2.1.1 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。

其次，当应用程序启动时，调用 SDK 的 API 初始化函数执行初始化操作。

再次，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）。当终止函数被调用时，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。

4.2.1.2 SDK 的 API 初始化

API 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。
[接口参考 5.3.1](#)

4.2.1.3 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 HDTUCAM 来做为其他函数的输入参数。[接口参考 5.3.2](#)

4.2.1.4 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。例如，相机型号、固件版本等。[接口参考 5.3.2](#)，产品信息参考 [TUCAM_IDINFO](#)

4.2.1.5 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。

4.2.1.6 示例代码

SDK 的 API 初始化参考以下代码，或者参考 Samples 目录 init_open 工程，相机产品信息参考 Samples 目录 get_info 工程。

```
1.  int main (int argc, char** argv)
2.  {
3.      TUCAM_INIT  itApi; // 初始化 SDK 环境参数
4.      TUCMA_OPEN  opCam; // 打开相机参数
5.
6.      itApi.pstrConfigPath = NULL;
7.      itApi.uiCamCount = 0;
8.      if (TUCAMRET_SUCCESS != TUCAM_Api_Init(&itApi))
9.      {
10.         // 初始化 SDK API 环境失败
11.         return 0;
12.     }
13.
14.     if (0 == itApi.uiCamCount)
15.     {
16.         // 没有相机
17.         return 0;
18.     }
19.
20.     opCam.hldxTUCam = 0;
21.     opCam.uiIdxOpen = 0;
22.
23.     if (TUCAMRET_SUCCESS != TUCAM_Dev_Open(&opCam))
24.     {
25.         // 打开相机失败
26.         return 0;
27.     }
28.
29.     // 应用程序可以使用 opCam.hldxTUCam 句柄
30.
31.     TUCAM_Dev_Close(opCam.hldxTUCam); // 关闭相机
32.     TUCAM_Api_Uninit(); // 反初始化 SDK API 环境
33.
34.     return 0;
35. }
36.
```

4.2.2 属性获取和设置

4.2.2.1 调用顺序

获取属性

TUCAM_GenICam_ElementAttr、TUCAM_GenICam_ElementAttrNext 函数，通过设置属性节点名称获取属性属性，其中包括最大值、最小值、默认值，参考 TUCAM_ELEMENT 查询、设置属性值

通过 TUCAM_GenICam_SetElementValue 和 TUCAM_GenICam_GetElementValue 函数来设置和获取属性值。设置和获取属性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，在某些情况下可能会返回错误的代码 TUCAMRET。 [接口参考 5.3.3](#)

4.2.2.2 属性节点名称

每个属性都有唯一的名称，通过 TUCAM_GenICam_ElementAttrNext 遍历接口可知节点名称、参数数据类型、读 / 写属性。

4.2.2.3 示例代码

参考以下代码:

```
1. // 属性节点访问权限字符串
2. static char s_access[][4] = {
3.     "NI",
4.     "NA",
5.     "WO",
6.     "RO",
7.     "RW"
8. };
9.
10. // 属性节点数据类型字符串
11. static char s_elemType[][16] = {
12.     "Value",
13.     "Base",
14.     "Integer",
15.     "Boolean",
16.     "Command",
```

```
17.     "Float",
18.     "String",
19.     "Register",
20.     "Category",
21.     "Enumeration",
22.     "EnumEntry",
23.     "Port"
24. };
25.
26. // 从根节点开始遍历
27. char *pData = NULL;
28. int nLevel = 0;          // 属性层级
29. TUCAM_ELEMENT node;     // 属性节点
30. node.pName = "Root";
31.
32. while ( TUCAMRET_SUCCESS ==
        TUCAM_GenICam_ElementAttrNext(opCam.hIdxTUCam, &node, node.pName) )
33. {
34.     if (NULL == node.pName)
35.         continue;
36.
37.     switch (node.Type)
38.     {
39.         case TU_ElemCategory: // 元素类别
40.         {
41.             nLevel = max(node.Level, 0);
42.             printf("%s[%d]%s\r\n", (nLevel << 1), "", nLevel, node.pName);
43.         }
44.         break;
45.
46.         case TU_ElemBoolean: // 布尔类型元素
47.             printf("%s[%d][%s][%s]%s, %d \r\n", (node.Level << 1), "", node.Level,
                s_access[node.Access], s_elemType[node.Type], node.pName, node.nVal);
48.             break;
49.
50.         case TU_ElemInteger: // 整型类型元素
51.             if (NULL != strstr(node.pName, "Timestamp"))
52.             {
53.                 printf("%s[%d][%s][%s]%s, %l64d \r\n", (node.Level << 1), "", node.Level,
                    s_access[node.Access], s_elemType[node.Type], node.pName, node.nVal);
54.             }
55.             else
```



```

56.     {
57.         printf("%s[%d][%s][%s]%s, %l64u \r\n", (node.Level << 1), "", node.Level,
s_access[node.Access], s_elemType[node.Type], node.pName, node.nVal);
58.     }
59.     break;
60.
61.     case TU_ElemFloat: // 浮点类型元素
62.         printf("%s[%d][%s][%s]%s, %.1f \r\n", (node.Level << 1), "", node.Level,
s_access[node.Access], s_elemType[node.Type], node.pName, node.dbVal);
63.         break;
64.
65.     case TU_ElemString: // 字符串类型元素
66.         pData = new char[node.nMax + 1];
67.         memset(pData, 0, node.nMax + 1);
68.         node.pTransfer = pData;
69.         TUCAM_GenICam_GetElementValue(s_opCam.hIdxTUCam, &node);
70.         printf("%s[%d][%s][%s]%s, %s \r\n", (node.Level << 1), "", node.Level,
s_access[node.Access], s_elemType[node.Type], node.pName, node.pTransfer);
71.         TUCAM_DELBUF(pData);
72.         break;
73.
74.     case TU_ElemEnumeration: // 枚举类型元素
75.     {
76. /*
77.         // 可以获取枚举的元素
78.         char str[1024] = { 0 };
79.         for (int j = 0; j <= node.nMax; ++j)
80.         {
81.             strcat_s(str, *(node.pEntries + j));
82.             strcat_s(str, ";");
83.         }
84. */
85.         // 值为 node.nVal, 对应字符串为 *(node.pEntries + node.nVal)
86.         pData = *(node.pEntries + node.nVal);
87.         printf("%s[%d][%s][%s]%s, %s \r\n", (node.Level << 1), "", node.Level,
s_access[node.Access], s_elemType[node.Type], node.pName, pData);
88.     }
89.     break;
90.
91.     case TU_ElemCommand: // 指令类型元素
92.         printf("%s[%d][%s][%s]%s, %d \r\n", (node.Level << 1), "", node.Level,
s_access[node.Access], s_elemType[node.Type], node.pName, node.nVal);

```



```
93.         break;
94.
95.         case TU_ElemRegister: // 寄存器类型元素
96.         default:
97.             printf("%*s[%d][%s][%s]%s, %s \r\n", (node.Level << 1), "", node.Level,
s_access[node.Access], s_elemType[node.Type], node.pName, "");
98.             break;
99.     }
100.
101.     printf("\r\n");
102. }
```

