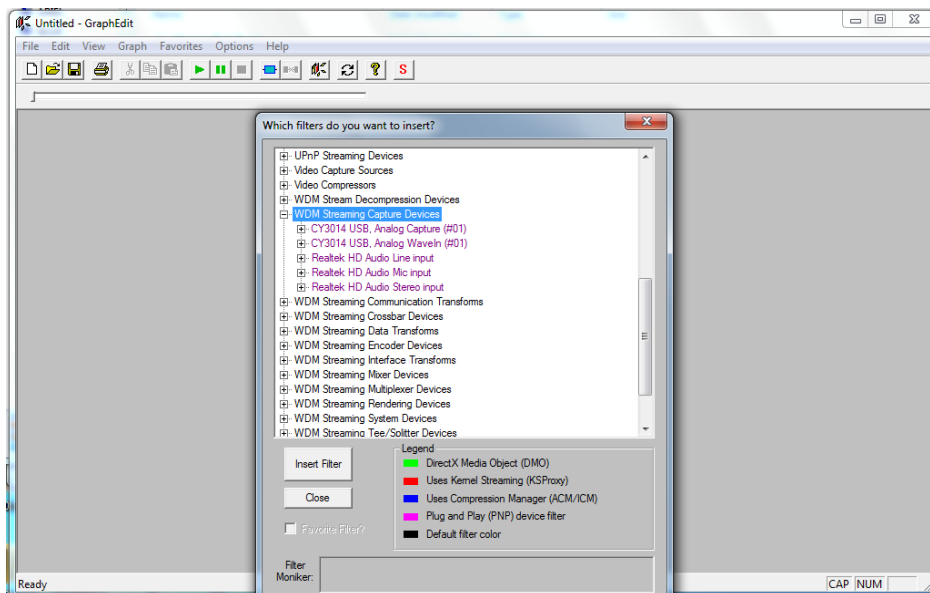


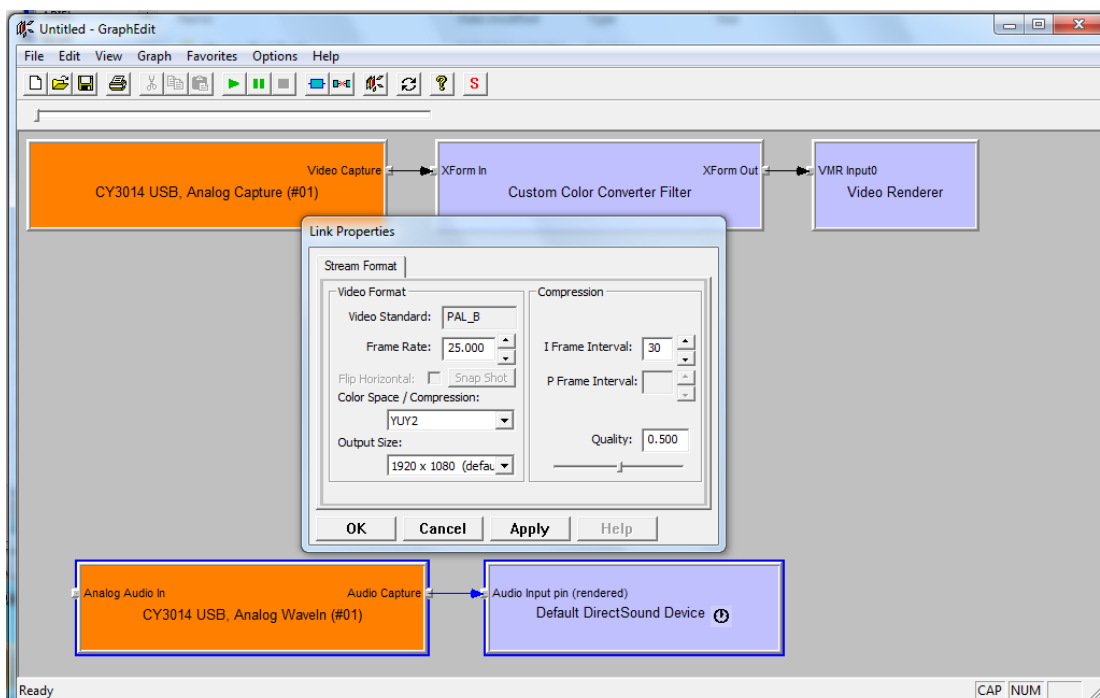
# SC530 DirectShow Software Programming Guide

