VT CMOS Series Products

SDK Instructions

Contents

Functions Overview 1
The Return Value of The Function 4
Use Function 5
CapInfoStruct Structure 5
VT_Initialize 6
VT_Uninitialize 7
VT_StartView 8
VT_StopView 9
VT_SetCapInfo 10
VT_GetCapInfo 10
VT_SetGainR 11
VT_GetGainR 11
VT_SetGainG 12
VT_GetGainG 12
VT_SetGainB 13
VT_GetGainB 13
VT_SetParamColor 14
VT_SetParamFlip 14
VT_SetDoAWB 15
VT_SetDoAE 16
VT_GetFrameRate 17
VT_GetRawFrame 18
VT_GetRawFrameToBmp 19
VT_GetRgbFrame 20
VT_GetRgbFrameToBmp 21
VT_GetRgbFrameToJpeg 22
VT_SetContrastValue 23
VT_GetContrastValue 23

VT_CaptureToAvi 24
VT _GetTrigger 25
VT _TriggerDoing 26
VT _SetScrollOffset 27
VT _SetVideoSize 28
VT _GetTotalDeviceNum 29
VT _GetDeviceID 30
VT _GetFriendlyName 30
VT _SetFrameCallBack 31
VT _SetClearView 32
VT _GetClearViewState 32
VT _SetIO 33
VT _GetIO 34
VT _PixelCalibration 35
VT _SetShowAutoMark 36
VT _SetShowVideo 37
VT _SelectAviCompressor 38
VT _SetAviCompressor 39
VT _SetAELevel 39
VT _SetGamma 40
VT _GetGamma 40
VT _SetTriggerMode 41
VT _SoftTrigger 41
VT _SetDeviceEven 42
VT _VideoPause 43
VT _GetRgbFrameToBmpZoom 44
VT _GetRgbFrameToJpegZoom 45
VT _SetHDR 46
VT _ResetLightAvgTable 47
VT _EnableLightAvg 47
VT _EEProm_Set 48

VT _EEProm_Get 49
VT _EEProm_SetByte 50
VT _EEProm_GetByte 51
VT _SetTriggerDelayTime 52
VT _SetCustomIO 53
VT _SetTriggerSource 54
VT _SetTriggerSourceTime 55
VT _SetHBlank 56
VT _SetVBlank 57
VT _AEParam 58
VT _GetNewCalibrationRawFrame 59
VT _RawFrameToRgb 61
VT _SetLight 62
VT _GetLight 62
VT _MCResolutionShow 65
VT _IsMCResolutionShow 66
VT _SetMCResolution 67
VT _GetMCResolution 68
VT _SetExpouseDelayTime 69
VT _SetExpouseUnit 70
VT _FrameToImage 71
VT _GetSerialNumber 73
VT _SetTriggerCheckSignalTime 75
VT _SetCameraLED 76
AWB/AE Callback Function 77
Frame Callback Function 78
SDK Use Examples 79
Initial CapInfoStruct structures 79
Initialization Camera 80
Set Camera Parameters 81
Video Preview 82

Acquisition of A Frame Image 83
Video and the Compression 84
Frame callback function 86
Automatic Exposure (AE) 88
Automatic White Balance (AWB) 89
Trigger 90
Appendix 91
Pins Definition and Use 91

Functions Overview

VT _Initialize	Initialization camera
VT _StartView	Open video preview
VT _StopView	Stop video preview
VT _SetCapInfo	Real-time video parameters set
VT _SetGainR	Set red gain
VT _GetGainR	Get red gain
VT _SetGainG	Set green gain
VT _GetGainG	Get green gain
VT _SetGainB	Set blue gain
VT _GetGainB	Get blue gain
VT _Uninitialize	Uninstall camera
VT _SetParamColor	Set the color of the image mode
VT _SetParamFlip	Set the flip mode image
VT _SetDoAWB	Open to be automatic white balance
VT _SetDoAE	Open automatic exposure
VT _GetFrameRate	Get frame rate
VT _GetRawFrame	To obtain a frame of the original image
VT _GetRawFrameToBmp	Will a frame the original image save into 8 (bit) BMP file
VT _GetRgbFrame	To obtain a frame 24 bits RGB image
VT _GetRgbFrameToBmp	Will a frame image save for 24 (bit) BMP file
VT _GetRgbFrameToJpeg	Will a frame image save for 24 (bit) JPEG files
VT _SetContrastValue	Set the contrast of images
VT _CaptureToAvi	Start/stop video
VT _GetTrigger	Open the trigger, automatic save images
VT _TriggerDoing	Open the trigger, functions by user defined callback function decision
VT _SetScrollOffset	Video images of window set offset
VT _SetVideoSize	Set the size of the window video images
VT _GetTotalDeviceNum	Get installed cameras number

VT _GetDeviceID To get the camera serial number

VT _GetFriendlyName To get the equipment device name

VT _SetFrameCallBack Set frame callback function

VT _SetClearView Open/close set clear video mode, in 24 bits RGB image effective

VT _GetClearViewState Inquires the clear video mode whether are open

VT _SetIO Set IO state

VT _GetIO Get IO state

VT _PixelCalibration Open bad point detection function

VT _SetShowAutoMark Set up AE/AWB mark displays a status

VT _SetShowVideo Setting is in the window to draw on video screen

VT _SelectAviCompressor Choose video compressor

VT _SetAviCompressor Set video compressor

VT _SetDeviceEvent Setting equipment news callback function

VT _VideoPause Suspended video

VT _GetRgbFrameToBmpZoom Will a frame image save for the wide high half the size of 24 (bit) BMP file

VT _GetRgbFrameToJpegZoom Will a frame image save for the wide high half the size of 24 (bit) JPEG file

VT _SetHDR Open camera high dynamic range effect (HDR)

VT _ResetLightAvgTable Reset brightness balanced table

VT _EnableLightAvg Open brightness balanced table

VT _EEProm_Set Set store the data

VT _EEProm_Get Get store the data

VT _EEProm_SetByte Set a fixed position of fixed size data

VT _EEProm_GetByte Get a fixed position of fixed size data

VT _SetTriggerDelayTime Set the camera trigger delay acquisition time

VT _SetCustomIO Set the user defined IO

VT _SetTriggerSource Set the camera that can trigger source

VT _SetTriggerSourceTime Set the camera trigger source time interval

VT _SetHBlank Set the camera line blank

VT _SetVBlank Set the camera frame blank

VT _AEParam	Set up automatic exposure parameters, real-time automatic exposure
VT _GetNewCalibrationRawFrame	According to original calibration image generation new calibration image
VT _RawFrameToRgb	The original data to create BMP image data
VT _SetLight	Set the brightness values of output image
VT _GetLight	Get the brightness values of output image
VT _MCResolutionShow	Set up if application custom cut resolution show preview
VT _IsMCResolutionShow	Get whether open cuting resolution display
VT _SetMCResolution	Setting cutting resolution display related parameters
VT _GetMCResolution	Getting cutting resolution display related parameters
VT _SetExpouseDelayTime	Set the exposure time delay
VT _SetExpouseUnit	Exposure unit set
VT _GetSerialNumber	Get camera serial number
VT _FrameToImage	Incoming data in memory is saved as an image
VT _SetTriggerCheckSignalTime	Set trigger testing time
VT _SetCameraLED	Set LED lights

The Return Value of The Function

ResSuccess	0x0000	Return to success
ResNullHandleErr	0x0001	Invalid handle
ResNullPointerErr	0x0002	Pointer is empty
ResFileOpenErr	0x0003	File open failure
ResNoDeviceErr	0x0004	No device to connect
ResInvalidParameterErr	0x0005	Invalid parameters
ResOutOfMemoryErr	0x0006	Memory leak
ResNoPreviewRunningErr	0x0007	Preview doesn't open
ResOSVersionErr	0x0008	Operating system version mistakes
ResUsbNotAvailableErr	0x0009	Equipment invalid
ResNotSupportedErr	0x000a	Device doesn't support this function
ResNoSerialString	0x000b	Read less than serial number
ResVerificationErr	0x000c	Failed to pass
ResTimeoutErr	0x000d	Time out
ResScaleModeErr	0x000f	Zoom error
ResUnknownErr	0x00ff	Unknown error
ResDisplayWndExist	0x0011	Should be closed preview window
ResAllocated	0x0012	Memory has distribution
ResAllocateFail	0x0013	Memory allocation failure
ResReadError	0x0014	USB read failure
ResWriteError	0x0015	USB command issued a failure
ResUsbOpen	0x0016	USB port is already open
ResCreateStreamErr	0x0017	Create AVI stream fail
ResSetStreamFormatErr	0x0018	Set AVI stream format fail

Use Function

CapInfoStruct Structure

CapInfoStruct , including the camera's basic control parameters. When initialization and to set the camera parameters, CapInfoStruct is to describe the current camera as parameters of the state.