4.2.3 内存管理

4.2.3.1 调用顺序

1) 内存的分配

内存的分配 TUCAM_Buf_Alloc 必须在 TUCAM_Cap_Start 数据开始捕获之前调用，且切换不同的分辨率都必须重新进行内存分配。

2) 数据的获取

有两种方式获取数据，都必须在 TUCAM_Cap_Start 数据开始捕获之后才能获取到数据。

① 通过注册回调函数 TUCAM_GenlCam_BuffDataCallBack，在有新的图像输出时触发回调，然后通过 TUCAM_GenlCam_GetBuffData 函数获取图像数据。

② 通过函数 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。

3) 结束等待

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) 内存释放

内存的释放 TUCAM_Buf_Release 必须在 TUCAM_Cap_Stop 停止数据捕获之后调用。

4.2.3.2 帧结构体

用于描述一帧图像数据的结构体。参考 [TUCAM_FRAME](#)。变量 pBuffer 是指向帧数据指针，帧数据包括帧头部数据、帧图像数据两部分。将帧数据偏移 usHeader 字节获取到帧图像数据。

4.2.3.3 示例代码

参考以下代码或 Samples 目录 wait_frame 工程：

```
1.  TUCAM_FRAME m_frame;           // 帧对象
2.  HANDLE m_hThdGrab;             // 取图线程事件句柄
3.  BOOL m_bLiving;                // 是否取图
4.
5.  BOOL CDlgTUCam::StartCapture()
6.  {
7.      m_frame.pBuffer      = NULL;
8.      m_frame.ucFormatGet = TUFRM_FMT_RGB888; // 帧数据格式 (RGB888)
9.      m_frame.uiRsdSize   = 1; // 一次捕获帧数 (TUCCM_TRIGGER_STANDARD 可大于 1)
10.
11.     if (TUCAMRET_SUCCESS != TUCAM_Buf_Alloc(m_opCam.hIdxTUCam, &m_frame))
12.     {
13.         return FALSE;
14.     }
15.
16.     if (TUCAMRET_SUCCESS != TUCAM_Cap_Start(m_opCam.hIdxCam,
        TUCCM_SEQUENCE))
17.     {
18.         TUCAM_Buf_Release(m_opCam.hIdxTUCam);
19.         return FALSE;
20.     }
21.
22.     m_bLiving = TRUE;
23.     m_hThdGrab = CreateEvent(NULL, TRUE, FALSE, NULL);
```

```
24.     _beginthread(GrabThread, 0, this);
25.
26.     return TRUE;
27. }
28.
29. Void __cdecl CDlgTUCam::GrabThread(LPVOID IParam)
30. {
31.     CDlgTUCam *pTuCam = (CDlgTUCam *)IParam;
32.
33.     While (pTUCam->m_bLiving)
34.     {
35.         pTUCam->m_frame.ucFormatGet = TUFRM_FMT_RGB888;
36.         if(TUCAMRET_SUCCESS ==
TUCAM_Buf_WaitForFrame(pTUCam->m_opCam.hIdxTUCam,\
37.                                     &pTUCam->m_frame))
38.         {
39.             // pTUCam->m_frame.pBuffer 返回捕获的图像数据, 格式为
TUFRM_FMT_RGB88
40.             // 该数据可以用于显示
41.
42.             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.
49.             if(TUCAMRET_SUCCESS==TUCAM_Buf_CopyFrame(pTUCam->m_opCam.hIdxTUCam,\
&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;
```

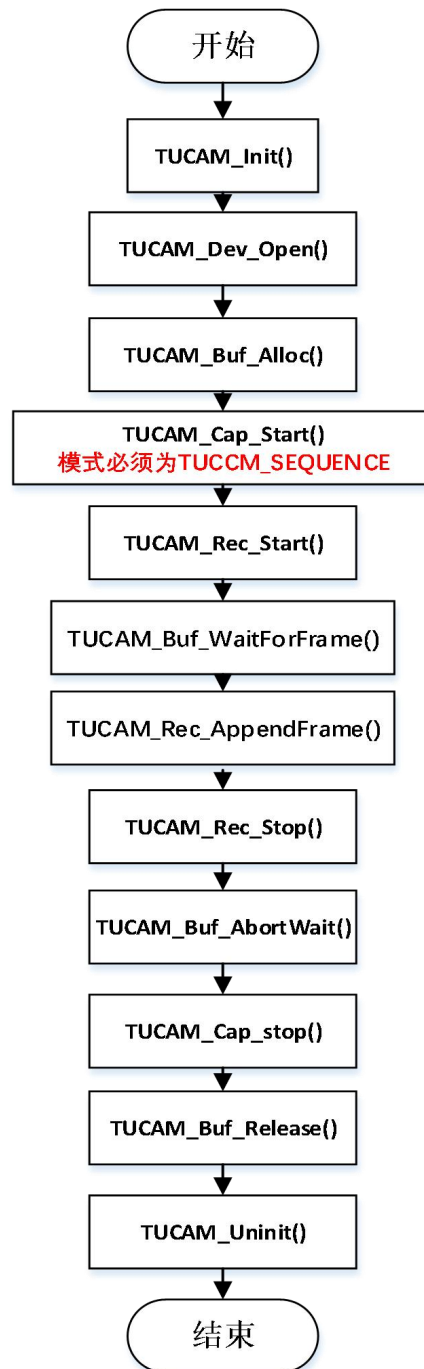
```
63.    TUCAM_BUF_AbortWait();    // 如果调用 TUCAM_Buf_WaitForFrame 接口
64.
65.    WaitForSingleObject(m_hThdGrab, INFINITE);    // 等待取图线程退出
66.    CloseHandle(m_hThdGrab);
67.    m_hThdGrab = NULL;
68.
69.    TUCAM_Cap_Stop(m_opCam.hIdxTUCam);    // 停止数据捕获
70.    TUCAM_Buf_Release(m_opCam.hIdxTUCam);    // 释放分配的内存
71. }
72.
```

4.2.4 文件管理

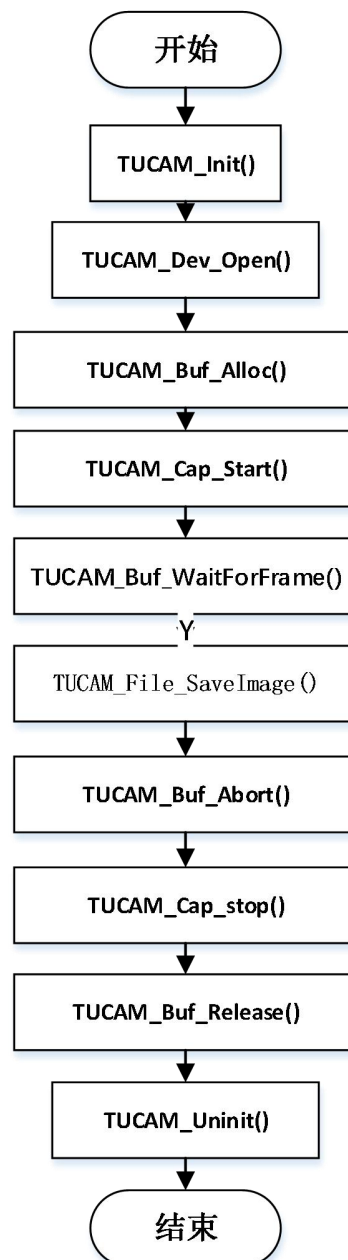
4.2.4.1 调用顺序

1) 视频流保存流程：

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 要将图像数据以追加的方式写入文件中，录像结束调用 TUCAM_Rec_Stop 来结束录像过程，再调用 TUCAM_CAP_STOP 结束采集。



