Simply Brighter

(In Canada)
111 Railside Road
Suite 201
Toronto, ON M3A 1B2
CANADA
Tel: 1-416-840 4991
Fax: 1-416-840 6541

(In US)
1241 Quarry Lane
Suite 105
Pleasanton, CA 94566
USA
Tel: 1-925-218 1885
Email: sales@mightex.com

Mightex Buffer USB CCD DirectShow Filter Description

Version 1.0.5

Dec. 27, 2018

Relevant Products

Part Numbers
CCN-B013-U, CCE-B013-U, CCN-C013-U, CCE-C013-U, CXN-B013-U, CXE-B013-U, CGN-B013-U, CGE-B013-U, CGN-C013-U, CGE-C013-U

Revision History

Revision	Date	Author	Description
1.0.0	Dec. 23, 2008	YH Wu	Initial Revision
1.0.1	Apr. 10, 2010	JT Zheng	Adding CGX, CXX modals
1.0.2	Apr. 16, 2010	JT Zheng	New Installation Description
1.0.3	Jul. 08, 2011	YH Wu	Multiple Camera Supports
1.0.4	Oct.21, 2011	YH Wu	X64 Supports
1.0.5	Dec. 27,2018	JT Zheng	New Mightex Logo

The Mightex Buffer CCD Camera DirectShow filter enables Mightex Buffer USB CCD Camera works as standard Video Capture Device on MS Windows, this enables Mightex Buffer CCD Camera works seamlessly in most DirectShow aware applications.

The driver provides property pages and the COM interfaces for DirectShow applications to control Mightex Buffer Camera. For applications to use the driver, it should add this Mightex filter to a filter graph, use the System Device Enumerator, which returns a unique moniker for the Mightex Buffer CCD Camera. The friendly name of the filter is **Mightex_BufferCCDSource**.

Note: while there're more than one cameras are attached, the filter is working with the first Mightex Buffer CCD camera enumerated on the USB Bus only.

Installation

The CD ROM contains directory "DirectShow" which has the following sub-directories:

The CD ROM contains directory "DirectShow" which has the following sub-directories:
\x86 : Contains all 32bit DirectShow files.
\x64: Contains all 64bit DirectShow files.
\Documents: It includes this document.

For **\x86** and **\x64** dir, each includes the following:
\Filter
It includes the filter file (.ax file) and "RegisterServer.bat" and "unRegisterServer.bat"
\MightexBufferCCDCameraEngine
It includes two DLL files which is Mightex New Classic Camera Runtime Engine.
\Application
Under this directory, a DirectShow application which utilizes the filter (and the customized interface methods) is stored, user might use the application to preview the video and capture the video stream to AVI file. Full source code of this application is provided as an example to use the filter, user might develop his own application based on it.
Note: while capturing AVI file to disk, the actual frames might be less than the setting frame rate due to the bandwidth limitation of PC resources (for video compression and disk operation). E.g. While setting 24fps @1280x1024, it's actually not achievable on most of the PC, as PC can NOT afford enough resources to compress and store the image to Hard disk with the bandwidth at this setting.

Please follow the steps below for installing the filter:

1). Make sure the device driver is installed properly prior to using DirectShow features. For driver installation, please refer to "**Camera Quick Start Guide**".
To confirm successful driver installation**,** go to your hard disk (where you have copied the content from the CD) and find the folder named **\Application,** where you will see the file **BUFCCDCameraApp.** Please run the program by double clicking on it – after you have confirmed that the camera is working well, close the application window and go to step 2.

2). The filter needs the DLL files in "\MightexBufferCCDCameraEngine" to be copied to the **\<WINDOWS>\system32** directory, these DLL files are Mightex Buffer Camera Runtime Engine. For installing 32bit DirectShow driver on 64bit windows, user should copy the DLL files under \x86\MightexBufferCameraEngine into **WINDOWS\SysWOW64** directory).

3). Now you need to register DirectShow Filter. Please go to your hard disk (where you have copied the content from the CD) and find the folder named **\DirectShow\x86\Filter (or**

\x64\filter), where you will see: **RegisterServer.bat** , and run the **.bat** file by double clicking on it.

You should get a message with: Registration successful.

