



TUCAM-API

Development Guide



Copyright (c) 2011-2023 Tucsen Photonics Co., Ltd.

All Rights Reserved.

Catalog

1. Before Use	4
2. General	4
3. Overview	5
3.1. Layer Structure	5
3.2. Principle	6
3.3. General Rules	6
3.3.1. Handles	6
3.3.2. Functions	6
3.3.3. Capability	8
3.3.4. Property	8
3.3.5. Trigger	9
4. Programming Guides	12
4.1. Programming Environment	12
4.2. Get Started	17
4.2.1. Start & End	18
4.2.2. Capability Getting and Setting	21
4.2.3. Property Getting and Setting	23
4.2.4. Memory Management	25
4.2.5. Capture	29
4.2.6. File Management	30
4.2.7. Extension Control	34
4.2.8. Others	36
5. Reference	41
5.1. Types and Constants	41
5.1.1. TUCAMRET error code	41
5.1.2. Product Information Code- TUCAM_IDINFO	43

5.1.3. Capability Code- TUCAM_IDCAPA	44
5.1.4. Property Code- TUCAM_IDPROP	48
5.1.5. Image Stitching Code	49
5.1.6. Region Calculation Code	50
5.1.7. Capture Mode Code	50
5.1.8. Image Format Code	51
5.1.9. Register Type Code	51
5.1.10. Trigger the Exposure Time Mode Code	52
5.1.11. Edge Trigger Type Code	52
5.1.12. Triggers the Read Output Reset Code	53
5.1.13. Trigger Output Direction Code	53
5.1.14. Frame Format Code	53
5.1.15. Gain Mode Code	53
5.1.16. Channel Selection Code	53
5.1.17. Trigger Output Port Code	54
5.1.18. Trigger Output Type Code	54
5.1.19. Trigger Output Edge Code	54
5.1.20. Image Drawing Mode Code	54
5.1.21. Record Keyframe Mode Code	55
5.1.22. Image Processing Mode Code	55
5.1.23. Image Stitching Mode Code	55
5.1.24. Focus Status Code	55
5.1.25. Vendor property (vendor only)	56
5.2. Structure	56
5.2.1. Initialization	57
5.2.2. Open camera	57
5.2.3. Open image	58

5.2.4. Camera Information	58
5.2.5. Capability Attribute	59
5.2.6. Property Attribute	59
5.2.7. Capability/ Property Value Text	60
5.2.8. The Attribute of the Vendor Property	61
5.2.9. Processing Image Attribute	61
5.2.10. ROI Attribute	62
5.2.11. Area Calculation Attribute	63
5.2.12. Trigger Attribute	64
5.2.13. Trigger Output Attribute	64
5.2.14. Any Bin Attribute	65
5.2.15. Frame Structure	66
5.2.16. File Saving	67
5.2.17. Video Saving	68
5.2.18. Read and Write Registers	69
5.2.19. Image Drawing Initialization	69
5.2.20. Image Drawing Parameters (Windows only)	70
5.3. Functions	71
5.3.1. API Initialization / Deinitialization	71
5.3.2. Open and Close the Camera	72
5.3.3. Set and Get Capability	74
5.3.4. Set and Get Property	78
5.3.5. Memory Management	80
5.3.6. Capture Control	86
5.3.7. File Management	93
5.3.8. Extended Control	96

1. Before Use

This document and sample code are internal documents published by TUCSEN to enable users to create applications that use TUCSEN digital cameras. This document and sample code are publicly available for the above purposes only and do not constitute a license, transfer or any other right of the owner. All risks and consequences of using this document still depend on the user.

This document may include technical errors or typographical errors. And no damage caused by such errors or texts can be guaranteed. TUCSEN makes no commitment to update or maintain the information contained in this document currently.

All brand and product names are trademarks or registered trademarks of their respective owners. TUCSEN All rights reserved to the copyright of the documentation. No part of the document may be copied, transmitted, transcribed, stored in a retrieval system or translated into any language or computer language, in any form, or by any means, such as electronic, mechanical, electromagnetic, optical, chemical, manual or otherwise, without the prior written permission of TUCSEN.

2. General

1) TUCAM-API

TUCAM-API is an application interface for controlling cameras manufactured by TUCSEN.

TUCAM-API supports USB, CameraLink, CoaXPress, Gige data interfaces, which can automatically match interfaces and work at initialization. Because the API is designed to be easy to understand, the number of function interfaces is limited to a minimum, and the function call format is written in C.

2) SDK

The TUCAM-API Software Development Kit (hereinafter referred to as the "SDK") provides resources for developers to make host software programs. The SDK

consists of header files, library files, TUCAM-API function reference files, the series camera capability and property files, and sample code. For example, partially modifying the source code and creating completely in-dependent programs, and/or making host software or plug-in modules available to everyone.

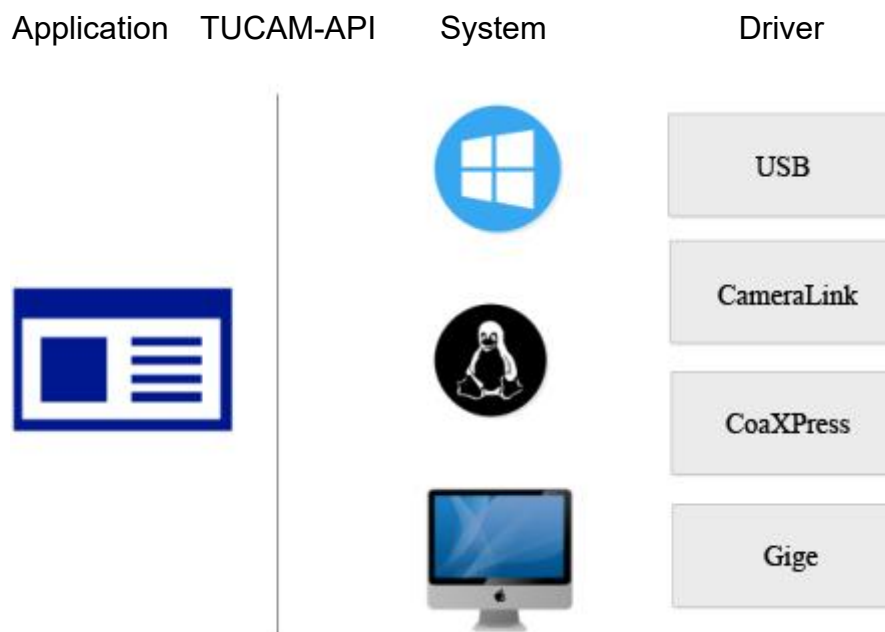
3) Description

Some extended functions are only available on certain camera models, and the values may vary for different camera models.

Values should simply be considered as guidelines, not exact values. For details, please refer to the series camera capability and property files.

3. Overview

3.1. Layer Structure



TUCSEN cameras use SDKs to connect drivers from different operating systems to control the camera and acquire image data.

3.2. Principle

The camera's bus interface and libraries are packaged as TUCAM-API, and the module layer provides more advanced TUCAM-API integration. You only need to access the TUCAM-API, and the modules can be constantly updated to access new cameras and provide new interface technologies without having to recompile your software.

The TUCAM-API does not contain a program for displaying images, because some methods of displaying images are difficult to predict, depending on the application, it is impossible to support all of these common modules. When the display program is called, it detects if the image is updated and draws the updated image. For more detailed information, see the sample source code.

3.3. General Rules

3.3.1.Handles

Name	Description
HDTUCAM	It is a handle on the API that specifies the target camera, and almost all APIs require this handle as a parameter. [TUCAM_Dev_Open] provides this handle, [TUCAM_Dev_Release] release the handle and stop the use of the device.
HDTUIMG	It is a handle that the API specifies to open images. [TUIMG_File_Open] provides this handle, [TUCAM_File_Close] closes the handle and released the image resource.

3.3.2.Functions

1) Name

All TUCAM-API functions are prefixed with [TUCAM_]. All functions of TUCAM-API are categorized by performance, and functions in the same group use the same prefix.

The following is a list of functions prefixes.

Prefix	Name	Description
TUCAM_Api_	Init, Uninit	Initialize, Deinitialization
TUCAM_Dev	Open, Close, GetInfo, GetInfoEx	Open, close camera, get information
TUCAM_Capa_	GetAttr, GetValue, SetValue, GetValueText	Get and set capabilities
TUCAM_Prop_	GetAttr, GetValue, SetValue, GetValueText	Get and set properties
TUCAM_Buf_	Alloc, Release, AbortWait, WaitForFrame, CopyFrame	Memory management
TUCAM_Cap_	SetROI, GetROI, SetTrigge, GetTrigger DoSoftwareTrigger, SetTriggerOut, GetTriggerOut, Start, Stop	Image data capture
TUCAM_File_	SaveImage, LoadProfiles, SaveProfiles, Open, Close	Operations for image files and profiles
TUCAM_Rec_	Start, AppendFrame, Stop	Video file saving interface
TUCAM_Reg_	Read, Write	Read/write camera configuration registers
TUCAM_Draw_	Init, Frame, UnInit	Drawing Image Interface (Windows)
TUCAM_Calc_	SetROI, GetROI	Read/write ROI calculation area for related functions
TUCAM_Vendor_	Config, ConfigEx, Update, Prop_GetAttr, Prop_SetValue, Prop_GetValue, Prop_GetValueText, ResetIndexFrame, WaitForIndexFrame, Buf_Attach, Buf_Detach	Vendor-controlled interfaces (not open)

2) Parameters

Except for functions such as TUCAM_Api_Init (initialization), TUCAM_Api_Uninit (deinitialization), TUCAM_Dev_Open (opening camera), etc., all other TUCAM-APIs require an HDTUCAM handle as the first parameter. Some functions also require a pointer to the struct as a parameter. In the struct parameter, [in] identifies the variable, meaning the application must set the value before the call, and [out] identifies the

variable, meaning that a value is populated when the function returns.

3) Returns

All functions of the TUCAM-API return a value defined starting with [TUCAMRET_] after execution. [TUCAMRET_SUCCESS] is the successful return of function execution, we recommend that you determine whether the return value is [TUCAMRET_SUCCESS] before proceeding to the next step during development, otherwise an error message is given to facilitate the problem finding.

3.3.3.Capability

1) What is capability in TUCAM?

[Capability] refers to the capabilities supported by the device. [TUCAM-API] defines [TUCAM_IDCAPA] capability constants, and each capability has a unique ID prefixed with [TUIDC_]. Users use the function prefixed with the [TUCAM_CAPA_] to query whether the device supports a capability, and obtain and set the capability value. Each value is processed as an integer, and each capability has its own ID, minimum, maximum, default value.

2) Multiple Capability Values

Some capabilities have multiple values, and their values have associated texts, which we call value texts. We can get value texts through [TUCAM_Capa_GetValueText].

3.3.4.Property

1) What is Property in TUCAM?

[Property] means device parameters. [TUCAM-API] defines [TUCAM_IDPROP] constants, and each property has a unique ID prefixed with [TUIDC_]. Users use the function prefixed with [TUCAM_Prop_] to obtain, set, and query device parameters. Each property is processed as a double precision floating-point type, and each

property has its own ID, minimum, maximum, default value.

2) Multiple Property Values

Some properties have multiple values, and their values have associated texts, which we call value texts. We can get value texts through [TUCAM_Prop_GetValueText].

3.3.5. Trigger

1) Capture Mode

There are two modes for capturing

Sequence mode (stream mode): it is used to capture continuous image data.

Trigger mode: The camera captures the images by the external signals. We call this option as "trigger mode", and you can call [TUCAM_Cap_SetTrigger] to configure this option. We also call the external signal as "external trigger".

2) Image Cell

It is generally two-dimensional, with vertical and horizontal directions.

Frame: it is a unit of image data. For one frame, the pixel data is aligned from left to right and from top to bottom. This is a series of image data unit.

3) Trigger Properties

Please refer to the [TUCAM_TRIGGER_ATTR] structure for Trigger Properties. Includes trigger mode, exposure mode, trigger level type, trigger delay, number of output frames. Different trigger modes support different parameters, for example, software triggering does not support setting edge trigger type, and the trigger exposure time is set by software (soft trigger has no rising edge or falling edge).

Mode	Trigger Mode (TUCAM_TRIGGER_EXP)	Edge Trigger (TUCAM_TRIGGER_EDGE)	Delay	Frame Number
Standard	•	•	•	•

Synchronization	●	●	○	○
Global	○	●	○	○
Software	○	○	○	○

●: supported ○: Not supported

4) Trigger Mode

Standard Mode: When the camera receives the signal (determined by the activation edge), it will start the image capturing of one frame or multiple frames. The number of captured frames is determined by the configuration parameters.

Synchronization Mode: When the camera receives the signal (determined by the activation edge), it will begin the exposure; after receiving the opposite signal, it will end the exposure, and start the capture of image data. The exposure and readout of each frame are fully synchronized with the external trigger signal.

Global Mode: All pixels are pre-triggered before the camera triggers, and all pixels are exposed simultaneously when the camera receives an external trigger signal. This mode is often used in scenarios where the light source can be controlled.

Soft Trigger Mode: Trigger signals are delivered via software.

5) Exposure Mode

Exposure Time: After receiving the trigger signal, the exposure time is determined by the value set by TUIDP_EXPOSURETM.

Level Width: After receiving the trigger signal, the exposure time is determined by the level width.

Note: Both options are supported in Standard mode and Global mode. Synchronization only supports Level Width. Soft trigger mode only supports Exposure Time.

6) Types of Trigger Signal

Rising Edge: The camera starts exposure when it receives a rising edge.

Falling Edge: The camera starts exposure when it receives a falling edge.

7) Trigger Delay

Input: When a trigger signal is received, you can set how long the delay is before exposure.

Output: When a output signal is received, you can set how long the delay is before the camera outputting signal.

8) Outputs

There are 3 trigger outputs: TRIG.OUT1, TRIG. OUT2 and TRIG. OUT3, corresponding to Port1, Port2 and Port3.

The three output signals are always on by default, and the camera outputs the level signal to a third-party device as its input signal. The three signals can work independently and output to different devices simultaneously.

The trigger output hardware has three pins to configure three ports, and the three ports do not interfere with each other.

9) Trigger output level

High: Always output with a high-level signal.

Low: Always output as a low-level signal.

Exposure Start: The signal output is the indicated level from the first line of exposure to the beginning of the last line.

Readout End: The signal output is the indicated level from the first line read out to the beginning of the last line.

Global Exposure: The signal output is indicated from the last line of exposure to the end of the first line (all lines are exposed).

4. Programming Guides

4.1. Programming Environment

Take Visual Studio 2013 as an example.

1) New Project

Select [MFC] when creating a project, click [OK], and then click [Next] in the dialog box, as Figure 4-1 and Figure 4-2 show.