2) 图片保存流程:



4.2.4.2 文件结构体

文件保存结构体参考 [TUCAM_File_Save](#)

录像保存结构体参考 [TUCAM_Rec_Save](#)

4.2.4.3 示例代码

参考以下代码或 Samples 目录 save_image/save_video 工程：

```
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,
6.         &m_frame))
7.     {
8.         TUCAM_FILE_SAVE fileSave;
9.         fileSave.nSaveFmt      = TUFMT_TIF;           // 保存 Tiff 格式
10.        fileSave.pFrame        = &m_frame;           // 需要保存的帧指针
11.        fileSave.pstrSavePath = "C:\\image";          // 路径包含文件名（不包含扩展名）
12.
13.        if (TUCAMRET_SUCCESS == TUCAM_File_SaveImage(m_opCam.hIdxTUCam,
14.            fileSave))
15.        {
16.            // 保存图像文件成功
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,
37.         &m_frame))
38.     {
39.         TUCAM_Rec_AppendFrame(m_opCam.hIdxTUCam, &m_frame);
40.     }
```

```
40. }  
41.  
42. void StopRecording()  
43. {  
44.     TUCAM_Rec_Stop(m_opCam.hIdxTUCam);  
45. }  
46.
```

4.2.5 其它

4.2.5.1 转 OpenCV 的 cvMat 格式

```
1. if(TUCAMRET_SUCCESS==TUCAM_Buf_WaitForFrame(m_opCam.hIdxTUCam,  
    &m_frame))  
2. {  
3.     int type = CV_MAKETYPE(m_frame.ucDepth, m_frame.ucChannels);  
4.     uchar *data = m_frame.pBuffer + m_frame.usHeader;  
5.     Mat dst = Mat(m_frame.usHeight, m_frame.usWidth, type, data);  
6. }  
7.
```

4.2.5.2 转 C# 的 Bitmap 格式

```
1. m_frame.ucFormatGet = TUFRM_FMT_RGB888;  
2. if (TUCAMRET.TUCAMRET_SUCCESS ==  
    TUCamAPI.TUCAM_Buf_WaitForFrame(m_opCam.hIdxTUCam, ref m_frame))  
3. {  
4.     int nWidthStep = m_frame.uiWidthStep;  
5.     int nSize = (int)(m_frame.uiImgSize + m_frame.usHeader);  
6.     byte[] buffer = new byte[nSize];  
7.     Marshal.Copy(m_frame.pBuffer, buffer, 0, nSize);  
8.     // 偏移头部数据  
9.     Buffer.BlockCopy(buffer, (int)(m_frame.usHeader), buffer, 0, (int)(m_frame.uiImgSize));  
10.  
11.    // 转换为 Bitmap  
12.    int stride = (int)(m_frame.uiWidthStep);  
13.    GCHandle handle = GCHandle.Alloc(buffer, GCHandleType.Pinned);  
14.    int scan0 = (int)handle.AddrOfPinnedObject();  
15.    scan0 += (m_frame.usHeight - 1) * stride;
```



```
16.     System.Drawing.Bitmap bitmap = new System.Drawing.Bitmap(m_frame.usWidth,
    m_frame.usHeight, -stride, System.Drawing.Imaging.PixelFormat.Format24bppRgb,
    (IntPtr)scan0);
17.     handle.Free();
18.
19.     if (null != bmpDraw)
20.     {
21.         bmpDraw.Dispose();
22.     }
23.
24.     if (null != bitmap)
25.     {
26.         // 这个数据就是 C#的 Bitmap, 可以转成 Image
27.         Bitmap bmpDraw = bitmap.Clone(new Rectangle(0, 0,
    bitmap.Width,bitmap.Height),bitmap.PixelFormat);
28.     }
29. }
30.
```

4.2.5.3 计算平均灰度值

```
1.  if(TUCAMRET_SUCCESS == TUCAM_Buf_WaitForFrame(m_opCam.hIdxTUCam,
    &m_frame))
2.  {
3.      // 计算全幅区域或 ROI 的平均灰度值
4.      bool isRoi = false;
5.
6.      // ROI 参数
7.      int roiX = 200;
8.      int roiY = 400;
9.      int roiW = 800;
10.     int roiH = 600;
11.
12.     int startX  = isRoi ? roiX : 0;
13.     int startY  = isRoi ? roiY : 0;
14.     int width   = isRoi ? roiW : m_frame.usWidth;
15.     int height  = isRoi ? roiH : m_frame.usHeight;
16.     int finishX = startX + width;
17.     int finishY = startY + height;
18.
```

```
19.     int pixels = width * height;
20.     double avg = 0;
21.     long long sum = 0;
22.
23.     // 16bit
24.     if ( 2 == m_frame.ucElemBytes )
25.     {
26.         unsigned short *data = (unsigned short *)(m_frame.pBuffer + m_frame.usHeader);
27.
28.         for (int i = startY; i < finishY; ++i)
29.         {
30.             for (int j = startX; j < finishX; ++j)
31.             {
32.                 sum += data[i * m_frame.usWidth + j];
33.             }
34.         }
35.     }
36.     else // 8bit
37.     {
38.         unsigned char *data = (unsigned char *)(m_frame.pBuffer + m_frame.usHeader);
39.
40.         for (int i = startY; i < finishY; ++i)
41.         {
42.             for (int j = startX; j < finishX; ++j)
43.             {
44.                 sum += data[i * m_frame.usWidth + j];
45.             }
46.         }
47.     }
48.
49.     // 计算平均值
50.     avg = sum * 1.0 / pixels;
51. }
52.
```

5. 参考

5.1 类型和常量

5.1.1 TUCAMRET 错误代码

错误代码常量 TUCAMRET_ 为前缀，按照类型分为初始化错误、状态错误、等待错误、调用错误、相机或接口错误。

名称	错误代码	描述
TUCAMRET_SUCCESS	0x00000001	没有错误，成功代码，应用程序应检查值为正
TUCAMRET_RECEIVE_FINISH	0x00000002	没有错误，厂商收到帧信息
TUCAMRET_EXTERNAL_TRIGGER	0x00000003	没有错误，接收到外部触发信号
TUCAMRET_FAILURE	0x80000000	调用 API 接口失败
初始化错误		
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	没有支持的子模块
TUCAMRET_NO_DRIVER	0x80000104	没有支持的驱动
TUCAMRET_NO_CAMERA	0x80000105	没有连接的相机
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（USB 接口）
TUCAMRET_FAILOPEN_BULKOUT	0x80000112	打开批传输输出端点失败（USB 接口）
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
TUCAMRET_FAILOPEN_CODEC	0x80000116	打开编码器失败
TUCAMRET_FAILOPEN_CONTEXT	0x80000117	打开上下文失败
状态错误		
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
等待错误		