Customer uses DirectShow to develop software can bypass our SDK to access CY3014 directly. Majority of device properties is implemented by Microsoft DirectShow standard interface. Software developer can refer Section 1 and Section 2 to control them. Other custom properties are implemented by IKsPropertySet interface. The interface can be queried from our capture source filter. Section 3 will describe how to access them in detail.

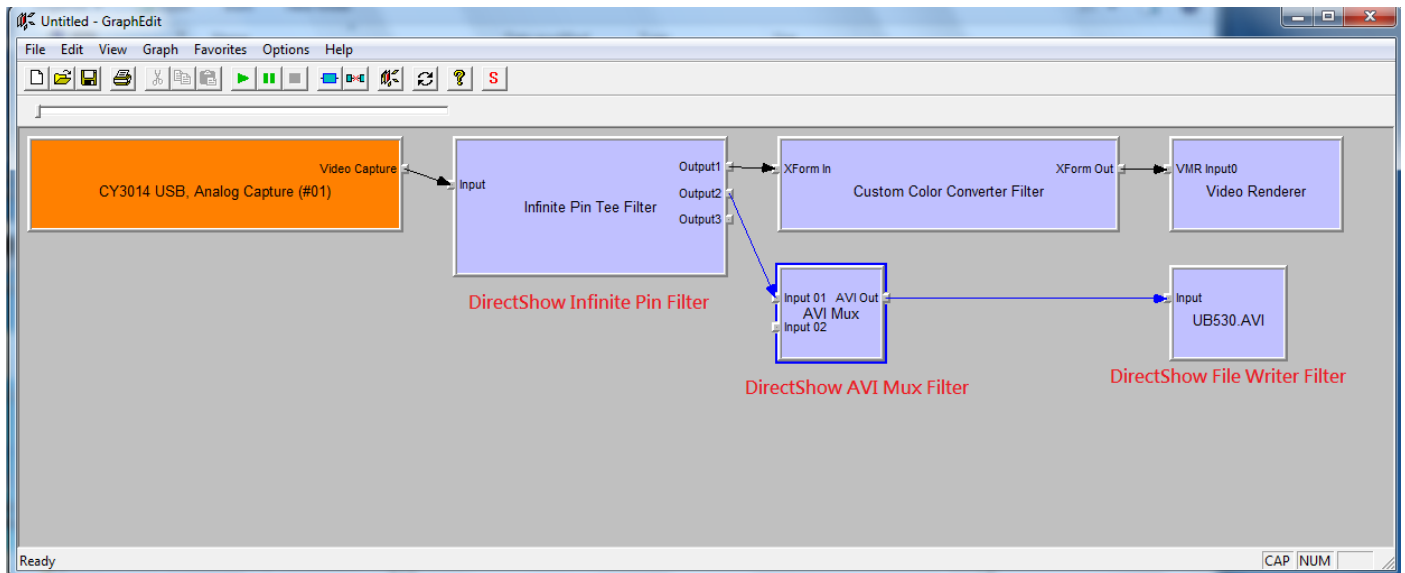
All filter names are "CY3014 USB, Analog Capture (#XX)" for video, and "CY3014 USB, Analog WaveIn (#XX)" for audio. They are registered at "WDM Streaming Captures Devices" category.



Here, the video format is YUY2 and audio format is PCM. The connection of filters is as:



Moreover, customer wants to use graphedit to save raw video stream into AVI can reference as below:



## 1. ACCESS VIDEO STANDARD (IAMAnalogVideoDecoder)

The video standard is implemented by IAMAnalogVideoDecoder interface. Customer must to setup the correct standard before accessing video format. For example, the 720X480@30fps format is only implemented under NTSC, and the 720x576@25fps format is only implemented under PALB.

