TUCAM-API
Guide for Development

Fuzhou Xintu Photonics Co., Ltd.

Table of Contents

1. Read before use

The document and software example code are the internal document and release content of TUCSEN, which are provided for the purpose of enabling the user to create and use the Apps of TUCSEN Digital Camera.

This document and software sample code are disclosed for the above purpose only, which shall not constitute the owner's permission, transfer or any other power.

All risks and the results of using software documentation still depend on the user.

This document may contain technical or typographical errors, and it does not guarantee any damage resulting from such mistakes or text.

TUCSEN does not make commitment to update or to keep the information contained in the current document.

All brands and product names are the trademarks or registered trademarks of their respective owners.

TUCSEN reserves all rights of the document copyright.

In the absence of prior written permission of TUCSEN, any part of the document can not be reproduced, transmitted, transcribed or stored in a retrieval system or translated into any language or computer language, in any form or by any means and any methods such as electronic, mechanical, electromagnetic, optical, chemical, manual or other ones.

2. Brief Introduction

This manual describes the detailed specification of TUCAM-API as how to use TUCSEN Digital Camera. TUCAM-API Software Development Kit is called "SDK". The part of TUCAM-API controls the digital camera is called "module".

SDK contains source code modules and one sample application that demonstrates how to access TUCAM-API. SDK user can freely use the software in any way they like, such as partial modification of the source code or the creation of a completely separate project.

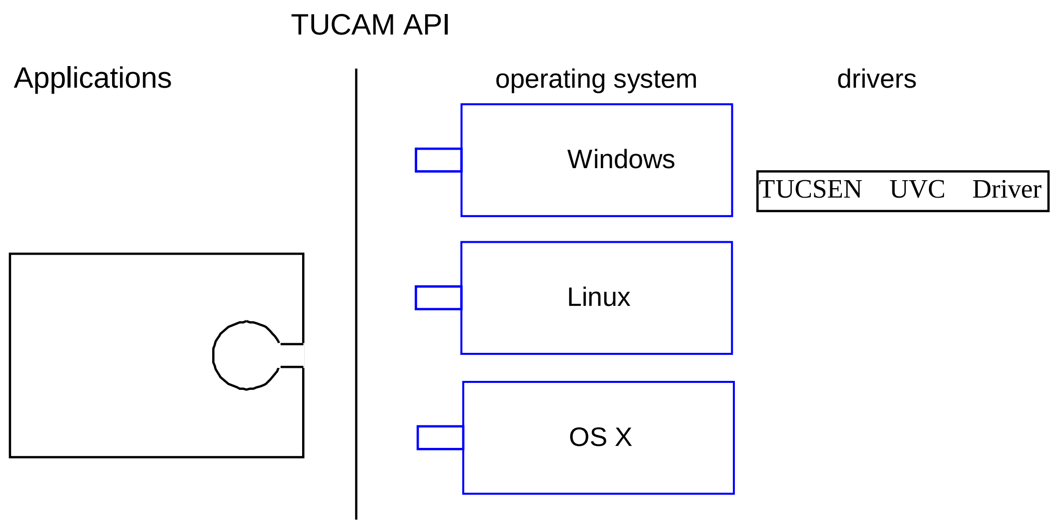
It is particularly easy to understand the SDK design. For this reason, it limits the number of function interfaces to a minimum, and the function call format is written in C language.

Parts of the extended functions are the additional features which can be used for certain digital cameras.

The values of the digital cameras may be different, which depend on the models of digital cameras used to capture images. The values should be simply seen as the guideline rather than the exact values.

3. Overview

3.1 Layer Structure



TUCSEN Digital Camera is connected with the drivers of digital camera of the different operating systems via SDK to achieve the functions of controlling the digital camera and capture of images and data.

The SDK currently supports only Windows systems.

3.2 Principle

The specific bus interfaces and libraries of digital cameras are packaged via TUCAM-API. You only need to access the TUCAM-API Layer. The Module Layer provides more advanced TUCAM-API integration. Modules can be constantly updated to access to the new cameras and provide new interface technology without recompiling of your software.

3.3 Interface Type

TUCAM-API functions can be divided into many types:
Start/End Processing;
Camera information collection;
Performance/property extraction and settings;
Memory management;
Capture control;
Document control;

Extending control.

TUCAM-API does not include the program for displaying images. Since it is difficult to predict some methods of displaying images, which depends on the application and it is impossible to support all of these common modules. When calling the display program, it will detect whether the image is updated, and it will draw the image after the update. For more detailed information, please refer to the sample source code.

3.4 Terminology

3.4.1 Capture Mode:

Camera capture modes are divided into the following 2 categories:

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".

3.4.2 Picture 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.

3.4.3 Trigger Mode:

Standard Mode: When the camera receives the level 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. Refer to TUCAM_TRIGGER_ATTR Structure.

Synchronization Mode: When the camera receives the level signal (determined by the activation edge), it will begin the exposure; after receiving the opposite level signal, it will end the exposure, and start the capture of image data. That is, the completion of exposure and read-out of each frame is completely synchronized with the external trigger signal.

Global Mode: Pre-trigger should be conducted prior to triggering the camera. When the camera receives a level signal (determined by the activation edge) or the exposure time set for the software, the reset operation currently in progress will be ended. When the exposure is ended, the image data will be captured, and the pre-trigger will be restarted. This mode is used to control the camera of shutter exposure mode to achieve global exposure mode.

Exposure Mode:
Exposure Time: after receiving the trigger signal, it is determined by the exposure time set by TUIDP_EXPOSURETM.
Level Width: after receiving the trigger signal, the exposure time is determined by the level width.
NOTE: The Standard Mode and Global Mode can configure both options while Synchronization Mode can only be the level width.

Types of Excitation Level:
Rising Edge: The trigger level starts exposure at the rising edge
Falling Edge: The trigger level starts exposure at the falling edge

3.4.4 Camera Status:

The camera status determines which functions can be called. Some functions will change the state of the camera. Four kinds of camera statuses are described as follows:

Unstable: parameter setting and call of other functions, but they are not in the status to be set.

Stable: the parameters and functions are set, but the image capture cannot be started as there is no frame memory created.

Preparation: frame memory has been created, and the image capture can be started.

Busy: image capturing is being executed.

3.5 Interface List

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

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

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

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

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

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

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

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

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

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

4. Application calls TUCAM-API

When TUCAM-API is being used to control camera, the function call should be executed in accordance with the following calling procedure:

- Initialize camera
- Set camera parameters
- Start capturing data
- Make sure that shooting has been completed and the data have been obtained
- Perform camera termination processing

4.1 Initialization and Termination of Procedures

4.1.1 Interface:

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

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

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

4.1.2 Calling Sequence:

First, the driver starts initialization. When the initialization of application installation and transmission handles has been successfully completed, the number of controllable cameras can be obtained.

When the application starts, it calls the camera initialization function to perform initialization. After the initialization function is successfully used, other functions can be called normally for execution.

The camera termination function is used for closing the program. 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, when exiting the application and the termination function is called, the calls of other performance functions will not be executed until after the initialization function is called again.

4.1.3 Driver Initialization:

Driver uses **TUCAM_Api_Init** function for the initialization operation. This function initialises frame grabbers and controls the camera.

4.1.4 Camera Initialization:

Camera initialization uses **TUCAM_Dev_Open** function. This function retrieves the necessary camera handles to use the input parameters for other functions.

4.1.5 Camera Product Information:

When the camera is opened after calling **TUCAM_Dev_Open** function, the camera product information can be obtained through the camera handle.

```
// enumerate information id
typedef enum
{
        TUIDI_BUS                   = 0x01,         // USB interface type: USB2.0/USB3.0
        TUIDI_VENDOR                = 0x02,         // Manufacturer ID
        TUIDI_PRODUCT               = 0x03,         // Product ID
        TUIDI_VERSION_API           = 0x04,         // TUCAM- API Version number
        TUIDI_VERSION_FRMW          = 0x05,         //Firmware version number
        TUIDI_VERSION_FPGA          = 0x06,         // FPGA Version number (reserved)
        TUIDI_VERSION_DRIVER        = 0x07,         // Driver version number (reserved)
        TUIDI_TRANSFER_RATE         = 0x08,         // USB transfer rate
        TUIDI_CAMERA_MODEL          = 0x09,         // Camera model (string type)
        TUIDI_ENDINFO               = 0x0A,         // Product information ID end bit
}TUCAM_IDINFO;
```

Example

```
        TUIDI_BUS                   = 0x300,        // USB3.0
        TUIDI_VENDOR                = 0x5453,       // TUCSEN
        TUIDI_PRODUCT               = 0x6404,       // Dhyana 400D
        TUIDI_VERSION_API           = “1.0.0.1”,        // “1.0.0.1”
        TUIDI_VERSION_FRMW          =-230244288 ,   // “f246c040”
        TUIDI_VERSION_FPGA          = 0,            // the FPGA version (reserved)
        TUIDI_VERSION_DRIVER        = “1.2.3.10”,   // “1.2.3.10”
        TUIDI_TRANSFER_RATE         = 292,          // 292MB / Sec
        TUIDI_CAMERA_MODEL          = “Dhyana 400D”, // “Dhyana 400D”
```

4.1.6 Termination Procedure:

The termination camera procedure uses **TUCAM_Dev_Close** function. Calling of this function releases the obtained port and the resources used for the camera frame. When this function is called, the camera will no longer be controlled.