Syntax

```
struct CapInfoStruct
{
    UCHAR    *Buffer;        // The user distribution, used for return to 8 raw data
    ULONG    Height;         // Acquisition height
    ULONG    Width;          // Acquisition width
    ULONG    OffsetX;        // horizontal offset
    ULONG    OffsetY;        // vertical offset
    ULONG    Exposure;       // Exposure
    UCHAR    Gain[3];    //Gain[0] Overall gain 1-63. Gain[1],Gain[2] Reserves.
    UCHAR    Control;
    UCHAR    InternalUse;    // Users don't bytes to operate
    UCHAR    ColorOff[3];    // Users from outside the array values don't change
    UCHAR    Reserved[4];    // Reserved Bit
};
```

In may, after the initial camera through a function " VT_SetCapInfo" to set the "CapInfoStruct", so as to realize the exposure setting, red, green, blue gain setting etc. function. The user can also through a function call " VT_GetCapInfo" to get the current video parameters.

VT_Initialize

Initialize equipment. Return to the camera's handle is used to other function calls

Syntax

```
VT_API    VT_Initialize(
        IN   LPCTSTR          pFilterName,
        IN   int              *pIndex,
        IN   CapInfoStruct    *pCapInfo,
        OUT HANDLE            *hCamera
);
```

Parameters

pFilterName

[in] Reserves

pIndex

[in] Camera serial number; The serial number of the decision by the enumeration equipment order. The first equipment serial number is 1.

pCapInfo

[in] Camera parameters structures pointer, Must give pCapInfo-> Buffer distribution enough memory space, in general to than the largest collection of space distribution are recommended for use, MAXWIDTH * MAXHEIGHT + 512, used to convey data collection.

hCamera

[out] If the function succeeds, the return calls camera handle, otherwise returns NULL.

VT_Uninitialize

Uninstall camera, in the program before exit or failed to initiate the call this function when used to release has allocated memory.

Syntax

```
VT_API    VT_Uninitialize(
            IN HANDLE    hCamera
);
```

Parameters

hCamera

[in] The camera will be release handle.

VT _StartView

Open the preview window, and start streaming video

Syntax

```
VT _API    VT_StartView(
            IN HANDLE     hCamera,
            IN LPCTSTR    lpszWindowName,
            IN DWORD      dwStyle,
            IN int        x,          IN int        y,
            IN int        nWidth,     IN int        nHeight,
            IN HWND       hwndParent,
            IN HMENU      nIDorHMenu
);
```

Parameters

hCamera

[in] Camera handle

lpszWindowName

[in] Video window name

dwStyle

[in] Video window style

x

[in] Video window x the starting position

y

[in] Video window y the starting position

nWidth

[in] Video window width

nHeight

[in] Video window height

hwndParent

[in] Parent window handle

nIDorHMenu

[in] Reserves

VT _StopView

Stop streaming video

Syntax

```
VT_API    VT_StopView(
            IN HANDLE            hCamera
);
```

Parameters

hCamera

[in] Camera handle

VT _SetCapInfo

Real-time video parameters set

Syntax

```
VT _API    VT_SetCapInfo(
            IN HANDLE              hCamera ,
            IN CapInfoStruct       *pCapInfo
);
```

Parameters

hCamera

[in] Camera handle

pCapInfo

[in] Camera parameters structures pointer

VT _GetCapInfo

Obtain image Parameters

Syntax

```
VT _API    VT_GetCapInfo(
            IN HANDLE              hCamera ,
            OUT CapInfoStruct      *pCapInfo
);
```

Parameters:

hCamera

[in] Camera handle

pCapInfo

[out] Return camera parameters structures pointer

VT _SetGainR

Set red gain

Syntax

```
VT _API       VT_SetGainR(
        IN HANDLE     hCamera,
        IN int        nIdxR
);
```

Parameters

hCamera

[in] Camera handle

nIdxR

[in] To set up the red gain value, range 1 ~ 6000.

VT _GetGainR

Get red gain

Syntax

```
VT _API       VT_GetGainR(
        IN HANDLE     hCamera,
        OUT int       *pIndexR
);
```

Parameters

hCamera

[in] Camera handle

pIndexR

[out] Returns the current red gain value

VT _SetGainG

Set green gain

Syntax

```
VT _API       VT_SetGainG(
        IN HANDLE     hCamera,
        IN int        nIdxR
);
```

Parameters

hCamera

[in] Camera handle

nIdxG

[in] To set up the green gain value, range 1 ~ 6000.

VT _GetGainG

Get green gain

Syntax

```
VT _API       VT_GetGainG(
        IN HANDLE     hCamera,
        OUT int       *pIndexG
);
```

Parameters

hCamera

[in] Camera handle

pIndexG

[out] Returns the current green gain value

VT _SetGainB

Set blue gain

Syntax

```
VT _API       VT_SetGainB(
        IN HANDLE     hCamera,
        IN int        nIdxB
);
```

Parameters

hCamera

[in] Camera handle

nIdxB

[in] To set up the blue gain value, range 1 ~ 6000.

VT _GetGainB

Get blue gain

Syntax

```
VT _API       VT_GetGainB(
        IN HANDLE     hCamera,
        OUT int       *pIndexB
);
```

Parameters

hCamera

[in] Camera handle

pIndexB

[out] Returns the current blue gain value

VT _SetParamColor

Set the color of the image mode

Syntax

```
VT _API    VT_SetParamColor (
        IN HANDLE          hCamera,
        COLOR_MODE         DLParam
);
```

Parameters

hCamera

[in] Camera handle

DLParam

[in] Color Mode, Enum COLOR_MODE

VT _SetParamFlip

Set the flip mode image

Syntax

```
VT _API    VT_SetParamFlip (
        IN HANDLE          hCamera,
        FLIP_MODE          DLParam
);
```

Parameters

hCamera

[in] Camera handle

DLParam

[in] Flip mode, Enum FLIP_MODE

VT _SetDoAWB

Open to be automatic white balance

Syntax

```
VT _API    VT_SetDoAWB(
        IN HANDLE                hCamera,
        IN BOOL                  bAWB,
        IN BYTE                  btTarget,
        IN DL_AUTOCALLBACK       pfAWBCallback,
        IN LPVOID                lpContext
);
```

Parameters

hCamera

[in] Camera handle

bAWB

[in] Whether open auto white balance, TRUE for open, FALSE to shut down

btTarget

[in] Reserves

pfAWBCallback

[in] Designated to be automatic white balance after the end of the callback function.

lpContext

[in] The user context.