4) Confirm successful installation. Please go to your hard disk (where you have copied the content from the CD) and find the folder named **\DirectShow\x86\Application (or \x64\Application),** where you will see **BUFCCDCameraApp.** Please run the program by double clicking on it, and you should now be able to control the camera using the software.

Important:

For 32bit Windows, user should always use the files under \x86 for the DirectShow filter (the ax file) and the camera engine files (the two DLL files) installation.

For 64bit Windows user, user might install 32bit DS filter OR 64bit DS filter OR both, depends on the DS application user intends to use (e.g. the Matlab software). For installing 64bit DS filter, user should use the files under \x64 sub-dir. For installing 32bit DS filter, user should use the files under \x86 sub-dir, and the camera engine (the two DLL files) should be copied into the \Windows\SysWow64 directory.

Note:

1). Please remember the camera has to be connected to the PC throughout the process.
2). There should be NO space in the directory name (e.g. a dir name of “Mightex Product” is not proper) for the installation directory on the local disk, as the bat files (e.g. “RegisterServer .bat”) will recognize the space as end of the path thus the filter registry will fail if there’s space in the directory name.

Multiple Camera Supports

When there’re more than one cameras in the application, user should edit the “RegisterServer.bat” file to: *regsvr32 /n /i:X "%~dp0\ Mightex_BufferCCDSource.ax"* and run it. X is the camera count(1-8).

Filter Description

The Mightex_BufferCCDSource Video Capture Filter for Mightex Buffer CCD Camera has the following attributes, mainly it exposes the interfaces listed in the first row of the table below. The application can use **QueryInterface** to find all of these interfaces:

Filter Interfaces	**IAMStreamConfig**, **IAMVideoProcAmp**, **IAMVideoControl**, **IMightex_BufferCCDSource**, **ISpecifyPropertyPages**, **IAMCameraControl.**
Input Pin Media Types	No input pin. Only one output pin.
Input Pin Interfaces	No input pin. Only one output pin.
Output Pin Media Types	MEDIATYPE_Video (RGB24)
Output Pin Interfaces	**IKsPropertySet**, **ISpecifyPropertyPages**
Filter CLSID	CLSID_Mightex_BufferCCDSource
Property Page CLSID	Driver-dependent.
Plug-in Executable	Mightex_BufferCCDSource.ax
Merit	MERIT_DO_NOT_USE
Filter Category	CLSID_VideoInputDeviceCategory

Interfaces:

IMightex_BufferCCDSource

The **IMightex_BufferCCDSource** interface is a customized interface defined by Mightex. The interface enables user to have detailed and efficient controls on the parameters of the Mightex buffer camera.

With this interface, User might set video format properties, such as the output dimensions and frame rate. Also use this interface to set the exposure time, the X, Y Start position (at a certain resolution) and the Red, Green and Blue pixels' gains. The interface also enables an application to adjust the qualities of an incoming video signal, such as brightness, contrast, gamma, and sharpness. The interface also enables user to flip a picture horizontally and/or vertically, set the camera to "Trigger" mode which can capture image while an external trigger asserts.

In addition to the methods inherited from **IUnknown**, the **IMightex_BufferCCDSource** interface exposes the following methods.

Method	Description
SetStreamFrameRate	User may set the frame rate by invoking this function, this actually set previewing and capturing video frame rate precisely.
SetPhysicalCameraFrameRate	User may set the frame rate by invoking this function, this actually set camera physical frame rate.
GetPhysicalCameraFrameRate	Retrieves the actual frame rate at which the Camera is streaming. This method is used with the Camera, where the maximum frame rate can be limited by bandwidth availability. The actual frame rate is affected by some factors, such as resolution itself, exposure time, maximum vertical blanking time of a frame…etc .This is only available during video streaming.
GetModuleNoSerialNo	For a present camera device, user might get its Module Number and Serial Number by invoking this function.
SetCameraWorkMode	By default, the Camera is working in "NORMAL" mode in which camera deliver frames to Host continuously, however, in some applications, user may set it to "TRIGGER" Mode, in which the camera is waiting for an external trigger signal and capture ONE frame for each trigger signal.
SetResolution	User may set the width / height size by

	invoking this function.
SetExposureTime	User may set the exposure time by invoking this function.
SetXYStart	User may set the X, Y Start position (at a certain resolution) by invoking this function.
SetGains	User may set the Red, Green and Blue pixels' gain by invoking this function.
SetGamma	User may set the Gamma, Contrast, Brightness and Sharpness level for Camera by invoking this function.
SetBWMode	User may set the processed Bitmap data as "Black and White" mode, "Horizontal Mirror" and "Vertical Flip" for camera by invoking this function.
Snapshot	User may grab still images from the stream and store in files by invoking this function.
GetCameraControl	User may get the camera control and global control parameters by invoking this function.
SetSelectCamera	User may set the camera id and select which camera.