4.1.7 Sample Code:

```
int main (int argc, char** argv)
{
    TUCAM_INIT    itApi;  // Initializing SDK environment parameters
    TUCMA_OPEN    opCam;  // Open camera parameters

    itApi.pstrConfigPath = NULL;
    itApi.uiCamCount = 0;
    if (TUCAMRET_SUCCESS != TUCAM_Api_Init(&itApi))
    {
        // Initializing SDK API environment failed
        return 0;
    }

    if (0 == itApi.uiCamCount)
    {
        // No camera
        return 0;
    }

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

    if (TUCAMRET_SUCCESS != TUCAM_Dev_Open(&opCam))
    {
        // Failed to open camera
        return 0;
    }

    // Application can use the handle of opCam.hIdxTUCam

    TUCAM_Dev_Close(opCam.hIdxTUCam);       // Close camera
    TUCAM_Api_Uninit();                     // Initializing SDK API environment

```

```
        return 0;
}

```

4.2 Performance Obtaining and Setting

4.2.1 Interface:

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

4.2.2 Calling Sequence:

The obtaining and setting are generally completed before or after the camera capture. If the function is set to be called when the data is captured, in some cases, it may return the error code TUCAMRET.

4.2.3 Performance Index:

```
// enumerate capability id
typedef enum
{
    TUIDC_RESOLUTION          = 0x00,        // Resolution
    TUIDC_PIXELCLOCK          = 0x01,        // Pixel clock
    TUIDC_BITOFDEPTH          = 0x02,        // Data bit wide
    TUIDC_ATEXPOSURE          = 0x03,        // Automatic exposure
    TUIDC_HORIZONTAL          = 0x04,        // Horizontal Flip
    TUIDC_VERTICAL            = 0x05,        // Vertical Flip
    TUIDC_ATWBBALANCE         = 0x06,        // Automatic white balance (color
camera)
    TUIDC_FAN_GEAR            = 0x07,        // Fan level (refrigeration camera)
    TUIDC_ENDCAPABILITY       = 0x08,        // Performance ID bit end
}TUCAM_IDCAPA;
```

Note: If the camera does not support the performance ID, it will return the error code TUCAMRET_NOT_SUPPORT.

4.2.3 Sample Code:

```
// Exampled with the resolution TUIDC_RESOLUTION
// Obtain the resolution scope
void GetResolutionRange()
{
    TUCAM_CAPA_ATTR attrCapa;
    TUCAM_VALUE_TEXT valText;

    char szRes[64] = {0};
    valText.nTextSize = 64;
    valText.pText = &szRes[0];
    attrCapa.idCapa = TUIDC_RESOLUTION;
    if (TUCAMRET_SUCCESS == TUCAM_Capa_GetAttr(opCam.hIdxTUCam, &attrCapa))
    {
        // Obtain the number of resolution
        int nCnt = attrCapa.nValMax - attrCapa.nValMin + 1;
        valText.nID = TUIDC_RESOLUTION;

        for (int i=0; i<nCnt; ++i)
        {
            valText.dbValue = i;
            TUCAM_Capa_GetValueText(opCam.hIdxTUCam, &valText);
            szRes = valText.pText;
            // Add resolution text to the drop-down menu
        }
    }
}

// Obtain the current resolution
void GetCurrentResolution()
{
    int nVal = 0;

    if (TUCAMRET_SUCCESS == TUCAM_Capa_GetValue(opCam.hIdxTUCam, \
                                                TUIDC_RESOLUTION, \
                                                &nVal))
    {
        // nVal Returns to the current resolution index
    }
}

// Set the current resolution
void SetCurrentResolution(int nIdxRes)
```

```
{
    TUCAM_Capa_SetValue(opCam.hIdxTUCam, TUIDC_RESOLUTION, nIdxRes);
}

```

4.3 Property Extraction and Settings

4.3.1 Interface:

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

4.3.2 Calling Sequence:

The obtaining and setting are generally completed before or after the camera capture. If the function is set to be called when the data is captured, in some cases, it may return the error code TUCAMRET.

4.3.3 Property Index:

```
// enumerate property id
typedef enum
{
    TUIDP_GLOBALGAIN          = 0x00,          // Global gain
    TUIDP_EXPOSURETM          = 0x01,          // Exposure time
    TUIDP_BRIGHTNESS          = 0x02,          // Brightness (valid under AE status)
    TUIDP_BLACKLEVEL          = 0x03,          // Black level
    TUIDP_TEMPERATURE         = 0x04,          // Temperature
    TUIDP_SHARPNESS           = 0x05,          // Sharpness
    TUIDP_NOISELEVEL          = 0x06,          // Noise level
    TUIDP_HDR_KVALUE          = 0x07,          // HDR K value (for sCMOS cameras)

    // image process property
    TUIDP_GAMMA               = 0x08,          // Gamma
    TUIDP_CONTRAST            = 0x09,          // Contrast
    TUIDP_LFTLEVELS           = 0x0A,          // Left Levels
    TUIDP_RGTLEVELS          = 0x0B,          // Right Levels
    TUIDP_CHNLGAIN           = 0x0C,          // Channel gain (for color cameras)
    TUIDP_SATURATION         = 0x0D,          // Saturation (for color cameras)
    TUIDP_ENDPROPERTY        = 0x0E,          // Property ID end bit
}TUCAM_IDPROP;
```

Note: If the camera does not support the Property ID, it will return the error code TUCAMRET_NOT_SUPPORT.

4.3.3 Sample Code:

```
// Exampled with exposure time
// Obtain exposure time range
void GetExposureTimeRange()
{
    TUCAM_PROP_ATTR attrProp;

    attrProp.nIdxChn = 0;       // Current channel
    attrProp.idProp = TUIDP_EXPOSURETM;

    if (TUCAMRET_SUCCESS == TUCAM_Prop_GetAttr(opCam.hIdxTUCam, &attrProp))
    {
        // Exposure time range
        attrProp.dbValMin;      // Minimum exposure time
        attrProp.dbValMax;      // Maximum exposure time
        attrProp.dbValDft;      // Default exposure time
        attrProp.dbValStep;     // Exposure time step
    }
}

// Obtain current exposure time
void GetCurrentExposureTime()
{
    double dbVal = 1.0f;

    if (TUCAMRET_SUCCESS == TUCAM_Prop_GetValue(opCam.hIdxTUCam, \
                                                TUIDP_EXPOSURETM, \
                                                &dbVal))
    {
        // dbVal returns the current exposure time in ms
    }
}

```

```
// Set current exposure time
void SetCurrentExposureTime(double dbVal)
{
    TUCAM_Prop_SetValue(opCam.hIdxTUCam, TUIDP_EXPOSURETM, dbVal);
}

```

4.4 Memory management

4.4.1 Interface:

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

4.4.2 Calling Sequence:

For memory allocation and release, the memory allocation **TUCAM_Buf_Alloc** must be called before **TUCAM_Cap_Start** starts data capture; while memory release **TUCAM_Buf_Release** must be called after **TUCAM_Cap_Stop** stops the data capture.

For the data capture, **TUCAM_Buf_WaitForFrame** must be called after **TUCAM_Cap_Start** starts the data capture
Wait for the completion of data capture, and data in different formats can be copied via **TUCAM_Buf_CopyFrame**.

If there are calls of data waiting and data copying, prior to stopping the data capture, **TUCAM_Buf_AbortWait** is called to stop data waiting, and then **TUCAM_Cap_Stop** is called to stop data capture.