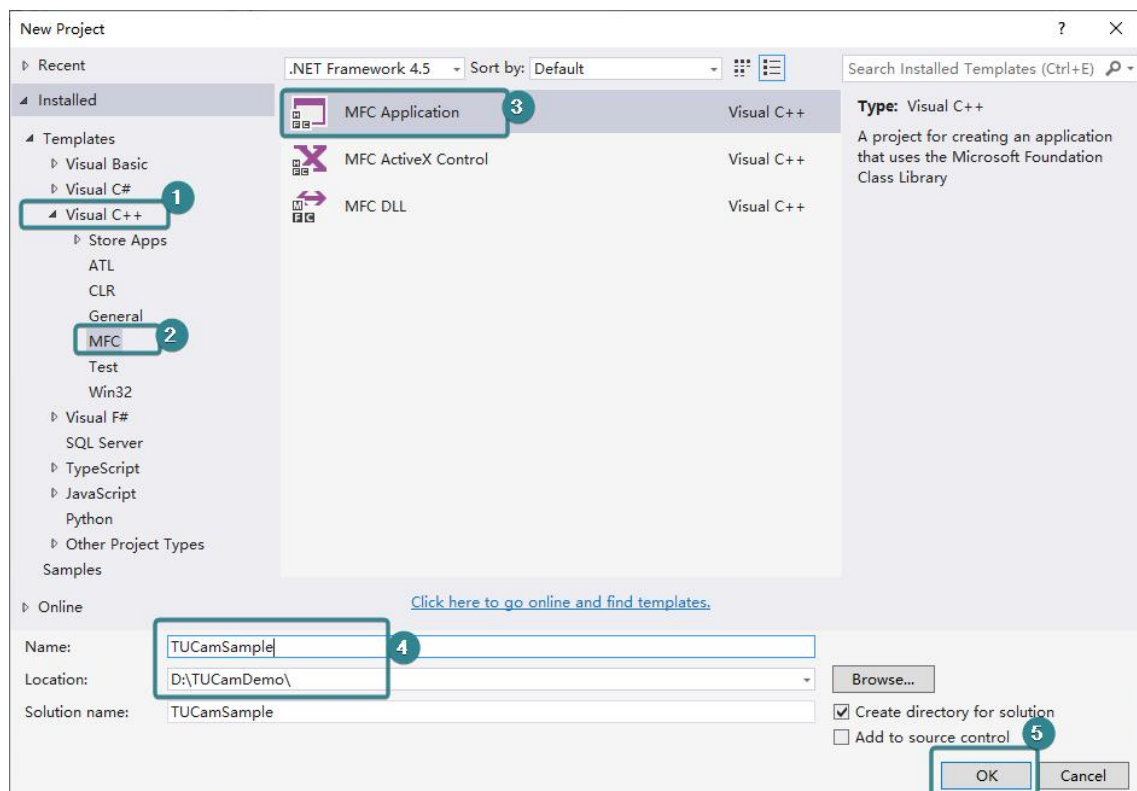


Figure 4-1

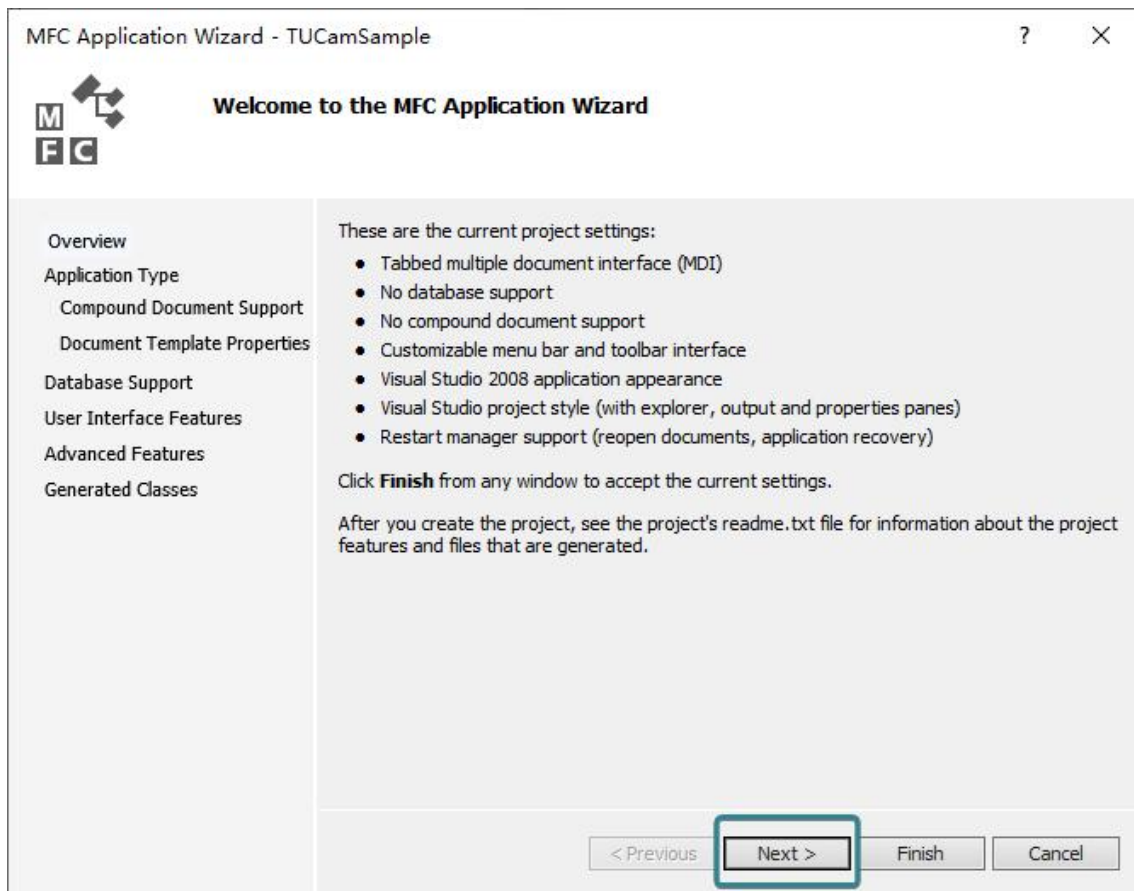


Figure 4-2

2) Application Wizard

Select [Dialog Based] on MFC Application Wizard, and then click [Finish] to complete the wizard, as Figure 4-3 shows. The MFC project is created in the specified directory.

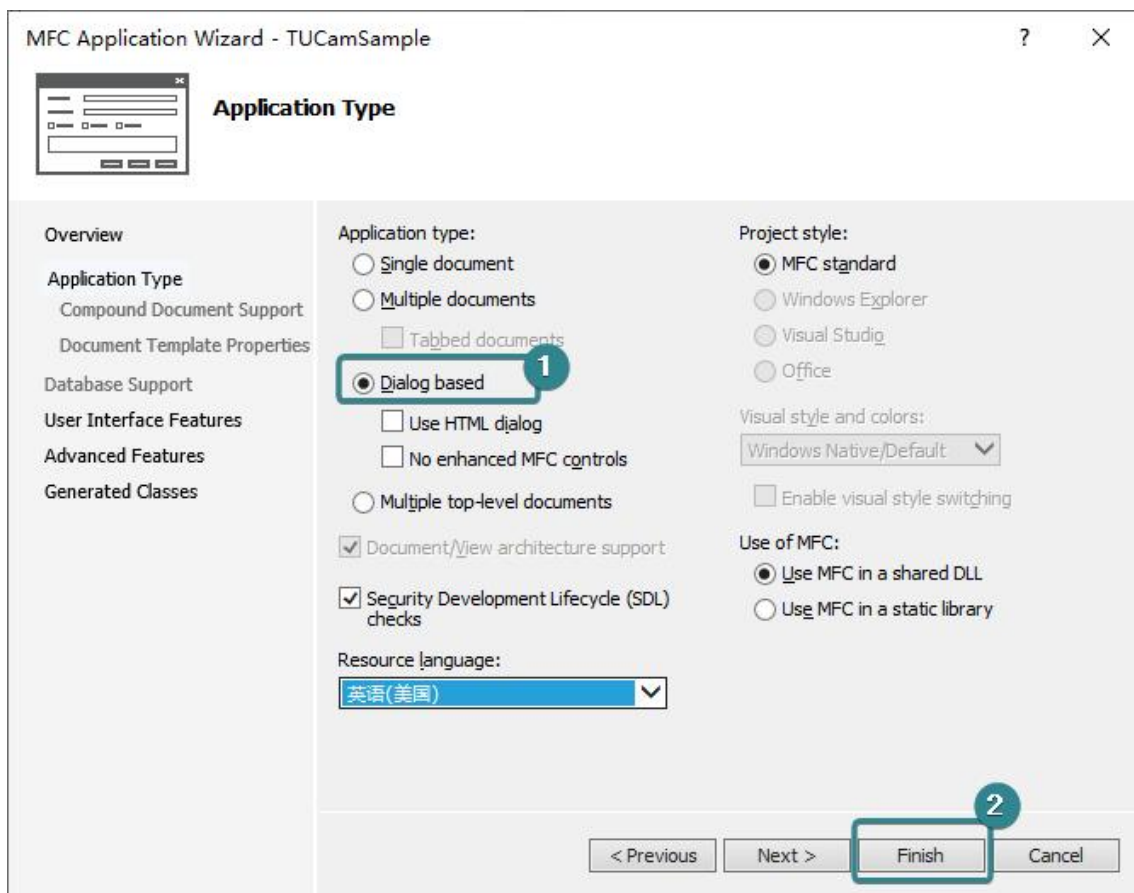


Figure 4-3

3) Reference Header Files

- ① Select the project on [Solution Explorer];
- ② Right click to open the menu, the select the [Properties];
- ③ Select [C/C++]->[General];
- ④ Fill in the directory path where TUCamApi.h is located (subject to the user installation directory), on [Additional Include Directories], as shown in the Figure 4-4:

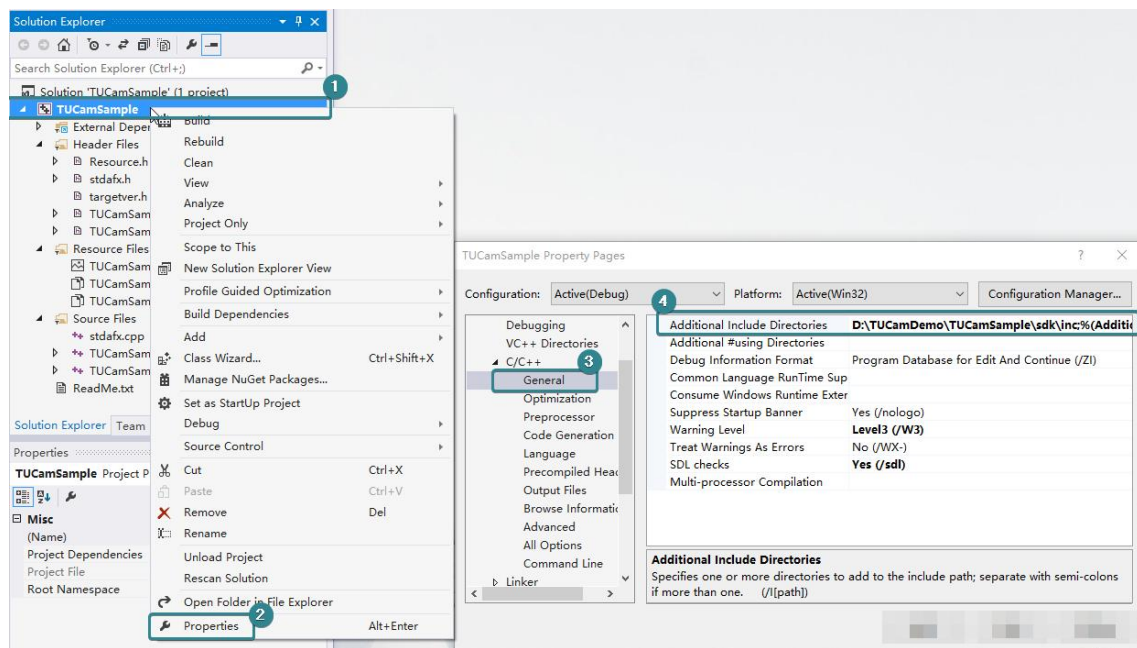


Figure 4-4

4) Configure the lib File

- ① Select [Configuration Properties]->[Linker]->[General] on TUCAMSample Property Pages;
- ② Fill in the directory path where TUCam.lib is located (subject to the user installation directory), on [Additional Library Directories], as shown in the Figure 4-5;
- ③ Select [Configuration Properties]->[Linker]->[Input] on TUCAMSample Property Pages;
- ④ Type [TUCam.lib] on [Additional Dependencies], as shown in the Figure 4-6;

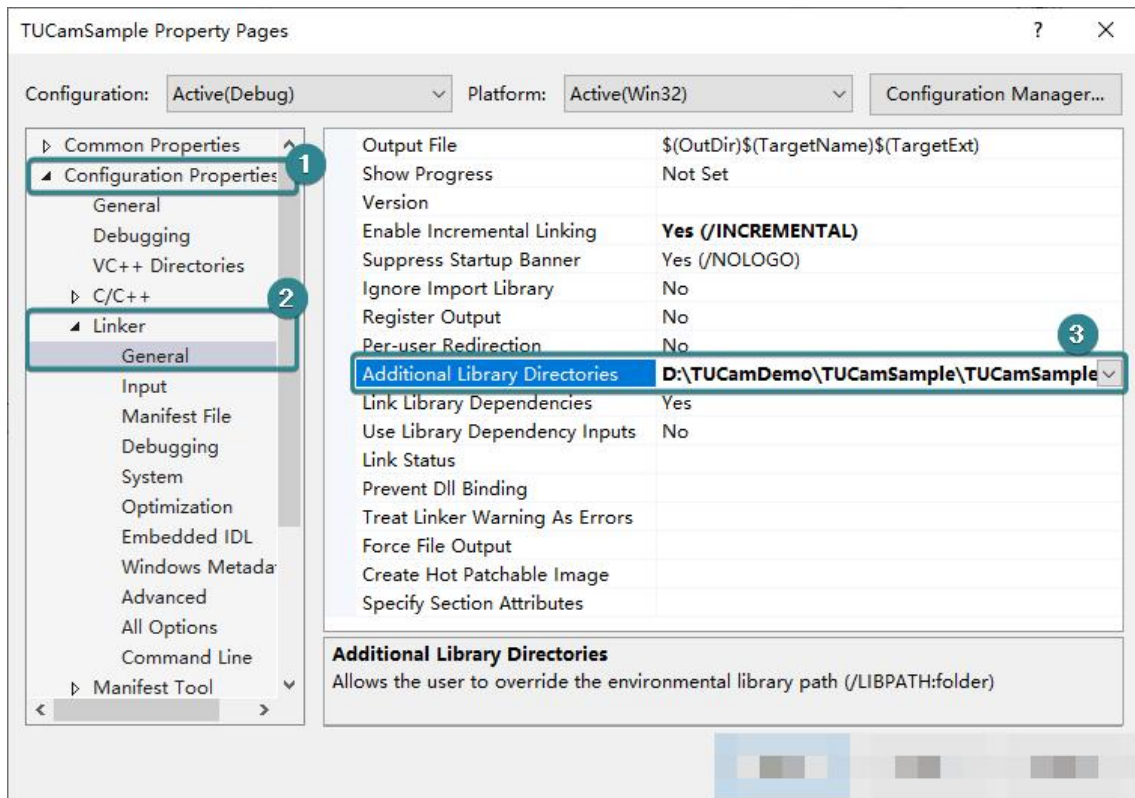


Figure 4-5

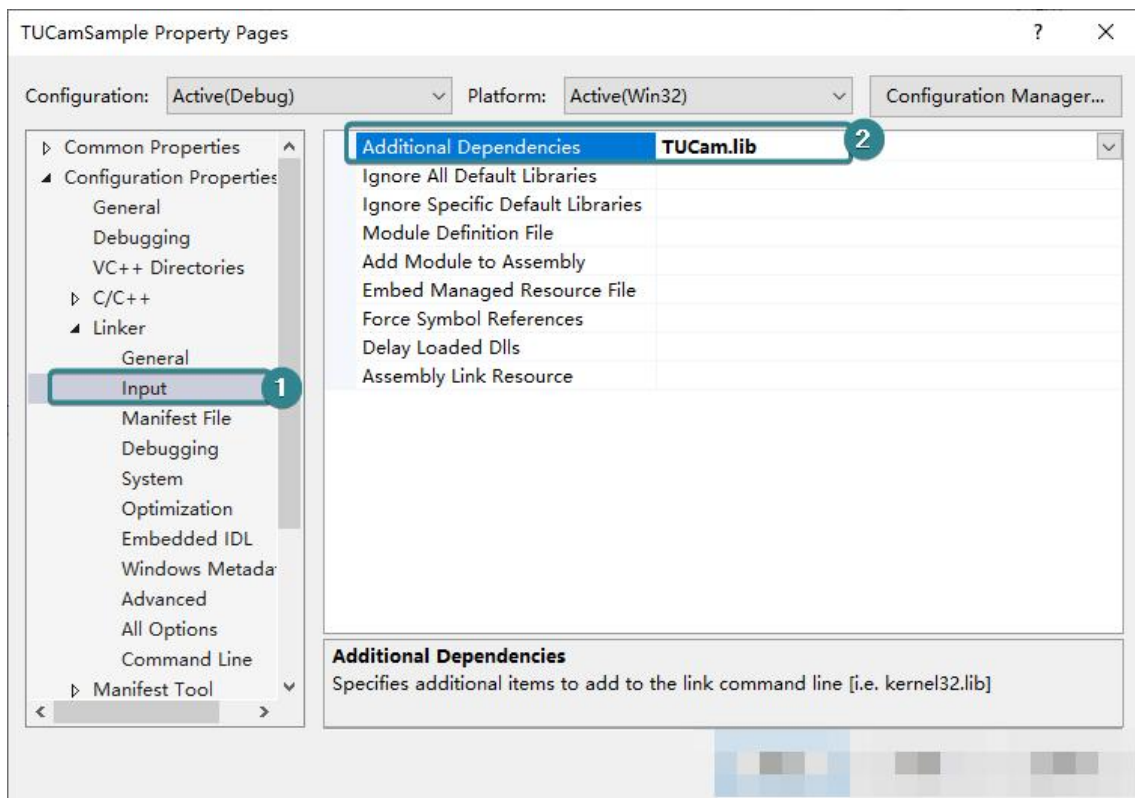


Figure 4-6

The reference header file is shown in the Figure 4-7;