HEADER FILE:

The “IMightex_BufferCCDSource.h” is as following:

```
//-------------------------------------------------------------------------
// File: IMightex_BufferCCDSource.h
//
// Desc: DirectShow code - custom interface allowing the user
//             to do some settings.
//
// Copyright (c) Mightex Corporation.    All rights reserved.
//-------------------------------------------------------------------------


#ifndef __H_IMightex_BufferCCDSource__
#define __H_IMightex_BufferCCDSource__

#ifdef __cplusplus
extern "C" {
#endif

      // {BA2CAB76-F2BE-4aaf-BED7-CEF05D8B9623}
      DEFINE_GUID(CLSID_Mightex_BufferCCDSource,
            0xba2cab76, 0xf2be, 0x4aaf, 0xbe, 0xd7, 0xce, 0xf0, 0x5d, 0x8b, 0x96, 0x23);

      // {035F169D-D936-4916-96DC-A45C9CA95E01}
      DEFINE_GUID(IID_IMightex_BufferCCDSource,
            0x35f169d, 0xd936, 0x4916, 0x96, 0xdc, 0xa4, 0x5c, 0x9c, 0xa9, 0x5e, 0x1);

      const GUID CLSID_Mightex_BufferCCDSources[8] = {
            // {BA2CAB76-F2BE-4aaf-BED7-CEF05D8B9623}
            { 0xba2cab76, 0xf2be, 0x4aaf, {0xbe, 0xd7, 0xce, 0xf0, 0x5d, 0x8b, 0x96, 0x23} },
            // {BA5491F0-62BD-43a8-AA11-1E25871F6F40}
            { 0xba5491f0, 0x62bd, 0x43a8, {0xaa, 0x11, 0x1e, 0x25, 0x87, 0x1f, 0x6f, 0x40} },
            // {BD0F418D-640F-44ff-8766-851C8E0B649F}
            { 0xbd0f418d, 0x640f, 0x44ff, {0x87, 0x66, 0x85, 0x1c, 0x8e, 0xb, 0x64, 0x9f} },
            // {DB68EA6A-414E-469f-91C3-22983C46A887}
            { 0xdb68ea6a, 0x414e, 0x469f, {0x91, 0xc3, 0x22, 0x98, 0x3c, 0x46, 0xa8, 0x87} },
            // {DD7BC8CE-3C14-4a93-A44F-8609FB12E79E}
            { 0xdd7bc8ce, 0x3c14, 0x4a93, {0xa4, 0x4f, 0x86, 0x9, 0xfb, 0x12, 0xe7, 0x9e} },
            // {E754A154-2890-424c-8C70-84224C9ACA58}
            { 0xe754a154, 0x2890, 0x424c, {0x8c, 0x70, 0x84, 0x22, 0x4c, 0x9a, 0xca, 0x58} },
            // {E9CD40E2-CBAC-463e-BF15-49C6F28AD12B}
            { 0xe9cd40e2, 0xcbac, 0x463e, {0xbf, 0x15, 0x49, 0xc6, 0xf2, 0x8a, 0xd1, 0x2b} },
            // {F4D40BAC-F0E2-4bbe-B8ED-2A996BB99567}
            { 0xf4d40bac, 0xf0e2, 0x4bbe, {0xb8, 0xed, 0x2a, 0x99, 0x6b, 0xb9, 0x95, 0x67} }
      };

      struct CCameraControl;
      struct CCameraGlobalControl;

      DECLARE_INTERFACE_(IMightex_BufferCCDSource, IUnknown)
      {
            STDMETHOD(SetStreamFrameRate) (THIS_
                  LONGLONG frameRate
```

```
            ) PURE;
        STDMETHOD(SetPhysicalCameraFrameRate) (THIS_
            LONGLONG frameRate
            ) PURE;
        STDMETHOD(GetPhysicalCameraFrameRate) (THIS_
            LONGLONG &actualFrameRate
            ) PURE;
        STDMETHOD(GetModuleNoSerialNo) (THIS_
            char *ModuleNo, char *SerialNo
            ) PURE;
        STDMETHOD(SetCameraWorkMode) (THIS_
            int WorkMode
            ) PURE;
        STDMETHOD(SetResolution) (THIS_
            int width, int height, int bin
            ) PURE;
        STDMETHOD(SetExposureTime) (THIS_
            int exposureTime
            ) PURE;
        STDMETHOD(SetXYStart) (THIS_
            int xStart, int yStart
            ) PURE;
        STDMETHOD(SetGains) (THIS_
            int redGain, int greenGain, int blueGain
            ) PURE;
        STDMETHOD(SetGamma) (THIS_
            int Gamma, int Contrast, int Bright, int Sharp
            ) PURE;
        STDMETHOD(SetBWMode) (THIS_
            int BWMode, int H_Mirror, int V_Flip
            ) PURE;
        STDMETHOD(Snapshot) (THIS_
            char * TargetFile, BOOL SaveAsJPEG, BOOL AppendDataTime, int SaveFileCount
            ) PURE;
        STDMETHOD(GetCameraControl) (THIS_
            CCameraControl &cameraCtrl, CCameraGlobalControl &cameraGlobalCtl
            ) PURE;
        STDMETHOD(SetSelectCamera) (THIS_
            int devID
            ) PURE;
    };

#ifdef __cplusplus
}
#endif

#endif // __ICONTRAST__
```