4.4.3 Frame Structure:

```
// the camera frame structure
typedef struct    _tagTUCAM_FRAME
{
    // TUCAM_Buf_WaitForFrame Using this structure enables a number of member variables of
different call directions.
    // "input" means that the application is called before setting
    // "output"means the value required is returned after the program calls the interface

    CHAR szSignature[8];        //   [out] Copyright information

    //   The based information
    USHORT usHeader;            //   [out] Head size of frame
    USHORT usOffset;            //   [out] Head offset of frame data (generally in the same size as the
head)
    USHORT usWidth;             //   [out] Frame width
    USHORT usHeight;            //   [out] Frame height
    UINT32 uiWidthStep;         //   [out] Frame width step

    UCHAR   ucDepth;            //   [out] Frame data bit depth
    UCHAR   ucFormat;           //   [out] Frame data format
    UCHAR   ucChannels;         //   [out] Number of frame data channel
    UCHAR   ucElemBytes;        //   [out] Frame pixel bytes
    UCHAR   ucFormatGet;        //   [in]   Obtain frame format (refer to TUFRM_FORMATS)

    UINT32 uiIndex;             //   [in/out] Current frame number
    UINT32 uiImgSize;           //   [out] Frame size
    UINT32 uiRsdSize;           //   [in]   Number of reserved frames (number of frames needed, trigger
use)
    UINT32 uiHstSize;           //   [out] Frame histogram frame size (reserved bit)

    PUCHAR pBuffer;             //   [in/out] Frame buffer
} TUCAM_FRAME, *PTUCAM_FRAME;

// enumerate frame format
typedef enum
{
    TUFRM_FMT_RAW               = 0x10,             // Data of Raw format
    TUFRM_FMT_USUAl             = 0x11,             // General data (8bit/16bit, monochrome,
color)
    TUFRM_FMT_RGB888            = 0x12,             // Data of RGB888 for display
}TUFRM_FORMATS;
```

4.4.4 Sample Code:

```
TUCAM_FRAME m_frame;          // Frame object
HANDLE m_hThdGrab;            // Event handler of picture grabbing
BOOL m_bLiving;               // Confirm to grab picture

BOOL CDlgTUCam::StartCapture()
{
        m_frame.pBuffer         = NULL;
        m_frame.ucFormatGet = TUFRM_FMT_RGB888;    // Frame data format (RGB888)
        m_frame.uiRsdSize      = 1;       // Number of frames captured once (TUCCM_TRIGGER_STANDARD may be greater than 1)

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

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

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

        return TRUE;
}

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

        While (pTUCam->m_bLiving)
        {
                pTUCam->m_frame.ucFormatGet = TUFRM_FMT_RGB888;
                if(TUCAMRET_SUCCESS == TUCAM_Buf_WaitForFrame(pTUCam->m_opCam.hIdxTUCam,\
                                                                                                                &pTUCam->m_frame))
                {
                        // pTUCam->m_frame.pBuffer Return the captured image data in the format of TUFRM_FMT_RGB88
                        // The data can be used to display

                        // Data in other formats can be obtained
                        pTUCam->m_frame.ucFormatGet = TUFRM_FMT_USUAL;

```

```
    if(TUCAMRET_SUCCESS==TUCAM_Buf_CopyFrame(pTUCam->m_opCam.hIdxTUCam,\
                                                &pTUCam->m_frame))
                {
                    // pTUCam->m_frame.pBuffer Return the captured image data
                }
            }
        }

        SetEvent(pTUCam->m_hThdGrab);
        _endthread();
    }

    void CDlgTUCam::StopCapture()
    {
        m_bLiving = FALSE;
        TUCAM_BUF_AbortWait();      // If calls TUCAM_Buf_WaitForFrame Interface

        WaitForSingleObject(m_hThdGrab, INFINITE);      // Wait for exiting of picture grabbing
        CloseHandle(m_hThdGrab);
        m_hThdGrab = NULL;

        TUCAM_Cap_Stop(m_opCam.hIdxTUCam);              // Stop data capture
        TUCAM_Buf_Release(m_opCam.hIdxTUCam);       // Release the allocated memory
    }

```

4.5 Capture Control

4.5.1 Interface:

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

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

4.5.2 Calling Sequence:

Set ROI property **TUCAM_Cap_SetROI** and trigger property **TUCAM_Cap_SetTrigger**, which needs to be called before start capturing data; if called when capturing data, the error code TUCAMRET may be returned.

4.5.2 Capture Mode Index:

```
// enumerate the capture mode
typedef enum
{
    TUCCM_SEQUENCE                  = 0x00,     // Sequence mode for data capture
    TUCCM_TRIGGER_STANDARD          = 0x01,     // Standard trigger mode for data capture
    TUCCM_TRIGGER_SYNCHRONOUS  = 0x02,     //    Synchronization trigger mode for data
capture
    TUCCM_TRIGGER_GLOBAL            = 0x03,     //    Global trigger mode for data capture
    TUCCM_TRIGGER_SOFTWARE          = 0x04,     //    Software trigger mode for data
capture
}TUCAM_CAPTURE_MODES;
```

4.5.3 Sample Code:

```
// Set ROI mode
void SetROIMode()
{
    TUCAM_ROI_ATTR roiAttr;
    roiAttr.bEnable = TRUE;
    roiAttr.nVOffset= 100;
    roiAttr.nHOffset = 100;
    roiAttr.nWidth    = 800;
    roiAttr.nHeight   = 600;

    TUCAM_Cap_SetROI(m_opCam.hIdxTUCam, roiAttr);
    TUCAM_Cap_Start(m_opCam.hIdxCam, TUCCM_SEQUENCE);        // Sequence mode
(stream mode)

    // Refer to memory management sample code for data obtaining
}

// Set trigger mode
```

```
void SetTriggerMode()
{
      TUCAM_TRIGGER_ATTR tgrAttr;

      tgrAttr.nTgrMode = TUCCM_TRIGGER_STANDARD;     // Standard trigger mode
      tgrAttr.nExpMode = TUCTE_EXPTM;                // Exposure mode
      tgrAttr.nEdgeMode= TUCTE_RISING;               // Stimulate rising edge
      tgrAttr.nFrames   = 1;                         // Trigger one frame
      tgrAttr.nDelayTm = 0;                          // Delay 0 ms

      TUCAM_Cap_SetTrigger(m_opCam.hIdxTUCam, tgrAttr);
      TUCAM_Cap_Start(m_opCam.hIdxCam, TUCCM_STANDARD);       // Standard trigger mode

      // Refer to memory management sample code for data obtaining
}

```

4.6 File Control

4.6.1 Interface:

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

4.6.2 Calling Sequence:

Starting of video recording **TUCAM_Rec_Start** needs to be called after **TUCAM_Cap_Start** starts capturing data, and the set capture mode must be TUCCM_SEQUENCE mode. In the video recording process, by calling **TUCAM_Rec_AppendFrame**, the image data are written into the file, the end of the video recording calls **TUCAM_Rec_Stop** to end the video recording process.

4.6.3 File Structure:

```
// the file save structure
```

```
typedef struct _tagTUCAM_FILE_SAVE
{
    INT32       nSaveFmt;                 //  [in] Format of saved file (refer to TUIMG_FORMATS)
    PCHAR       pstrSavePath;             //  [in] Path of saved file (including file name, but not including the extension name)
    PTUCAM_FRAME pFrame;                  //  [in] Frame structure pointer
} TUCAM_FILE_SAVE, *PTUCAM_FILE_SAVE;

// the record save structure
typedef struct _tagTUCAM_REC_SAVE
{
    INT32       nCodec;                   //  [in] Codec type
    PCHAR       pstrSavePath;             //  [in] Path of saved file (including file name, but not including the extension name)
    Float       fFps;                     //  [in] Current frame rate (video frame rate)
} TUCAM_REC_SAVE, *PTUCAM_REC_SAVE;
```

4.6.4 Sample Code:

```
// Save image file
void SaveImage()
{
    m_frame.ucFormatGet = TUFRM_FMT_USUAL;
    if(TUCAMRET_SUCCESS==TUCAM_Buf_WaitForFrame(m_opCam.hIdxTUCam, &m_frame))
    {
        TUCAM_FILE_SAVE fileSave;
        fileSave.nSaveFmt       = TUFMT_TIF;        // Save Tiff format
        fileSave.pFrame         = &m_frame;         // Frame pointer needs to be saved
        fileSave.pstrSavePath = "C:\\image";        // Path including file name (not including extension name)

        if (TUCAMRET_SUCCESS == TUCAM_File_SaveImage(m_opCam.hIdxTUCam, fileSave))
        {
            // Image file saved successfully
        }
    }
}

// Save video file
void StartRecording()
{
    TUCAM_REC_SAVE    recSave;
    recSave.fFps              = 15.0f;                     // Frame rate needs to be saved
    recSave.nCodec            = m_dwFccHandler;
    recSave.pstrSavePath = "C:\\TUVideo.avi"  // Full path

    if (TUCAMRET_SUCCESS == TUCAM_Rec_Start(m_opCam.hIdxTUcam, recSave))
    {
        // Start video recording. . .
    }
}

Void AppendFrame()
{
    m_frame.ucFormatGet = TUFRM_FMT_RGB888;
    if(TUCAMRET_SUCCESS==TUCAM_Buf_WaitForFrame(m_opCam.hIdxTUCam,
&m_frame))
    {
        TUCAM_Rec_AppendFrame(m_opCam.hIdxTUCam, &m_frame);
    }
}

void StopRecording()
{
    TUCAM_Rec_Stop(m_opCam.hIdxTUCam);
}

```

4.7 Extending Control

4.7.1 Interface:

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

4.7.2 Calling Sequence:

Read/write register of **TUCAM_Reg_Read / TUCAM_Reg_Write** must be called after **TUCAM_Dev_Open** opens the camera; if the register cannot be read and written after **TUCAM_Dev_Close** turns off the camera, the camera must be reopened.