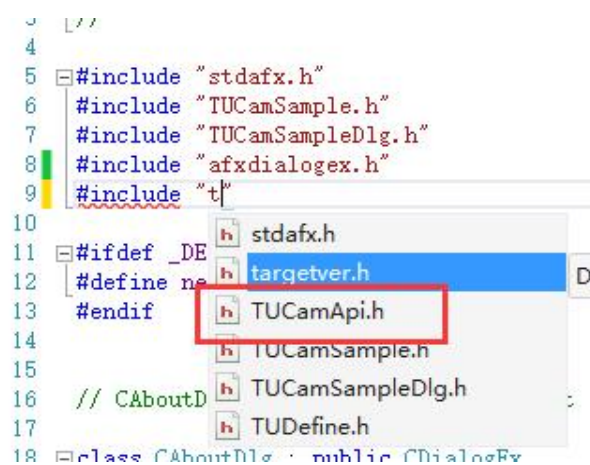


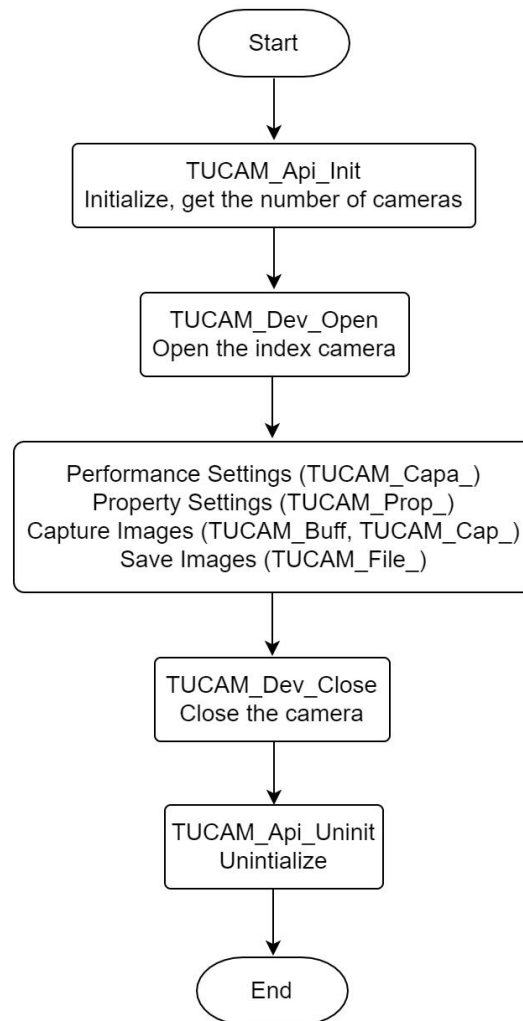
Figure 4-7

Note:

Select the x86 or x64 according to the application.

4.2. Get Started

Working Process of TUCAM-API:



4.2.1. Start & End

4.2.1.1. Calling Sequence

- 1) The driver starts initialization. When the initialization of the device handle has completed, the number of controllable cameras can be obtained.
- 2) When the application starts, it calls the API to perform initialization.
- 3) After the initialization function is used, other functions can be called for execution.

The termination function is used for closing the application. It is the function to be executed when a camera is suspended, or the resources are released and no longer

control the camera (For example, exiting the application). When the termination function is called, other functions will not be executed, until the initialization function is called again.

4.2.1.2. API Initialization

The API uses [TUCAM_Api_Init] for initial operations. This function initializes frame acquisition and controls the camera. For more information about the interface, [refer to 5.3.1](#)

4.2.1.3. Camera Initialization

The camera uses [TUCAM_Dev_Open] for initialization. This function gets the necessary camera handle HDTUCAM as an input parameter to other functions. For more information about the interface, [refer to 5.3.2](#)

4.2.1.4. Camera Information

After calling [TUCAM_Dev_Open] to open the camera, you can get the product information through the handle. For example, model, firmware version etc. For more information about the interface, [refer to 5.3.2](#), For camera information, [refer to TUCAM_IDINFO](#).

4.2.1.5. Terminate Procedures

Terminate the camera program using [TUCAM_Dev_Close]. This function is called to release the port and resources that were used for camera frame acquisition. After this function is called, the camera will no longer be controlled.

4.2.1.6. Sample code

Please refer to the following code for SDK API initialization, or refer to the [init_open] project on Samples directory. For camera information, please refer to the [get_info/get_infoEx] project on Sample directory.

```

1.  int main (int argc, char** argv)
2.  {
3.      TUCAM_INIT  itApi; // Initialize the SDK environment parameters
4.      TUCMA_OPEN  opCam; // Open camera parameters
5.
6.      itApi.pstrConfigPath = NULL;
7.      itApi.uiCamCount = 0;
8.      if (TUCAMRET_SUCCESS != TUCAM_Api_Init(&itApi))
9.      {
10.         // Failed to initialize SDK API environment
11.         return 0;
12.      }
13.
14.      if (0 == itApi.uiCamCount)
15.      {
16.         // No camera
17.         return 0;
18.      }
19.
20.      opCam.hIdxTUCam = 0;
21.      opCam.uiIdxOpen = 0;
22.
23.      if (TUCAMRET_SUCCESS != TUCAM_Dev_Open(&opCam))
24.      {
25.         // Failed to open the camera
26.         return 0;
27.      }
28.
29.         // Applications can use opCam.hIdxTUCam handle
30.
31.      TUCAM_Dev_Close(opCam.hIdxTUCam); // Close the camera
32.      TUCAM_Api_Uninit(); // uninitialize the SDK API environment
33.

```

```
34.     return 0;  
35. }  
36.
```

4.2.2. Capability Getting and Setting

4.2.2.1. Calling Sequence

Getting Capability

The [TUCAM_Capa_GetAttr] is used to get the capability by setting the ID, which includes the maximum, minimum, and default.

Getting&Setting Capability Value

[TUCAM_Capa_GetValue] and [TUCAM_Capa_SetValue] functions are used to set and get capability values. All values are processed as an integer, and some capabilities have multiple values. Setting and getting are usually done before or after the image has been captured. If the set function is called at the time of data capture, an error code [TUCAMRET] may be returned in some cases. For more information about the interface, [refer to 5.3.3](#)

Get Capability Value Text

[TUCAM_Capa_GetValueText] gets capability value text, which describes the specific meaning of each capability value.

4.2.2.2. Index

Each capability has a unique ID, refer to TUCAM_IDCAPA

Note:

If this capability ID is not supported, the error code [TUCAMRET_NOT_SUPPORT] or [TUCAMRET_INVALID_IDCAPA] will be returned.

4.2.2.3. Sample Code

Please refer to the following code, or refer to the [capa_list] project on Samples directory.

```

1.  // take [TUIDC_RESOLUTION] as an example
2.  // Get resolution range
3.  void GetResolutionRange()
4.  {
5.      TUCAM_CAPA_ATTR attrCapa;
6.      TUCAM_VALUE_TEXT valText;
7.
8.      char szRes[64] = {0};
9.      valText.nTextSize = 64;
10.     valText.pText = &szRes[0];
11.     attrCapa.idCapa = TUIDC_RESOLUTION;
12.     if (TUCAMRET_SUCCESS == TUCAM_Capa_GetAttr(opCam.hIdxTUCam, &attrCapa))
13.     {
14.         // Get the number of resolutions
15.         int nCnt = attrCapa.nValMax - attrCapa.nValMin + 1;
16.         valText.nID = TUIDC_RESOLUTION;
17.
18.         for (int i=0; i<nCnt; ++i)
19.         {
20.             valText.dbValue = i;
21.             TUCAM_Capa_GetValueText(opCam.hIdxTUCam, &valText);
22.             szRes = valText.pText;
23.             // Add resolution text to the drop-down menu
24.         }
25.     }
26. }
27.
28. // Get current resolution
29. void GetCurrentResolution()
30. {

```

```

31.     int nVal = 0;
32.
33.     if (TUCAMRET_SUCCESS == TUCAM_Capa_GetValue(opCam.hIdxTUCam, \
34.                                                  TUIDC_RESOLUTION, \
35.                                                  &nVal))
36.     {
37.         // nVal Return the current resolution index
38.     }
39. }
40.
41. // Set current resolution
42. void SetCurrentResolution(int nIdxRes)
43. {
44.     TUCAM_Capa_SetValue(opCam.hIdxTUCam, TUIDC_RESOLUTION, nIdxRes);
45. }

```

4.2.3. Property Getting and Setting

4.2.3.1. Calling Sequence

Getting Properties

The [TUCAM_Prop_GetAttr] is used to get the Properties by setting the ID, which includes the maximum, minimum, and default.

Getting&Setting Property Value

The [TUCAM_Prop_GetValue] and [TUCAM_Prop_SetValue] are used to set and get Property values. All values are processed as a double precision floating-point, and some Properties have multiple values. Setting and getting are usually done before or after the image has been captured. If the set function is called at the time of data capture, an error code [TUCAMRET] may be returned in some cases. For more information about the interface, [refer to 5.3.4](#)

Get Property Value Text

[TUCAM_Prop_GetValueText] gets value text, which describes the specific meaning of each property value.

4.2.3.2. Index

Each property has a unique ID, refer to TUCAM_IDPROP

Note:

If this property ID is not supported, the error code [TUCAMRET_NOT_SUPPORT] or [TUCAMRET_INVALID_IDPROP] will be returned.

4.2.3.3. Sample Code

Please refer to the following code, or refer to the [prop_list] project on Samples directory.

```

1.  // Take exposure as an example
2.  // Get exposure range
3.  void GetExposureTimeRange()
4.  {
5.      TUCAM_PROP_ATTR attrProp;
6.
7.      attrProp.nIdxChn = 0;    // Current channel
8.      attrProp.idProp = TUIDP_EXPOSURETM;
9.
10.     if (TUCAMRET_SUCCESS == TUCAM_Prop_GetAttr(opCam.hIdxTUCam, &attrProp))
11.     {
12.         // Exposure range
13.         attrProp.dbValMin;    // Minimum exposure
14.         attrProp.dbValMax;    // Maximum exposure
15.         attrProp.dbValDft;    // Default exposure
16.         attrProp.dbValStep;   // Exposure time step
17.     }
18. }
```

```
19.  
20. // Get current exposure  
21. void GetCurrentExposureTime()  
22. {  
23.     double dbVal = 1.0f;  
24.  
25.     if (TUCAMRET_SUCCESS == TUCAM_Prop_GetValue(opCam.hIdxTUCam, \  
26.                                                  TUIDP_EXPOSURETM, \  
27.                                                  &dbVal))  
28.     {  
29.         // dbVal returns the current exposure in ms  
30.     }  
31. }  
32.  
33. // Set current exposure  
34. void SetCurrentExposureTime(double dbVal)  
35. {  
36.     TUCAM_Prop_SetValue(opCam.hIdxTUCam, TUIDP_EXPOSURETM, dbVal);  
37. }
```

4.2.4. Memory Management

4.2.4.1. Calling Sequence

Allocation of Memory

Memory allocation [TUCAM_Buf_Alloc] must be called before [TUCAM_Cap_Start] capturing data, and memory allocation must be done again when switching to a different resolution.

Data Acquisition

[TUCAM_Buf_WaitForFrame] must be called after [TUCAM_Cap_Start] data capture to wait for the completion of data capture. And you can copy the data of current frame

in different format by [TUCAM_Buf_CopyFrame], once [TUCAM_Buf_WaitForFrame] is called again, [TUCAM_Buf_CopyFrame] will copy the data to the next frame as well.

Ending the Wait

If there is data wait and data copy, first call [TUCAM_Buf_AbortWait] to end data wait, then call [TUCAM_Cap_Stop] to stop data capture.

Memory Release

The memory release [TUCAM_Buf_Release] must be called after the [TUCAM_Cap_Stop].

4.2.4.2. Frame Structure

A structure that describes the image data of a frame. [Refer to TUCAM_FRAME.](#)

The variable [pBuffer] is a pointer to the frame data, which consists of frame header data and frame image data. The frame image data is obtained by offsetting the [usHeader] byte to the frame image data.

4.2.4.3. Sample Code

Please refer to the following code, or refer to the [wait_frame] project on Samples directory.

```
1.  TUCAM_FRAME m_frame;           // Frame Object
2.  HANDLE m_hThdGrab;             // Event handle of the capture image thread
3.  BOOL m_bLiving;                // Capture Image or not?
4.
5.  BOOL CDlgTUCam::StartCapture()
6.  {
7.      m_frame.pBuffer      = NULL;
8.      m_frame.ucFormatGet = TUFrm_FMT_RGB888; // Data format of frame (RGB888)
9.      m_frame.uiRsdSize    = 1;    // Number of frames captured at a time
```

```

10.
11.     if (TUCAMRET_SUCCESS != TUCAM_Buf_Alloc(m_opCam.hIdxTUCam,
12.         &m_frame))
13.     {
14.         return FALSE;
15.     }
16.     if (TUCAMRET_SUCCESS != TUCAM_Cap_Start(m_opCam.hIdxCam,
17.         TUCCM_SEQUENCE))
18.     {
19.         TUCAM_Buf_Release(m_opCam.hIdxTUCam);
20.         return FALSE;
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 ==
37.             TUCAM_Buf_WaitForFrame(pTUCam->m_opCam.hIdxTUCam,\
38.                                     &pTUCam->m_frame))
39.         {

```

```

39.          // pTUCam->m_frame.pBuffer returns the captured image data in the format
of TUFrm_FMT_RGB88
40.          // This data can be used to display
41.
42.          TUCAM_IMG_HEADER    frmhead;
43.          memcpy(&frmhead, pln->m_frame.pBuffer, sizeof(TUCAM_IMG_HEADER));
44.          // This data captures the headers
45.
46.          // Obtain data in other formats
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 returns the captured image data
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();    // If you call the TUCAM_Buf_WaitForFrame interface
64.
65.     WaitForSingleObject(m_hThdGrab, INFINITE);    // Waiting for the thread to exit
66.     CloseHandle(m_hThdGrab);
67.     m_hThdGrab = NULL;
68.