HRESULT SetStreamFrameRate (LONGLONG frameRate);

User may set the frame rate by invoking this function, this sets previewing and capturing video frame rate precisely. User may also use **IAMStreamConfig**::**SetFormat** to do this.

Argument:

frameRate – The frame rate is expressed as frame duration in 100-nanosecond units.

Return: Always return NOERROR.

HRESULT SetPhysicalCameraFrameRate (LONGLONG frameRate);

User may set the physical camera frame rate by invoking this function.

Argument:

frameRate – the frame rate is actually frame time which is in 100-nanosecond units. Camera will do"best effort" to achieve this frame rate, however, under a certain resolution, the minimum and maximum frame rate is also affected by other parameters, such as resolution itself, exposure time, maximum vertical blanking time of a frame…etc. The actual frame rate might not be the set rate.

Return: E_FAIL: If the function fails (e.g. invalid device number)
NOERROR: if the call succedds.

HRESULT GetPhysicalCameraFrameRate (LONGLONG &actualFrameRate);

Retrieves the actual camera frame rate at which the Camera is streaming. This method is used with the Camera, where the maximum frame rate can be limited by bandwidth availability. The actual frame rate is affected by some parameters, such as resolution itself, exposure time, maximum vertical blanking time of a frame…etc. This is only available during video streaming. User might also use **IAMVideoControl**::**GetCurrentActualFrameRate** for the similar purpose.

Argument:

actualFrameRate – Pointer to the frame rate in frame duration in 100-nanosecond units.

Return: Always return NOERROR.

Note:

With the customized interface, user might set the frame rate in two levels:

***.** At the physical device level, user might set the frame rate on the camera itself, so the camera will generate frames at this setting interval, please also note that the actual camera frame rate might not exactly the one user set to it, as it might also be affected by other parameters such as resolution and exposure time..etc.

***.** At Filter level, user might set a frame rate for the generating of the video stream, for example, this frame rate can be set to 30fps always, no matter what the actual camera frame rate it, the filter will do automatic frame insertion or ignorance according to user's settings.

HRESULT GetModuleNoSerialNo(char *ModuleNo, char *SerialNo);

For any present device, user might get its Module Number and Serial Number by invoking this function.

Argument:

ModuleNo – the pointer to a character buffer, the buffer should be available for at least 16 characters.

SerialNo – the pointer to a character buffer, the buffer should be available for at least 16 characters.

Return: Always return NOERROR.

HRESULT SetCameraWorkMode (int WorkMode);

By default, the Camera is working in "NORMAL" mode in which camera deliver frames to Host continuously, however, in some applications, user may set it to "TRIGGER" Mode, in which the camera is waiting for an external trigger signal and capture ONE frame for each trigger signal. User might also use **IAMVideoControl**::**SetMode** to do this.