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的问题
TUCAMRET_USB_STATUS_ERROR	0x8000020B	USB 状态错误
调用错误		
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	无效帧序列号
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，但通道无效
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_INVALID_IDVPROP	0x8000030F	无效的厂商属性 ID
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	相机不支持性能或属性
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 API 支持
TUCAMRET_ACCESSDENY	0x80000412	访问拒绝（可能是没有足够权限）
TUCAMRET_NO_CORRECTIONDATA	0x80000501	没有彩色校点数据
TUCAMRET_INVALID_PRFSETS	0x80000601	配置文件设置名称无效
TUCAMRET_INVALID_IDPPROP	0x80000602	无效的属性 ID
TUCAMRET_DECODE_FAILURE	0x80000701	解码失败
TUCAMRET_COPYDATA_FAILURE	0x80000702	拷贝数据失败
TUCAMRET_ENCODE_FAILURE	0x80000703	编码失败
TUCAMRET_WRITE_FAILURE	0x80000704	写入失败
相机或总线错误		
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	光学部件已拆下，请检查

5.1.2 TUCAM_IDINFO 产品信息代码

名称	代码	描述
TUIDI_BUS	0x00	USB 接口类型。0x200、0x210 对应 USB2.0。
TUIDI_VENDOR	0x01	厂商 ID
TUIDI_PRODUCT	0x02	产品 ID
TUIDI_VERSION_API	0x03	当前使用 TUCAM-API 接口版本
TUIDI_VERSION_FRMW	0x04	当前相机固件版本号
TUIDI_VERSION_FPGA	0x06	当前相机 FPGA 版本号
TUIDI_VERSION_DRIVER	0x07	当前相机驱动版本号
TUIDI_TRANSFER_RATE	0x08	传输速率
TUIDI_CAMERA_MODEL	0x09	当前相机型号，字符串类型（例如 Dhyana 400BSI V3）
TUIDI_CURRENT_WIDTH	0x0A	相机图像数据宽度（必须使用 TUCAM_Dev_GetInfoEx，并在 TUCAM_Buf_Alloc 后调用）
TUIDI_CURRENT_HEIGHT	0x0B	相机图像数据高度（必须使用 TUCAM_Dev_GetInfoEx，并在 TUCAM_Buf_Alloc 后调用）
TUIDI_CAMERA_CHANNELS	0x0C	相机图像数据通道数。彩色相机为 3，黑白相机为 1。
TUIDI_BCDDEVICE	0x0D	BCD 码
TUIDI_TEMPALARMFLAG	0x0E	温度预警标志
TUIDI_UTCTIME	0x0F	获取 UTC（世界统一时间、世界标准时间）
TUIDI_LONGITUDE_LATITUDE	0x10	获取经纬度
TUIDI_WORKING_TIME	0x11	获取工作时间
TUIDI_FAN_SPEED	0x12	获取风扇速度
TUIDI_FPGA_TEMPERATURE	0x13	获取 FPGA 温度
TUIDI_PCBA_TEMPERATURE	0x14	获取 PCBA 温度
TUIDI_ENV_TEMPERATURE	0x15	获取环境温度
TUIDI_DEVICE_ADDRESS	0x16	USB 设备地址
TUIDI_USB_PORT_ID	0x17	USB 设备端口号
TUIDI_ENDINFO	0x18	产品信息 ID 结束位

5.1.3 TUELEM_TYPE 属性节点数据类型

名称	代码	描述
----	----	----

TU_ElemValue	0x00	对应 GenICam 协议的 IValue 类型
TU_ElemBase	0x01	对应 GenICam 协议的 IBase 类型
TU_ElemInteger	0x02	对应 GenICam 协议的 IInteger 类型
TU_ElemBoolean	0x03	对应 GenICam 协议的 IBoolean 类型
TU_ElemCommand	0x04	对应 GenICam 协议的 ICommand 类型
TU_ElemFloat	0x05	对应 GenICam 协议的 IFloat 类型
TU_ElemString	0x06	对应 GenICam 协议的 IString 类型
TU_ElemRegister	0x07	对应 GenICam 协议的 IRegister 类型
TU_ElemCategory	0x08	对应 GenICam 协议的 ICategory 类型
TU_ElemEnumeration	0x09	对应 GenICam 协议的 IEnumeration 类型
TU_ElemEnumEntry	0x0A	对应 GenICam 协议的 IEnumEntry 类型
TU_ElemPort	0x0B	对应 GenICam 协议的 IPort 类型

5.1.4 TUACCESS_MODE 属性节点的访问权限

名称	代码	描述
TU_AM_NI	0x00	未实现的节点
TU_AM_NA	0x01	无法使用的节点
TU_AM_WO	0x02	只写权限的节点
TU_AM_RO	0x03	只读权限的节点
TU_AM_RW	0x04	读写权限的节点

5.1.5 TU_VISIBILITY 节点可见

名称	代码	描述
TU_VS_Beginner	0x00	始终可见
TU_VS_Expert	0x01	专家或导师可见
TU_VS_Guru	0x02	导师可见
TU_VS_Invisible	0x03	不可见
TU_VS_UndefinedVisibility	0x04	对象尚未初始化

5.1.6 TU_REPRESENTATION 控件的呈现形式

名称	代码	描述
TU_REPRESENTATION_LINEAR	0x00	具有线性行为的滑块
TU_REPRESENTATION_LOGARITHMIC	0x01	具有对数行为的滑块
TU_REPRESENTATION_BOOLEAN	0x02	复选框

N		
TU_REPRESENTATION_PURE_NUMBER	0x03	编辑控件中的十进制数
TU_REPRESENTATION_HEX_NUMBER	0x04	编辑控件中的十六进制数
TU_REPRESENTATION_UNDEFINED	0x05	未找到的表示
TU_REPRESENTATION_IPV4ADDRESS	0x06	IP 地址（IP 版本 4）
TU_REPRESENTATION_MACADDRESS	0x07	MAC 地址
TU_REPRESENTATION_TIMESTAMP	0x08	时间戳
TU_REPRESENTATION_PTPFRAMECNT	0x09	结束位

5.1.7 TUXML_DEVICE XML 文件对应的设备类型

名称	代码	描述
TU_CAMERA_XML	0x00	相机的 XML 表
TU_CAMERALINK_XML	0x01	采集卡的 XML 表

5.1.8 TUIMG_FORMATS 图像格式代码

名称	代码	描述
TUFMT_RAW	0x01	保存图像为 RAW 格式。Tucsen 自定义的一种相机图像文件格式（包含头部 + 图像数据）
TUFMT_TIF	0x02	保存图像为 TIF 格式，也叫 TIF 格式。支持不同颜色模式、路径、透明度和通道，是打印文档最常用的格式。 优点：保存丰富的图像层次和细节，画面质量无损。 缺点：占用存储空间大。
TUFMT_PNG	0x04	保存图像为 PNG 格式。 优点：支持高级别无损压缩；支持透明背景。 缺点：对旧浏览器和软件兼容性较差。
TUFMT_JPG	0x08	保存图像为 JPG 格式。 优点：能将图片压缩至很小的存储空间，对色彩信息的保留较好。 缺点：有损压缩会降低图片数据质量。
TUFMT_BMP	0x10	保存图像为 BMP 格式。Windows 系统中标准图像文件格式，支持多种 Windows 应用程序使用。