EXAMPLE#01: SET STANDARD TO NTSC.

```
m_pCommonCaptureGraphBuilder2->FindInterface( NULL,
                                                NULL,
                                                m_pVideoCaptureSourceBaseFilter,
                                                IID_IAMAnalogVideoDecoder,
                                                (VOID **) (&m_pAMAnalogVideoDecoder) );
m_pAMAnalogVideoDecoder->put_TVFormat( AnalogVideo_NTSC_M );
```

## 2. ACCESS OUTPUT FORMAT OF CAPTURE PIN (IAMStreamConfig)

To get/set output format of capture pin, customer can use IAMStreamConfig interface.

EXAMPLE#01: SET VIDEO OUTPUT FORAMT TO 1920X1080 AT 30FPS.

```
m_pCommonCaptureGraphBuilder2->FindInterface( &LOOK_DOWNSTREAM_ONLY,
                                                NULL,
                                                m_pVideoCaptureSourceBaseFilter,
                                                IID_IAMStreamConfig,
                                                (VOID **)(&m_pAMStreamConfig) );

AM_MEDIA_TYPE * pmt = NULL;
m_pAMStreamConfig->GetFormat( &pmt );
((VIDEOINFOHEADER *) (pmt->pbFormat))->bmiHeader.biCompression = MAKEFOURCC('Y', 'U', 'Y', '2');
((VIDEOINFOHEADER *) (pmt->pbFormat))->bmiHeader.biHeight = 1920;
((VIDEOINFOHEADER *) (pmt->pbFormat))->bmiHeader.biWidth = 1080;
((VIDEOINFOHEADER *) (pmt->pbFormat))->bmiHeader.biBitCount = 16;
((VIDEOINFOHEADER *) (pmt->pbFormat))->bmiHeader.biSizeImage = 1920 * 1080 * 16 / 8;
((VIDEOINFOHEADER *) (pmt->pbFormat))->AvgTimePerFrame = (ULONG)(INT)(10000000.0 / 30.000);
((VIDEOINFOHEADER *) (pmt->pbFormat))->dwBitRate = (ULONG)(INT)(1920 * 1080 * 16 * 30.000);
m_pAMStreamConfig->SetFormat( pmt );
DeleteMediaType( pmt );
```

EXAMPLE#02: SET AUDIO OUTPUT FORAMT TO SETERO, 16BITS, AND 48000HZ.

```
m_pCommonCaptureGraphBuilder2->FindInterface( &LOOK_DOWNSTREAM_ONLY,
                                                NULL,
                                                m_pAudioCaptureSourceBaseFilter,
                                                IID_IAMStreamConfig,
                                                (VOID **)(&m_pAMStreamConfig) );

AM_MEDIA_TYPE * pmt = NULL;
m_pAMStreamConfig->GetFormat( &pmt );
((WAVEFORMATEX *) (pmt->pbFormat))->nChannels = (USHORT)(2);
((WAVEFORMATEX *) (pmt->pbFormat))->wBitsPerSample = (USHORT)(16);
((WAVEFORMATEX *) (pmt->pbFormat))->nSamplesPerSec = (ULONG)(48000);
((WAVEFORMATEX *) (pmt->pbFormat))->nBlockAlign = (USHORT)(2 * 16 / 8);
((WAVEFORMATEX *) (pmt->pbFormat))->nAvgBytesPerSec = (ULONG)(2 * 16 * 48000 / 8);
m_pAMStreamConfig->SetFormat( pmt );
DeleteMediaType( pmt );
```

### 3 Customer Property Access

Customer can access all custom properties by IKsPropertySet, the parameter rguidPropSet of IKsPropertySet::Set/Get function, is defined as below:

```
GUID PROPSETID_AMEBDAD_CUSTOM_PROP =  
{ 0xD1E5209F, 0x68FD, 0x4529, 0xBE, 0xE0, 0x5E, 0x7A, 0x1F, 0x47, 0x92, 0x1F };
```

All custom properties are defined as below:

```
typedef enum {  
    KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_INPUT                = 201,  
    KSPROPERTY_CUSTOM_GET_ANALOG_VIDEO_MACROVISION          = 202,  
    KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_FRAME_RATE           = 208,  
    KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_RESOLUTION           = 210,  
    KSPROPERTY_CUSTOM_XET_ANALOG_AUDIO_SAMPLE_FREQUENCY      = 253,  
    KSPROPERTY_CUSTOM_XET_ANALOG_AUDIO_INPUT                 = 255,  
} KSPROPERTY_AMEBDAD_CUSTOM;
```

### 3.1. KSPROPERTY\_CUSTOM\_XET\_ANALOG\_VIDEO\_INPUT (201)

The property allows you to get/change current video input source. We can support total 5 kinds of video input sources, HDMI, DVI-D, Components, DVI-A, and SDI.