```

```
69.    TUCAM_Cap_Stop(m_opCam.hIdxTUCam);           // Stop data capture
70.    TUCAM_Buf_Release(m_opCam.hIdxTUCam);        // Release allocated memory
71. }
```

4.2.5. Capture

4.2.5.1. Calling Sequence

The [TUCAM_Cap_SetROI] is used to set ROI properties, and the [TUCAM_Cap_SetTrigger] is used to set trigger properties. They need to be called before data capture. If called at the time of data capture may return the TUCAMRET error code

4.2.5.2. Capture Pattern Index

Please refer to the [TUCAM_CAPTURE_MODES](#)

4.2.5.3. Sample Code

Please refer to the following code, or refer to the [roi_mode] project on Samples directory.

```
1.  // set ROI mode
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); // Sequence mode
```

```

13.
14.     // Refer to the memory management sample code for data capture
15. }
16.
17. // Set trigger mode
18. void SetTriggerMode()
19. {
20.     TUCAM_TRIGGER_ATTR tgrAttr;
21.
22.     tgrAttr.nTgrMode = TUCCM_TRIGGER_STANDARD;    // Standard trigger mode
23.     tgrAttr.nExpMode = TUCTE_EXPTM;              // Exposure mode
24.     tgrAttr.nEdgeMode= TUCTE_RISING;              // Rising edge trigger
25.     tgrAttr.nFrames  = 1;                         // The number of frame is 1
26.     tgrAttr.nDelayTm = 0;                         // Delay 0 ms
27.
28.     TUCAM_Cap_SetTrigger(m_opCam.hIdxTUCam, tgrAttr);
29.     TUCAM_Cap_Start(m_opCam.hIdxCam, TUCCM_STANDARD);    // Standard
    trigger mode
30.
31.     // Refer to the memory management sample code for data capture
32. }

```

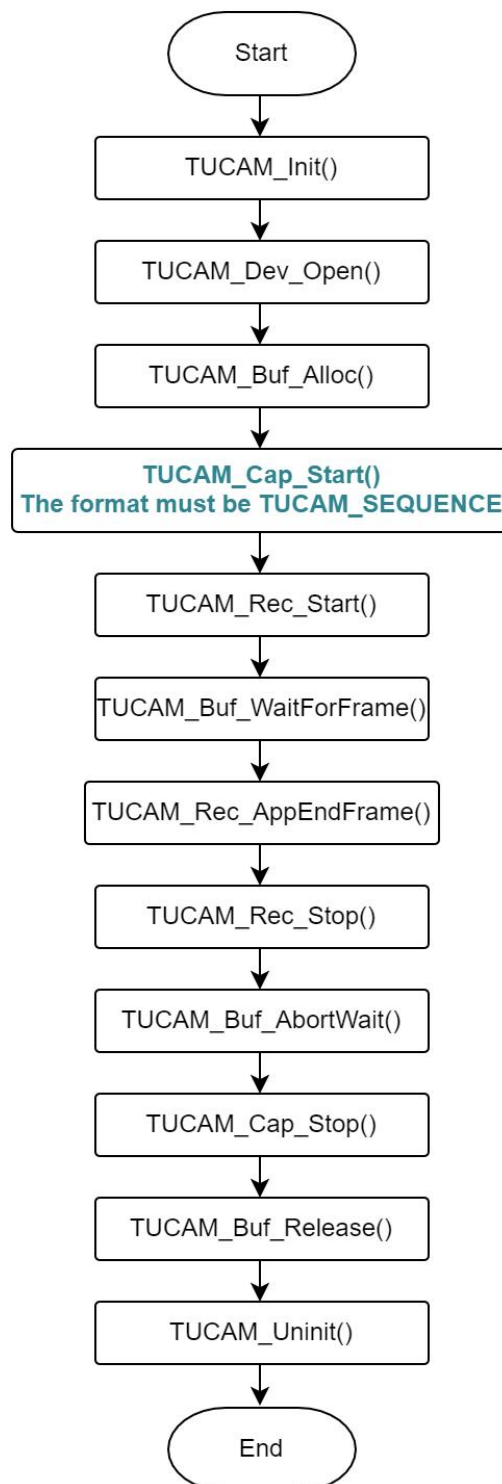
4.2.6. File Management

4.2.6.1. Calling Sequence

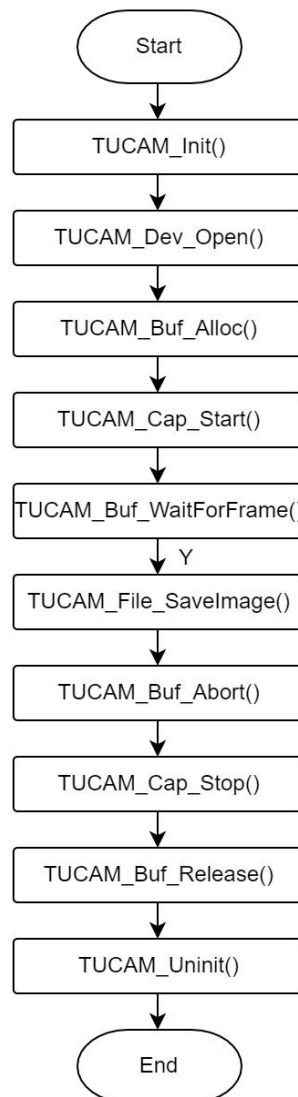
1) Process of Saving Video Stream

The [TUCAM_Rec_Start] needs to be called after the [TUCAM_Cap_Start] starts capturing data, and the capture mode [TUCCM_SEQUENCE] is necessary. The [TUCAM_Rec_AppendFrame] is used to append image data to a file during recording, the [TUCAM_Rec_Stop] is used to end the recording process, and the

[TUCAM_CAP_STOP] is used to end the acquisition.



2) Process of Saving Images



4.2.6.2. File Structures

File save structure reference [TUCAM_File_Save](#)

Video save structure reference [TUCAM_Rec_Save](#)

4.2.6.3. Sample Code

Please refer to the following code, or refer to the [save_image/save_video] project on

Samples directory

```

1.  // Save the image file
2.  void SaveImage()
3.  {
4.      m_frame.ucFormatGet = TUFRTM_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; // Save Tiff format
9.          fileSave.pFrame = &m_frame; // Pointer to the frame to be saved
10.         fileSave.pstrSavePath = "C:\\image"; // path contains filename (without extension)
11.
12.         if (TUCAMRET_SUCCESS == TUCAM_File_SaveImage(m_opCam.hIdxTUCam,
            fileSave))
13.         {
14.             // Saved image file
15.         }
16.     }
17. }
18.
19. // Save the video
20. void StartRecording()
21. {
22.     TUCAM_REC_SAVE recSave;
23.     recSave.fFps = 15.0f; // the frame rate to be saved
24.     recSave.nCodec = m_dwFccHandler;
25.     recSave.pstrSavePath = "C:\\TUVideo.avi" // Full path
26.
27.     if (TUCAMRET_SUCCESS == TUCAM_Rec_Start(m_opCam.hIdxTUcam, recSave))
28.     {
29.         // Start the video.
30.     }

```

```
31. The }
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.
```

4.2.7.Extension Control

4.2.7.1. Calling Sequence

[TUCAM_Reg_Read] and [TUCAM_Reg_Write] are used to read and write registers, they must be called after the [TUCAM_Dev_Open], if [TUCAM_Dev_Close] is called, they will not be possible to read or write registers.

4.2.7.2. Registers Structure

Please refer to the [\[TUCAM_REG_RW\]](#)

4.2.7.3. Sample Code

```

1.  // Read Registers
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.         // Get SN information
14.     }
15. }
16.
17. // Write register
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.         // Write SN to register
30.     }
31. }

```

4.2.8. Others

4.2.8.1. Cooling Temperature Settings

```

1.  // Get Temperature
2.  void GetTemperature()
3.  {
4.      double dblval = 1.0f;
5.      TUCAM_Prop_GetValue(m_opCam.hIdxTUCam, TUIDP_TEMPERATURE, &dblval);
6.  }
7.
8.  // Set Temperature
9.  void SetTemperature()
10. {
11.     int nVal = 40; // -10°C (40 - 50 = -10)
12.     TUCAM_Prop_SetValue(m_opCam.hIdxTUCam, TUIDP_TEMPERATURE, nVal);
13. } // Set the temperature to the SDK interface, the value 0~100 corresponds to the actual
    temperature -50°C~50°C
14.

```

4.2.8.2. Exposure Settings

```

1.  // Auto exposure
2.  TUCAM_Capa_SetValue(m_opCam.hIdxTUCam, TUIDC_ATEXPOSURE, 1);
3.
4.  // once exposure
5.  TUCAM_Capa_SetValue(m_opCam.hIdxTUCam, TUIDC_ATEXPOSURE, 2);
6.  // Waiting
7.  int nVal = 0;
8.  TUCAM_Capa_GetValue(m_opCam.hIdxTUCam, TUIDC_ATEXPOSURE, &nVal);
9.  // once exposure complete
10. if (0 == nVal)
11.
12. // Default is right exposure mode (only supported by MIchrome), which cannot set auto

```

```

    exposure target values (TUIDP_BRIGHTNESS)
13. TUCAM_Capa_SetValue(m_opCam.hIdxTUCam, TUIDC_ATEXPOSURE_MODE, 0);
14.
15. // Set the centered exposure mode, which can set auto exposure target values
16. TUCAM_Capa_SetValue(m_opCam.hIdxTUCam, TUIDC_ATEXPOSURE_MODE, 3);
17. TUCAM_Prop_SetValue(m_opCam.hIdxTUCam, TUIDP_BRIGHTNESS, 128);
18.

```

4.2.8.3. Auto Level Setting

```

1. // Setting Mode (0: close,1: auto min(left level),2: auto max(right level),3: auto min and max
   (left and right))
2. int nVal = 0;
3.
4. If ( 0 == nVal )
5.     TUCAM_Capa_SetValue(m_opCam.hIdxTUCam, TUIDC_HISTC, 0); // disable the
   histogram statistics
6. else
7.     TUCAM_Capa_SetValue(m_opCam.hIdxTUCam, TUIDC_HISTC, 1); // enable the
   histogram statistics
8.
9. // Set the Auto Levels mode
10. TUCAM_Capa_SetValue(m_opCam.hIdxTUCam, TUIDC_ATLEVELS, nVal);
11.

```

4.2.8.4. Converting to cvMat format in OpenCV

```

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.

```

4.2.8.5. Converting Bitmap Format in C#

```

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.uilmgSize + m_frame.usHeader);
6.      byte[] pBuf = new byte[nSize];
7.      Marshal.Copy(m_frame.pBuffer, buffer, 0, nSize);
8.      // Offset header data
9.      Buffer.BlockCopy(buffer, (int)(m_frame.usHeader), buffer, 0,
    (int)(m_frame.uilmgSize));
10.
11.     // Convert to 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.     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.        // This data is Bitmap (in C# ), which can be converted to an image
27.        Bitmap bmpDraw = bitmap.Clone(new Rectangle(0, 0,
        bitmap.Width,bitmap.Height),bitmap;)
28.    }
29. }
30.

```

4.2.8.6. The average gray value

```

1.  if(TUCAMRET_SUCCESS == TUCAM_Buf_WaitForFrame(m_opCam.hIdxTUCam,
    &m_frame))
2.  {
3.      // Calculate the average gray value of the full-frame area or ROI
4.      bool isRoi = false;
5.
6.      // ROI parameters
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.     // Calculate the average
50.     avg = sum * 1.0 / pixels;
51. }
52.

```

5. Reference

5.1. Types and Constants

5.1.1.TUCAMRET error code

Error code constants are prefixed with [TUCAMRET_], and are classified by types as initialization error, state error, wait error, and Call error, camera or interface error.

Name	Error Code	Description
TUCAMRET_SUCCESS	0x00000001	No error, success code, the application should check the value is positive
TUCAMRET_RECEIVE_FINISH	0x00000002	No errors, frame information received by vendor
TUCAMRET_EXTERNAL_TRIGGER	0x00000003	No error, external trigger signal received
TUCAMRET_FAILURE	0x80000000	Failed to call the API interface
Initialization Errors		
TUCAMRET_NO_MEMORY	0x80000101	Not enough memory
TUCAMRET_NO_RESOURCE	0x80000102	Not enough resources (not including memory)
TUCAMRET_NO_MODULE	0x80000103	No supported sub-modules
TUCAMRET_NO_DRIVER	0x80000104	No supported drivers
TUCAMRET_NO_CAMERA	0x80000105	No camera
TUCAMRET_NO_GRABBER	0x80000106	No image taken
TUCAMRET_NO_PROPERTY	0x80000107	No substitute for the property ID
TUCAMRET_FAILOPEN_CAMERA	0x80000110	Failed to open the camera
TUCAMRET_FAILOPEN_BULKIN	0x80000111	Failed to open input endpoint for bulk transfer (USB interface)
TUCAMRET_FAILOPEN_BULKOUT	0x80000112	Failed to open output endpoint for bulk transfer (USB interface)
TUCAMRET_FAILOPEN_CONTROL	0x80000113	Failed to open the control endpoint
TUCAMRET_FAILCLOSE_CAMERA	0x80000114	Failed to close the camera
TUCAMRET_FAILOPEN_FILE	0x80000115	Failed to open file
TUCAMRET_FAILOPEN_CODEC	0x80000116	Failed to open encoder

TUCAMRET_FAILOPEN_CONTEXT	0x80000117	Failed to open the context
State Errors		
TUCAMRET_INIT	0x80000201	API requires initialization
TUCAMRET_BUSY	0x80000202	API is busy
TUCAMRET_NOT_INIT	0x80000203	API is not initialized
TUCAMRET_EXCLUDED	0x80000204	Some resources are used exclusively
TUCAMRET_NOT_BUSY	0x80000205	API is not busy
TUCAMRET_NOT_READY	0x80000206	API is not in ready
Waiting Errors		
TUCAMRET_ABORT	0x80000207	Aborted
TUCAMRET_TIMEOUT	0x80000208	Timeout
TUCAMRET_LOSTFRAME	0x80000209	Frame loss
TUCAMRET_MISSFRAME	0x8000020A	Frame loss due to underlying driver issues
TUCAMRET_USB_STATUS_ERROR	0x8000020B	USB state error
Call Errors		
TUCAMRET_INVALID_CAMERA	0x80000301	Invalid camera
TUCAMRET_INVALID_HANDLE	0x80000302	Invalid camera handle
TUCAMRET_INVALID_OPTION	0x80000303	Invalid configuration values
TUCAMRET_INVALID_IDPROP	0x80000304	Invalid Property ID
TUCAMRET_INVALID_IDCAPA	0x80000305	Invalid capability ID
TUCAMRET_INVALID_IDPARAM	0x80000306	Invalid parameter ID
TUCAMRET_INVALID_PARAM	0x80000307	Invalid parameters
TUCAMRET_INVALID_FRAMEIDX	0x80000308	Invalid frame sequence number
TUCAMRET_INVALID_VALUE	0x80000309	Invalid value
TUCAMRET_INVALID_EQUAL	0x8000030A	Equal values, invalid parameters
TUCAMRET_INVALID_CHANNEL	0x8000030B	The Property ID specifies the channel, but the channel is invalid
TUCAMRET_INVALID_SUBARRAY	0x8000030C	Invalid subarray value
TUCAMRET_INVALID_VIEW	0x8000030D	Invalid display window handle
TUCAMRET_INVALID_PATH	0x8000030E	Invalid file path
TUCAMRET_INVALID_IDVPROP	0x8000030F	Invalid vendor Property ID
TUCAMRET_NO_VALUETEXT	0x80000310	Property has no value for the text

TUCAMRET_OUT_OF_RANGE	0x80000311	Value out of range
TUCAMRET_NOT_SUPPORT	0x80000312	The imager does not support capability or Property
TUCAMRET_NOT_WRITABLE	0x80000313	Properties are not writable
TUCAMRET_NOT_READABLE	0x80000314	Properties are not readable
TUCAMRET_WRONG_HANDSHAKE	0x80000410	The error occurs when getting the error code
TUCAMRET_NEWAPI_REQUIRED	0x80000411	Old API is not supported, only new API is supported
TUCAMRET_ACCESSDENY	0x80000412	Access denied (probably not enough permissions)
TUCAMRET_NO_CORRECTIONDATA	0x80000501	There is no color correction data
TUCAMRET_INVALID_PRFSETS	0x80000601	Invalid profile setting name
TUCAMRET_INVALID_IDPPROP	0x80000602	Invalid property ID
TUCAMRET_DECODE_FAILURE	0x80000701	Failed to decode
TUCAMRET_COPYDATA_FAILURE	0x80000702	Failed to copy data
TUCAMRET_ENCODE_FAILURE	0x80000703	Failed to code
TUCAMRET_WRITE_FAILURE	0x80000704	Failed to write
Camera or Bus Error		
TUCAMRET_FAIL_READ_CAMERA	0x83001001	Failed to read from camera
TUCAMRET_FAIL_WRITE_CAMERA	0x83001002	Failed to write camera
TUCAMRET_OPTICS_UNPLUGGED	0x83001003	Optical parts have been removed, please check it