4.7.3 File Structure:

```
// the register read/write struct
typedef struct _tagTUCAM_REG_RW
{
    INT32      nRegFmt;                 //   [in] Type of read/write register (refer to TUREG_TYPE)
    PCHAR      pBuf;                    //   [in/out] Pointer to the buffer
    INT32      nBufSize;                //   [in] Buffer size
} TUCAM_REG_RW, *PTUCAM_REG_RW;
```

4.7.4 Sample Code:

```
// Read register data
void ReadRegisterData()
{
    char cSN[TUSN_SIZE] = {0};
    TUCAM_REG_RW regRW;

    regRW.nRegType    = TUREG_SN;
    regRW.pBuf        = &cSN[0];
    regRW.nBufSize    = TUSN_SIZE;

    if (TUCAMRET_SUCCESS == TUCAM_Reg_Read(m_opCam.hIdxTUcam, regRW))
    {
        // Obtain SN data
    }
}

// Write register data
void WriteRegisterData()
{
    char cSN[TUSN_SIZE] = {‘S’, ‘N’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’};// “SN123456”
    TUCAM_REG_RW regRW;

    regRW.nRegType    = TUREG_SN;
    regRW.pBuf        = &cSN[0];
    regRW.nBufSize    = TUSN_SIZE;

    if (TUCAMRET_SUCCESS == TUCAM_Reg_Write(m_opCam.hIdxTUcam, regRW))
    {
        // SN successfully written into register
    }
}
```

5. Reference

5.1 Types and Constants

5.1.1 TUCAMRET Error code:

```
    TUCAMRET_SUCCESS                = 0x00000001,      // No error, general success code
    TUCAMRET_FAILURE                = 0x80000000,      // Error

    //  initialization error
    TUCAMRET_NO_MEMORY              = 0x80000101,      // Not enough memory
    TUCAMRET_NO_RESOURCE            = 0x80000102,      // Not enough resources (not including
memory)
    TUCAMRET_NO_MODULE              = 0x80000103,      // No sub-module
    TUCAMRET_NO_DRIVER              = 0x80000104,      // No driver
    TUCAMRET_NO_CAMERA              = 0x80000105,      // No camera
    TUCAMRET_NO_GRABBER             = 0x80000106,      // No picture grabber
    TUCAMRET_NO_PROPERTY            = 0x80000107,      // No alternative property ID

    TUCAMRET_FAILOPEN_CAMERA        = 0x80000110,       // Failed to open camera
    TUCAMRET_FAILOPEN_BULKIN             = 0x80000111,       // Failed to open batch
transmission input terminal
    TUCAMRET_FAILOPEN_BULKOUT       = 0x80000112,       // Failed to open batch transmission
output terminal
    TUCAMRET_FAILOPEN_CONTROL       = 0x80000113,       // Failed to open control endpoint
    TUCAMRET_FAILCLOSE_CAMERA       = 0x80000114,       // Failed to close camera
    TUCAMRET_FAILOPEN_FILE          = 0x80000115,       // Failed to open file

    //  status error
    TUCAMRET_INIT                   = 0x80000201,      //  API needs to initialize state.
    TUCAMRET_BUSY                   = 0x80000202,      //  API 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 standby

    //  wait error
    TUCAMRET_ABORT                  = 0x80000207,       // Processing aborted
    TUCAMRET_TIMEOUT                = 0x80000208,       // Timeout
    TUCAMRET_LOSTFRAME              = 0x80000209,       // Frame loss
    TUCAMRET_MISSFRAME              = 0x8000020A,       // Frame loss but it is caused by
underlying driver issue

    // calling error
    TUCAMRET_INVALID_CAMERA        = 0x80000301,     // Invalid camera
    TUCAMRET_INVALID_HANDLE        = 0x80000302,     // Invalid camera handle
    TUCAMRET_INVALID_OPTION        = 0x80000303,     // Invalid configuration value
    TUCAMRET_INVALID_IDPROP        = 0x80000304,     // Invalid property ID
    TUCAMRET_INVALID_IDCAPA        = 0x80000305,    // Invalid performance ID
    TUCAMRET_INVALID_IDPARAM       = 0x80000306,    // Invalid parameter ID
    TUCAMRET_INVALID_PARAM         = 0x80000307,    // Invalid parameter
    TUCAMRET_INVALID_FRAMEIDX      = 0x80000308,    // Invalid frame IDX
    TUCAMRET_INVALID_VALUE         = 0x80000309,    // Invalid value
    TUCAMRET_INVALID_EQUAL         = 0x8000030A,    // Values are equal, but the parameter is
invalid
    TUCAMRET_INVALID_CHANNEL       = 0x8000030B,    // Specified channel of property ID, but
the channel is not valid
    TUCAMRET_INVALID_SUBARRAY      = 0x8000030C,    // Value of the sub-array is invalid

    TUCAMRET_INVALID_VIEW          = 0x8000030D,    // Invalid display window handle
    TUCAMRET_INVALID_PATH          = 0x8000030E,    // Invalid file path

    TUCAMRET_NO_VALUETEXT          = 0x80000310,     // Text without property value
    TUCAMRET_OUT_OF_RANGE          = 0x80000311,     // Value is out of range

    TUCAMRET_NOT_SUPPORT           = 0x80000312,     // Unsupported features or properties
    TUCAMRET_NOT_WRITABLE          = 0x80000313,     // Property unwritable
    TUCAMRET_NOT_READABLE          = 0x80000314,     // Property unreadable


    TUCAMRET_WRONG_HANDSHAKE  = 0x80000410,          // Error occurred while retrieving the
error code
    TUCAMRET_NEWAPI_REQUIRED       = 0x80000411,     // Old API does not support, only new
API supports
    TUCAMRET_ACCESSDENY            = 0x80000412,     // Camera cannot be accessed under
certain state

    TUCAMRET_NO_CORRECTIONDATA = 0x80000501,        // No color dot correction of data.

    //   camera or bus trouble
    TUCAMRET_FAIL_READ_CAMERA      = 0x83001001,     // Failed to read from camera
    TUCAMRET_FAIL_WRITE_CAMERA     = 0x83001002,     // Failed to write into camera
    TUCAMRET_OPTICS_UNPLUGGED      = 0x83001003,     // Unplugged
```

5.1.2 TUCAM_IDINFO Product Information Code:

```
     TUIDI_BUS                     = 0x01,          // USB interface type:   USB2.0/USB3.0
     TUIDI_VENDOR                  = 0x02,          // Manufacturer ID
```

```
TUIDI_PRODUCT                 = 0x03,          // Product ID
TUIDI_VERSION_API             = 0x04,          // TUCAM- API Version number
TUIDI_VERSION_FRMW            = 0x05,          // Firmware version number
TUIDI_VERSION_FPGA            = 0x06,          // FPGA Version number (reserved)
TUIDI_VERSION_DRIVER          = 0x07,          // Driver version number (reserved)
TUIDI_TRANSFER_RATE           = 0x08,          // USB transfer rate
TUIDI_CAMERA_MODEL            = 0x09,          // Camera model (string type)
TUIDI_ENDINFO                 = 0x0A,          // Product information ID end bit
```

5.1.3 TUCAM_IDCAPA Performance Code:

```
    TUIDC_RESOLUTION              = 0x00,          // Resolution
    TUIDC_PIXELCLOCK              = 0x01,          // Pixel clock
    TUIDC_BITOFDEPTH              = 0x02,          // Data bit wide
    TUIDC_ATEXPOSURE              = 0x03,          // Automatic exposure
    TUIDC_HORIZONTAL              = 0x04,          // Horizontal Flip
    TUIDC_VERTICAL                = 0x05,          // Vertical Flip
    TUIDC_ATWBBALANCE             = 0x06,          // Automatic white balance (color
camera)
    TUIDC_FAN_GEAR                = 0x07,          // Fan level (refrigeration camera)
    TUIDC_ENDCAPABILITY           = 0x08,          // Performance ID bit end
```

5.1.4 TUCAM_IDPROP Property Code:

```
TUIDP_GLOBALGAIN              = 0x00,          // Global gain
TUIDP_EXPOSURETM              = 0x01,          // Exposure time
TUIDP_BRIGHTNESS              = 0x02,          // Brightness (valid under AE status)
TUIDP_BLACKLEVEL              = 0x03,          // Black level
TUIDP_TEMPERATURE             = 0x04,          // Temperature
TUIDP_SHARPNESS               = 0x05,          // Sharpness
TUIDP_NOISELEVEL              = 0x06,          // Noise level
TUIDP_HDR_KVALUE              = 0x07,          // HDR K value (for sCMOS cameras)

// image process property
TUIDP_GAMMA                   = 0x08,          // Gamma
TUIDP_CONTRAST                = 0x09,          // Contrast
TUIDP_LFTLEVELS               = 0x0A,          // Left Levels
TUIDP_RGTLEVELS               = 0x0B,          // Right Levels
TUIDP_CHNLGAIN                = 0x0C,          // Channel gain (for color cameras)
TUIDP_SATURATION              = 0x0D,          // Saturation (for color cameras)
TUIDP_ENDPROPERTY             = 0x0E,          // Property ID end bit
```

5.1.5 TUCAM_CAPTURE_MODES Capture Mode Code:

```
    TUCCM_SEQUENCE                  = 0x00,          // Sequence mode (stream mode)
    TUCCM_TRIGGER_STANDARD          = 0x01,          // Standard trigger mode
    TUCCM_TRIGGER_SYNCHRONOUS       = 0x02,          // Synchronization trigger mode
    TUCCM_TRIGGER_GLOBAL            = 0x03,          // Global trigger mode
    TUCCM_TRIGGER_SOFTWARE          = 0x04,          // Software trigger mode
```