		优点：无损压缩；图像画质优秀。 缺点：占用存储空间大。
--	--	--------------------------------

5.1.9 TUFM_FORMATS 帧格式代码

名称	代码	描述
TUFM_FMT_RAW	0x10	RAW 格式数据
TUFM_FMT_USUAL	0x11	通常格式数据（8bit/16bit、黑白/彩色）
TUFM_FMT_RGB888	0x12	RGB888 格式数据（可用于显示）

5.1.10 TUDRAW_MODE 图像绘制模式代码

注意：TUDRAW_DFT 模式仅在 windows 系统中可用，并且 windows 系统只能支持 TUDRAW_DFT 模式。

名称	代码	描述
TUDRAW_DFT	0x00	默认绘制模式
TUDRAW_DIB	0x01	DIB 绘制模式
TUDRAW_DX9	0x02	DirectX 9.0

5.1.11 TUREC_MODE 录制文件关键帧模式代码

名称	代码	描述
TUREC_TIMESTAMP	0x01	根据时间戳确定关键帧位置（To Disk 模式使用）
TUREC_SEQUENCE	0x02	根据图像序列确定关键帧位置（To Ram 模式使用）

5.1.12 TUDATA_RW 数据读写标志符

名称	代码	描述
TUDATA_READ	0x00	读取数据
TUDATA_WRITE	0x01	写入数据

5.2 结构体

一些函数需要一个指向结构体的指针作为参数。对于结构体参数，结构体中有[in]标识变量，意味着应用程序必须在调用之前设置值，有[out]标识变量，意味着函数返回

时填充一个值。

5.2.1 初始化

```
31. typedef struct _tagTUCAM_INIT
32. {
33.     UINT32    uiCamCount;
34.     PCHAR     pstrConfigPath;
35. }TUCAM_INIT, *PTUCAM_INIT;
36.
```

接口 TUCAM_API_Init 输入参数：

名称	描述
uiCamCount	[out] 返回当前连接的相机个数
pstrConfigPath	[in] 输入保存相机参数的路径，当执行 TUCAM_File_SaveProfiles 函数时保存该路径下

5.2.2 打开相机

```
1. typedef struct _tagTUCAM_OPEN
2. {
3.     UINT32    uiIdxOpen;
4.     HDTUCAM   hIdxTUCam;
5. }TUCAM_OPEN, *PTUCAM_OPEN;
```

接口 TUCAM_Dev_Open 输入参数：

名称	描述
uiIdxOpen	[in] 打开指定索引相机
hIdxTUCam	[out] 已打开相机的句柄

5.2.3 打开图像

```
1. typedef struct _tagTUIMG_OPEN
```

```

2. {
3.     PCHAR    pszfileName;
4.     HDTUIMG  hldxTUImg;
5. }TUIMG_OPEN, *PTUIMG_OPEN;
6.

```

接口 TUIMG_File_Open 输入参数：

名称	描述
pszfileName	[in] 打开指定索引相机
hldxTUImg	[out] 已打开图像的句柄

5.2.4 回调函数缓冲区头部信息

```

1. typedef struct _tagTUCAM_CB_BUFHEADER
2. {
3.     UINT32 uiWidth;
4.     UINT32 uiHeight;
5.     UINT32 uiXOffset;
6.     UINT32 uiYOffset;
7.     UINT32 uiXPadding;
8.     UINT32 uiYPadding;
9.     UINT32 uilmgOffset;
10.    UINT32 uiPixelFormat;
11.    UINT32 uilmgSize;
12.    UINT32 uiBufHandle;
13. } TUCAM_CB_BUFHEADER, *PTUCAM_CB_BUFHEADER;
14.

```

接口 TUCAM_GenlCam_GetBuffData 输入参数：

名称	描述
uiWidth	[out] 水平像素数
uiHeight	[out] 垂直像素数
uiXOffset	[out] X 偏移量（目前为 0）
uiYOffset	[out] Y 偏移量（目前为 0）
uiXPadding	[out] X 填充
uiYPadding	[out] Y 填充

uilmgOffset	[out] 数据偏移量
uiPixelFormat	[out] 像素格式
uilmgSize	[out] 数据大小
uiBufHandle	[out] 缓存区句柄

5.2.5 属性节点信息

```

1.  typedef struct _tagTUCAM_ELEMENT
2.  {
3.      BYTE IsLocked;
4.      BYTE Level;
5.      WORD Representation;
6.      TUELEM_TYPE Type;
7.      TUACCESS_MODE Access;
8.      TU_VISIBILITY Visibility;
9.      INT32 nReserve;
10.     union {
11.         struct {
12.             INT64 nVal;
13.             INT64 nMin;
14.             INT64 nMax;
15.             INT64 nStep;
16.             INT64 nDefault;
17.         };
18.         struct {
19.             DOUBLE dbVal;
20.             DOUBLE dbMin;
21.             DOUBLE dbMax;
22.             DOUBLE dbStep;
23.             DOUBLE dbDefault;
24.         };
25.     };
26.     PCHAR pName;
27.     PCHAR pDisplayName;
28.     PCHAR pTransfer;
29.     PCHAR pDesc;
30.     PCHAR pUnit;
31.     PCHAR *pEntries;
32.     INT64 PollingTime;
33.     INT64 DisplayPrecision;

```

```
34. }TUCAM_ELEMENT, *PTUCAM_ELEMENT;
35.
```

以下接口输入参数：

TUCAM_GenICam_ElementAttr 、 TUCAM_GenICam_ElementAttrNext 、
TUCAM_GenICam_SetElementValue、TUCAM_GenICam_GetElementValue

名称	描述
IsLocked	[out] 属性是否锁定，锁定状态下不可写
Level	[out] 节点级别
Representation	[out] 控件呈现方式
Type	[out] [RO]属性数据类型
Access	[out] 节点访问模式，读/写
Visibility	[out] 可见性
nReserve	[out] 保留字段
nVal / dbVal	[in/out] 属性当前值或字符串长度
nMin / dbMin	[out] 属性最小值或字符串长度
nMax / dbMax	[out] 属性最大值或字符串长度
nStep / dbStep	[out] 属性步进或字符串长度
nDefault / dbDefault	[out] 属性默认值或字符串长度
pName	[in/out] [RO]属性 ID 值
pDisplayName	[out] [RO]属性显示名称
pTransfer	[out] 寄存器地址（字符串）
pDesc	[out] 描述
pUnit	[out] 单位
*pEntries	[out] 枚举条目数组
PollingTime	[out] 轮询时间
DisplayPrecision	[out] 展示精度

5.2.6 处理图像属性

```
1. typedef struct _tagTUCAM_PPROP_ATTR
2. {
3.     INT32    idVProp;
4.     INT32    procType;
5.     DOUBLE   dbValMin;
6.     DOUBLE   dbValMax;
7.     DOUBLE   dbValDft;
8.     DOUBLE   dbValStep;
9. }TUCAM_PPROP_ATTR, *PTUCAM_PPROP_ATTR;
10.
```