VT _SetDoAE

Open automatic exposure

Syntax

```
VT _API    VT_SetDoAE(
        IN HANDLE                hCamera,
        IN BOOL                  bAE,
        IN BYTE                  btTarget,
        IN DL_AUTOCALLBACK       pfAECallback,
        IN LPVOID                lpContext
);
```

Parameters

hCamera

[in] Camera handle

bAE

[in] Whether open automatic exposure, TRUE for Open, FALSE to shut down.

btTarget

[in] Reserves

pfAECallback

[in] Designated to be automatic exposure after the end of the callback function, Function type DL_AUTOCALLBACK.

```
typedef VOID (WINAPI *DL_AUTOCALLBACK )(
    DWORD dw1,
    DWORD dw2,
    LPVOID lpContext );
```

dw1 is automatic exposure after the end of the exposure value.

lpContext

[in] The user context

VT _GetFrameRate

Get frame rate

Syntax

```
VT _API    VT_GetFrameRate(
            IN HANDLE          hCamera,
            OUT float          *pfFrameRate
);
```

Parameters

hCamera

[in] Camera handle

pfFrameRate

[out] Return to the camera frame rate

VT _GetRawFrame

To obtain a frame of the original image, and will this frame data stored in pCapInfo-> Buffer parameters.

Syntax

```
VT _API    VT_GetRawFrame(
        IN HANDLE          hCamera,
        IN CapInfoStruct   *pCapInfo
);
```

Parameters

hCamera

[in] Camera handle

pCapInfo

[in][out] If have open video preview, then the current video parameter acquisition of a frame original data to the pCapInfo-> Buffer; If no open video preview, then the pCapInfo description of the image size acquisition of a frame raw data to pCapInfo-> Buffer.

VT _GetRawFrameToBmp

Will a frame the original image save into 8 (bit) BMP file

Syntax

```
VT _API    VT_GetRawFrameToBmp(
        IN HANDLE          hCamera,
        IN CapInfoStruct   *pCapInfo,
        LPCTSTR            strFileName
);
```

Parameters

hCamera

[in] Camera handle

pCapInfo

[in][out] If have open video preview, then the current video parameter acquisition of a frame original data to the pCapInfo-> Buffer; If no open video preview, then the pCapInfo description of the image size acquisition of a frame raw data to pCapInfo-> Buffer.

[in] BMP file name

VT _GetRgbFrame

To obtain a frame 24 bits RGB image

Syntax

```
VT _API    VT_GetRgbFrame(
        IN HANDLE          hCamera,
        IN CapInfoStruct   *pCapInfo,
        OUT BYTE           *pDest,
        IN BOOL            bFlip
);
```

Parameters

hCamera

[in] Camera handle

pCapInfo

[in][out] If have open video preview, then the current video parameter acquisition of a frame original data to the pCapInfo-> Buffer; If no open video preview, then the pCapInfo description of the image size acquisition of a frame raw data to pCapInfo-> Buffer.

pDest

[out] Used to store the data of the memory address RGB

bFlip

[in] If TRUE, the output of the image data will be turned up and down.

VT _GetRgbFrameToBmp

Will a frame image save for 24 (bit) BMP file

Syntax

```
VT _API    VT_GetRgbFrameToBmp(
        IN HANDLE           hCamera,
        IN CapInfoStruct    *pCapInfo,
        OUT BYTE            *pDest,
        LPCTSTR             strFileName,
        IN BOOL             SaveData
);
```

Parameters

hCamera

[in] Camera handle

pCapInfo

[in][out] If have open video preview, then the current video parameter acquisition of a frame original data to the pCapInfo-> Buffer; If no open video preview, then the pCapInfo description of the image size acquisition of a frame raw data to pCapInfo-> Buffer.

pDest

[out] Used to store data memory address, RGB if SaveData set to FALSE, the parameters can be NULL

strFileName

[in] BMP file name

SaveData

[in] Whether it will be saved to the selected memory RGB data (pDest).

.

VT _GetRgbFrameToJpeg

Will a frame image save for 24 (bit) JPEG files

Syntax

```
VT _API    VT_GetRgbFrameToJpeg(
        IN HANDLE          hCamera,
        IN CapInfoStruct   *pCapInfo,
        OUT BYTE           *pDest,
        LPCTSTR            strFileName,
        int                nQuality
);
```

Parameters

hCamera

[in] Camera handle

pCapInfo

[in][out] If have open video preview, then the current video parameter acquisition of a frame original data to the pCapInfo-> Buffer; If no open video preview, then the pCapInfo description of the image size acquisition of a frame raw data to pCapInfo-> Buffer.

pDest

[in] Reserves

strFileName

[in] Jpeg file name

nQuality

[in] Jpeg files file quality, value (1-100)

VT_SetContrastValue

Set the contrast of images

Syntax

```
VT _API VT_SetContrastValue(
        IN HANDLE     hCamera,
        IN int        nValue
);
```

Parameters

hCamera

[in] Camera handle

nValue

[in] Contrast,Value(1-32)

VT_GetContrastValue

Get the contrast of images

Syntax:

```
VT _API    VT_GetContrastValue(
        IN HANDLE hCamera,
        OUT int *pValue
);
```

Parameters:

hCamera

[in] Camera handle

pValue

[out] Return to the contrast

VT _CaptureToAvi

Start/stop AVI video, video file is no compression AVI files.

Syntax

```
VT _API VT _CaptureToAvi(
        IN HANDLE     hCamera,
        BOOL          bStart,
        LPCTSTR       strFileName
);
```

Parameters

hCamera

[in] Camera handle

bStart

[in] Whether to begin an AVI video, TRUE or FALSE on video, to start to stop the video.

strFileName

[in] Video file name

Note

Function does not support eight gray figure video, if the current color mode using is not 24 bits data, then function returns to the failure, the return value is ResNotSupportedErr.

In the video, the SDK to open after some of the function calls, will stop the current video operation, and close and save has been incorporated into avi files. Call the following function will stop ongoing video operation: VT_Uninitialize, VT_StopView, VT_SetParamColor, VT_SetCapInfo. What is worth mentioning, VT_SetCapInfo function call, only as change resolution will stop ongoing video operation.

VT _GetTrigger

Open trigger mode , to receive a trigger signal, a frame image collection.

For VT - FF series do not provide this function.

Syntax

```
VT _API       VT _GetTrigger(
        IN HANDLE           hCamera,
        BOOL                bStart,
        IN   VTIMAGETYPR    ImageType,
        IN   LPCTSTR        pFileNameStart,
        IN   int            JpegCompressibility
);
```

Parameters

hCamera

[in] Camera handle

bStart

[in] Whether the trigger mode, TRUE for open trigger mode, FALSE for close trigger mode.

ImageType

[in] Save image of type, support BMP, JPEG two image format

pFileNameStart

[in] The file name set, to the input file named after the beginning of the file name according to the order and digital label arrangement. If input is empty, then the system time naming.

JpegCompressibility

[in] If the selected image save way as JPEG, the parameters given the compression ratio value (1 ~ 100).

Note

Call this function, the signal effectively triggered more than a time frame to collect the data of time, otherwise may not be able to detect the trigger signal.

VT _TriggerDoing

Open the trigger, functions by user defined.

Syntax

```
VT _API    VT _TriggerDoing(
        IN HANDLE                   hCamera,
        IN DL_FRAMECALLBACK         pCallBack,
        IN LPVOID                   lpContext ,
        IN BOOL                     bOn
);
```

Parameters

hCamera

[in] Camera handle

pCallBack

[in] The trigger callback function

lpContext

[in] Context

bOn

[in] Whether the trigger mode, TRUE for open trigger mode, FALSE for close trigger mode.