Argument:

WorkMode – 0: NORMAL Mode, 1: TRIGGER Mode.

Return: E_FAIL: If the function fails (e.g. invalid device number)
NOERROR: if the call succeeds.

Important:

NORMAL mode and **TRIGGER** mode have the same features, but:

NORMAL mode – Camera will always grab frame as long as there's available buffer for a new frame, For example, when host (in most cases, it's a PC) is keeping to get frame from the camera, the camera is continuously grabbing frames from CMOS sensor.

TRIGGER mode – Camera will only grab a frame from CMOS sensor while there's an external trigger asserted (and there's available buffer for a new frame).

HRESULT SetResolution (int width, int height, int bin);

User may set the width / height size by invoking this function. Also use **IAMStreamConfig**::**SetFormat**.

Argument:

width– the customer defined width size.

height– the customer defined height size.

bin – 0: Normal mode, 1: 1:2 BIN mode, 2: 1:3 BIN mode , 3: 1:4 BIN mode.

Return: E_FAIL: If the function fails (e.g. invalid device number)

E_INVALIDARG: The customer defined width or height size is out of range.

The valid range for width / height sizes are:

Width: 4 – Maximum width Size (Camera dependant, refer to camera spec.)

Height: 4 – Maximum height Size (Camera dependant, refer to camera spec.)

E_PROP_ID_UNSUPPORTED: The granularity and alignment of customer defined width / height size is 4, otherwise it return –3.

NOERROR: if the call succedds.

HRESULT SetExposureTime (int exposureTime);

User may set the exposure time by invoking this function.

Argument:

exposureTime – the Exposure Time is set in 50 Microsecond UNIT, e.g. if it's 4, the exposure time of the camera will be set to 200us. The valid range of exposure time is 50us – 750ms, So the exposure time value here is from 1 – 4000,000.

Return: E_FAIL: If the function fails (e.g. invalid device number)

NOERROR: if the call succedds.

HRESULT SetXYStart (int xStart, int yStart);

User may set the X, Y Start position (at a certain resolution) by invoking this function.

Argument:

Xstart, YStart – the start position of the ROI (in pixel).

Return: E_FAIL: If the function fails (e.g. invalid device number)

NOERROR: if the call succedds.

Important: While the resolution is NOT set to the Maximum, the setting Region Of Interesting (ROI) is an active region of the sensor pixels, user may use this function to set the left top point of the ROI. Note that under a certain ROI, the Xstart and Ystart have a valid settable range. If any value out of this range is set to device, device firmware will
check it and set the closest valid value.

HRESULT SetGains (int redGain, int greenGain, int blueGain);

User may set the Red, Green and Blue pixels' gain by invoking this function.

Argument:

redGain, greenGain, blueGain – the gain value to be set.

Return: E_FAIL: If the function fails (e.g. invalid device number)

NOERROR: if the call succedds.

Important: The gain value should be from 6 – 41, represents 2x – 112x of analog amplifying Multiples. For Monochrome sensor modules, only GreenGain value is used by the camera firmware.

1). For setting proper exposure for an image, it's recommended to adjust exposure time prior to the gain, as

setting high gain will increase the noise (Gain is similar to the ISO settings on consumer camera), for applications which the SNR is important, it's recommended to set Gain not more than 16dB.

2). For CCX-B013-U camera, although the minimum Gain is 6dB, user might has to set it to 14dB when the camera is NOT in BIN mode, with the current hardware/firmware design, the CCD output (Sony ICX205) is only up to 0.45V as its saturation voltage, even with 6dB gain (2x), it's ~0.9V signal, while the CCD processor is with a 2V reference ADC, only set the Gain to 14dB will let the ADC generate full range data, however, we leave this feasibility to users as in some cases, user might still want to set Gain to 6dB to get optimized SNR (rather than the ADC range).

HRESULT SetGamma (int Gamma, int Contrast, int Bright, int Sharp);

User may set the Gamma, Contrast, Brightness and Sharpness level for ALL Cameras by invoking this function. Also use **IAMVideoProcAmp**::**Set**.

Argument:

Gamma – Gamma value, valid range 1 – 20, represent 0.1 – 2.0

Contrast – Contrast value, valid range 0 – 100, represent 0% -- 100%.