5.1.6 TUIMG_FORMATS Image Format Code:

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

5.1.7 TUREG_TYPE Register Type Code:

```
     TUREG_SN                       = 0x01,          // Read and write SN code register
     TUREG_DATA                     = 0x02,          // Read and write DATA register
(reserved)
```

5.1.8 TUCAM_TRIGGER_EXP Trigger Exposure Mode Code:

```
     TUCTE_EXPTM                    = 0x00,          // Exposure time mode for trigger
     TUCTE_WIDTH                    = 0x01,          // Level width mode for trigger
```

5.1.9 TUCAM_TRIGGER_EDGE Trigger Excitation Edge Code:

```
     TUCTD_RISING                   = 0x01,          // Stimulate rising edge
     TUCTD_FAILING                  = 0x00,          // Stimulate falling edge
```

5.1.10 TUFRM_FORMATS Frame Format Code:

```
     TUFRM_FMT_RAW                  = 0x10,          // RAW data format
     TUFRM_FMT_USUAl                = 0x11,          // Common format data (8bit/16bit,
monochrome/color)
     TUFRM_FMT_RGB888               = 0x12,          // RGB888 format data (can be used for
display)
```

5.2 Structure

5.2.1 Initialization:

```
// the camera initialize structure
typedef struct _tagTUCAM_INIT
{
      UINT32    uiCamCount;                     //   [out] Return the number of the current
connected cameras
      PCHAR    pstrConfigPath;                  //   [in] Camera parameters input saving path
}TUCAM_INIT, *PTUCAM_INIT;
```

5.2.2 Open Camera:

```
//   the camera open structure
typedef struct _tagTUCAM_OPEN
{
      UINT32    uiIdxOpen;                      //   [in]   Input serial number of camera to be
opened
      HDTUCAM hIdxTUCam;                        //   [out] Output handles of opened cameras
}TUCAM_OPEN, *PTUCAM_OPEN;
```

5.2.3 Camera Information:

```
// the camera value text structure
typedef struct _tagTUCAM_VALUE_INFO
{
      INT32     nID;                            //   [in] Information ID TUCAM_IDINFO
      INT32     nValue;                                //   [in] Information value
      PCHAR    pText;                           //   [in/out] Pointer pointing to text data
      INT32          nTextSize;                        //   [in] Text buffer size
}TUCAM_VALUE_INFO, *PTUCAM_VALUE_INFO;
```

5.2.4 Performance/Property Value Text:

```
// the camera value text structure
typedef struct _tagTUCAM_VALUE_TEXT
{
      INT32      nID;                           //   [in]   ID TUCAM_IDPROP / TUCAM_IDCAPA
      DOUBLE   dbValue;                         //   [in] Performance/property value
      PCHAR     pText;                                 //   [in/out] Pointer pointing to text data
      INT32           nTextSize;                       //   [in] Text buffer size
}TUCAM_VALUE_TEXT, *PTUCAM_VALUE_TEXT;
```

5.2.5 Performance Property:

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

    INT32     nValMin;                   //  [out] Minimum value
    INT32     nValMax;                   //  [out] Maximum value
    INT32     nValDft;                        //  [out] Default value
    INT32     nValStep;                  //  [out] Step length
}TUCAM_CAPA_ATTR, *PTUCAM_CAPA_ATTR;
```

5.2.6 Property Attribute:

```
// the camera property attribute
typedef struct _tagTUCAM_PROP_ATTR
{
    INT32     idProp;                    //  [in]   ID TUCAM_IDPROP
    INT32     nIdxChn;                   //  [in/out] Index number of current channel
    DOUBLE    dbValMin;                  //  [out] Minimum value
    DOUBLE    dbValMax;                  //  [out] Maximum value
    DOUBLE    dbValDft;                  //  [out] Default value
    DOUBLE    dbValStep;                 //  [out] Step length
}TUCAM_PROP_ATTR, *PTUCAM_PROP_ATTR;
```

5.2.7 ROI Attribute:

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

    INT32     nHOffset;                  //  [in/out]   Horizontal offset
    INT32     nVOffset;                  //  [in/out]   Vertical offset
    INT32     nWidth;                    //  [in/out]   ROI width
    INT32     nHeight;                   //  [in/out]   ROI height
}TUCAM_ROI_ATTR, *PTUCAM_ROI_ATTR;
```

5.2.8 Trigger Attribute:

```
// the camera trigger attribute
typedef struct _tagTUCAM_TRIGGER_ATTR
{
    INT32     nTgrMode;                        //   [in/out] Trigger mode
    INT32     nExpMode;                        //   [in/out] Exposure mode value [0,1] 0: Exposure
Time    1: level width
    INT32     nEdgeMode;                       //   [in/out] Edge excitation mode [0, 1] 0: Falling
edge    1: Rising edge
    INT32     nDelayTm;                        //   [in/out] Trigger delay time ms
    INT32     nFrames;                         //   [in/out] Number of output frames of one trigger
}TUCAM_TRIGGER_ATTR, *PTUCAM_TRIGGER_ATTR;
```

5.2.9 Frame Structure:

```
// the camera frame structure
typedef struct _tagTUCAM_FRAME
{
    CHAR szSignature[8];                       //   [out] Copyright information

    //   The based information
    USHORT usHeader;                           //   [out] Head size of frame
    USHORT usOffset;                           //   [out] Offset size of frame data
    USHORT usWidth;                            //   [out] Width of frame image
    USHORT usHeight;                           //   [out] Height of frame image
    UINT32 uiWidthStep;                        //   [out] Frame image width step

    UCHAR    ucDepth;                          //   [out] Frame image data bit depth
    UCHAR    ucFormat;                         //   [out] Frame image data format
    UCHAR    ucChannels;                       //   [out] Number of frame image channel
    UCHAR    ucElemBytes;                      //   [out] Frame image data byte
    UCHAR    ucFormatGet;                      //   [in/out] Image format needs to be obtained
TUFRM_FORMATS

    UINT32 uiIndex;                            //   [out] Frame image serial number (reserved)
    UINT32 uiImgSize;                          //   [out] Frame image data size
    UINT32 uiRsdSize;                          //   [in]    Number of frame needs to be obtained
    UINT32 uiHstSize;                          //   [out] Reserved field of frame image

    PUCHAR pBuffer;                            //   [in/out] Buffer pointing to frame data
} TUCAM_FRAME, *PTUCAM_FRAME;
```

5.2.10 File Saving:

```
// the file save structure
typedef struct _tagTUCAM_FILE_SAVE
{
    INT32     nSaveFmt;                        //   [in] Format of saved file refer to
TUIMG_FORMATS
        PCHAR       pstrSavePath;                   //  [in] Path of saving (Not including the extension
name)

        PTUCAM_FRAME pFrame;                        //  [in] Structure pointing to the frame
} TUCAM_FILE_SAVE, *PTUCAM_FILE_SAVE;
```

5.2.11 Video Recording Saving:

```
// the record save structure
typedef struct _tagTUCAM_REC_SAVE
{
        INT32       nCodec;                         //  [in] Codec type
        PCHAR       pstrSavePath;                   //  [in] Path of saved file including file name
        float       fFps;                           //  [in]    Frame rate needs to be saved
} TUCAM_REC_SAVE, *PTUCAM_REC_SAVE;
```

5.2.12 Read/Write of Register

```
// the register read/write structure
typedef struct _tagTUCAM_REG_RW
{
        INT32       nRegType;                       //  [in] Type of read/write register refer to
TUREG_TYPE

        PCHAR       pBuf;                           //  [in/out] Point to the buffer of read/write contents
        INT32       nBufSize;                       //  [in]    Buffer size
} TUCAM_REG_RW, *PTUCAM_REG_RW;
```

5.3 Functions

TUCAM_Api_Init

Description

Initialization of TUCAM-API library includes binding of driver and initialization of some internal resources, which is used before calling the other interfaces. The whole program only needs to call once.

Statement

```
TUCAMRET    TUCAM_Api_Init(PTUCAM_INIT pInitParam);
```

Parameters

PTUCAM_INIT pInitParam   Initialization of structure pointer, refer to structure TUCAM_INIT

Error codes

TUCAMRET_INIT	TUCAM-API has been initialized

Related Interfaces

TUCAM_Api_Uninit

TUCAM_Api_Uninit

Description

Uninstallation of TUCAM-API library includes the release of driver binding and some internal resources. It will be called once when the entire program is ended.

Statement

```
TUCAMRET    TUCAM_Api_Uninit ();
```

Parameters

No parameters input

Error codes

TUCAMRET_NOT_INIT TUCAM-API not initialized

Related Interfaces

TUCAM_Api_Uninit

TUCAM_Dev_Open

Description

Open the camera, the camera is in work mode after the call, which can respond to the calls of other interfaces. The camera should be ensured prior to that, that is, it should be after the initialization of calling **TUCAM_Api_Init**.