VT _SetScrollOffset

By changing the video window, the starting point of the video window position, set offset

Syntax

```
VT _API    VT_SetScrollOffset(
        IN HANDLE      hCamera,
        int            Xoffset,
        int            Yoffset
);
```

Parameters

hCamera

[in] Camera handle

Xoffset

[in] Video window x direction offset

Yoffset

[in] Video window y direction offset

VT _SetVideoSize

Set the size of the window video images

Syntax

```
VT _API    VT _SetVideoSize(
            IN HANDLE hCamera,
            int nWidth,
            int nHeight
);
```

Parameters

hCamera

[in] Camera handle

nWidth

[in] Video window width

nHeight

[in] Video window height

VT _GetTotalDeviceNum

Get installed cameras number

Syntax

```
VT _API    VT _GetTotalDeviceNum(
            IN HANDLE hCamera,
            int * pNum
);
```

Parameters

hCamera

[in] Camera handle

pNum

[out] Return to cameras number

VT _GetDeviceID

To get the camera serial number

Syntax

```
VT _API    VT _GetDeviceID(
            IN HANDLE              hCamera,
            IN USHORT              *pDeviceID
);
```

Parameters

hCamera

[in] Camera handle

pDeviceID

[out] Back to hCamera designated by the camera's serial number

VT _GetFriendlyName

To get the equipment device name

Syntax

```
VT _API    VT _GetFriendlyName(
            IN HANDLE          hCamera,
            OUT TCHAR          *pName
);
```

Parameters

hCamera

[in] Camera handle

pName

[out] Back to hCamera designated by the camera's friendly name

VT _SetFrameCallBack

Set frame callback function, in open video preview, each frame image in drawn call before frame callback function, through the frame callback function change video image, directly will be reaction comes out. In addition, frame callback function of the video collection efficiency influence speed, so please as soon as possible from the frame callback function returns.

Syntax

```
VT _API    VT _SetFrameCallBack(
        IN HANDLE                hCamera,
        IN DL_FRAMECALLBACK      FrameCB,
        LPVOID                   lpContext
);
```

Parameters

hCamera

[in] Camera handle

FrameCB

[in] Frame callback function

lpContext

[in] The user context

VT _SetClearView

Open/close set clear video mode, in 24 bits RGB image effective.

Syntax

```
VT _API    VT _SetClearView(
        IN HANDLE           hCamera,
        IN BOOL             bOn
);
```

Parameters

hCamera

[in] Camera handle

bOn

[in] Clear video mode of open mark, TRUE for open, FALSE for closed.

VT _GetClearViewState

Inquires the clear video mode whether are open

Syntax

```
VT _API    VT _GetClearViewState(
        IN HANDLE           hCamera,
        OUT BOOL            &bOn
);
```

Parameters

hCamera

[in] Camera handle

bOn

[out] Return clear video mode of open mark, TRUE for open, FALSE for closed..

VT _SetIO

Set IO control bit, this function need firmware support

Syntax

```
VT _API    VT _SetIO(
        IN HANDLE          hCamera,
        IN BYTE            Data
);
```

Parameters

hCamera

[in] Camera handle

Data

[in] IO control, see Pins Definition and Use.

Data	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
Pins							[1]	(4)

VT _GetIO

Get IO control bit, this function need firmware support

Syntax

```
VT _API    VT _GetIO(
        IN HANDLE          hCamera,
        IN OUT BYTE        *pData
);
```

Parameters

hCamera

[in] Camera handle

Data

[out] Return to length is for 1 bytes of IO control bit , see Pins Definition and Use.

Data	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
Pins					(7)	(8)	(5)	(6)

VT _PixelCalibration

Open bad point detection function. Use this function, please put the aperture of the lens of the camera lens to close or cover. This can more accurately detection bad points. In the process, will end in applications directory generates a log file, the file *. bpt The current record of the camera, and will be in bad points at the next initialize the camera to be automatic loading the document. So for the same camera, only need to do a bad point detection can.

Syntax

VT _API VT _PixelCalibration(

IN HANDLE hCamera

);

Parameters

hCamera

[in] Camera handle

VT _SetShowAutoMark

Set up automatic exposure mark displays a status

Syntax

```
VT _API    VT_SetShowAutoMark(
            IN HANDLE          hCamera,
            IN BOOL            bShow
);
```

Parameters

hCamera

[in] Camera handle

bShow

[in] Shows signs

VT _SetShowVideo

In the use of VT_StartView began after the acquisition, set up if display images

Syntax

```
VT _API VT _SetShowVideo(
        IN HANDLE       hCamera,
        IN BOOL         bShow
);
```

Parameters

hCamera

[in] Camera handle

bShow

[in] Video shows state

VT _SelectAviCompressor

Enumeration in the system has been installed in a video of the compressor, return to users to choose the compression of the four bytes.

Syntax

```
VT _API      VT _SelectAviCompressor(
        IN HANDLE          hCamera,
        OUT DWORD          *fccHandler
);
```

Parameters

hCamera

[in] Camera handle

fccHandler

[out] Back to user choice of video compressor four bytes, if a user cancels the choice, return zero.

VT _SetAviCompressor

Set video compressor

Syntax

```
VT _API      VT _SetAviCompressor(
        IN HANDLE          hCamera,
        IN DWORD           fccHandler
);
```

Parameters

hCamera

[in] Camera handle

fccHandler

[in] Will the video compressor is set up to four byte, if the parameter value is 0, said it does not use video compressor..

VT _SetAELevel

Set up automatic exposure brightness effect

Syntax

```
VT _API      VT _SetAELevel (
        IN HANDLE          hCamera,
        IN VT_AE_LEVEL     AELevel
);
```

Parameters

hCamera

[in] Camera handle

AELevel

[in] Automatic exposure brightness effect

VT _SetGamma

Set Gamma value

Syntax

```
VT _API       VT _SetGamma (
        IN HANDLE          hCamera,
        IN int             _nGammaIdx
);
```

Parameters

hCamera

[in] Camera handle

nGammaIdx

[in] Gamma value, effective value 0-32

VT _GetGamma

Get Gamma value

Syntax

```
VT _API       VT _GetGamma (
        IN HANDLE          hCamera,
        OUT int            *_nGammaIdx
);
```

Parameters

hCamera

[in] Camera handle

*_nGammaIdx

[out] Gamma value

VT _SetTriggerMode

Set state trigger, in this mode camera do not upload image, only in the camera to trigger signal after receiving a frame to upload images.

Syntax

```
VT _API    VT _SetTriggerMode(
            IN HANDLE hCamera,
            IN BOOL bEnable
            );
```

Parameters

hCamera

[in] Camera handle

bEnable

[in] TRUE for open trigger mode, FALSE for close trigger mode.

VT _SoftTrigger

Soft trigger collection, will need to hardware PA2 output and PE0 connected, and applied to trigger VT_SetTriggerMode open trigger collection mode. In the open trigger collection mode, you can use this function in the upper application software collection of images.

Syntax

```
VT _API    VT _SoftTrigger(
            IN HANDLE hCamera
            );
```

Parameters

hCamera

[in] Camera handle

VT _SetDeviceEven

Setting equipment news callback function

Syntax

```
VT _API    VT _SetDeviceEvent(
            IN HANDLE hCamera,
            IN DL_AUTOCALLBACK Event,
            IN LPVOID lpContext
            );
```

Parameters

hCamera

[in] Camera handle

Event

[in] Callback function

lpContext

[in] Context

VT _VideoPause

Suspended video

Syntax

```
VT _API    VT _VideoPause(
            IN HANDLE hCamera,
            IN BOOL bPause
            );
```

Parameters

hCamera

[in] Camera handle

bPause

[in] Whether video, TRUE for suspended video, FALSE for close suspended video

VT _GetRgbFrameToBmpZoom

Will a frame image save for the wide high half the size of 24 (bit) BMP file

Syntax

```
VT _API    VT _GetRgbFrameToBmpZoom(
        IN HANDLE           hCamera,
        IN CapInfoStruct    *pCapInfo,
        OUT BYTE            *pDest,
        LPCTSTR             strFileName,
        IN BOOL             SaveData
);
```

Parameters

hCamera

[in] Camera handle

pCapInfo

[in][out] If have open video preview, then the current video parameter acquisition of a frame original data to the pCapInfo-> Buffer; If no open video preview, then the pCapInfo description of the image size acquisition of a frame raw data to pCapInfo-> Buffer.

pDest

[out] Used to store data memory address, RGB if SaveData set to FALSE, the parameters can be NULL

strFileName

[in] BMP file name

SaveData

[in] Whether it will be saved to the selected memory RGB data (pDest).