SUPPORT VALUE: 0: HDMI  
1: DVI-Digital  
2: Components (YCbCr)  
3: DVI-Analog (RGB) (VGA)  
4: SDI

EXAMPLE#01: SET INPUT TO HDMI.

```
ULONG input = 0;
m_pKsPropertySet->Set( PROPSETID_AMEBDAD_CUSTOM_PROP,
                        KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_INPUT,
                        NULL, 0,
                        &input, sizeof(ULONG) );
```

EXAMPLE#02: CHANGE INPUT TO SDI.

```
ULONG input = 4;
m_pKsPropertySet->Set( PROPSETID_AMEBDAD_CUSTOM_PROP,
                        KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_INPUT,
                        NULL, 0,
                        &input, sizeof(ULONG) );
```

EXAMPLE#03: GET CURRENT VIDEO INPUT SOURCE.

```
ULONG input = 0;
m_pKsPropertySet->Get( PROPSETID_AMEBDAD_CUSTOM_PROP,
                        KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_INPUT,
                        NULL, 0,
                        &input, sizeof(ULONG), &temp );
```

### 3.2. KSPROPERTY\_CUSTOM\_XET\_ANALOG\_AUDIO\_INPUT (255)

The property allows you to get/change current audio input source. You can select audio from embedded audio data or from extra line-in cable.

SUPPORT VALUE: 0: Embedded Audio  
1: Line In

**Note!! The property is enabled only by HDMI, DVI-D, and SDI input mode.**

EXAMPLE#01: CHANGE TO EMBEDDED AUDIO INPUT.

```
ULONG input = 0;
m_pKsPropertySet->Set( PROPSETID_AMEBDAD_CUSTOM_PROP,
                        KSPROPERTY_CUSTOM_XET_ANALOG_AUDIO_INPUT,
                        NULL, 0,
                        &input, sizeof(ULONG) );
```

EXAMPLE#02: CHANGE TO LINE-IN INPUT.

```
ULONG input = 1;
m_pKsPropertySet->Set( PROPSETID_AMEBDAD_CUSTOM_PROP,
                        KSPROPERTY_CUSTOM_XET_ANALOG_AUDIO_INPUT,
                        NULL, 0,
                        &input, sizeof(ULONG) );
```

EXAMPLE#03: GET CURRENT AUDIO INPUT SOURCE.

```
ULONG input = 0;
m_pKsPropertySet->Get( PROPSETID_AMEBDAD_CUSTOM_PROP,
                       KSPROPERTY_CUSTOM_XET_ANALOG_AUDIO_INPUT,
                       NULL, 0,
                       &input, sizeof(ULONG), &temp );
```

### 3.3. KSPROPERTY\_CUSTOM\_GET\_ANALOG\_VIDEO\_RESOLUTION (210) (READ ONLY)

### 3.3. KSPROPERTY\_CUSTOM\_GET\_ANALOG\_VIDEO\_FRAME\_RATE (208) (READ ONLY)

Our driver can auto detect video format and can report the current input format to your software. The both properties can help to obtain current video format's resolution and frame rate. Some supported formats are described in the table. The format table keeps on increasing into the new driver. Please check our sales to obtain the latest one.