Statement

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

Parameters

PTUCAM_OPEN pOpenParam Open camera structure pointer, refer to structure TUCAM_OPEN

Error Codes

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 camera index needs to be opened exceeds the range of connected cameras
TUCAMRET_FAILOPEN_CAMERA	Failed to open camera
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit

TUCAM_Dev_Close

TUCAM_Dev_Close

Description

Close the camera and the camera is in standby mode after called, and does not respond to calls from other interfaces.

Statement

```
TUCAMRET    TUCAM_Dev_Close ();
```

Parameters

No parameters input

Error Codes

TUCAMRET_NOT_INIT	TUCAM-API not initialized

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Open

TUCAM_Dev_GetInfo

Description

Obtain the related information of camera, such as a USB port type, camera product number, API version number, firmware version number, camera type, etc. The camera should be ensured prior to that, and make sure the camera is opened, that is, it should be done after calling **TUCAM_Api_Init** initialization and **TUCAM_Api_Open**.

Statement

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

Parameters

HDTUCAM hTUCam Camera handle

PTUCAM_VALUE_INFO pInfo Structure pointer of camera information value, refer to TUCAM_VALUE_INFO

Error Codes

TUCAMRET_NOT_INIT TUCAM-API not initialized

TUCAMRET_INVALID_PARAM Invalid parameter, when product information code does not exist, refer to TUCAM_IDINFO

TUCAMRET_INVALID_CAMERA Invalid camera, when the camera handle does not exist

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit

TUCAM_Dev_Open, TUCAM_Dev_Close

TUCAM_Capa_GetAttr

Description

Obtain the attribute value of performance parameters. The attributes obtained include the minimum value, maximum value, the default value and step of the parameter. The specific supported performance type can be referred to TUCAM_IDCAPA.

Statement

```
TUCAMRET   TUCAM_Capa_GetAttr (HDTUCAM hTUCam, PTUCAM_CAPA_ATTR pAttr);
```

Parameters

HDTUCAM hTUCam Camera handle

PTUCAM_CAPA_ATTR pAttr Camera performance attribute structure pointer, refer to TUCAM_CAPA_ATTR

Error Codes

TUCAMRET_NOT_INIT TUCAM-API not initialized

TUCAMRET_INVALID_IDCAPA Invalid parameter, when product information code does not exist, refer to TUCAM_IDCAPA

TUCAMRET_INVALID_VALUE Invalid value, when 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 Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Open, TUCAM_Dev_Close
TUCAM_Capa_GetValue, TUCAM_Capa_SetValue, TUCAM_Capa_GetValueText

TUCAM_Capa_GetValue

Description

Obtain the current attribute value of performance parameters. The specific supported performance type can be referred to TUCAM_IDCAPA.

Statement

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

Parameters

HDTUCAM hTUCam	Camera handle
INT32 nCapa	Camera performance attribute ID, refer to TUCAM_IDCAPA
INT32 *pnVal	Return the current value

Error Codes

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_INVALID_IDCAPA	Invalid parameter, when product information code does not exist, refer to TUCAM_IDCAPA
TUCAMRET_INVALID_VALUE	Invalid value, when 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 Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Open, TUCAM_Dev_Close
TUCAM_Capa_GetAttr, TUCAM_Capa_SetValue, TUCAM_Capa_GetValueText

TUCAM_Capa_SetValue

Description

Set the current attribute value of performance parameters. The specific supported performance type can be referred to TUCAM_IDCAPA.

Statement

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

Parameters

HDTUCAM hTUCam	Camera handle
INT32 nCapa	Camera performance attribute ID, refer to TUCAM_IDCAPA
INT32 nVal	Current value needs to be set

Error Codes

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_INVALID_IDCAPA	Invalid parameter, when product information code does not exist, refer to TUCAM_IDCAPA
TUCAMRET_INVALID_VALUE	Invalid value, when 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 Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Open, TUCAM_Dev_Close
TUCAM_Capa_GetAttr, TUCAM_Capa_GetValue, TUCAM_Capa_GetValueText

TUCAM_Capa_GetValueText

Description

Obtain the text information of current attribute value of performance parameters. The specific supported performance type can be referred to TUCAM_IDCAPA.

Statement

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

Parameters

HDTUCAM hTUCam	Camera handle
PTUCAM_VALUE_TEXT pVal	Obtain the text information structure pointer of performance parameters, TUCAM_VALUE_TEXT

Error Codes

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

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit

TUCAM_Dev_Open, TUCAM_Dev_Close

TUCAM_Capa_GetAttr, TUCAM_Capa_SetValue, TUCAM_Capa_GetValue

TUCAM_Prop_GetAttr

Description

Obtain the attribute value of attribute parameters. The attributes obtained include the minimum value, maximum value, the default value and step of the parameter. The specific supported performance type can be referred to TUCAM_IDPROP.

Statement

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

Parameters

HDTUCAM hTUCam	Camera handle
PTUCAM_PROP_ATTR pAttr	Camera performance attribute structure pointer, refer to TUCAM_PROP_ATTR

Error Codes

TUCAMRET_NOT_INIT	TUCAM-API not initialized

TUCAMRET_INVALID_IDPROP	Invalid parameter, when product information code does not exist, refer to TUCAM_IDPROP
TUCAMRET_INVALID_VALUE	Invalid value, when pAttr pointer is empty
TUCAMRET_NOT_SUPPORT	When the underlying request is not supported
TUCAMRET_OUT_OF_RANGE	When the channel needs to be obtained exceeds the range
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit

TUCAM_Dev_Open, TUCAM_Dev_Close

TUCAM_Prop_GetValue, TUCAM_Prop_SetValue, TUCAM_Prop_GetValueText

TUCAM_Prop_GetValue

Description

Obtain the current attribute value of attribute parameters. The specific supported performance type can be referred to TUCAM_IDPROP.

Statement

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

Parameters

HDTUCAM hTUCam	Camera handle
INT32 nProp	Camera performance attribute ID, refer toTUCAM_IDPROP
DOUBLE *pdbVal	Return the current value
INT32 nChn	Current channel needs to be obtained (the default is 0, monochrome camera is 0)

Error Codes

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_INVALID_IDPROP	Invalid parameter, when product information code does not exist, refer to TUCAM_IDPROP
TUCAMRET_INVALID_VALUE	Invalid value, when pAttr pointer is empty
TUCAMRET_NOT_SUPPORT	When the underlying request is not supported
TUCAMRET_OUT_OF_RANGE	When the channel needs to be obtained exceeds the range
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit

TUCAM_Dev_Open, TUCAM_Dev_Close

TUCAM_Prop_GetAttr, TUCAM_Prop_SetValue, TUCAM_Prop_GetValueText

TUCAM_Prop_SetValue

Description

Set the current attribute value of attribute parameters. The specific supported performance type can be referred to TUCAM_IDPROP.

Statement

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

Parameters

HDTUCAM hTUCam	Camera handle
INT32 nProp	Camera attribute ID, refer to TUCAM_IDPROP
DOUBLE dbVal	Current value needs to be set
INT32 nChn	Current channel needs to be obtained (the default is 0, monochrome camera is 0)

Error Codes

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_INVALID_IDPROP	Invalid parameter, when product information code does not exist, refer to TUCAM_IDPROP
TUCAMRET_INVALID_VALUE	Invalid value, when pAttr pointer is empty
TUCAMRET_NOT_SUPPORT	When the underlying request is not supported
TUCAMRET_OUT_OF_RANGE	When the channel needs to be obtained exceeds the range
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit

TUCAM_Dev_Open, TUCAM_Dev_Close

TUCAM_Prop_GetAttr, TUCAM_Prop_GetValue, TUCAM_Prop_GetValueText

TUCAM_Prop_GetValueText

Description

Obtain the text information of current attribute value of attribute parameters. The specific supported performance type can be referred to TUCAM_IDPROP.

Statement

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

Parameters

HDTUCAM hTUCam	Camera handle
PTUCAM_VALUE_TEXT pVal	Obtain the text information structure pointer of performance parameters, TUCAM_VALUE_TEXT

Error Codes

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

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit

TUCAM_Dev_Open, TUCAM_Dev_Close

TUCAM_Prop_GetAttr, TUCAM_Prop_SetValue, TUCAM_Prop_GetValue

TUCAM_Buf_Alloc

Description

Allocate memory for data capture. When the application calls this interface, SDK will allocate the necessary internal buffer to buffer image acquisition. Capture does not start from this moment. When the acquisition is started, the application must call **TUCAM_Cap_Start** interface. If the buffer is no longer necessary, the application should call **TUCAM_Buf_Release** interface to release the internal buffer.