VT _GetRgbFrameToJpegZoom

Will a frame image save for the wide high half the size of 24 (bit) JPEG file

Syntax

```
VT _API    VT _GetRgbFrameToJpegZoom(
        IN HANDLE          hCamera,
        IN CapInfoStruct   *pCapInfo,
        OUT BYTE           *pDest,
        LPCTSTR            strFileName,
        int                nQuality
);
```

Parameters

hCamera

[in] Camera handle

pCapInfo

[in][out] If have open video preview, then the current video parameter acquisition of a frame original data to the pCapInfo-> Buffer; If no open video preview, then the pCapInfo description of the image size acquisition of a frame raw data to pCapInfo-> Buffer.

pDest

[in] Reserves

strFileName

[in] Jpeg file name

nQuality

[in] Jpeg files file quality, value (1-100)

VT _SetHDR

Open camera high dynamic range effect (HDR)

Syntax

```
VT _API    VT _SetHDR(
            IN HANDLE hCamera,
            IN BOOL bEnable
            );
```

Parameters

hCamera

[in] Camera handle

bEnable

[in] TURE for open, FALSE for closed.

Note

Currently only support color camera and VT36M

VT _ResetLightAvgTable

Reset brightness balanced table

Syntax

```
VT _API    VT _ResetLightAvgTable(
            IN HANDLE hCamera
            );
```

Parameters

hCamera

[in] Camera handle

VT _EnableLightAvg

Open brightness balanced table

Syntax

```
VT _API    VT _EnableLightAvg(
            IN HANDLE hCamera,
            BOOL bOn
            );
```

Parameters

hCamera

[in] Camera handle

bOn

[in] Set up open brightness balanced table, TURE for open, FALSE for closed.

VT _EEProm_Set

Set store the data

Syntax

VT _API VT _EEProm_Set(

IN HANDLE hCamera,

IN BYTE *pData

);

Parameters

hCamera

[in] Camera handle

pData

[in] Set data content

Note

To VT -F300CF, VT -F130MF, VT -F500CF have 40 bytes is available, to VT -F130C, VT -F300C, VT -F500C, VT 36M have 240 bytes is available.

VT _EEProm_Get

Get store the data

Syntax

VT _API VT _EEProm_Get(

IN HANDLE hCamera,

OUT BYTE *pData

);

Parameters

hCamera

[in] Camera handle

pData

[out] Get data content

Note

To VT-F300CF, VT-F130MF, VT-F500CF have 40 bytes is available, to VT -F130C, VT -F300C, VT -F500C, VT36M have 240 bytes is available.

VT_EEProm_SetByte

Set a fixed position of fixed size data

Syntax

VT_API VT_EEProm_SetByte(

IN HANDLE hCamera,

IN int nIndex,

IN int nCount,

IN BYTE *pData

);

Parameters

hCamera

[in] Camera handle

nIndex

[in] Start bytes, value range is 0-39/239

nCount

[in] Data size, value range is 1-40/240, nIndex + nCount < = 40/240

pData

[in] data

Note

To VT-F300CF, VT-F130MF, VT-F500CF have 40 bytes is available, to VT-F130C, VT-F300C, VT-F500C, VT36M have 240 bytes is available.

VT_EEProm_GetByte

Get a fixed position of fixed size data

Syntax

VT_API VT_EEProm_GetByte(

IN HANDLE hCamera,

IN int nIndex,

IN int nCount,

OUT BYTE *pData

);

Parameters

hCamera

[IN] Camera handle

nIndex

[IN] Start bytes, value range is 0-39/239

nCount

[IN] Data size, value range is 1-40/240, nIndex + nCount < = 40/240

pData

[OUT] data

Note

To VT-F300CF, VT-F130MF, VT-F500CF have 40 bytes is available, to VT-F130C, VT-F300C, VT-F500C, VT36M have 240 bytes is available.

VT_SetTriggerDelayTime

Set the camera trigger delay acquisition time

Syntax

VT_API VT_SetTriggerDelayTime(

IN HANDLE hCamera,

IN LONG nTime

);

Parameters

hCamera

[IN] Camera handle

nTime

[IN] Delay time, unit: ms

Note

At present, only for VT-F300CF, VT-F130MF, VT-F500CF effective

VT_SetCustomIO

Set the user defined IO

Syntax

VT_API VT_SetCustomIO(

IN HANDLE hCamera,

IN BYTE data

);

Parameters

hCamera

[IN] Camera handle

data

[IN] 0 low level, 1 high level

Note

At present, only for VT-F300CF, VT-F130MF, VT-F500CF effective

VT_SetTriggerSource

Set the camera that can trigger source

Syntax

VT_API VT_SetTriggerSource(

IN HANDLE hCamera,

IN BYTE data

);

Parameters

hCamera

[IN] Camera handle

data

[IN] Control, 0 close，1 open

Note

At present, only for VT-F300CF, VT-F130MF, VT-F500CF effective

VT_SetTriggerSourceTime

Set the camera trigger source time interval

Syntax

VT_API VT_SetTriggerSourceTime(

IN HANDLE hCamera,

IN LONG nTime

);

Parameters

hCamera

[IN] Camera handle

nTime

[IN] time, unit: ms

Note

At present, only for VT-F300CF, VT-F130MF, VT-F500CF effective

VT_SetHBlank

Set the camera line blank

Syntax

```
VT_API    VT_SetHBlank(
    IN HANDLE hCamera,
    IN int nHBlank
    );
```

Parameters

hCamera

[in] Camera handle

nHBlank

[in] line blank value

VT _SetVBlank

Set the camera frame blank

Syntax

```
VT _API    VT _SetVBlank(
    IN HANDLE hCamera,
    IN int nVBlank
    );
```

Parameters

hCamera

[in] Camera handle

nHBlank

[in] frame blank value

VT_AEParam

Set up automatic exposure parameters, real-time automatic exposure

Syntax

```
VT_API    VT_AEParam(
    IN HANDLE hCamera,
    IN BOOL bLiving,
    IN VT _AE_LEVEL AELevel,
    IN DL_AUTOCALLBACK pfAECallback,
    IN LPVOID lpContext
    );
```

Parameters

hCamera

[in] Camera handle

bLiving

[in] Whether open real-time automatic exposure

AELevel

[in] Exposure levels

pfAECallback

[in] Exposure callback function

lpContext

[in] The user context

VT _GetNewCalibrationRawFrame

According to primitive calibration image generation new calibration image, function is set up according to the new gain and contrast, by the original image generation new calibration image, the new calibration image is still as the primary data format (bit).