5.1.2. Product Information Code- TUCAM_IDINFO

Name	Code	Description
TUIDI_BUS	0x00	USB interface type. 0x200 and 0x210 correspond to USB2.0.
TUIDI_VENDOR	0x01	Manufacturer ID
TUIDI_PRODUCT	0x02	Product ID
TUIDI_VERSION_API	0x03	TUCAM-API interface version

TUIDI_VERSION_FRMW	0x04	Current camera firmware version
TUIDI_VERSION_FPGA	0x06	Current camera FPGA version
TUIDI_VERSION_DRIVER	0x07	Current camera driver version
TUIDI_TRANSFER_RATE	0x08	Transmission rate
TUIDI_CAMERA_MODEL	0x09	Current camera model, string type (e.g. Dhyana 400BSI V3)
TUIDI_CURRENT_WIDTH	0x0A	Image data width (must be used with [TUCAM_Dev_GetInfoEx], and called after [TUCAM_Buf_Alloc])
TUIDI_CURRENT_HEIGHT	0x0B	Image data height (must be used with [TUCAM_Dev_GetInfoEx], and called after [TUCAM_Buf_Alloc])
TUIDI_CAMERA_CHANNELS	0x0C	The number of data channels. Color camera:3 Mono:1
TUIDI_BCDDEVICE	0x0D	BCD Code
TUIDI_TEMPALARMFLAG	0x0E	Temperature Warning
TUIDI_UTCTIME	0x0F	Get UTC (Universal Time, Universal Standard Time)
TUIDI_LONGITUDE_LATITUDE	0x10	Get latitude and longitude
TUIDI_WORKING_TIME	0x11	Get working time
TUIDI_FAN_SPEED	0x12	Get fan speed
TUIDI_FPGA_TEMPERATURE	0x13	Get FPGA temperature
TUIDI_PCBA_TEMPERATURE	0x14	Get PCBA temperature
TUIDI_ENV_TEMPERATURE	0x15	Get ambient temperature
TUIDI_DEVICE_ADDRESS	0x16	USB device address
TUIDI_USB_PORT_ID	0x17	USB device port number
TUIDI_ENDINFO	0x18	Product information ID end bit

5.1.3.Capability Code- TUCAM_IDCAPA

Name	Code	Description	R/W
TUIDC_RESOLUTION	0x00	Resolution	R/W

TUIDC_PIXELCLOCK	0x01	Pixel clock	R/W
TUIDC_BITOFDEPTH	0x02	Data bit (8-bit, 16-bit)	R/W
TUIDC_ATEXPOSURE	0x03	Auto exposure(0: disable 1:enable)	R/W
TUIDC_HORIZONTAL	0x04	Horizontal mirror flip(0: disable 1:enable)	R/W
TUIDC_VERTICAL	0x05	Vertical mirror flip(0: disable 1:enable)	R/W
TUIDC_ATWBALANCE	0x06	Auto White Balance (supported by color cameras) (0: disable 1: enable)	R/W
TUIDC_FAN_GEAR	0x07	Fan gears (supported by cooling cameras)	R/W
TUIDC_ATLEVELS	0x08	Auto level [0-3] (0:disable 1: LevelL 2:LevelR 3: enable)	R/W
TUIDC_SHIFT	0x09	Display shift [supported by 16-bit], select 8 consecutive bits (15~8, 14~7, 13~6, 12~5, 11~4, 10~3, 9~2, 8~1, 7~0)	R/W
TUIDC_HISTC	0x0A	Enable the histogram statistics	R/W
TUIDC_CHANNELS	0x0B	Current Channel Index (refer to TUCHN_SELECT, color cameras only)	R/W
TUIDC_ENHANCE	0x0C	Enable image enhancement	R/W
TUIDC_DFTCORRECTION	0x0D	Bad spot correction (supported by some models) (0: no correction 1: calculation 2: correction)	R/W
TUIDC_ENABLEDENOISE	0x0E	Enable noise reduction, affected by noise reduction level (Refer to TUIDP_NOISELEVEL)	R/W
TUIDC_FLTCORRECTION	0x0F	Flat field correction (0: off 1: get data 2: calculation 3: correction)	R/W
TUIDC_RESTARTLONGTM	0x10	Restart long exposure(supported by CCD)	R/W
TUIDC_DATAFORMAT	0x11	Data format (supported by some models) (0: YUV 1:RAW)	R/W
TUIDC_DRCORRECTION	0x12	Dynamic range correction	R/W
TUIDC_VERCORRECTION	0x13	Vertical mirror correction (correction data is displayed vertically, the default value is 1 in Windows systems)	R/W
TUIDC_MONOCHROME	0x14	Enable monochrome	R/W
TUIDC_BLACKBALANCE	0x15	Enable black balance	R/W

TUIDC_IMGMODESELECT	0x16	Select image mode(enable CMS)	R/W
TUIDC_CAM_MULTIPLE	0x17	Multiple camera settings (Set the number of cameras that capture data at the same time)	RO
TUIDC_ENABLEPOWEEFREQUENCY	0x18	Disable flicker enable (0:50Hz 1:60Hz)	R/W
TUIDC_ROTATE_R90	0x19	Enable rotate 90 degrees clockwise	R/W
TUIDC_ROTATE_L90	0x1A	Enable rotate 90 degrees counterclockwise	R/W
TUIDC_NEGATIVE	0x1B	Enable negative	R/W
TUIDC_HDR	0x1C	Enable HDR	R/W
TUIDC_ENABLEIMGPRO	0x1D	Enable image process	R/W
TUIDC_ENABLELED	0x1E	Enable camera LED	R/W
TUIDC_TUIDC_ENABLETIMESTAMP	0x1F	Enable timestamp	R/W
TUIDC_ENABLEBLACKLEVEL	0x20	Enable black level	R/W
TUIDC_ATFOCUS	0x21	Enable auto focus (0: Manual 1: Auto 2: Once focus)	R/W
TUIDC_ATFOCUS_STATUSES	0x22	State of auto focus (refer to TUFOCUS_STATUS)	R/W
TUIDC_PGAGAIN	0x23	Sensor gain (supported by some model, which sensor supports either high-gain or low-gain modes)	R/W
TUIDC_ATEXPOSURE_MODE	0x24	Exposure mode (0:centered exposure 3: right exposure, target is supported)	R/W
TUIDC_BINNING_SUM	0x25	Sum binning mode (Processing through software)	R/W
TUIDC_BINNING_AVG	0x26	Average binning mode (Processing through software)	R/W
TUIDC_FOCUS_C_MOUNT	0x27	C-Mount focusing mode (0: Standard 1: C-mount)	R/W
TUIDC_ENABLEPI	0x28	Enable heating sheet	R/W
TUIDC_ATEXPOSURE_ST	0x29	Auto exposure state (0: Processing 1:	RO

ATUS		Completed)	
TUIDC_ATWBALANCE_ST ATUS	0x2A	Auto white balance (0: Processing 1: Completed)	RO
TUIDC_TESTIMGMODE	0x2B	Test image mode (multiple test images, independent of the capture scene)	R/W
TUIDC_SENSORRESET	0x2C	Reset sensor (Once)	WO
TUIDC_PGAAHIGH	0x2D	High gain of the sensor	R/W
TUIDC_PGALOW	0x2E	Low gain of the sensor	R/W
TUIDC_PIXCLK1_EN	0x2F	Enable pixel clock 1	R/W
TUIDC_PIXCLK2_EN	0x30	Enable pixel clock 2	R/W
TUIDC_ATLEVELGEAR	0x31	Automatic levels gear	R/W
TUIDC_ENABLEDSNU	0x32	Enable dark signal non-uniformity(DSNU)	R/W
TUIDC_ENABLEOVERLAP	0x33	Enable exposure time overlap mode	R/W
TUIDC_CAMSTATE	0x34	Camera states	R/W
TUIDC_ENABLETRIOUT	0x35	Enable trigger output	R/W
TUIDC_ROLLINGSCANMO DE	0x36	Enable rolling scan mode	R/W
TUIDC_ROLLINGSCANLTD	0x37	Line delay time of rolling scan	R/W
TUIDC_ROLLINGSCANSLI T	0x38	Interval height of rolling scan	R/W
TUIDC_ROLLINGSCANDIR	0x39	Rolling scan direction (0: down 1: up 2: bottom-up cycle	R/W
TUIDC_ROLLINGSCANRE SET	0x3A	Reset rolling scan direction	R/W
TUIDC_ENABLETEC	0x3B	Enable TEC(cooling)	R/W
TUIDC_ENABLEBLC	0x3C	Enable backlight compensation	R/W
TUIDC_ENABLETHROUGH FOG	0x3D	Enable electronic fog penetration(with grades) Image processing	R/W
TUIDC_ENABLEGAMMA	0x3E	Enable gamma	R/W
TUIDC_ENABLEFILTER	0x3F	Enable filter	R/W
TUIDC_ENABLEHLC	0x40	Enable high light suppression	R/W
TUIDC_CAMPARASAVE	0x41	Save camera parameter	R/W
TUIDC_CAMPARALOAD	0x42	Load camera parameter	R/W

TUIDC_ENABLEISP	0x43	Enable camera ISP	R/W
TUIDC_ENDCAPABILITY	0x44	Camera performance ID end bit	R/W

5.1.4.Property Code- TUCAM_IDPROP

Name	Code	Description	R/W
TUIDP_GLOBALGAIN	0x00	Global gain	R/W
TUIDP_EXPOSURETM	0x01	Exposure time (ms)	R/W
TUIDP_BRIGHTNESS	0x02	Target value of auto exposure (Supported by right exposure)	R/W
TUIDP_BLACKLEVEL	0x03	Black level	R/W
TUIDP_TEMPERATURE	0x04	Camera (sensor) temperature	R/W
TUIDP_SHARPNESS	0x05	Sharpness level	R/W
TUIDP_NOISELEVEL	0x06	Noise reduction level	R/W
TUIDP_HDR_KVALUE	0x07	K value of HDR (supported by sCMOS cameras)	R/W
Image Process			
TUIDP_GAMMA	0x08	Gamma	R/W
TUIDP_CONTRAST	0x09	Contrast	R/W
TUIDP_LFTLEVELS	0x0A	Left levels	R/W
TUIDP_RGTLEVELS	0x0B	Right levels	R/W
TUIDP_CHNLGAIN	0x0C	Channel gain (supported by color cameras)	R/W
TUIDP_SATURATION	0x0D	Saturation (supported by color cameras)	R/W
TUIDP_CLRTEMPERATURE	0x0E	Color temperature	R/W
TUIDP_CLRMATRIX	0x0F	Color matrix	R/W
TUIDP_DPCLEVEL	0x10	Dead pixel correction	R/W
TUIDP_BLACKLEVELHG	0x11	Black level high gain	R/W
TUIDP_BLACKLEVELLG	0x12	Black level low gain	R/W
TUIDP_POWEEFREQUENCY	0x13	Anti-strobe (50HZ OR 60HZ)	R/W
TUIDP_HUE	0x14	hue	R/W
TUIDP_LIGHT	0x15	light	R/W
TUIDP_ENHANCE_STRENGTH	0x16	enhance	R/W

TUIDP_NOISELEVEL_3D	0x17	3D noise reduction	R/W
TUIDP_FOCUS_POSITION	0x18	Focus position	R/W
TUIDP_FRAME_RATE	0x19	Frame rate	R/W
TUIDP_START_TIME	0x1A	Start time of GPS	R/W
TUIDP_FRAME_NUMBER	0x1B	Frame number of GPS	R/W
TUIDP_INTERVAL_TIME	0x1C	Interval time of GPS	R/W
TUIDP_GPS_APPLY	0x1D	Enable GPS	R/W
TUIDP_AMB_TEMPERATURE	0x1E	Ambient temperature (from external device)	R/W
TUIDP_AMB_HUMIDITY	0x1F	Ambient humidity (from external device)	R/W
TUIDP_AUTO_CTRLTEMP	0x20	Enable auto temperature control	R/W
TUIDP_AVERAGEGRAY	0x21	Set average grayscale value	R/W
TUIDP_AVERAGEGRAYTHD	0x22	Average gray value threshold	R/W
TUIDP_ENHANCETHD	0x23	Set enhanced threshold	R/W
TUIDP_ENHANCEPARA	0x24	Set enhanced parameters	R/W
TUIDP_EXPOSUREMAX	0x25	Set the maximum exposure time	R/W
TUIDP_EXPOSUREMIN	0x26	Set the minimum exposure time	R/W
TUIDP_GAINMAX	0x27	Set the maximum gain	R/W
TUIDP_GAINMIN	0x28	Set the minimum gain	R/W
TUIDP_THROUGHFOGPARA	0x29	Set fog penetration parameters	R/W
TUIDP_ENDPROPERTY	0x2A	Property ID end bit	R/W

5.1.5. Image Stitching Code

Name	Code	Description
TUIDPP_EDF_QUALITY	0x00	Extend depth of field quality (0: Low 1: Medium 2: High)
TUIDPP_STITCH_SPEED	0x01	Stitching speed (0: Fast 1: Normal)
TUIDPP_STITCH_BGC_RED	0x02	Red channel values for image stitching background
TUIDPP_STITCH_BGC_GREEN	0x03	Green channel values for image stitching background
TUIDPP_STITCH_BGC_BLUE	0x04	Blue channel values for image stitching background
TUIDPP_STITCH_VALID	0x05	Invalid image stitching result(read-only)

TUIDPP_STITCH_AREA_X	0x06	Current X coordinate value of image stitching result (read-only)
TUIDPP_STITCH_AREA_Y	0x07	Current Y coordinate value of image stitching result (read-only)
TUIDPP_STITCH_NEXT_X	0x08	Next X coordinate value of image stitching result (read-only)
TUIDPP_STITCH_NEXT_Y	0x09	Next Y coordinate value of image stitching result (read-only)
TUIDPP_ENDPPROPERTY	0x0A	Image stitching property ID end bit

5.1.6.Region Calculation Code

Name	Code	Description
TUIDCR_WBALANCE	0x00	Calculate area white balance
TUIDCR_BBALANCE	0x01	Calculate area black balance
TUIDCR_BOFFSET	0x02	Calculate area black balance offset
TUIDCR_FOCUS	0x03	Calculate area focus
TUIDCR_EXPOSURETM	0x04	Calculate area exposure time
TUIDCR_END	0x05	Calculate area ID end bit

5.1.7.Capture Mode Code

Name	Code	Description
TUCCM_SEQUENCE	0x00	Sequential mode (streaming mode), which captures continuous image data
TUCCM_TRIGGER_STANDARD	0x01	Standard trigger mode
TUCCM_TRIGGER_SYNCHRONOUS	0x02	Synchronous trigger mode
TUCCM_TRIGGER_GLOBAL	0x03	Global trigger mode
TUCCM_TRIGGER_SOFTWARE	0x04	Software trigger mode
TUCCM_TRIGGER_GPS	0x05	GPS trigger mode

5.1.8. Image Format Code

Name	Code	Description
TUFMT_RAW	0x01	Save the image in RAW format. An image file format customized by Tucsen (contains header + image data)
TUFMT_TIF	0x02	Save images in TIF format. Supports different color modes, paths, transparency, and channels, and is the most common format for printing. Advantages: Saves rich image layers and details, and the image quality is lossless. Disadvantages: Takes up a lot of storage space.
TUFMT_PNG	0x04	Save the image in PNG format. Advantages: Supports high-level lossless compression and transparent backgrounds. Disadvantages: poor compatibility with older browsers and software.
TUFMT_JPG	0x08	Save the image in JPG format. Advantages: compresses the image to a small storage space, better retention of color information. Disadvantage: Lossy compression will reduce the quality of image data.
TUFMT_BMP	0x10	Save the image in BMP format. Standard image format for Windows systems, supporting a wide range of applications. Advantages: Lossless compression; excellent image quality. Disadvantages: Takes up a lot of storage space.