Bright – Brightness value, valid range 0 – 100, represent 0% -- 100%.

Sharp – Sharp Level, valid range 0 – 3, 0: No Sharp, 1: Sharp, 2: Sharper, 3: Sharpest.

Return: E_FAIL: If the function fails,

NOERROR: if the call succedds.

Important: In most cases, the default values (Gamma = 1.0, Contrast and Brightness = 50%, Sharp = 0) are proper.

HRESULT SetBWMode (int BWMode, int H_Mirror, int V_Flip);

User may set the processed Bitmap data as "Black and White" mode, "Horizontal Mirror" and "Vertical Flip" for ALL Cameras by invoking this function. Also use **IAMVideoControl**::**SetMode**.

Argument:

BWMode : 0: Normal, 1: Black and White mode.

H_Mirror: 0: Normal, 1: Horizontal Mirror.

V_Flip: 0: Normal, 1: Vertical Flip.

Return: E_FAIL: If the function fails,
NOERROR: if the call succedds.

HRESULT Snapshot (char *TargetFile, BOOL SaveAsJPEG, BOOL AppendDataTime, int SaveFileCount);

User may grab still image and store them to files by invoking this function.

Argument:

TargetFile: The target files' path and name.

SaveAsJPEG: 0: BMP file, 1: JPG. File.

AppendDataTime: 0: Normal, 1: The files name append date and time.

SaveFileCount:The files count.

Return: Always return NOERROR.

HRESULT GetCameraControl (CCameraControl &cameraCtrl, CCameraGlobalControl &cameraGlobalCtl);

User may get the camera control and global control parameters by invoking this function.

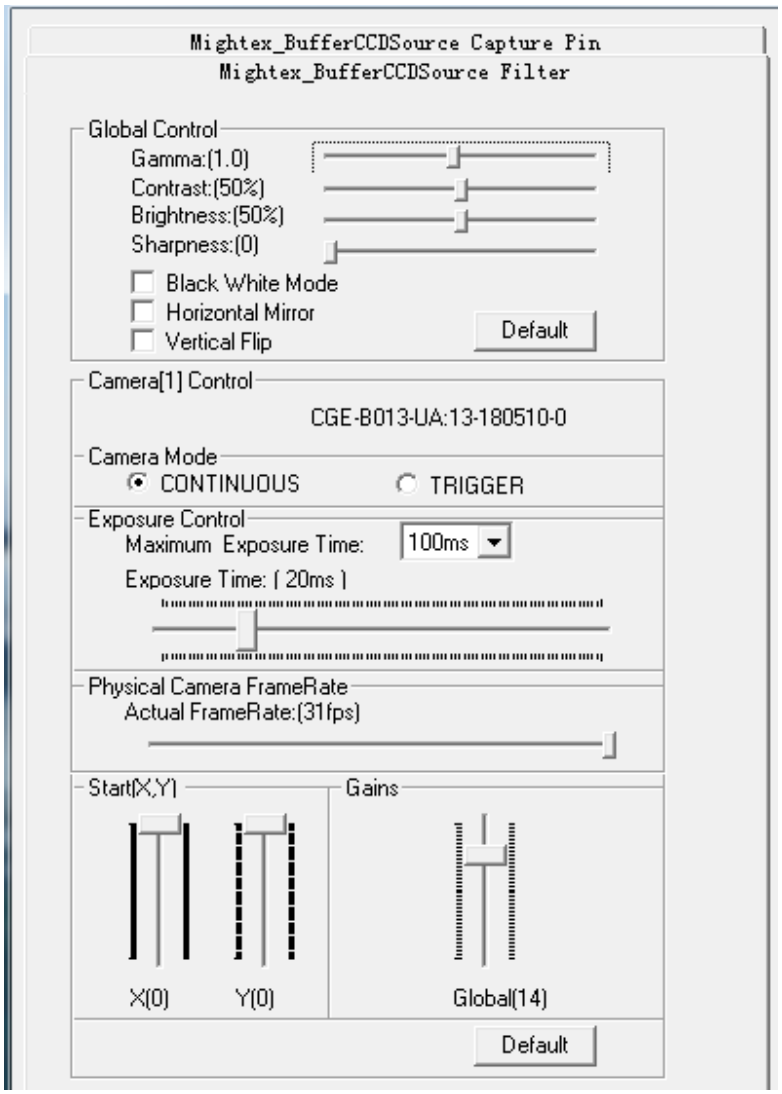
Argument:

cameraCtrl: The Camera Control Parameters structure variable.