Syntax

```
VT_API    VT_GetNewCalibrationRawFrame(
              IN HANDLE hCamera,
              IN ULONG lHeight_In,
              IN ULONG lWidth_In,
              IN int iColourType_In,
              IN int iRawDateType_In,
              IN UCHAR cTotalGain_In,
              IN int iContrastValue_In,
              IN UCHAR *pRawFrame_In,
              IN UCHAR cNewTotalGain_In,
              IN int iNewContrastValue_In,
              OUT UCHAR *pRawFrame_Out
              );
```

Parameters

hCamera

[in] Camera handle

lHeight_In

[in] Data acquisition height

lWidth_In

[in] Data acquisition width

iColourType_In

[in] Reserves

iRawDateType_In

[in] Reserves

cTotalGain_In

[in] Reserves

iContrastValue_In

[in] Reserves

pRawFrame_In

[in] Need to convert image data pointer

cNewTotalGain_In

[in] After converting the brightness of the image data

iNewContrastValue_In

[in] After converting image data contrast

pRawFrame_Out

[OUT] After converting the image of the pointer

VT_RawFrameToRgb

The original data to create BMP image data

Syntax

```
VT_API    VT_RawFrameToRgb(IN HANDLE hCamera,
                IN ULONG lHeight_In,
                IN ULONG lWidth_In,
                IN int iColourType_In,
                IN int iRawDateType_In,
                IN UCHAR *pRawFrame_In,
                OUT UCHAR *pRgbData_Out
                );
```

Parameters

hCamera

[in] Camera handle

lHeight_In

[in] Data acquisition height

lWidth_In

[in] Data acquisition width

iColourType_In

[in] Reserves

iRawDateType_In

[in] Reserves

pRawFrame_In

[in] A pointer to the original data

pRgbData_Out

[out] The output of the image data area RGB pointer

Note

This method is currently only support VT-F500CF, old VT-F500CF original image sequence (BGR), for new VT-F500CF for (GRB)

VT_SetLight

Set the brightness values of output image

Syntax

VT_API VT_SetLight(

IN HANDLE hCamera,

IN UCHAR cTotalGain_In

);

Parameters

hCamera

[in] Camera handle

cTotalGain_In

[in] the brightness values of output image

VT_GetLight

Get the brightness values of output image

Syntax

VT_API VT_GetLight(

IN HANDLE hCamera,

OUT UCHAR * cTotalGain_Out

);

Parameters

hCamera

[IN] Camera handle

cTotalGain_Out

[OUT] Return to the brightness values of current output image

VT_WriteDataToDevice

Save data to a camera device (data length 40 byte)

Syntax

VT_API VT_WriteDataToDevice(

IN HANDLE hCamera,

IN BYTE *bData_In,

IN int iOffSet_In,

IN int iNum_In

);

Parameter

hCamera

[IN] Camera handle

bData_In

[IN] Writing to the device data pointer

iOffSet_In

[IN] Write down the position offset data (unit: BYTE)

iNum_In

[IN] Write data length (unit: BYTE)

Note

This method is currently only support VT-EX500CF 、VT-EX500MF.

VT_ReadDataFromDevice

Read the stored in the camera equipment in the data (data length byte)

Syntax

VT_API VT_ReadDataFromDevice(

IN HANDLE hCamera,

OUT BYTE *bData_In,

IN int iOffSet_In,

IN int iNum_In

);

Parameters

hCamera

[IN] Camera handle

bData_Out

[OUT] The output data pointer.

iOffSet_In

[IN] Read first offset data (unit: BYTE)

iNum_In

[IN] To read the data length (unit: BYTE)

Note

This method is currently only support VT-EX500CF 、VT-EX500MF.

.VT_DeviceIoControl

Method for controlling the lamp, byte (24bit) before the number corresponding to the underlying IO output control, a bit control to a LED, 1 bytes are used to control on-off of 24V, 0:close;1:open.

Syntax

VT_API VT_DeviceIoControl(

IN HANDLE hCamera,

IN BYTE *bIoData_In

);

Parameters

hCamera

[IN] Camera handle

bIoData_In

[IN] Control device for the output data of IO (bit + 8bit)

Note

This method is currently only support VT-F500CF 、VT-F500MF.

VT_MCResolutionShow

Set up if application custom cut resolution show preview

Syntax

VT_API VT_MCResolutionShow(

IN HANDLE hCamera,

IN BOOL nEnable

);

Parameters

hCamera

[in] Camera handle

nEnable

[in] TRUE, open; FALSE, close.

Note

In the display resolution after cutting open the window, preview images to display the long image data original image data is specified in the location of the image

VT_IsMCResolutionShow

Get whether open cuting resolution display

Syntax

VT_API VT_IsMCResolutionShow(

IN HANDLE hCamera,

OUT BOOL *nEnable

);

Parameters

hCamera

[in] Camera handle

nEnable

[out] TRUE, open; FALSE, close.

VT_SetMCResolution

Setting cutting resolution display related parameters

Syntax

```
VT_API    VT_SetMCResolution(
      IN HANDLE hCamera,
      IN int nOffX,
      IN int nOffY,
      IN int nWidth,
      IN int nHeight
);
```

Parameters

hCamera

[IN] Camera handle

nOffX

[IN] Setting cutting resolution relative to the current camera to the bottom left corner of the image collection of X offset

nOffY

[IN] Setting cutting resolution relative to the current camera to the bottom left corner of the image collection of Y offset

nWidth

[IN] Set the width of cutting resolution

nHeight

[IN] Set the height of cutting resolution

Note

In the application of the set the current function setting resolution after resolution or change the color mode, cutting the migration will resume for resolution, size will set to the current setting and the resolution of the same. In the set please ensure that data set the legitimacy of the nOffX + nWidth < = : (the current camera Settings of the resolution of the wide), nOffY + nHeight < = (the current camera Settings of the resolution of the high), nWidth and nHeight try to set to integer times. Coordinates with the origin in the callback function of image data obtained in the coordinates of origin is consistent

VT_GetMCResolution

Getting cutting resolution display related parameters

Syntax

VT_API

VT_GetMCResolution(

IN HANDLE hCamera,

OUT int *nOffX,

OUT int *nOffY,

OUT int *nWidth,

OUT int *nHeight

);

Parameters

hCamera

[IN] Camera handle

nOffX

[IN] Getting cutting resolution relative to the current camera to the bottom left corner of the image collection of X offset

nOffY

[IN] Getting cutting resolution relative to the current camera to the bottom left corner of the image collection of Y offset

nWidth

[IN] Get the width of cutting resolution

nHeight

[IN] Get the height of cutting resolution

VT _SetExpouseDelayTime

Set the exposure time delay

Syntax

VT_API VT_SetExpouseDelayTime(

IN HANDLE hCamera,

IN int nTime

);

Parameters

hCamera

[in] Camera handle.

nTime

[in] the exposure time delay，range 0-2047

Note

Currently only support VT-F130C, VT-F300C, VT36M, VT-F500C

VT_SetExpouseUnit

Exposure unit set，default for line number

Syntax

VT_API　VT_SetExpouseUnit(

IN HANDLE hCamera,

IN int nUnit

);

Parameters

hCamera

　　[in] Camera handle.

nUnit

　　[in] Exposure unit, 0 for line number, 1 for us

VT_GetSerialNumber

Gets the camera's serial number, the sequence number cannot be modified.

Syntax

VT_API　　VT_GetSerialNumber(

　　IN HANDLE hCamera,

　　BYTE *pData

　　);

Parameters

hCamera

　　[IN]　Camera handle.

pData

　　[OUT]　To obtain the data, ensure the pData size of at least 16 bytes.

Note

　　Only support the cameras that without the frame memory series :6w、130w、300w、500w.