接口 TUCAM_Proc_Prop_GetValueText 输入参数：

名称	描述
idPProp	[in] 图像处理 ID，参考 TUCAM_IDPPROP
procType	[in] 图像处理类型，参考 TUPROC_TYPE
dbValMin	[out] 可取最小值
dbValMax	[out] 可取最大值
dbValDft	[out] 设备默认值
dbValStep	[out] 可取范围内最小步进

5.2.7 帧结构

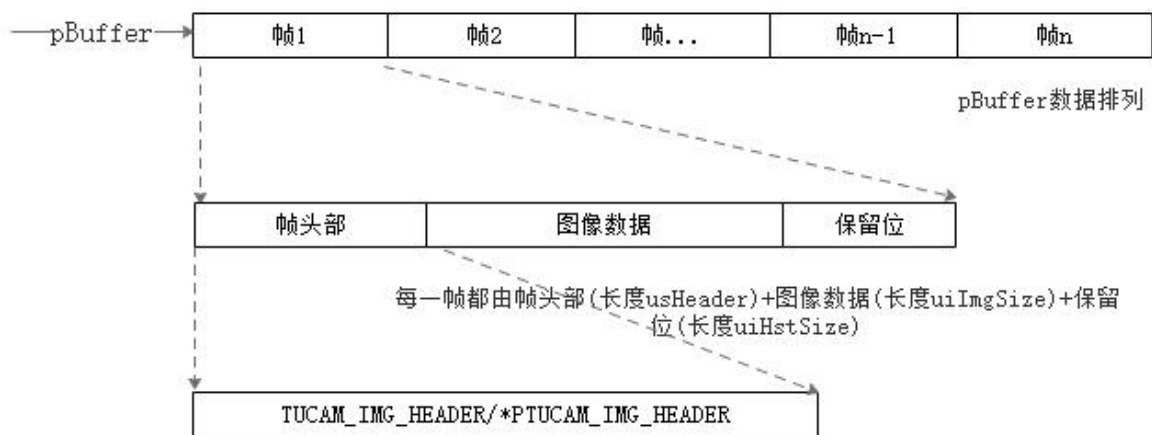
```
1. typedef struct _tagTUCAM_FRAME
2. {
3.     CHAR szSignature[8];
4.     USHORT usHeader;
5.     USHORT usOffset;
6.     USHORT usWidth;
7.     USHORT usHeight;
8.     UINT32 uiWidthStep;
9.     UCHAR  ucDepth;
10.    UCHAR  ucFormat;
11.    UCHAR  ucChannels;
12.    UCHAR  ucElemBytes;
13.    UCHAR  ucFormatGet;
14.    UINT32 uiIndex;
```

```

15.     UINT32 uiImgSize;
16.     UINT32 uiRsdSize;
17.     UINT32 uiHstSize;
18.     PCHAR pBuffer;
19. }TUCAM_FRAME, *PTUCAM_FRAME;
20.

```

pBuffer 数据排列如下，可以根据帧头部大小获取到每一帧的头部信息，也可以根据帧数据偏移获取到图像数据，进行绘制展示：



帧头部信息结构体，能够获取到图片宽度、高度、色阶、图片格式等信息

TUCAM_Buf_WaitForFrame 输入参数：

名称	描述
szSignature[8]	[out] 版权信息
usHeader	[out] 帧头部大小
usOffset	[out] 帧数据偏移大小
usWidth	[out] 水平像素数
usHeight	[out] 垂直像素数
uiWidthStep	[out] 帧图像的宽度步长
ucDepth	[out] 帧图像数据位深
ucFormat	[out] 帧图像数据格式
ucChannels	[out] 帧图像的通道数
ucElemBytes	[out] 帧图像的数据字节数
ucFormatGet	[in/out] 需要获取的图像格式。参考 TUFrm_FORMATS
uiIndex	[out] 帧图像序列号(保留)
uiImgSize	[out] 帧图像数据的大小

uiRsdSize	[in] 需要获取的帧数
uiHstSize	[out] 帧图像保留字段
pBuffer	[out] 指向帧数据缓存的指针

5.2.8 文件保存

```

1. typedef struct _tagTUCAM_FILE_SAVE
2. {
3.     INT32    nSaveFmt;
4.     PCHAR    pstrSavePath;
5.     PTUCAM_FRAME pFrame;
6. }TUCAM_FILE_SAVE, *PTUCAM_FILE_SAVE;

```

TUCAM_File_SaveImage 图像保存用到这个结构体：

名称	描述
nSaveFmt	[in] 保存图像的格式 TUIMG_FORMATS
pstrSavePath	[in] 保存图像路径(不包括扩展名)
pFrame	[in] 指向帧数据指针

5.2.9 录像保存

```

1. typedef struct _tagTUCAM_REC_SAVE
2. {
3.     INT32    nCodec;
4.     PCHAR    pstrSavePath;
5.     float    fFps;
6. }TUCAM_REC_SAVE, *PTUCAM_REC_SAVE;
7.

```

接口 TUCAM_Rec_Start 输入参数：

名称	描述
nCodec	[in] 编码解码类型，默认为 0
pstrSavePath	[in] 保存路径(包含文件名)
fFps	[in] 需要保存的帧率

5.2.10 图像绘制初始化

```
1. typedef struct _tagTUCAM_DRAW_INIT
2. {
3.     #ifdef TUCAM_TARGETOS_IS_WIN32;
4.         HWND    hWnd;
5.     #endif;
6.         INT32    nMode;
7.         UCHAR    ucChannels;
8.         INT32    nWidth;
9.         INT32    nHeight;
10. }TUCAM_DRAW_INIT, *PTUCAM_DRAW_INIT;
```

接口 TUCAM_Draw_Init 输入参数：

名称	描述
hWnd	[in] 绘制窗口句柄，仅支持 Windows 操作系统
nMode	[in] 是否使用硬件加速（预留 GPU 支持），默认值 TUDRAW_DFT
ucChannels	[in] 相机通道数
nWidth	[in] 绘制数据宽度
nHeight	[in] 绘制数据高度

5.2.11 图像绘制参数（仅 Windows）

```
1. typedef struct _tagTUCAM_DRAW
2. {
3.     INT32    nSrcX;
4.     INT32    nSrcY;
5.     INT32    nSrcWidth;
6.     INT32    nSrcHeight;
7.     INT32    nDstX;
8.     INT32    nDstY;
9.     INT32    nDstWidth;
10.    INT32    nDstHeight;
11.    PTUCAM_FRAME pFrame;
12. }TUCAM_DRAW, *PTUCAM_DRAW;
13.
```


接口 TUCAM_Draw_Frame 输入参数：

名称	描述
nSrcX	[in/out] 源矩形左上角的 x 坐标（以像素为单位）
nSrcY	[in/out] 源矩形左上角的 y 坐标（以像素为单位）
nSrcWidth	[in/out] 源矩形的宽度（以像素为单位）
nSrcHeight	[in/out] 源矩形的高度（以像素为单位）
nDstX	[in/out] 目标矩形左上角的 x 坐标，以 MM_TEXT 客户端坐标表示。
nDstY	[in/out] 目标矩形左上角的 y 坐标，以 MM_TEXT 客户端坐标表示。
nDstWidth	[in/out] 目标矩形的宽度，以 MM_TEXT 客户端坐标表示。
nDstHeight	[in/out] 目标矩形的高度，以 MM_TEXT 客户端坐标表示。
pFrame	[in] 指向待绘制帧结构体指针