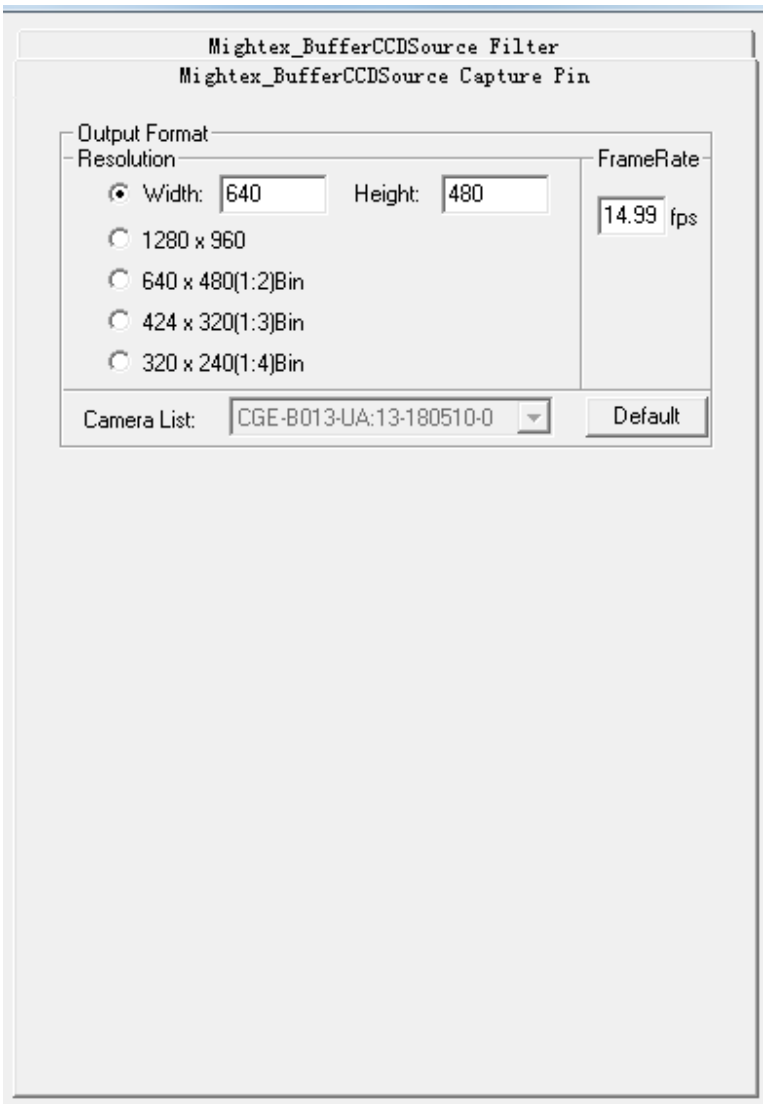
cameraGlobalCtl: The Camera Global Control Parameters structure variable.

Return: Always return NOERROR.

The interfaces of property pages:



Note: In this example, the camera is C013, so it only has Global Gain Control, for 3M Color camera, there're individual gain control for Red, Green and Blue.


Mightex_BufferCCDSource Filter
Mightex_BufferCCDSource Capture Pin
Output Format
Resolution
Width: 640
Height: 480
1280 x 960
640 x 480(1:2)Bin
424 x 320(1:3)Bin
320 x 240(1:4)Bin
FrameRate
14.99 fps
Camera List:
CGE-B013-UA:13-180510-0
Default

All the following interfaces are standard DirectShow interfaces for video capture device, please refer to Microsoft DirectShow documents for the detailed description of the interfaces and their methods.

IAMStreamConfig Interface

The **IAMStreamConfig** interface enables an application to set the output format on certain capture and compression filters, for both audio and video. In our case, it's a video capture filter.

Use this interface to set format properties, such as the output dimensions and frame rate (for video) or the sample rate and number of channels (for audio).

Filters expose this interface on their output pins. To use the interface, enumerate the filter's pins and query for **IAMStreamConfig**. Or, if you are using the Capture Graph Builder object to build the filter graph, you can call the **ICaptureGraphBuilder2::FindInterface** method.