FORMAT	RESOLUTION	FRAME RATE
1920×1080p@60fps	0x07800438	60
1920×1080p@50fps	0x07800438	50
1920×1080p@30fps	0x07800438	30
1920×1080p@25fps	0x07800438	25
1920×1080p@24fps	0x07800438	24
1920×1080i@60fps	0x0780021C	60
1920×1080i@50fps	0x0780021C	50
1280×720P@60fps	0x050002D0	60
1280×720P@50fps	0x050002D0	50
1280×720P@30fps	0x050002D0	30
1280×720P@25fps	0x050002D0	25
1280×720P@24fps	0x050002D0	24
720×480P@60fps	0x02D001E0	60
720×576P@50fps	0x02D00240	50
720×480i@60fps	0x02D000F0	60
720×576i@50fps	0x02D00120	50
720×240P@60fps	0x05A001E0	60
720×288P@50fps	0x05A00240	50
1440×900p@60fps	0x05A00384	60
1280×1024p@60fps	0x05000400	60
1280×960p@60fps	0x050003C0	60
1280×800p@60fps	0x05000320	60
1280×768p@60fps	0x05000300	60
1024×768p@60fps	0x04000300	60
800×600p@60fps	0x03200258	60
640×480p@60fps	0x028001E0	60
640×400p@60fps	0x02800190	60
640×384p@60fps	0x02800180	60

\*<sub>1</sub> THE FORMAT IS USED BY SONY PS1/PS2 GAME MACHINE.

\*<sub>2</sub> THE FORMAT IS USED BY MICROSOFT XBOX360 GAME MACHINE (640×480p@60fps).

\*<sub>3</sub> THE FORMAT IS USED BY NEC IPC MACHINE (640×400p@56.4fps).



Here, the resolution property can be described as below:

RESOLUTION = (WIDTH << 16) | (HEIGHT << 0)

**Note!! Developer should design one polling operation in one background thread to obtain/update current input format.**

EXAMPLE#01: GET CURRENT VIDEO FORMAT.

```
ULONG resolution = 0;
```

```
ULONG framerate = 0;
```

```
m_pKsPropertySet->Get( PROPSETID_AMEBDAD_CUSTOM_PROP,  
                        KSPROPERTY_CUSTOM_GET_ANALOG_VIDEO_RESOLUTION,  
                        NULL, 0,  
                        &resolution, sizeof(ULONG), &temp );  
  
m_pKsPropertySet->Get( PROPSETID_AMEBDAD_CUSTOM_PROP,  
                        KSPROPERTY_CUSTOM_GET_ANALOG_VIDEO_FRAME_RATE,  
                        NULL, 0,  
                        &framerate, sizeof(ULONG), &temp );
```

### 3.4. KSPROPERTY\_CUSTOM\_XET\_ANALOG\_AUDIO\_SAMPLE\_FREQUENCY (253) (READ ONLY)

The driver also can auto detect current audio format and can report it to upper software. Currently, all audio formats are stereo and 16bits quality. The only difference is their sample frequency, so you can use the property to obtain the input's sample frequency.

**Note!! Developer should design one polling operation in one background thread to obtain/update current input format.**

SUPPORT VALUE: 48000 - STEREO / 16BITS / 48000HZ  
44100 - STEREO / 16BITS / 44100HZ  
32000 - STEREO / 16BITS / 32000HZ

EXAMPLE#01: GET CURRENT AUDIO SAMPLE FREQUENCY.

```
ULONG frequency = 0;
m_pKsPropertySet->Get( PROPSETID_AMEBDAD_CUSTOM_PROP,
                        KSPROPERTY_CUSTOM_XET_ANALOG_AUDIO_SAMPLE_FREQUENCY,
                        NULL, 0,
                        &frequency, sizeof(ULONG), &temp );
```

### 3.5. KSPROPERTY\_CUSTOM\_GET\_ANALOG\_VIDEO\_MACROVISION (202) (READ ONLY)

The property allows you to detect if the input's media content owns HDCP or MarcoVision protection.

**Note!! To protect the content license, all behaviors in software porting should be complied with HDCP rules. Detect in any registered content of HDCP or MarcoVision, please disable the recording function in software.**

SUPPORT VALUE: 0, 1 - NO ~ YES

EXAMPLE#01: GET HDCP PROTECT STATUS.

```
ULONG HDCP = 0;
```

[illegible]

#### **4. Application Note for DirectShow Developer**

The developer who uses DirectShow to access our capture source filter need check the frame size in the callback function of your SampleGrabber class. If the frame size is 0 bytes, it means the frame is one bad frame. You should drop it. More detail, please check with our engineer team directly.