5.1.9. Register Type Code

Name	Code	Description
TUREG_SN	0x01	Read and write SN code registers
TUREG_DATA	0x02	Read and write data content register
TUREG_BAD_ROW	0x03	Read and write bad line register (vendor-used)
TUREG_BAD_COL	0x04	Read and write bad column register (vendor-used)
TUREG_BGC	0x05	Read and write background register (vendor-used)

TUREG_HDR	0x06	Read and write HDR exposure parameter register (vendor-used)
TUREG_HBG	0x07	Read and write HBG exposure parameter register (vendor-used)
TUREG_CMS	0x08	Read and write CMS exposure parameter register (vendor-used)
TUREG_CBG	0x09	Read and write CBG exposure parameter register (vendor-used)
TUREG_CODE	0x0A	Read and write code parameter register (vendor-used)
TUREG_DPC	0x0B	Read and write DPC parameter register (vendor-used)
TUREG_TEMPERATUREO FFSET	0x0C	Read and write temperature register (vendor-used)
TUREG_HIGHSPEEDBGY LIST	0x0D	Read and write cut the background register (vendor-used)

5.1.10. Trigger the Exposure Time Mode Code

Name	Code	Description
TUCTE_EXPTM	0x00	After receiving the trigger signal, the exposure time is determined by the TUIDP_EXPOSURETM
TUCTE_WIDTH	0x01	After receiving the trigger signal, the exposure time is determined by the width of the level

5.1.11. Edge Trigger Type Code

Name	Code	Description
TUCTD_RISING	0x00	Exposure begins when a rising edge trigger level is received
TUCTD_FAILING	0x01	Exposure begins when a failing edge trigger level is received

5.1.12. Triggers the Read Output Reset Code

Name	Code	Description
TUCTD_YES	0x00	Reset
TUCTD_NO	0x01	Do not reset

5.1.13. Trigger Output Direction Code

Name	Code	Description
TUCTD_DOWN	0x00	Down
TUCTD_UP	0x01	Up
TUCTD_DOWNUPCYC	0x02	Loop down

5.1.14. Frame Format Code

Name	Code	Description
TUFRM_FMT_RAW	0x10	RAW format data
TUFRM_FMT_USUAL	0x11	Usual format data (8bit/16bit, mono/color)
TUFRM_FMT_RGB888	0x12	RGB888 format data (For display)

5.1.15. Gain Mode Code

Name	Code	Description
TUGAIN_HDR	0x00	HDR mode
TUGAIN_HIGH	0x01	High gain mode
TUGAIN_LOW	0x02	Low gain mode

5.1.16. Channel Selection Code

Name	Code	Description
TUCHN_SHARE	0x00	Shared channel (gray or RGB)

TUCHN_RED	0x01	Channel 1 (Red channel)
TUCHN_GREEN	0x02	Channel 2 (Green channel)
TUCHN_BLUE	0x03	Channel 3 (Blue channel)

5.1.17. Trigger Output Port Code

Name	Code	Description
TUPORT_ONE	0x00	Use the port 1
TUPORT_TWO	0x01	Use the port 2
TUPORT_THREE	0x02	Use the port 3

5.1.18. Trigger Output Type Code

Name	Code	Description
TUOPT_GND	0x00	Low level
TUOPT_VCC	0x01	High level
TUOPT_IN	0x02	Trigger input
TUOPT_EXPSTART	0x03	Exposure start reference point
TUOPT_EXPGLOBAL	0x04	Global exposure
TUOPT_READEND	0x05	Read end

5.1.19. Trigger Output Edge Code

Name	Code	Description
TUOPT_RISING	0x00	Trigger the rising edge
TUOPT_FAILING	0x01	Trigger the failing edge

5.1.20. Image Drawing Mode Code

Note: TUDRAW_DFT mode is only available on windows systems, and windows systems can only support TUDRAW_DFT mode.

Name	Code	Description
TUDRAW_DFT	0x00	Default drawing mode
TUDRAW_DIB	0x01	DIB drawing mode
TUDRAW_DX9	0x02	DirectX 9.0

5.1.21. Record Keyframe Mode Code

Name	Code	Description
TUREC_TIMESTAMP	0x01	Determine keyframe position based on timestamp (to Disk mode)
TUREC_SEQUENCE	0x02	Determine keyframe positions based on image sequences (to Ram mode)

5.1.22. Image Processing Mode Code

Name	Code	Description
TUPROC_EDF	0x00	Depth of field fusion mode
TUPROC_STITCH	0x01	Image stitching

5.1.23. Image Stitching Mode Code

Name	Code	Description
TUSM_FINE	0x00	Stitching normal quality (fast)
TUSM_EXCELLENT	0x01	Stitching high quality (slow)

5.1.24. Focus Status Code

Name	Code	Description
TUFS_STOP	0x00	Focus stops
TUFS_FOCUSING	0x01	Focusing
TUFS_COMPLETED	0x02	Focus complete

TUFS_DEFOCUS	0x03	Out of focus state
--------------	------	--------------------

5.1.25. Vendor property (vendor only)

Name	Code	Description
TUIDV_ADDR_FLASH	0x00	Vendor flash address
TUIDV_ODDEVENH	0x01	High parity values
TUIDV_ODDEVENL	0x02	Low parity values
TUIDV_HDRHGBOFFSET	0x03	HDR high gain bit offset
TUIDV_HDRLGBOFFSET	0x04	HDR low gain bit offset
TUIDV_CMSHGBOFFSET	0x05	CMS high gain bit offset
TUIDV_CMSLGBOFFSET	0x06	CMS low gain bit offset
TUIDV_FPNENABLE	0x07	Enable FPN reduction
TUIDV_WORKING_TIME	0x08	Working time
TUIDV_CALC_DSNU	0x09	Calculate DSNU
TUIDV_CALC_PRNU	0x0A	Calculate PSNU
TUIDV_CALC_DPC	0x0B	Calculate DPC
TUIDV_CALC_STOP	0x0C	Stop calculation
TUIDV_CALC_STATE	0x0D	Get the calculation status 0: Interface idle 2: Interface busy 3: Computation busy
TUIDV_HDR_LVALUE	0x0E	HDR low value
TUIDV_HDR_HVALUE	0x0F	HDR high value
TUIDV_FW_CHECK	0x10	Firmware check value
TUIDV_HIGHSPEEDHGBOFFSET	0x11	High speed high gain bit offset
TUIDV_HIGHSPEEDLGBOFFSET	0x12	High speed low gain bit offset
TUIDV_TEMPERATURE_OFFSET	0x13	Temperature offset
TUIDV_ENDVPROPERTY	0x14	Vendor property ID end bit

5.2. Structure

Some functions require pointers as parameters, the pointer to the structure. For

structure, if an [in] identifier variable means that the application must set the value before the call, and an [out] identifier variable means that the function returns with a value filled.

5.2.1.Initialization

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

[TUCAM_API_Init] input parameters

Name	Description
uiCamCount	[out] returns the number of cameras currently connected
pstrConfigPath	[in] enters the path where the camera parameters will be saved when the [TUCAM_File_SaveProfiles] is executed.

5.2.2.Open camera

```
1. typedef struct _tagTUCAM_OPEN
2. {
3.     UINT32 uidxOpen;
4.     HDTUCAM hIdxTUCam;
5. }TUCAM_OPEN, *PTUCAM_OPEN;
6.
```

[TUCAM_Dev_Open] input parameters

Name	Description
uidxOpen	[in] Open the specified camera
hldxTUCam	[out] Handle of the opened camera

5.2.3.Open image

```

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

```

[TUIMG_File_Open] input parameters

Name	Description
pszfileName	[in] Open the specified camera
hldxTUImg	[out] Handle of the opened image

5.2.4.Camera Information

```

1. typedef struct _tagTUCAM_VALUE_INFO
2. {
3.     INT32    nID;
4.     INT32    nValue;
5.     PCHAR    pText;
6.     INT32    nTextSize;
7. }TUCAM_VALUE_INFO, *PTUCAM_VALUE_INFO;
8.

```

[TUCAM_Get_Info \ TUCAM_Get_InfoEx] input parameters

Name	Description
nID	[in] Device information ID, refer to TUCAM_IDINFO
nValue	[in] The value of the device information
pText	[in / out] A pointer to the device information text
nTextSize	[in] Text cache size

5.2.5.Capability Attribute

```

1. typedef struct _tagTUCAM_CAPA_ATTR
2. {
3.     INT32    idCapa;
4.     INT32    nValMin;
5.     INT32    nValMax;
6.     INT32    nValDft;
7.     INT32    nValStep;
8. }TUCAM_CAPA_ATTR, *PTUCAM_CAPA_ATTR;
9.

```

[TUCAM_Capa_GetAttr] input parameters:

Name	Description
idCapa	[in] Camera capability ID, refer to TUCAM_IDCAPA
nValMin	[out] Minimum
nValMax	[out] Maximum
nValDft	[out] Default value
nValStep	[out] Minimum step in the range

5.2.6.Property Attribute

```

1. typedef struct _tagTUCAM_PROP_ATTR

```

```

2.  {
3.      INT32    idProp;
4.      INT32    nIdxChn;
5.      DOUBLE   dbValMin;
6.      DOUBLE   dbValMax;
7.      DOUBLE   dbValDft;
8.      DOUBLE   dbValStep;
9.  }TUCAM_PROP_ATTR, *PTUCAM_PROP_ATTR;

```

[TUCAM_Prop_GetAttr] input parameters:

Name	Description
idProp	[in] Camera property ID, refer to TUCAM_IDPROP
nIdxChn	[in/out] Channel index, the default channel index is 0
dbValMin	[out] Minimum
dbValMax	[out] Maximum
dbValDft	[out] Default value
dbValStep	[out] Minimum step in the range

5.2.7.Capability/ Property Value Text

```

1.  typedef struct _tagTUCAM_VALUE_TEXT
2.  {
3.      INT32    nID;
4.      DOUBLE   dbValue;
5.      PCHAR    pText;
6.      INT32    nTextSize;
7.  }TUCAM_VALUE_TEXT, *PTUCAM_VALUE_TEXT;
8.

```

[TUCAM_Capa_GetValueText] and [TUCAM_Prop_GetValueText] input parameters:

Name	Description
nID	[in] Capability/ Property ID, refer to TUCAM_IDPROP / TUCAM_IDCAPA

dbValue	[in] Valid value
pText	[in/out] A pointer to the value text
nTextSize	[out] Value text cache size

5.2.8.The Attribute of the Vendor Property

```

1. typedef struct _tagTUCAM_VPROP_ATTR
2. {
3.     INT32    idVProp;
4.     INT32    nIdxChn;
5.     DOUBLE   dbValMin;
6.     DOUBLE   dbValMax;
7.     DOUBLE   dbValDft;
8.     DOUBLE   dbValStep;
9. }TUCAM_VPROP_ATTR, *PTUCAM_VPROP_ATTR;
10.

```

[TUCAM_Proc_Prop_GetValueText] input parameters:

Name	Description
idVProp	[in] Vendor property ID, refer to TUCAM_IDVPROP
nIdxChn	[in/out] Channel index, the default channel index is 0
dbValMin	[out] Minimum
dbValMax	[out] Maximum
dbValDft	[out] Default value
dbValStep	[out] Minimum step in the range

5.2.9.Processing Image Attribute

```

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.

```

[TUCAM_Proc_Prop_GetValueText] input parameters.

Name	Description
idPProp	[in] Image processing ID, refer to TUCAM_IDPPROP
procType	[in] Image processing type, refer to TUPROC_TYPE
dbValMin	[out] Minimum
dbValMax	[out] Maximum
dbValDft	[out] Device default
dbValStep	[out] Minimum step in the range

5.2.10. ROI Attribute

```

1. typedef struct _tagTUCAM_ROI_ATTR
2. {
3.     BOOL    bEnable;
4.     INT32   nHOffset;
5.     INT32   nVOffset;
6.     INT32   nWidth;
7.     INT32   nHeight;
8. }TUCAM_ROI_ATTR, *PTUCAM_ROI_ATTR;
9.

```

[TUCAM_Cap_SetROI] and [TUCAM_Cap_GetROI] input parameters:

Name	Description
bEnable	[in/out] Enable ROI
nHOffset	[in/out] The horizontal offset, in pixels
nVOffset	[in/out] The vertical offset, in pixels
nWidth	[in/out] ROI width
nHeight	[in/out] ROI height

5.2.11. Area Calculation Attribute

```

1. typedef struct _tagTUCAM_CALC_ROI_ATTR
2. {
3.     BOOL    bEnable;
4.     INT32   idCalc;
5.     INT32   nHOffset;
6.     INT32   nVOffset;
7.     INT32   nWidth;
8.     INT32   nHeight;
9. }TUCAM_CALC_ROI_ATTR, *PTUCAM_CALC_ROI_ATTR;
10.

```

[TUCAM_Cap_SetROI] and [TUCAM_Cap_GetROI] input parameters:

Name	Description
bEnable	[in/out] Enable ROI
idCalc	[in] Area calculation type, refer to TUCAM_IDCROI
nHOffset	[in/out] The horizontal offset, in pixels
nVOffset	[in/out] The vertical offset, in pixels
nWidth	[in/out] ROI width
nHeight	[in/out] ROI height

5.2.12. Trigger Attribute

```

1. typedef struct _tagTUCAM_TRIGGER_ATTR
2. {
3.     INT32    nTgrMode;
4.     INT32    nExpMode;
5.     INT32    nEdgeMode;
6.     INT32    nDelayTm;
7.     INT32    nFrames;
8.     INT32    nBufFrames;
9. }TUCAM_TRIGGER_ATTR, *PTUCAM_TRIGGER_ATTR;
10.

```

[TUCAM_Cap_SetTrigger] and [TUCAM_Cap_GetTrigger] input parameters:

Name	Description
nTgrMode	[in/out] Trigger mode, refer to TUCAM_CAPTURE_MODES
nExpMode	[in/out] Exposure mode[0,1] , refer to TUCAM_TRIGGER_EXP 0: Exposure time, 1: Level width
nEdgeMode	[in/out] Edge trigger mode [0,1], refer to TUCAM_TRIGGER_EDGE 0: Rising edge, 1: Falling edge,
nDelayTm	[in/out] Trigger delay time in milliseconds
nFrames	[in/out] The number of output frames triggered at one time
nBufFrames	[in/out] The number of frames in the buffer

5.2.13. Trigger Output Attribute

```

1. typedef struct _tagTUCAM_TRGOUT_ATTR
2. {
3.     INT32    nTgrOutPort;
4.     INT32    nTgrOutMode;
5.     INT32    nEdgeMode;

```

```

6.      INT32    nDelayTm;
7.      INT32    nWidth;
8.  }TUCAM_TRGOUT_ATTR, *PTUCAM_TRGOUT_ATTR;
9.

```

[TUCAM_Cap_SetTriggerOut] and [TUCAM_Cap_GetTriggerOut] input parameters:

Name	Description
nTgrOutPort	[in/out] Trigger output, refer to TUCAM_OUTPUTTRG_PORT
nTgrOutMode	[in/out] Trigger output mode, refer to TUCAM_OUTPUTTRG_KIND
nEdgeMode	[in/out] Edge trigger mode
nDelayTm	[in/out] Trigger output delay time
nWidth	[in/out] Trigger output pulse width

5.2.14. Any Bin Attribute

```

1.  typedef struct _tagTUCAM_BIN_ATTR
2.  {
3.      BOOL    bEnable;
4.      INT32    nWidth;
5.      INT32    nHeight;
6.  }TUCAM_BIN_ATTR, *PTUCAM_BIN_ATTR;
7.