Statement

TUCAMRET TUCAM_Buf_Alloc (HDTUCAM hTUCam, PTUCAM_FRAME pFrame);

Parameters

HDTUCAM hTUCam	Camera handle
PTUCAM_FRAME pFrame	Picture time frame structure, refer to TUCAM_FRAME

Error Codes

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_INVALID_PARAM	when pFrame pointer is empty
TUCAMRET_EXCLUDED	When TUCAM_Buf_Alloc is called and not released
TUCAMRET_OUT_OF_RANGE	When the number of frame need to be obtained exceeds the range
TUCAMRET_NO_MEMORY	When memory is low
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit

TUCAM_Dev_Open, TUCAM_Dev_Close

TUCAM_Buf_Release

TUCAM_Buf_AbortWait, TUCAM_Buf_WaitForFrame, TUCAM_Buf_CopyFrame

TUCAM_Cap_Start, TUCAM_Cap_Stop

TUCAM_Buf_Release

Description

Free up memory space for data capture. If the interface is called during capture, the interface will return to the state that the camera is busy.

Statement

TUCAMRET TUCAM_Buf_Release (HDTUCAM hTUCam);

Parameters

HDTUCAM hTUCam	Camera handle

Error Codes

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

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit

TUCAM_Dev_Open, TUCAM_Dev_Close

TUCAM_Buf_Alloc

TUCAM_Buf_AbortWait, TUCAM_Buf_WaitForFrame, TUCAM_Buf_CopyFrame

TUCAM_Cap_Start, TUCAM_Cap_Stop

TUCAM_Buf_AbortWait

Description

It is used for the waiting when stopping the data capture. After calling **TUCAM_Buf_WaitForFrame** for data capture waiting, use this interface to abort waiting.

Statement

```
TUCAMRET   TUCAM_Buf_AbortWait (HDTUCAM hTUCam);
```

Parameters

HDTUCAM hTUCam	Camera handle

Error Codes

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

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit

TUCAM_Dev_Open, TUCAM_Dev_Close

TUCAM_Buf_Alloc, TUCAM_Buf_Release

TUCAM_Buf_WaitForFrame, TUCAM_Buf_CopyFrame

TUCAM_Cap_Start, TUCAM_Cap_Stop

TUCAM_Buf_WaitForFrame

Description

It is used for the completion of data capture. By calling **TUCAM_Buf_Alloc** the space allocated, the captured frame data is obtained. It must be used after calling **TUCAM_Cap_Start** to start capturing; otherwise it will return the state of not ready.

This function belongs to choke function, until the data capture is completed or call **TUCAM_Buf_AbortWait** to abort.

Set the frames need to be captured in uiRsdSize of frame structure, and it is valid for the trigger mode. For example: When 5 frames need to be returned in one trigger, this function will be returned after waiting for the end of capture of five frames of data.

Note: The data ordering of the returned frame structure pBuffer is frame head (usHeader) + image data (uiImgSize) + reserved bits (uiHstSize). If the multi-frame is returned, then it is arranged in this order.

Statement

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

Parameters

HDTUCAM hTUCam	Camera handle
PTUCAM_FRAME pFrame	Frame structure pointer

Error Codes

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_NOT_READY	When starting capture without callingTUCAM_Cap_Start
TUCAMRET_NO_MEMORY	When TUCAM_Buf_Alloc is not called to create memory space
TUCAMRET_NO_RESOURCE	When pFrame pointer is empty
TUCAMRET_OUT_OF_RANGE	When the number of frames need to be obtained is greater than 1 and the format obtained is different from TUCAM_Buf_Alloc
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit

TUCAM_Dev_Open, TUCAM_Dev_Close

TUCAM_Buf_Alloc, TUCAM_Buf_Release

TUCAM_Buf_AbortWait, TUCAM_Buf_CopyFrame

TUCAM_Cap_Start, TUCAM_Cap_Stop

TUCAM_Buf_CopyFrame

Description

It is used for the image format data copied after the completion of waiting data capture different from **TUCAM_Buf_Alloc**. It must be called after **TUCAM_Buf_WaitForFrame** is returned; otherwise the correct image data cannot be obtained.

For example: when the allocated image format is TUFRM_FMT_RGB888, data of other formats can be copied by using this function (i.e., TUFRM_FMT_RAW), and this interface cannot copy the data larger than 1 frame, that is, uiRsdSize in the frame structure cannot be larger than 1.

Note: The data ordering of the returned frame structure pBuffer is frame head (usHeader) + image data (uiImgSize) + reserved bits (uiHstSize). Return of the multi-frame data is not supported.

Statement

Statement

TUCAMRET TUCAM_Buf_CopyFrame (HDTUCAM hTUCam, PTUCAM_FRAME pFrame);

Parameters

HDTUCAM hTUCam	Camera handle
PTUCAM_FRAME pFrame	Frame structure pointer

Error Codes

TUCAMRET_NOT_INIT TUCAM-API not initialized

TUCAMRET_NOT_READY when starting capture without callingTUCAM_Cap_Start

TUCAMRET_NO_MEMORY When TUCAM_Buf_Alloc is not called to create memory space

TUCAMRET_NO_RESOURCE when pFrame pointer is empty

TUCAMRET_OUT_OF_RANGE When the number of frames need to be obtained is greater than 1 and the format obtained is different from TUCAM_Buf_Alloc

TUCAMRET_INVALID_CAMERA Invalid camera, when the camera handle does not exist

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Open, TUCAM_Dev_Close
TUCAM_Buf_Alloc, TUCAM_Buf_Release
TUCAM_Buf_AbortWait, TUCAM_Buf_WaitForFrame
TUCAM_Cap_Start, TUCAM_Cap_Stop

TUCAM_Cap_SetROI

Description

It is used to set the interested areas of image, with the upper left corner as the origin of coordinates. The set horizontal offset, vertical offset, width and height must be in multiples of 4.

Statement

```
TUCAMRET   TUCAM_Cap_SetROI (HDTUCAM hTUCam, TUCAM_ROI_ATTR roiAttr);
```

Parameters

HDTUCAM hTUCam	Camera handle
TUCAM_ROI_ATTR roiAttr	Object of ROI attribute structure

Error Codes

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

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Open, TUCAM_Dev_Close
TUCAM_Buf_Alloc, TUCAM_Buf_Release
TUCAM_Buf_AbortWait, TUCAM_Buf_WaitForFrame
TUCAM_Cap_Start, TUCAM_Cap_Stop
TUCAM_Cap_GetROI

TUCAM_Cap_GetROI

Description

It is used to set the interested areas of image, with the upper left corner as the origin of coordinates. The set horizontal offset, vertical offset, width and height must be in multiples of 4.

Capture does not start from this moment. Acquisition starts after **TUCAM_Cap_Start** interface is called.

Statement

```
TUCAMRET    TUCAM_Cap_GetROI (HDTUCAM hTUCam, PTUCAM_ROI_ATTR pRoiAttr);
```

Parameters

HDTUCAM hTUCam	Camera handle
PTUCAM_ROI_ATTR pRoiAttr	Pointer of ROI attribute structure

Error Codes

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

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Open, TUCAM_Dev_Close
TUCAM_Buf_Alloc, TUCAM_Buf_Release
TUCAM_Buf_AbortWait, TUCAM_Buf_WaitForFrame
TUCAM_Cap_Start, TUCAM_Cap_Stop
TUCAM_Cap_SetROI

TUCAM_Cap_SetTrigger

Description

It is used for setting the trigger attribute. Capture does not start from this moment.

Acquisition starts after **TUCAM_Cap_Start** interface is called.

Exposure Mode:

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

Stimulation edge mode:

TUCTD_RISING	means the trigger signal is rising edge valid
TUCTD_FAILING	means the trigger signal is falling edge valid

Number of frames: it means after receiving a trigger signal, the exposure time of each image is the same no matter how many images are shot, which depends on the software settings. (When choosing level width, the function is invalid.)

Delay: it means after receiving a trigger signal, the delay time of the desired delay time can be set to trigger the camera exposure.

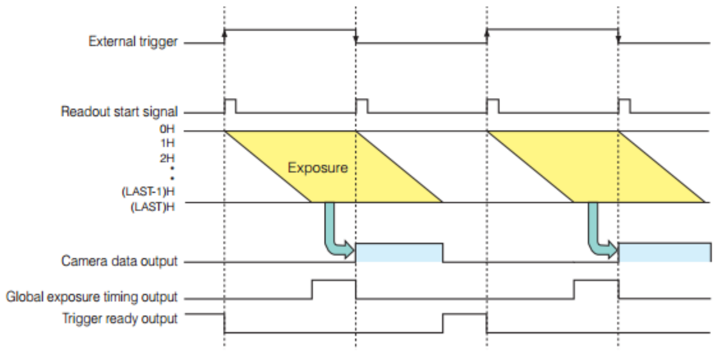


Fig. 19: Normal level trigger mode (rising edge)

Parameters supported by trigger mode:

Mode	TUCAM_TRIGGER_EXP	TUCAM_TRIGGER_EDGE	Delay	Number of frames
Standard trigger	Yes	Yes	Yes	Yes
Synchronization trigger	Yes	Yes	No	No
Global trigger	No	Yes	No	No
Software trigger	No	No	No	No

Synchronization trigger: namely the synchronous picture grabbing, first trigger to start, and the second trigger to output synchronization image.