VT_FrameToImage

Incoming data in memory is saved as an image

Syntax

```
VT_API VT_FrameToImage(
        IN LPCTSTR strFileName,
        IN LPVOID pImageData,
        IN ULONG _ulWidth,
        IN ULONG _ulHeight,
        IN ULONG nColor,
        IN VTIMAGETYPR ImgType
);
```

Parameters

strFileName

[IN] To save the image name

pImageData

[IN] Save the image memory data pointer

_ulWidth

[IN] Image width

_ulHeight

[IN] Image height

nColor

[IN] Color bits (8 or 24)

ImgType

[IN] Image type

Note

nColor parameters need to format the image data to set specific 8-bit grayscale or 24-bit RGB images. See the supported image types defined VTIMAGETYPR

VT_SetSaturation

Set the saturation

Syntax

```
VT_API       VT_SetSaturation (
        IN HANDLE                hCamera,
        IN int                   nSaturation
        );
```

Parameters

hCamera

[IN] Camera handle.

nSaturation

[IN] Saturation value, the range of 1-16, the default 8.

.VT_GetSaturation

Gain saturation

Syntax

```
VT_API       VT_GetSaturation (
        IN HANDLE                hCamera,
        OUT int*                 nSaturation
        );
```

Parameters

hCamera

[IN] Camera handle.

nSaturation

[IN] Saturation value, the range of 1-16, the default 8.

.VT_SetSharpness

Set the sharpness.

.Syntax

```
VT_API       VT_SetSharpness (
        IN HANDLE                hCamera,
        IN int                   nSharp
```

);

Parameters

hCamera

[IN] Camera handle.

nSharp

[IN] The Sharpness value, the range of 1-8, the default 1.

Note

The camera of VT 130SC_YUV ,the range of 1-8, the default 1.

.VT_GetSharpness

Get the Sharpness.

Syntax

```
VT_API      VT_GetSharpness (
    IN HANDLE           hCamera,
    OUT int*            fValue
    );
```

Parameters

hCamera

[IN] Camera handle.

fValue

[IN] The Sharpness value，the range of 1-8, the default 1.

Note

The camera of VT 130SC_YUV ,the range of 0-7, the default 0.

VT_GetSerialNumber

Get camera serial number, this serial number do not change

Syntax

```
VT_API   VT_GetSerialNumber(
        IN HANDLE hCamera,
        BYTE *pData
        );
```

Parameters

hCamera

[IN] Camera handle.

pData

[OUT] Get the data, ensure pData size at least 16 bytes.

Note

For VT- FF series do not provide this function.

VT_SetTriggerCheckSignalTime

Set trigger testing time

Syntax

VT_API VT_SetTriggerCheckSignalTime(

IN HANDLE hCamera,

IN USHORT nTime

);

Parameters

hCamera

[in] Camera handle.

nTime

[in] testing time, units: ms.

Note

Currently only support VT36M.

VT_SetCameraLED

Set LED lights

Syntax

VT_API VT_SetCameraLED(

IN HANDLE hCamera,

IN int nLED,

IN int nLight

);

Parameters

hCamera

[in] Camera handle.

nLED

[in] serial number, 0-3

nLight

[in]the state of LED lights. When nLED equal to 0, 1 or 2, nLight is equal to 0 lights off, equal to 127 lights on. When nLED equal to 3, nLight is equal to 1 lights off, equal to 0 lights on.

Note

This function needs hardware support.

AWB/AE Callback Function

In the end to be automatic white balance/automatic exposure was called after the callback function, used to return to the result of the automatic process

Syntax

```
typedef  VOID      (WINAPI *DL_AUTOCALLBACK )(
         DWORD     dw1,
         DWORD     dw2,
         LPVOID    lpContext
         );
```

Parameters

dw1

Return to the result of the automatic process

Dw2

Return to the result of the automatic process

lpContext

In the open, the automatic process set user context, used to convey parameters.

Frame Callback Function

Through the VT_SetFrameCallBack set frame callback function, used to frame the operation of the data

Syntax

```
typedef    VOID    (WINAPI *DL_FRAMECALLBACK)(
            LPVOID lpParam1,
            LPVOID lpReserved,
            LPVOID lpContext
);
```

Parameters

lpParam1

Store the data of the memory address frame.

lpReserved

Reserves

lpContext

Set frame callback function, setting user context, used to convey parameters.

SDK Use Examples

Initial CapInfoStruct structures

```
BYTE * m_pRawData = NULL;
BYTE * m_pRgbData = NULL;
CapInfoStruct m_CapInfo;
// ...
//To VT-F300C for example, according to the current camera maximum resolution of support for the original data memory,
// In order to avoid the spill, usually the length of the application for more than than the actual line.

m_pRawData = ( BYTE* ) new BYTE [ 2048 * 1536 + 2048 ];

// Apply for 24 bits of data in memory, for RGB collection needs to use the pictures.
m_pRgbData = ( BYTE* ) new BYTE [ 2048 * 1536 * 3 +512];

memset( &m_CapInfo, 0, sizeof( CapInfoStruct ) );
m_CapInfo.Buffer = m_pRawData;     // A specified data buffer address
m_CapInfo.Width      = 1280;       // Sets the default width resolution
m_CapInfo.Height     = 1024;       // Sets the default high resolution
m_CapInfo.Exposure = 100;          // Sets the default exposure value
memset( m_CapInfo.Gain, 30, 3 );   // Sets the default gain
```

Initialization Camera

```
HANDLE    m_hDevice;             //Camera handle
int   nIndex = 1;              // Designated camera serial numbers, the first camera to 1
if( ResSuccess != VT_Initialize(    //
                  _T("VT_DEMO"),    //
                  &nIndex,          // camera serial numbers
                  &m_CapInfo,       // Has been initialized structures
                  &m_hDevice ) )    //If the call is successful, return Camera handle.
{
     VT_Uninitialize( m_hDevice );     // If not, the uninstall camera success
     m_hDevice = NULL;
     ::MessageBox( NULL, _T("Initial Error"), _T("Demo"), 0 );
     return;
}
```

Set Camera Parameters

```
VT_SetCapInfo( m_hDevice, &m_CapInfo );          //
VT_SetParamColor(m_hDevice, COLOR_RGB24 );       // Set Color Mode
VT_SetParamFlip(m_hDevice, FLIP_NATURAL );       // Set Flip Mode
VT_SetContrastValue(m_hDevice,m_nContrast);      // Set Contrast

VT_SetGainR(m_hDevice, m_nR);      //Set red gain
VT_SetGainG(m_hDevice, m_nG);      // Set green gain
VT_SetGainB(m_hDevice, m_nB);      // Set blue gain
```

Video Preview

```
        RECT       rect;
    ::GetClientRect(m_hWnd, &rect);
    VT_StartView(
            m_hDevice,                          //Camera handle
            _T("Digital Lab"),                  // Reserves
            WS_CHILD | WS_VISIBLE,              // Reserves
            0, 0,                   // Preview the upper left corner of the image coordinates
            rect.right-rect.left,               // Preview images the width of the drawing
            rect.bottom-rect.top,               // Preview images the height of the drawing
            m_hWnd,                             // Preview image drawing window handle
            NULL );                             // Reserves

    // Stop video preview
    VT_StopView( m_hDevice );
```

Acquisition of A Frame Image

```
CString strPath;
// Will a frame data save BMP file
strPath = _T("Sample.bmp");
VT_GetRgbFrameToBmp(
        m_hDevice,      //Camera handle
        &m_CapInfo,     // CapInfoStruct structures, if not open preview,
    //as the structure of the body image size acquisition a describe images
        m_pRgbData,     // RGB data buffer
        strPath,        // filename
    TRUE); // If TRUE, it will be the frame RGB data replication to RGB buffer.

// Will a frame for data save JPG document
strPath = _T("Sample.jpg");
VT_GetRgbFrameToJpeg(
        m_hDevice,      //Camera handle
        &m_CapInfo,     // CapInfoStruct structures, if not open preview,
    //as the structure of the body image size acquisition a describe images
        NULL,           // Reserves
        strPath,        // filename
        100);           // JPG image quality
```

Video and the Compression

```
CString strPath = _T("sample.avi");

// Begin to video
VT_CaptureToAvi(
    m_hDevice,  //Camera handle
    TRUE,       // If TRUE, the start of the video
    strPath);   //

// Stop video
VT_CaptureToAvi(
    m_hDevice,  //Camera handle
    FALSE,      // If be FALSE, then stop the video
    _T(""))     //

// Choose the compression
DWORD m_dwCompressor = 0;
DWORD dwCode = 0;
// This function has been installed in the system of the enumeration compressor
if(0 == VT_SelectAviCompressor(
        m_hDevice,      //Camera handle
        &dwCode))       // Return to the compression of the four bytes
{
    m_dwCompressor = dwCode; // If successful, save the compression
}
```

```
// Set the compression
// If set to success, the video using the compressor many video compression
VT_SetAviCompressor(
    m_hDevice,          //Camera handle
    m_dwCompressor);

// Cancel the compression
VT_SetAviCompressor(
    m_hDevice,          //Camera handle
    0);                 // Do not use the compression
```

Frame callback function

Frame callback functions, is in the image data to be before drawing and to offer users of image data interface operation.