```

[TUCAM_Cap_SetBIN] and [TUCAM_Cap_GetBIN] input parameters:

Name	Description
bEnable	[in/out] Enable any bin
nWidth	[in/out] Any bin width
nHeight	[in/out] Any bin height

5.2.15. Frame Structure

```

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.

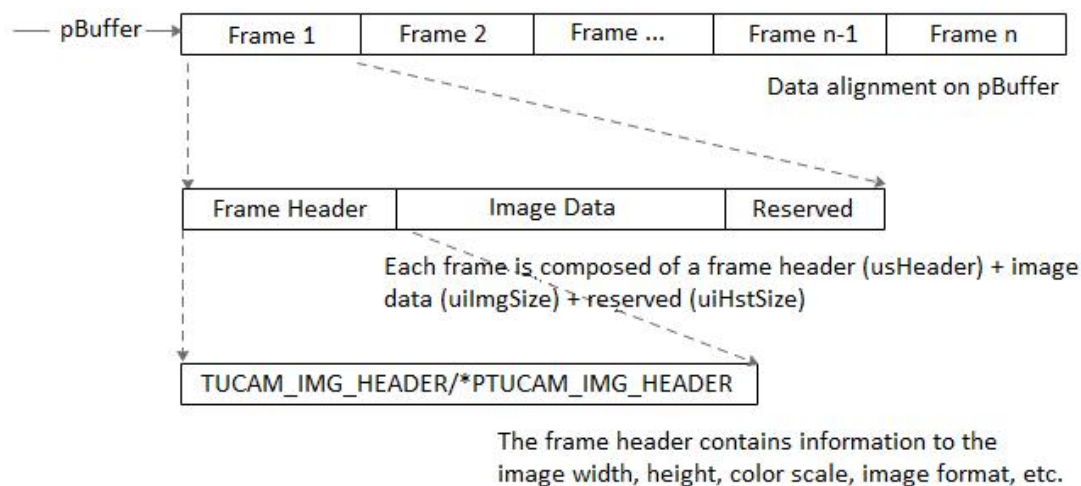
```

[TUCAM_Buf_WaitForFrame] input parameter

Name	Description
szSignature[8]	[out] Copyright information
usHeader	[out] Frame header size
usOffset	[out] Frame data offset size
usWidth	[out] Number of horizontal pixels
usHeight	[out] Number of vertical pixels
uiWidthStep	[out] Frame image width step
ucDepth	[out] Frame image data bit depth

ucFormat	[out] Frame image data format
ucChannels	[out] Number of channels of frame image
ucElemBytes	[out] Number of data bytes of frame image
ucFormatGet	[in/out] The format of the image to be acquired. Refer to TUFrm_FORMATS
uiIndex	[out] Frame image sequence number (reserved)
uiImgSize	[out] The data size of the frame image
uiRsdSize	[in] Number of frames to get
uiHstSize	[out] Retention field of frame image
pBuffer	[out] Pointer to the frame data cache

The pBuffer data is arranged as follows, and the header information of each frame can be obtained according to the frame header size. And the image data can also be obtained according to the frame data offset for drawing display.



5.2.16. File Saving

```

1. typedef struct _tagTUCAM_FILE_SAVE
2. {
3.     INT32 nSaveFmt;
4.     PCHAR pstrSavePath;
5.     PTUCAM_FRAME pFrame;
6. }TUCAM_FILE_SAVE, *PTUCAM_FILE_SAVE;
7.

```



[TUCAM_File_SaveImage] structure:

Name	Description
nSaveFmt	[in] Format for saving images TUIMG_FORMATS
pstrSavePath	[in] Save image path (without extension)
pFrame	[in] Pointer to frame data

5.2.17. Video Saving

```

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

```

[TUCAM_Rec_Start] input parameters:

Name	Description
nCodec	[in] Code type, default is 0
pstrSavePath	[in] Save path (with file name)
fFps	[in] Frame rate to be saved

5.2.18. Read and Write Registers

```

1. typedef struct _tagTUCAM_REG_RW
2. {
3.     INT32    nRegType;
4.     PCHAR    pBuf;
5.     INT32    nBufSize;
6. }TUCAM_REG_RW, *PTUCAM_REG_RW;
7.

```

[TUCAM_Reg_Read] and [TUCAM_Reg_Write] input parameters:

Name	Description
nRegType	[in] Read and write register type, refer to TUREG_TYPE
pBuf	[in/out] A pointer to the read-write content buffer
nBufSize	[in] Buffer size

5.2.19. Image Drawing Initialization

```

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;
11.

```

[TUCAM_Draw_Init] input parameters:

Name	Description
hWnd	[in] Draw window handle, only supported for Windows OS
nMode	[in] Whether to use hardware acceleration (reserved for GPU support), default TUDRAW_DFT
ucChannels	[in] Number of camera channels
nWidth	[in] Draw data width
nHeight	[in] Draw data height

5.2.20. Image Drawing Parameters (Windows only)

```

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.

```

[TUCAM_Draw_Frame] input parameters:

Name	Description
nSrcX	[in/out] x-coordinate of the upper-left corner of the source rectangle (in pixels)
nSrcY	[in/out] y-coordinate of the upper-left corner of the source rectangle (in pixels)
nSrcWidth	[in/out] Width of the source rectangle in pixels
nSrcHeight	[in/out] Height of the source rectangle in pixels
nDstX	[in/out] The x-coordinate of the upper-left corner of the target rectangle, expressed

	in MM_TEXT client coordinates.
nDstY	[in/out] The y-coordinate of the upper-left corner of the target rectangle, expressed in MM_TEXT client coordinates.
nDstWidth	[in/out] The width of the target rectangle, expressed in MM_TEXT client coordinates.
nDstHeight	[in/out] The height of the target rectangle, expressed in MM_TEXT client coordinates.
pFrame	[in] Pointer to the structure of the frame to be drawn

5.3. Functions

5.3.1.API Initialization / Deinitialization

5.3.1.1. [TUCAM_Api_Init] and [TUCAM_Api_Uninit]

Description

Unloads the TUCAM-API library, including releasing the driver bindings and some internal resources. Called once at the end of the entire program.

Statement

```
TUCAMRET TUCAM_Api_Uninit ();
```

Parameters

No input parameters

Error Code

TUCAMRET_NOT_INIT	TUCAM-API not initialized
-------------------	---------------------------

Related APIs

TUCAM_Api_Uninit

5.3.2. Open and Close the Camera

5.3.2.1. TUCAM_Dev_Open

Description

After calling [TUCAM_Dev_Open], the camera is in working mode and can respond to other interface calls. Before calling it, make sure that the camera already exists, i.e. call it after [TUCAM_Api_Init initialization].

Statement

```
TUCAMRET TUCAM_Dev_Open (PTUCAM_OPEN pOpenParam);
```

Parameters

PTUCAM_OPEN pOpenParam	Open camera structure pointer, refer to structure TUCAM_OPEN
------------------------	--

Error Code

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_INVALID_PARAM	Invalid parameter, when pOpenParam pointer is empty
TUCAMRET_OUT_OF_RANGE	Out of range, when the index number of the camera to be opened is out of range of the connected camera
TUCAMRET_FAILOPEN_CAMERA	Failed to open the camera
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Close

5.3.2.2. TUCAM_Dev_Close

Description

After calling [TUCAM_Dev_Close], the camera is in standby mode and can't respond to calls from other interfaces.

Statement

```
TUCAMRET TUCAM_Dev_Close ();
```

Parameters

No input parameters

Error Code

TUCAMRET	TUCAM-API not initialized
----------	---------------------------

Related APIs

TUCAM_Api_Init, TUCAM_Api_Uninit

TUCAM_Dev_Open

5.3.2.3. TUCAM_Dev_GetInfo

Description

Get information about the camera, such as USB port type, camera information, API version, firmware version, etc. Before calling it, make sure that the camera already exists and on, i.e. call it after [TUCAM_Api_Init initialization] and [TUCAM_Api_Open]

Statement

```
TUCAMRET TUCAM_Dev_GetInfo (HDTUCAM hTUCam, PTUCAM_VALUE_INFO pInfo);
```

Parameters

HDTUCAM hTUCam	The handle of the camera
PTUCAM_VALUE_INFO pInfo	Pointer to the structure of the camera information, refer to TUCAM_VALUE_INFO

Error Code

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_INVALID_PARAM	Invalid parameter, when the product information code does not exist, refer to TUCAM_IDINFO
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Open, TUCAM_Dev_Close

5.3.2.4. TUCAM_Dev_GetInfoEx

Description

Get information about the camera, such as USB port type, camera information, API version, firmware version, etc. Before calling it, make sure that the camera already exists and on, i.e. call it after [TUCAM_Api_Open]. [TUIDI_CAMERA_MODEL], [TUIDI_BUS, TUIDI_VENDOR], [TUIDI_PRODUCT], [TUIDI_VERSION_API], [TUIDI_BCDDEVICE] are only supported.

Statement

TUCAMRET TUCAM_Dev_GetInfoEx (UINT32 uilCam, PTUCAM_VALUE_INFO pInfo);

Parameters

UINT32 uilCam	The index number of the camera
PTUCAM_VALUE_INFO pInfo	Pointer to the structure of the camera information, refer to TUCAM_VALUE_INFO

Error Code

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_INVALID_PARAM	Invalid parameter, when the product information code does not exist, refer to TUCAM_IDINFO
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

TUCAM_Api_Init, TUCAM_Api_Uninit

5.3.3. Set and Get Capability

5.3.3.1. TUCAM_Capa_GetAttr

Description

Gets the attribute value of the capability parameter, contains the minimum, maximum, default and

step values. Refer to TUCAM_IDCAPA.

Statement

TUCAMRET TUCAM_Capa_GetAttr (HDTUCAM hTUCam, PTUCAM_CAPA_ATTR pAttr);

Parameters

HDTUCAM hTUCam	The handle of the camera
PTUCAM_CAPA_ATTR pAttr	Camera capability attribute structure pointer, refer to TUCAM_CAPA_ATTR

Error Code

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_INVALID_IDCAPA	Invalid parameter, when the product information code does not exist, refer to TUCAM_IDINFO
TUCAMRET_INVALID_VALUE	Invalid value when the pAttr pointer is empty
TUCAMRET_NOT_SUPPORT	When the underlying request is not supported
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

TUCAM_Api_Init, TUCAM_Api_Uninit

TUCAM_Dev_Open, TUCAM_Dev_Close

TUCAM_Capa_GetValue, TUCAM_Capa_SetValue, TUCAM_Capa_GetValueText

5.3.3.2. TUCAM_Capa_GetValue

Description

Gets the attribute value of the capability parameter. Refer to TUCAM_IDCAPA for the supported capability types

Statement

TUCAMRET TUCAM_Capa_GetValue (HDTUCAM hTUCam, INT32 nCapa, INT32 *pnVal);

Parameters

HDTUCAM hTUCam	The handle of the camera
----------------	--------------------------

INT32 nCapa	Camera capability attribute ID, refer to TUCAM_IDCAPA
INT32 *pnVal	Return current value

Error Code

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_INVALID_IDCAPA	Invalid parameter, when the product information code does not exist, refer to TUCAM_IDINFO
TUCAMRET_INVALID_VALUE	Invalid value when the pAttr pointer is empty
TUCAMRET_NOT_SUPPORT	When the underlying request is not supported
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Open, TUCAM_Dev_Close
TUCAM_Capa_GetAttr, TUCAM_Capa_SetValue, TUCAM_Capa_GetValueText

5.3.3.3. TUCAM_Capa_SetValue

Description

Gets the attribute value of the capability parameter. Refer to TUCAM_IDCAPA for the supported capability types.

Statement

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

Parameters

HDTUCAM hTUCam	The handle of the camera
INT32 nCapa	Camera capability attribute ID, refer to TUCAM_IDCAPA
INT32 nVal	The current value that needs to be set

Error Code

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_INVALID_IDCAPA	Invalid parameter, when the product information code does not exist, refer to TUCAM_IDINFO

TUCAMRET_INVALID_VALUE	Invalid value when the pAttr pointer is empty
TUCAMRET_NOT_SUPPORT	When the underlying request is not supported
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

TUCAM_Api_Init, TUCAM_Api_Uninit

TUCAM_Dev_Open, TUCAM_Dev_Close

TUCAM_Capa_GetAttr, TUCAM_Capa_GetValue, TUCAM_Capa_GetValueText

5.3.3.4. TUCAM_Capa_GetValueText

Description

Gets text information about the current attribute value of the capability parameter. For specific supported capability types, refer to TUCAM_IDCAPA.

Statement

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

Parameters

HDTUCAM hTUCam	The handle of the camera
PTUCAM_VALUE_TEXT pVal	Get the text information structure pointer for the capability parameter, TUCAM_VALUE_TEXT

Error Code

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_FAILURE	When the text buffer size is 0 or the pText pointer is empty
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

TUCAM_Api_Init, TUCAM_Api_Uninit

TUCAM_Dev_Open, TUCAM_Dev_Close

TUCAM_Capa_GetAttr, TUCAM_Capa_SetValue, TUCAM_Capa_GetValue

5.3.4. Set and Get Property

5.3.4.1. TUCAM_Prop_GetAttr

Description

Gets the attribute value of the property parameter, contains the minimum, maximum, default and step values. Refer to TUCAM_IDPROP.

Statement

```
TUCAMRET TUCAM_Prop_GetAttr (HDTUCAM hTUCam, PTUCAM_PROP_ATTR pAttr);
```

Parameters

HDTUCAM hTUCam	The handle of the camera
PTUCAM_PROP_ATTR pAttr	Camera property structure pointer, refer to TUCAM_PROP_ATTR

Error Code

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_INVALID_IDPROP	Invalid property code, refer to TUCAM_IDPROP
TUCAMRET_INVALID_VALUE	Invalid value when the pAttr pointer is empty
TUCAMRET_NOT_SUPPORT	When the underlying request is not supported
TUCAMRET_OUT_OF_RANGE	The acquired channel is out of range
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

TUCAM_Api_Init, TUCAM_Api_Uninit

TUCAM_Dev_Open, TUCAM_Dev_Close

TUCAM_Prop_GetValue, TUCAM_Prop_SetValue, TUCAM_Prop_GetValueText

5.3.4.2. TUCAM_Prop_GetValue

Description

It gets the current value of the property parameters. For specific supported property types, refer to TUCAM_IDPROP.

Statement

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

Parameters

HDTUCAM hTUCam	The handle of the camera
INT32 nProp	ID of the camera property, refer to TUCAM_IDPROP
DOUBLE *pdbVal	Returns the current value
INT32 nChn	The current channel (default is 0, monochrome camera is 0)

Error codes

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_INVALID_IDPROP	Invalid property code, refer to TUCAM_IDPROP
TUCAMRET_INVALID_VALUE	Invalid value when the pAttr pointer is empty
TUCAMRET_NOT_SUPPORT	When the underlying request is not supported
TUCAMRET_OUT_OF_RANGE	The acquired channel is out of range
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Open, TUCAM_Dev_Close
TUCAM_Prop_GetAttr, TUCAM_Prop_SetValue, TUCAM_Prop_GetValueText

5.3.4.3. TUCAM_Prop_GetValueText

Description

Gets the text information of the current value of the property parameter. For specific supported property types, refer to TUCAM_IDPROP.

Statement

TUCAMRET TUCAM_Prop_GetValueText (HDTUCAM hTUCam, PTUCAM_VALUE_TEXT pVal);

Parameters

HDTUCAM hTUCam	The handle of the camera
----------------	--------------------------

PTUCAM_VALUE_TEXT pVal Get the text information structure pointer for the capability parameter, TUCAM_VALUE_TEXT

Error Codes

TUCAMRET_NOT_INIT TUCAM-API not initialized
TUCAMRET_FAILURE When the text buffer size is 0 or the pText pointer is empty
TUCAMRET_INVALID_CAMERA Invalid camera, when the camera handle does not exist

Related APIs

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Open, TUCAM_Dev_Close
TUCAM_Prop_GetAttr, TUCAM_Prop_SetValue, TUCAM_Prop_GetValue

5.3.5.Memory Management

5.3.5.1. TUCAM_Buf_Alloc

Description

The [TUCAM_Buf_Alloc] is used to allocate buffer for data capture. When it is called, the SDK allocates the necessary internal buffers to capture the image data. But data capture starts from calling [TUCAM_Cap_Start].

If the buffer is no longer necessary, the application should call the TUCAM_Buf_Release interface to release the internal buffer.

Statement

TUCAMRET TUCAM_Buf_Alloc (HDTUCAM hTUCam, PTUCAM_FRAME pFrame);

Parameters

HDTUCAM hTUCam The handle of the camera
PTUCAM_VALUE_TEXT pVal Pointer to the image frame structure, referencing TUCAM_FRAME

Error Code

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_INVALID_PARAM	When the pFrame pointer is empty
TUCAMRET_EXCLUDED	When TUCAM_Buf_Alloc is called and not freed
TUCAMRET_OUT_OF_RANGE	When the number of frames to be acquired is out of range
TUCAMRET_NO_MEMORY	When memory is low
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

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

5.3.5.2. TUCAM_Buf_Release

Description

The [TUCAM_Buf_Release] is used to release the buffer used for data capture. If this interface is called during a capture, it will return the state that the camera is busy.