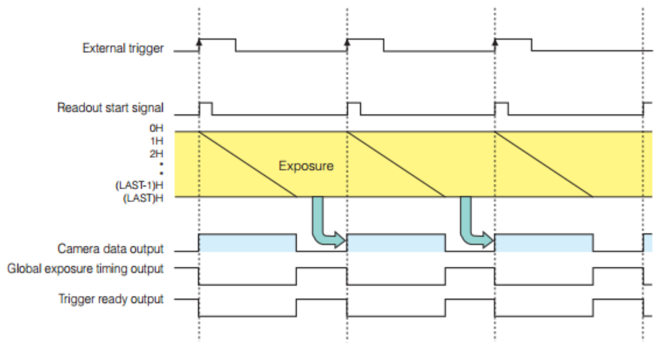


Fig. 21: Normal synchronous readout trigger mode (rising edge)

Global trigger: generally used for the scenes of controllable light source.

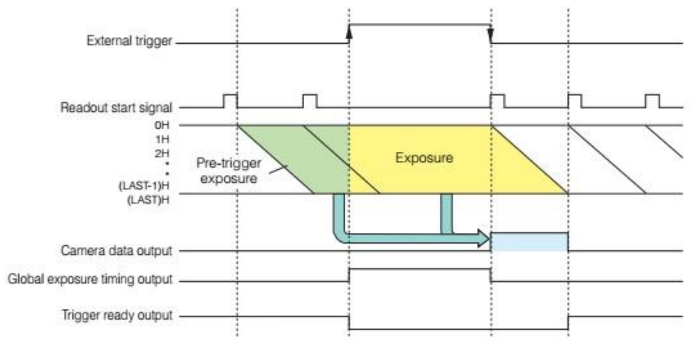


Fig. 20: Global exposure level trigger mode

Software trigger: Simulate trigger signal via the software command.

Statement

TUCAMRET TUCAM_Cap_SetTrigger (HDTUCAM hTUCam, TUCAM_TRIGGER_ATTR tgrAttr);

Parameters

HDTUCAM hTUCam	Camera handle
TUCAM_TRIGGER_ATTR tgrAttr	Object of trigger attribute structure

Error Codes

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

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit

TUCAM_Dev_Open, TUCAM_Dev_Close

TUCAM_Buf_Alloc, TUCAM_Buf_Release
TUCAM_Buf_AbortWait, TUCAM_Buf_WaitForFrame
TUCAM_Cap_Start, TUCAM_Cap_Stop
TUCAM_Cap_GetTrigger, TUCAM_Cap_DoSoftwareTrigger

TUCAM_Cap_GetTrigger

Description

It is used to obtain the trigger attributes.

Statement

```
TUCAMRET   TUCAM_Cap_GetTrigger (HDTUCAM hTUCam, PTUCAM_TRIGGER_ATTR pTgrAttr);
```

Parameters

HDTUCAM hTUCam	Camera handle
PTUCAM_TRIGGER_ATTR pTgrAttr	Pointer of trigger attribute structure

Error Codes

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

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Open, TUCAM_Dev_Close
TUCAM_Buf_Alloc, TUCAM_Buf_Release
TUCAM_Buf_AbortWait, TUCAM_Buf_WaitForFrame
TUCAM_Cap_Start, TUCAM_Cap_Stop
TUCAM_Cap_SetTrigger

TUCAM_Cap_DoSoftwareTrigger

Description

Execute software trigger commands.

Statement

```
TUCAMRET   TUCAM_Cap_DoSoftwareTrigger(HDTUCAM hTUCam);
```

Parameters

HDTUCAM hTUCam	Camera handle

Error Codes

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_FAILURE	Failed to execute trigger commands
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Open, TUCAM_Dev_Close
TUCAM_Buf_Alloc, TUCAM_Buf_Release
TUCAM_Buf_AbortWait, TUCAM_Buf_WaitForFrame
TUCAM_Cap_Start, TUCAM_Cap_Stop
TUCAM_Cap_SetTrigger

TUCAM_Cap_Start

Description

Start data capture. Prior to capture, the interested areas and trigger mode should be configured.

Statement

```
TUCAMRET   TUCAM_Cap_Start(HDTUCAM hTUCam, UINT32 uiMode);
```

Parameters

HDTUCAM hTUCam	Camera handle
UINT32 uiMode	Camera capture mode

Error Codes

TUCAMRET_NOT_INIT	TUCAM-API not initialized

TUCAMRET_FAILOPEN_BULKIN	Failed to open camera capture
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Open, TUCAM_Dev_Close
TUCAM_Buf_Alloc, TUCAM_Buf_Release
TUCAM_Buf_AbortWait, TUCAM_Buf_WaitForFrame
TUCAM_Cap_Start
TUCAM_Cap_SetTrigger, TUCAM_SetROI

TUCAM_Cap_Stop

Description

Stop data capture.

Statement

```
TUCAMRET   TUCAM_Cap_Stop (HDTUCAM hTUCam);
```

Parameters

HDTUCAM hTUCam	Camera handle

Error Codes

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

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Open, TUCAM_Dev_Close
TUCAM_Buf_Alloc, TUCAM_Buf_Release
TUCAM_Buf_AbortWait, TUCAM_Buf_WaitForFrame
TUCAM_Cap_Stop

TUCAM_File_SaveImage

Description

Save frame data.

Statement

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

Parameters

HDTUCAM hTUCam	Camera handle
TUCAM_FILE_SAVE fileSave	File save structure

Error Codes

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_INVALID_PARAM	Parameters entered invalid
TUCAMRET_INVALID_PATH	Path entered does not exist
TUCAMRET_FAILURE	Failed to save file
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit

TUCAM_Dev_Open, TUCAM_Dev_Close

TUCAM_Buf_Alloc, TUCAM_Buf_Release

TUCAM_Buf_WaitForFrame, TUCAM_Buf_CopyFrame

TUCAM_Rec_Start

Description

Open the video file, and save the video frame data, the data are not written at this time. The frame rate set should be greater than 1fps, and those less than 1fps will be regarded as 1fps to create video files.

Statement

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

Parameters

HDTUCAM hTUCam	Camera handle
TUCAM_REC_SAVE recSave	Video file save structure

Error Codes

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_INVALID_PARAM	Parameters entered invalid
TUCAMRET_INVALID_PATH	Path entered does not exist
TUCAMRET_FAILOPEN_FILE	Failed to open files
TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Open, TUCAM_Dev_Close
TUCAM_Buf_Alloc, TUCAM_Buf_Release
TUCAM_Buf_WaitForFrame
TUCAM_Cap_Start, TUCAM_Cap_Stop
TUCAM_Rec_Stop, TUCAM_Rec_AppendFrame

TUCAM_Rec_AppendFrame

Description

Write the image data into the file, and call the interface at **TUCAM_Buf_WaitForFrame**.

Statement

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

Parameters

HDTUCAM hTUCam	Camera handle
PTUCAM_FRAME pFrame	Frame structure pointer

Error Codes

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_NOT_READY	TUCAM_Rec_Start interface not called
TUCAMRET_OUT_OF_RANGE	The image width and height are inconsistent with the

ones when created

TUCAMRET_INVALID_CAMERA	Invalid camera, when the camera handle does not exist

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Open, TUCAM_Dev_Close
TUCAM_Buf_Alloc, TUCAM_Buf_Release
TUCAM_Buf_WaitForFrame
TUCAM_Cap_Start, TUCAM_Cap_Stop
TUCAM_Rec_Start, TUCAM_Rec_Stop

TUCAM_Rec_Stop

Description

Close video file, and calling of **TUCAM_Rec_AppendFrame** at this time will not be able to write data.

Statement

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

Parameters

HDTUCAM hTUCam	Camera handle

Error Codes

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

Related Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit
TUCAM_Dev_Open, TUCAM_Dev_Close
TUCAM_Buf_Alloc, TUCAM_Buf_Release
TUCAM_Buf_WaitForFrame
TUCAM_Cap_Start, TUCAM_Cap_Stop
TUCAM_Rec_Start, TUCAM_Rec_AppendFrame

TUCAM_Reg_Read

Description

Contents read from the register. Refer to TUREG_TYPE for types of reading.

Statement

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

Parameters

HDTUCAM hTUCam	Camera handle
TUCAM_REG_RW regRW	Register read/write structure

Error Codes

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_NO_MEMORY	The incoming buffer did not allocate memory space
TUCAMRET_NOT_SUPPORT	Reading of this type 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 Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit

TUCAM_Dev_Open, TUCAM_Dev_Close

TUCAM_Reg_Write

TUCAM_Reg_Write

Description

Contents written into the register. Refer to TUREG_TYPE for types of writing.

Statement

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

Parameters

HDTUCAM hTUCam	Camera handle
TUCAM_REG_RW regRW	Register read/write structure

Error Codes

TUCAMRET_NOT_INIT	TUCAM-API not initialized
TUCAMRET_NO_MEMORY	The incoming buffer did not allocate memory space
TUCAMRET_NOT_SUPPORT	Reading of this type 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 Interfaces

TUCAM_Api_Init, TUCAM_Api_Uninit

TUCAM_Dev_Open, TUCAM_Dev_Close

TUCAM_Reg_Read