5.3 函数

5.3.1 API 初始化 / 反初始化

5.3.1.1 TUCAM_Api_Init 和 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

5.3.2 打开、关闭相机

5.3.2.1 TUCAM_Dev_Open

描述

用于开始相机，调用它之后，相机处于工作模式，可以响应其他接口的调用。调用之前要保证

相机已存在，即要在 TUCAM_Api_Init 初始化之后调用它。

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_CAMER	打开相机失败
A	
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

5.3.2.2 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	TUCAM-API 未初始化
----------	----------------

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit
TUCAM_Dev_Open

5.3.2.3 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

5.3.3 GenICam 相关接口

5.3.3.1 TUCAM_GenICam_ElementAttr

描述

获取指定节点属性参数的属性。获取的属性包含该参数的最小值、最大值、默认值和步长。

声明

```
TUCAMRET TUCAM_GenICam_ElementAttr(HDTUCAM hTUCam, PTUCAM_ELEMENT pNote, PCHAR pName, TUXML_DEVICE Xml = TU_CAMERA_XML);
```

参数

HDTUCAM hTUCam	相机的句柄
PTUCAM_ELEMENT pNote	相机属性节点信息结构体指针
PCHAR pName	相机属性节点名称
TUXML_DEVICE Xml	查询的 XML 类型，相机或者采集卡

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_GenICam_ElementAttrNext
TUCAM_GenICam_SetElementValue、TUCAM_GenICam_GetElementValue

5.3.3.2 TUCAM_GenICam_ElementAttrNext

描述

获取指定节点下一属性参数的属性。获取的属性包含该参数的最小值、最大值、默认值和步长。

声明

```
TUCAMRET TUCAM_GenICam_ElementAttrNext(HDTUCAM hTUCam, PTUCAM_ELEMENT
pNote, PCHAR pName, TUXML_DEVICE Xml = TU_CAMERA_XML);
```

参数

HDTUCAM hTUCam	相机的句柄
PTUCAM_ELEMENT pNote	相机属性节点信息结构体指针
PCHAR pName	相机属性节点名称
TUXML_DEVICE Xml	查询的 XML 类型，相机或者采集卡

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_GenICam_ElementAttr
TUCAM_GenICam_SetElementValue、TUCAM_GenICam_GetElementValue

5.3.3.3 TUCAM_GenICam_SetElementValue

描述

设置节点属性参数的属性值。

声明

```
TUCAMRET TUCAM_GenICam_SetElementValue(HDTUCAM hTUCam, PTUCAM_ELEMENT
pNote, TUXML_DEVICE Xml = TU_CAMERA_XML);
```

参数

HDTUCAM hTUCam	相机的句柄
PTUCAM_ELEMENT pNote	相机属性节点信息结构体指针
TUXML_DEVICE Xml	查询的 XML 类型，相机或者采集卡

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_GenICam_ElementAttr、TUCAM_GenICam_ElementAttrNext
TUCAM_GenICam_GetElementValue

5.3.3.4 TUCAM_GenICam_GetElementValue

描述

获取节点属性参数的属性值。

声明

TUCAMRET TUCAM_GenICam_GetElementValue(HDTUCAM hTUCam, PTUCAM_ELEMENT pNote, TUXML_DEVICE Xml = TU_CAMERA_XML);

参数

HDTUCAM hTUCam	相机的句柄
PTUCAM_ELEMENT pNote	相机属性节点信息结构体指针
TUXML_DEVICE Xml	查询的 XML 类型，相机或者采集卡

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_GenICam_ElementAttr、TUCAM_GenICam_ElementAttrNext
TUCAM_GenICam_SetElementValue

5.3.3.5 TUCAM_GenICam_BuffDataCallBack

描述

用于获取原始数据流注册回调函数，注册回调之后通过调用 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 TUCAM_GenICam_GetBuffData 函数便可获取原始数据流。

声明

```
TUCAMRET TUCAM_GenICam_BuffDataCallBack(HDTUCAM hTUCam, BUFFER_CALLBACK
cbBuffer, void *pUserContext);
```

参数

HDTUCAM hTUCam	相机的句柄
BUFFER_CALLBACK cbBuffer	获取原始数据流回调函数
void *pUserContext	用户数据回调函数注册

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
TUCAMRET_NOT_SUPPORT	设备不支持

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit
TUCAM_Dev_Open、TUCAM_Dev_Close
TUCAM_Buf_Alloc、TUCAM_Buf_Release
TUCAM_GenICam_GetBuffData
TUCAM_Cap_Start、TUCAM_Cap_Stop

5.3.3.6 TUCAM_GenICam_GetBuffData

描述

用于获取原始数据流函数，当使用 TUCAM_Buf_DataCallBack 函数。

注册回调之后通过调用 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。

声明

```
TUCAMRET TUCAM_GenICam_GetBuffData(HDTUCAM hTUCam, PCHAR *pBuffer,
PTUCAM_CB_BUFHEADER pBufInfo);
```

参数

HDTUCAM hTUCam	相机的句柄
----------------	-------

PUCHAR *pBuffer	图像缓存指针
PTUCAM_CB_BUFHEADER	图像缓存头部信息结构体指针
pBufInfo	

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_DataCallBack
TUCAM_Cap_Start、TUCAM_Cap_Stop

5.3.4 内存管理

5.3.4.1 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_VALUE_TEXT pVal	图像帧结构的指针，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

5.3.4.2 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

5.3.4.3 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

5.3.4.4 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 才会终止。

帧结构体中 uiRsdSize 用于设置需要捕获的帧数，仅触发模式有效。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。

注意：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, INT32 nTimeOut = 1000);

参数

HDTUCAM hTUCam	相机的句柄
PTUCAM_FRAME pFrame	帧结构体的指针
INT32 nTimeOut	超时退出的时间，默认 1S

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

5.3.4.5 TUCAM_Buf_CopyFrame

描述

在等待数据捕获完成之后，用于拷贝不同于 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 TUFPM_FMT_RGB888，通过该函数可以拷贝其他格式的数据（如 TUFPM_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。

注意：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

5.3.5 捕获控制

5.3.5.1 TUCAM_Cap_AnnounceBuffer

描述

用于声明和注册用户申请的内存空间，绑定用户申请的内存空间作为图像采集的缓冲区，使用 TUCAM_Cap_ClearBuffer 解绑，由用户负责管理和释放。

此接口作为 TUCAM_Buf_Alloc 的扩展接口，可以支持用户自定义缓冲区大小，每调用一次将注册一个图像缓冲区，如果要注册多个图像缓冲区需要多次调用。在 TUCAM_Cap_Start 开始捕获之前调用。

声明

```
TUCAMRET TUCAM_Cap_AnnounceBuffer (HDTUCAM hTUCam, UINT32 uiSize, void *pBuf);
```