Statement

TUCAMRET TUCAM_Buf_Release (HDTUCAM hTUCam);

Parameters

HDTUCAM hTUCam	The handle of the camera
----------------	--------------------------

Error Code

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_BUSY	The camera is busy
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

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

5.3.5.3. TUCAM_Buf_AbortWait

Description

The [TUCAM_Buf_AbortWait] is used to stop waiting while data is being captured. The [TUCAM_Buf_WaitForFrame] must be called after the data capture wait.

Statement

TUCAMRET TUCAM_Buf_AbortWait (HDTUCAM hTUCam);

Parameters

HDTUCAM hTUCam	The handle of the camera
----------------	--------------------------

Error Code

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

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

5.3.5.4. TUCAM_Buf_WaitForFrame

Description

The [TUCAM_Buf_WaitForFrame] is used to wait for data capture to complete, it must be called after [TUCAM_Cap_Start], otherwise it will return an unprepared state. And the [TUCAM_Buf_Alloc] is used to allocate space to capture data frames.

This function is a blocking function and will not terminate until data capture is complete or

[TUCAM_Buf_AbortWait] is called.

The uiRsdSize in the frame structure is used to set the number of frames to be captured, valid only for trigger mode. For example, if the trigger needs to return 5 frames at a time, the function waits for all 5 frames to be captured before returning.

Note: The data arrangement of the returned frame structure pBuffer is frame header (usHeader) + image data (uiImgSize) + reserved bits (uiHstSize). If multiple frames are returned then they are arranged in this order backwards.

Statement

```
TUCAMRET TUCAM_Buf_WaitForFrame (HDTUCAM hTUCam, PTUCAM_FRAME pFrame,
INT32 nTimeOut = 1000);
```

Parameters

HDTUCAM hTUCam	The handle of the camera
PTUCAM_FRAME pFrame	Pointer to the frame structure
INT32 nTimeOut	Timeout, default 1S

Error Code

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_NOT_READY	When [TUCAM_Cap_Start] is not called to start capturing
TUCAMRET_NO_MEMORY	When [TUCAM_Buf_Alloc] is not called to create memory space
TUCAMRET_NO_RESOURCE	When the pFrame pointer is empty
TUCAMRET_OUT_OF_RANGE	When the number of frames acquired is greater than 1 and the acquisition format is different from [TUCAM_Buf_Alloc]
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

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

5.3.5.5. TUCAM_Buf_CopyFrame

Description

The [TUCAM_Buf_CopyFrame] is used to copy image data and save it in a format different from the one allocated by [TUCAM_Buf_Alloc]. It must be called after the [TUCAM_Buf_WaitForFrame] returning, otherwise it cannot get the correct image data.

For example, the allocated image format is TUFRM_FMT_RGB888, through this function you can copy data of other formats (e.g. TUFRM_FMT_RAW), this interface can only copy data for 1 frame, that is, the value of uiRsdSize in the frame structure cannot be greater than 1.

Note: The data arrangement of the returned frame structure pBuffer is the frame header (usHeader) + image data (uiImgSize) + reserved bits (uiHstSize). Returning multiple frames of data is not supported.

Statement

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

Parameters

HDTUCAM hTUCam	The handle of the camera
PTUCAM_FRAME pFrame	Pointer to the frame structure

Error Code

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_NOT_READY	When TUCAM_Cap_Start is not called to start capturing
TUCAMRET_NO_MEMORY	When TUCAM_Buf_Alloc is not called to create memory space
TUCAMRET_NO_RESOURCE	When the pFrame pointer is empty
TUCAMRET_OUT_OF_RANGE	When the number of frames acquired is greater than 1 and the acquisition format is different from TUCAM_Buf_Alloc
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

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

5.3.5.6. TUCAM_Buf_DataCallBack

Description

The [Tucam_Buf_DataCallBack] is used to get the raw data flow registration callback, it must be called after [TUCAM_Cap_Start]. And the [TUCAM_Buf_Alloc] is used to allocate space to capture data frames.

Statement

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

Parameters

HDTUCAM hTUCam	The handle of the camera
BUFFER_CALLBACK cbBuffer	Gets the raw data flow callback function
void *pUserContext	User data callback function registration

Error Codes

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Open, TUCAM_Dev_Close
TUCAM_Buf_Alloc, TUCAM_Buf_Release
TUCAM_Buf_GetData
TUCAM_Cap_Start, TUCAM_Cap_Stop

5.3.5.7. TUCAM_Buf_GetData

Description

The [TUCAM_Buf_GetDate] is used to get the raw data stream of [TUCAM_Buf_DataCallBack], it must be called after [TUCAM_Cap_Start]. And the [TUCAM_Buf_Alloc] is used to allocate space to capture data frames.

Statement

TUCAMRET TUCAM_Buf_GetData (HDTUCAM hTUCam, PTUCAM_RAWIMG_HEADER pFrame);

Parameters

HDTUCAM hTUCam	The handle of the camera
PTUCAM_RAWIMG_HEADER pFrame	The original frame structure pointer

Error Codes

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_FAILURE	Failed to get frame
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

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

5.3.6. Capture Control

5.3.6.1. TUCAM_Cap_SetROI

Description

The [TUCAM_Cap_Start] is used to set the area of interest of the image, with the upper left corner as the coordinate origin. The horizontal offset, vertical offset, width, and height must be set in multiples of 4. (In 11Bit mode, width is greater than 32, width times height must be a multiple of 32)

Statement

TUCAMRET TUCAM_Cap_SetROI (HDTUCAM hTUCam, TUCAM_ROI_ATTR roiAttr);

Parameters

HDTUCAM hTUCam	The handle of the camera
TUCAM_ROI_ATTR roiAttr	Objects of the ROI property structure

Error Code

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_NOT_SUPPORT	ROI not supported
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

APIs

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

5.3.6.2. TUCAM_Cap_GetROI

Description

Used to set the area of interest of the image, with the upper left corner as the coordinate origin.
The horizontal offset, vertical offset, width, and height must be set in multiples of 4. (In 11Bit mode, width is greater than 32, width times height must be a multiple of 32)
Capture after the [TUCAM_Cap_Start] interface is called.

Statement

TUCAMRET TUCAM_Cap_GetROI (HDTUCAM hTUCam, PTUCAM_ROI_ATTR pRoiAttr);

Parameters

HDTUCAM hTUCam	The handle of the camera
PTUCAM_ROI_ATTR pRoiAttr	A pointer to the ROI attribute structure

Error Code

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_NOT_SUPPORT	ROI not supported
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

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

5.3.6.3. TUCAM_Cap_SetTrigger

Description

Used to set the properties of the trigger. Capture after the [TUCAM_Cap_Start] interface is called.

Exposure Mode

TUCTE_EXPTM	The exposure time is set by the software.
TUCTE_WIDTH	The exposure time is set by the input level width.

Edge Trigger Mode

TUCTD_RISING	The trigger signal is a rising edge.
TUCTD_FAILING	The trigger signal is a falling edge.

Number of frames: How many images are taken after receiving a trigger signal, the exposure time of each image is the same, depending on the software settings. (This feature has no effect when selecting a level width.)

Delay: After receiving a trigger signal, you can set how long the delay time is before the camera is exposed.

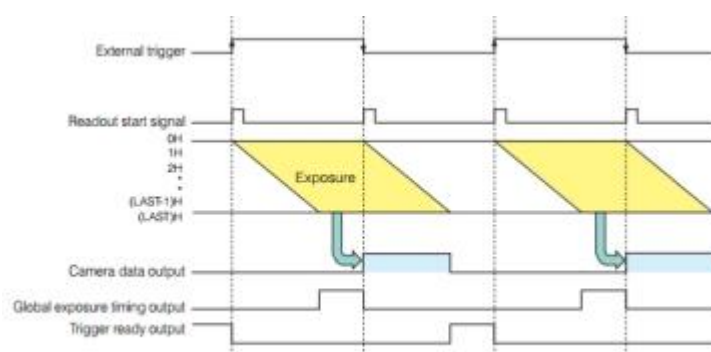


Fig. 19: Normal level trigger mode (rising edge)

Parameters:

Mode	TUCAM_TRIGGER_	TUCAM_TRIGGER_	Delay	Frame Number
	EXP	EDGE		
Standard	●	●	●	●
Synchronous	●	●	○	○
Global	○	●	○	○
Software	○	○	○	○

●: Supported ○: Not supported

Synchronous: synchronous image capture, start when the first trigger, output the synchronized image when the second trigger.

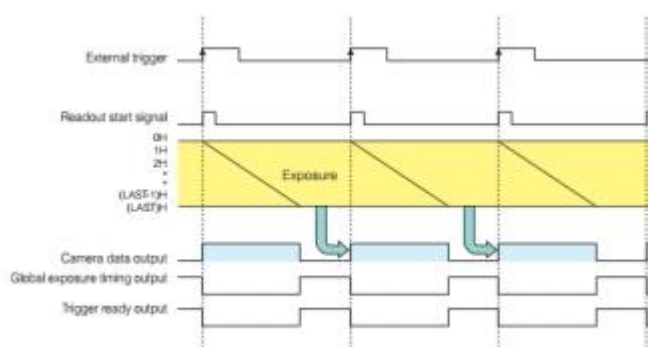
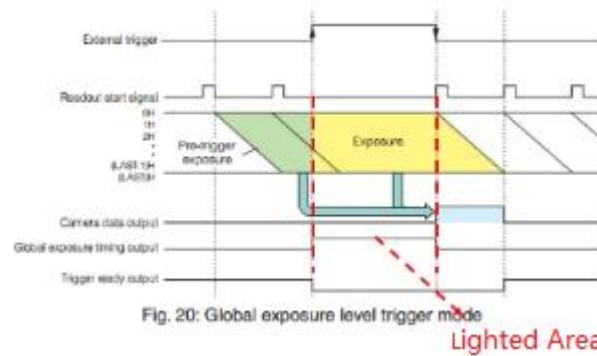


Fig. 21: Normal synchronous readout trigger mode (rising edge)

Global: Generally used in scenes where the light source can be controlled.



Software: Sending trigger commands through software.

Statement

```
TUCAMRET TUCAM_Cap_SetTrigger (HDTUCAM hTUCam, TUCAM_TRIGGER_ATTR tgrAttr);
```

Parameters

HDTUCAM hTUCam	The handle of the camera
TUCAM_TRIGGER_ATTR tgrAttr	A pointer to the ROI attribute structure

Error Codes

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

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

5.3.6.4. TUCAM_Cap_GetTriggerOut

Description

The [TUCAM_Cap_GetTriggerOut] is used to get the attributes of the trigger output.

Statement

TUCAMRET TUCAM_Cap_GetTriggerOut (HDTUCAM hTUCam, PTUCAM_TRGOUT_ATTR pTgrOutAttr);

Parameters

HDTUCAM hTUCam	The handle of the camera
PTUCAM_TRGOUT_ATTR pTgrOutAttr	Trigger output attribute structure pointer

Error Codes

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_NOT_SUPPORT	Settings are not supported
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Open, TUCAM_Dev_Close
TUCAM_Buf_Alloc, TUCAM_Buf_Release

5.3.6.5. TUCAM_Cap_Start

Description

The [TUCAM_Cap_Start] is used to start the data capture. It should be called after the capture buffer has been prepared by [TUCAM_Buf_alloc]. If [TUCCM_SEQUENCE] mode is used, the capture will continue until the [TUCAM_Cap_Stop] is called. This function is suitable for configuring ROI and triggering mode

Statement

TUCAMRET TUCAM_Cap_Start(HDTUCAM hTUCam, UINT32 uiMode);

Parameters

HDTUCAM hTUCam	The handle of the camera
UINT32 uiMode	Mode of camera capture, refer to TUCAM_CAPTURE_MODES

Error Code

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_FAILOPEN_BULKIN	Failed to camera capture
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

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

5.3.6.6. TUCAM_Cap_Stop

Description

The [TUCAM_Cap_Stop] is used to stop data capture

Statement

TUCAMRET TUCAM_Cap_Stop (HDTUCAM hTUCam);

Parameters

HDTUCAM hTUCam	The handle of the camera
----------------	--------------------------

Error Code

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_FAILOPEN_BULKIN	Failed to camera capture
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

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

5.3.7. File Management

5.3.7.1. TUCAM_File_SaveImage

Description

The [TUCAM_File_SaveImage] is used to save the frame data.

Statement

```
TUCAMRET TUCAM_File_SaveImage (HDTUCAM hTUCam, TUCAM_FILE_SAVE fileSave);
```

Parameters

HDTUCAM hTUCam	The handle of the camera
TUCAM_FILE_SAVE fileSave	File saving structure

Error Code

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_INVALID_PARAM	Invalid parameters
TUCAMRET_INVALID_PATH	The input path does not exist
TUCAMRET_FAILURE	Failed to save files
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Open, TUCAM_Dev_Close
TUCAM_Buf_Alloc, TUCAM_Buf_Release
TUCAM_Buf_WaitForFrame, TUCAM_Buf_CopyFrame

5.3.7.2. TUCAM_Rec_Start

Description

The [TUCAM_Rec_Start] is used to open the video file and save the frame data as a video. But no data is written when it is called.

The frame rate needs to be greater than 1fps, if it is less than 1fps, the recording file will be

created by 1fps.

Statement

```
TUCAMRET TUCAM_Rec_Start(HDTUCAM hTUCam, TUCAM_REC_SAVE recSave);
```

Parameters

HDTUCAM hTUCam	The handle of the camera
TUCAM_REC_SAVE recSave	Video file saving structure

Error Code

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_INVALID_PARAM	Invalid parameters
TUCAMRET_INVALID_PATH	The input path does not exist
TUCAMRET_FAILOPEN_FILE	Failed to open the file
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

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

5.3.7.3. TUCAM_Rec_AppendFrame

Description

The [TUCAM_Rec_AppendFrame] is used to write the image data to a file, it must be called after the [TUCAM_Buf_WaitForFrame].

Statement

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

Parameters

HDTUCAM hTUCam	The handle of the camera
PTUCAM_FRAME pFrame	Pointer to the frame structure

Error Code

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_NOT_READY	TUCAM_Rec_Start interface is not called
TUCAMRET_OUT_OF_RANGE	The width and height of the written image does not match the created one
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

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

5.3.7.4. TUCAM_Rec_Stop

Description

The [TUCAM_Rec_Stop] is used to close the video file, when called it, the [TUCAM_Rec_AppendFrame] will not be able to write data.

Statement

TUCAMRET TUCAM_Rec_Stop (HDTUCAM hTUCam);

Parameters

HDTUCAM hTUCam	The handle of the camera
----------------	--------------------------

Error Code

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

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

5.3.8.Extended Control

5.3.8.1. TUCAM_Reg_Read

Description

Read the contents of the registers. The type of read refers to TUREG_TYPE.

Statement

```
TUCAMRET TUCAM_Reg_Read (HDTUCAM hTUCam, TUCAM_REG_RW regRW);
```

Parameters

HDTUCAM hTUCam	The handle of the camera
TUCAM_REG_RW regRW	Register read-write structure

Error Codes

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_NO_MEMORY	The incoming buffer has no memory space allocated
TUCAMRET_NOT_SUPPORT	This type of read is not supported
TUCAMRET_INVALID_IDPARAM	Invalid type, refer to TUREG_TYPE
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Open, TUCAM_Dev_Close
TUCAM_Reg_Write

5.3.8.2. TUCAM_Reg_Write

Description

Write the contents of the registers. The type of write, please refer to TUREG_TYPE.

Statement

```
TUCAMRET TUCAM_Reg_Write(HDTUCAM hTUCam, TUCAM_REG_RW regRW);
```

Parameters

HDTUCAM hTUCam	The handle of the camera
TUCAM_REG_RW regRW	Register read-write structure

Error Codes

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_NO_MEMORY	The incoming buffer has no memory space allocated
TUCAMRET_NOT_SUPPORT	This type of read is not supported
TUCAMRET_INVALID_IDPARAM	Invalid type, refer to TUREG_TYPE
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related APIs

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Open, TUCAM_Dev_Close
TUCAM_Reg_Read