First define a callback function as follows:

```
VOID CALLBACK FrameCallback(
    LPVOID pData,           // Image data address
    LPVOID lpReserve,       // Reserves
    LPVOID lpContext )      /Set the callback functions of the context
{
    // Frame callback function, please return as soon as possible.
    TRACE(_T("--- frame callback --- \n"));

    // For example, in RGB24 mode will be the image up 50 points set to red
    int n;
    for(n=0; n<50; n++)
    {
        *(pData+n*3+0) = 0;     //B
        *(pData+n*3+1) = 0;     //G
        *(pData+n*3+2) = 255;   //R
    }
};
```

Set frame callback function

```
VT_SetFrameCallBack(
    m_hDevice,          //Camera handle
    FrameCallback,      // frame callback function
    LPVOID(NULL));  // Context, in the callback functions, from the third parameter redirect
```

```
Cancel frame callback function
VT_SetFrameCallBack(
    m_hDevice,
    NULL,
    LPVOID(NULL));
```

Automatic Exposure (AE)

// First need to define a automatic exposure callback functions, in the automatic exposure processes completed is invoked.

```
VOID CALLBACK AECallBack(
    DWORD dw1,          // Automatic process results
    DWORD dw2,          // Automatic process results
    LPVOID lpContext)
// In the opening set when automatic exposure of the context
{
    // Among them dw1 for ae process, the current exposure value
    // With the results of the dw1 update user interface
};

// Open automatic exposure
VT_SetDoAE(
    m_hDevice,      //Camera handle
    bAE,            // If for TURE, it is opened automatic exposure
    0,              // Reserves
    AECallBack,     // Automatic exposure callback function
    lpContext );    // context
```

Automatic White Balance (AWB)

```
// First need to define an automatic white balance callback functions, in the automatic white balance after the finish was called.

VOID CALLBACK AWBCallBack(
    DWORD dw1,          // Automatic process results
    DWORD dw2,          // Automatic process results
    LPVOID lpContext)  //Context
{
    // One of the dw1 high 16 for the current R gain value, low 16 for the current G gain value
    // Dw2 high 16 for the current B gain value

    int r, g, b;
    r = (dw1>>16) & 0x0000FFFF;
    g = dw1 & 0x0000FFFF;
    b = (dw2>>16) & 0x0000FFFF;
};

// Open to be automatic white balance
VT_SetDoAWB(
    m_hDevice,      //Camera handle
    bAWB,           // If for TRUE, it is loading automatic white balance
    0,              // Reserves
    AWBCallBack,    // AWB callback functions
    lpContext );    //Context
```

Trigger

State trigger:

In the startup image, if open the preview after the state trigger, then, when triggered signals received automatically, and save the image of a frame.

/Set the state trigger

```
VT_GetTrigger(
    m_hDevice,      //Camera handle
    bOn,            // TRUE for open trigger mode, FALSE for close trigger mode.
    BMP,            // image types
    lpPath,         // The file name, do not contain extensions
    100)            // As the picture type is JPEG, set image quality
```

Trigger collection mode:

When opening the trigger collection model, after only in signal acquisition data triggered. Even if open preview, if not trigger signal, also won't collect data.

```
// Set trigger collection mode
// By CapInfoStruct trigger collection structure in the body of the Control bytes minimum the second to Control.
// When this one is 1, trigger collection mode is effective.
void SetTriggerMode(BOOL bOn)
{
    if(bOn)
        m_CapInfo.Control |=    0x02;   // 0000 0010
    else
        m_CapInfo.Control &=    0xFD;   // 1111 1101
    VT_SetCapInfo(m_hDevice,&m_CapInfo);
}
```

Appendix

Pins Definition and Use

Table 1 test pins position

9	TXD0	PE3	PE1	GND	PA2	1
10	RXD0	PE2	PE0	PC2	VCC	2

Definition

1	PA2	General IO (output) can be used for soft trigger the output
2	VCC	+5V
3	GND	grounding
4	PC2	General IO (output)
5	PE1	General IO (input)
6	PE0	General IO (input) As with the trigger pins, low level and effective
7	PE3	General IO (input)
8	PE2	General IO (input)
9	TXD0	invalid
10	RXD0	invalid

The use of general IO pins

General IO tube feet can be used as a signal input or signal feedback and therefore can realize the response or trigger certain external function.

For example, through the button control the function of light LED. Will pick up in the general IO tube LED feet up in PA2, switch general IO tube feet PC2, when switch press, will PC2 set to low level, this time the firmware program to this signal detection, the response, light and general IO tube feet PA2 connection LED.

That is worth the, the particular tube feet detection and control by the program need to realize the corresponding function. If the program did not realize these functions, even if the connection switch or LED, these tube feet also do not rise to any role.

Connection PA2 and PE0 soft trigger function can be realized.

Note

1．If a IO tube from external to the feet, send signals to ensure that the frequency of

the signal not more than 48 MHz can be effectively to detect.

2．PE0 ~ PE3 tube feet all have through a 330Ω resistance to pull 3.3 V. (figure 1)

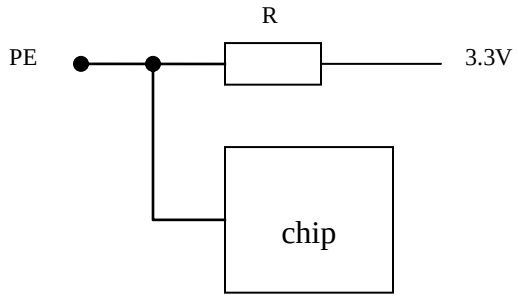


figure 1

3．Signal response time

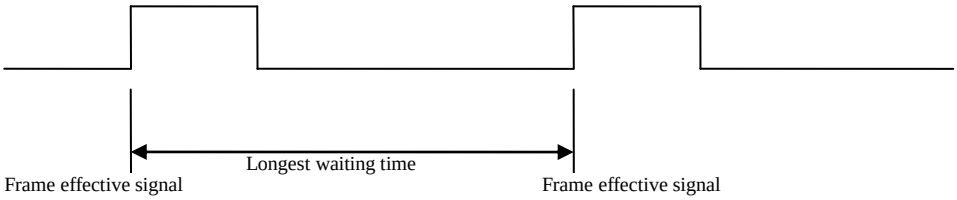


figure 2

As in the collection of data frame does not respond to pipe of the signal, the foot to foot the response signal in the collection of data frame only intervals, so the longest signal response time is from a frame effective signal to the next frame effective signal interval of time (figure 2). And this time DuoGe factors by effects, including: the resolution of the image collection, exposure value, etc. In the application of, frame rate directly reflected the various factors that work together. For example the current frame rate for five frames per second, the longest waiting time for 200 milliseconds.