参数

HDTUCAM hTUCam	相机的句柄
UINT32 uiSize	单帧图像缓冲区大小
void *pBuf	用户申请的缓冲区指针

错误代码

TUCAMRET_NOT_INIT	TUCAM-API 未初始化
TUCAMRET_FAILOPEN_BULKIN	注册内存失败
TUCAMRET_INVALID_CAMERA	无效的相机，相机句柄不存在时

相关接口

TUCAM_Cap_ClearBuffer
TUCAM_Buf_Alloc、TUCAM_Buf_Release

5.3.5.2 TUCAM_Cap_ClearBuffer

描述

用于反注册图像采集缓冲区，解绑用户申请的图像缓冲区内存空间，该接口不会释放用户申请的内存空间，由用户负责管理和释放。在 TUCAM_Cap_Stop 关闭捕获之后调用。

声明

```
TUCAMRET TUCAM_Cap_ClearBuffer (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_Cap_AnnounceBuffer

TUCAM_Buf_Alloc、TUCAM_Buf_Release

5.3.5.3 TUCAM_Cap_Start

描述

开始进行数据捕获。在开始捕获之前，应使用 TUCAM_Buf_alloc 准备捕获缓冲区，如果使用 TUCAM_SEQUENCE 模式将持续捕获，直到调用 TUCAM_Cap_Stop 被调用。适用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，参考 TUCAM_CAPTURE_MODES

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

5.3.5.4 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

5.3.6 文件管理

5.3.6.1 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

5.3.6.2 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

5.3.6.3 TUCAM_Rec_AppendFrame

描述

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

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

5.3.6.4 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

5.3.7 扩展控制

5.3.7.1 TUCAM_GenICam_GetRegisterValue

描述

读取寄存器的内容。

声明

```
TUCAMRET TUCAM_GenICam_GetRegisterValue(HDTUCAM hTUCam, PUCHAR pBuffer,
INT64 nAddress, INT64 nLength);;
```

参数

HDTUCAM hTUCam	相机的句柄
PUCHAR pBuffer	作为返回值使用，保存读到内存值
INT64 nAddress	待读取的内存地址
INT64 nLength	待读取的内存长度

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_GenICam_SetRegisterValue

5.3.7.2 TUCAM_GenICam_SetRegisterValue

描述

写入寄存器的内容。

声明

```
TUCAMRET TUCAM_GenICam_SetRegisterValue(HDTUCAM hTUCam, PUCHAR pBuffer,
INT64 nAddress, INT64 nLength);;
```

参数

HDTUCAM hTUCam	相机的句柄
PUCHAR pBuffer	待写入内存值
INT64 nAddress	待写入的内存地址
INT64 nLength	待写入的内存长度

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_GenICam_GetRegisterValue

5.3.7.3 TUCAM_GenICam_DSNUDataRW

描述

读取/写入 DSNU 寄存器的内容。

声明

TUCAMRET TUCAM_GenICam_DSNUDataRW(HDTUCAM hTUCam, PCHAR pTextFile, TUDATA_RW rwFlag);

参数

HDTUCAM hTUCam	相机的句柄
PCHAR pTextFile	文件路径
TUDATA_RW rwFlag	读取/写入标志

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
TUCAMRET_FAILOPEN_FILE	打开文件失败
TUCAMRET_FAILURE	读取/写入寄存器失败

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit
TUCAM_Dev_Open、TUCAM_Dev_Close

5.3.7.4 TUCAM_GenICam_PRNUDataRW

描述

读取/写入 PRNU 寄存器的内容。

声明

TUCAMRET TUCAM_GenICam_PRNUDataRW(HDTUCAM hTUCam, PCHAR pTextFile, TUDATA_RW rwFlag);

参数

HDTUCAM hTUCam	相机的句柄
PCHAR pTextFile	文件路径
TUDATA_RW rwFlag	读取/写入标志

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
TUCAMRET_FAILOPEN_FILE	打开文件失败
TUCAMRET_FAILURE	读取/写入寄存器失败

相关接口

TUCAM_Api_Init、TUCAM_Api_Uninit
TUCAM_Dev_Open、TUCAM_Dev_Close

6. 常见问题解答（FAQ）

6.1 问题排查思路

1. 基于 SDK 开发的程序有异常，建议先运行 SamplePro 软件，查看相应功能是否能够正常运行。
2. 如果 SamplePro 软件正常，而开发程序异常，建议重点排查二次开发的程序代码。
3. 如果 SamplePro 也异常，建议查看如下常见问题解决方法，看是否有帮助。
4. 如果以上排查都不能解决，建议请记录相关问题现象和保存图片数据、以及我们 SamplePro 与 SDK 的文件版本和产品版本号，联系本公司的技术支持同事获取帮助支持。

6.2 常见问题解决方法

问题 1：用 SDK 开发存图异常。

问题原因：TUCAM_FRAME 结构体中的 pBuffer 需要偏置到 usHeader 头部长度后才是真正的帧数据指针。

解决方法：uchar *pBuf = m_frame.pBuffer+m_frame.usHeader，即 pBuf 指针才是帧数据指针。

问题 2：用 TUCAM_Buf_GetData 获取到的 TUCAM_RAWIMG_HEADER 结构体帧数据偏移头部图像异常。

问题原因：TUCAM_RAWIMG_HEADER 中的 plmgData 是没有头部的，直接就是图像数据了，所以不必要偏移头部。

解决方法：直接从 plmgData 指针拿到的数据就是帧数据。

问题 3：调用 TUCAM_Buf_Alloc 后报错（0x80000204）。

问题原因：由于多次重复调用了此接口。

解决方法：此接口只调用一次即可，如果还要调需先调用 TUCAM_Buf_Release 后再调。

问题 4：设置自动色阶图像不生效。

问题原因：直方图计算没有开启，并且直方图开启需要在 CaptureStart 后才能设置生效。另外一方面注意设置参数接口是否用错了（TUCAM_Capa_SetValue、TUCAM_Prop_SetValue）。

解决方法：先开启直方图，然后设置自动色阶参数即可。

问题 5：16bit 相机为何拿到帧灰度值最大值是 255。

问题原因：我们帧数据都是按最小字节单位数据输出，16bit 帧需要两个字节拼接成一个像素，小端排列即低字节在前高字节在后。

解决方法：ushort *pBuf = (ushort*)(m_frame.pBuffer+m_frame.usHeader),接口获取到灰度值转换。

问题 6：二次开发亮度和我们软件显示的亮度不一致问题。

问题原因：我们软件是显示有效位的高八位数据，还有曝光、增益、色阶、伽马、对比度、锐度等功能参数是否一致。

解决方法：相机功能参数值需要设置一致，界面显示数据有效位需要一致。

问题 7：TUCAM_Buf_WaitForFrame 函数获取图像超时失败。

问题原因：接口获取数据超时退出了。

解决方法：可以将超时时间适当的设大。

问题 8：通过带 TUCAM_Capa_和 TUCAM_Prop_的接口设置功能不生效（只有色阶、伽马、对比度可通过带 TUCAM_Prop_的接口设置生效）。

问题原因：GeniCam 协议相机对这些自定义的接口不生效。

解决方法：需要通过带_GeniCam_接口设置功能生效。