In addition to the methods inherited from **IUnknown**, the **IAMStreamConfig** interface exposes the following methods.

Method	Description
GetFormat	Retrieves the current or preferred output format.
GetNumberOfCapabilities	Retrieves the number of format capabilities that this pin supports.
GetStreamCaps	Retrieves a set of format capabilities.
SetFormat	Sets the output format on the pin.

IAMVideoProcAmp Interface

The **IAMVideoProcAmp** interface enables an application to adjust the qualities of an incoming video signal, such as brightness, contrast, hue, saturation, gamma, and sharpness.

The Video Capture filter exposes this interface if the hardware supports image adjustment.

In addition to the methods inherited from **IUnknown**, the **IAMVideoProcAmp** interface exposes the following methods.

Method	Description
GetRange	Retrieves minimum, maximum, and default values for setting properties.
Set	Sets video quality for a specified property.
Get	Retrieves video quality for a specified property.

IAMVideoControl Interface

The **IAMVideoControl** interface enables you to flip a picture horizontally and/or vertically, set up a stream so it can capture from an external trigger (such as a camera button that the user pushes), simulate an external trigger in software, and list the available frame rates.

Use this interface when your application runs on hardware that is capable of flipping and external triggering. For Mightex Buffer CCD camera, it's capable to be set to Trigger mode, for details of Trigger mode, please refer to Mightex Buffer CCD Camera User manual.

In addition to the methods inherited from **IUnknown**, the **IAMVideoControl** interface exposes the following methods.

Method	Description
GetCaps	Retrieves the capabilities of the underlying hardware.
SetMode	Sets the video control mode of operation.
GetMode	Retrieves the video control mode of operation.
GetCurrentActualFrameRate	Retrieves the actual frame rate at which the device is streaming. This method is used with devices, such as the Universal Serial Bus (USB) or cameras that use the IEEE 1394 serial standard, where the maximum frame rate can be limited by bandwidth availability. This is only available during video streaming.
GetMaxAvailableFrameRate	Retrieves the maximum frame rate currently available based on bus bandwidth usage for connections such as USB and IEEE 1394 camera devices where the maximum frame rate can be limited by bandwidth availability.
GetFrameRateList	Retrieves a list of available frame rates.

Note that the last two methods are NOT implemented on Mightex filter.

IKsPropertySet Interface

The **IKsPropertySet** interface was originally designed as an efficient way to set and retrieve device properties on WDM drivers, using KSProxy to translate the user-mode COM method calls into the kernel-mode property sets used by WDM streaming class drivers. This interface is now also used to pass information strictly between software components.

In some cases, software components must implement either this interface, or else the IKsControl interface (documented in the DirectShow DDK). For example, if you are writing a software MPEG-2 decoder for use with the Microsoft® DVD Navigator, you must implement one of these interfaces and also support the DVD-related property sets that the Navigator will send to the decoder. Pins may support one of these interfaces to allow other pins or filters to set or retrieve their properties.

Note Another interface by this name exists in the dsound.h header file. The two interfaces are not compatible. The **IKsControl** interface, documented in the DirectShow DDK, is now the recommended interface for passing property sets between WDM drivers and user mode components.

Methods in Vtable Order

In addition to the methods inherited from **IUnknown**, the interface exposes the following methods.

Method	Description
Set	Sets a property identified by a property set **GUID** and a property ID.
Get	Retrieves a property identified by a property set **GUID** and a property ID.
QuerySupported	Determines whether an object supports a specified property set.

IAMCameraControl Interface

The **IAMCameraControl** interface enables you to flip a picture horizontally and/or vertically, set up a stream so it can capture from an external trigger (such as a camera button that the user pushes), simulate an external trigger in software, and list the available frame rates.

Use this interface when your application runs on hardware that is capable of flipping and external triggering. For Mightex New Classic camera, it's capable to be set to Trigger mode, for details of Trigger mode, please refer to Mightex Buffer CCD Camera User manual.

In addition to the methods inherited from **IUnknown**, the **IAMCameraControl** interface exposes the following methods.

Method	Description
Get	Retrieves the current setting of a camera property.
GetRange	retrieves the range and default value of a specified camera property.
Set	Sets a specified property on the camera.