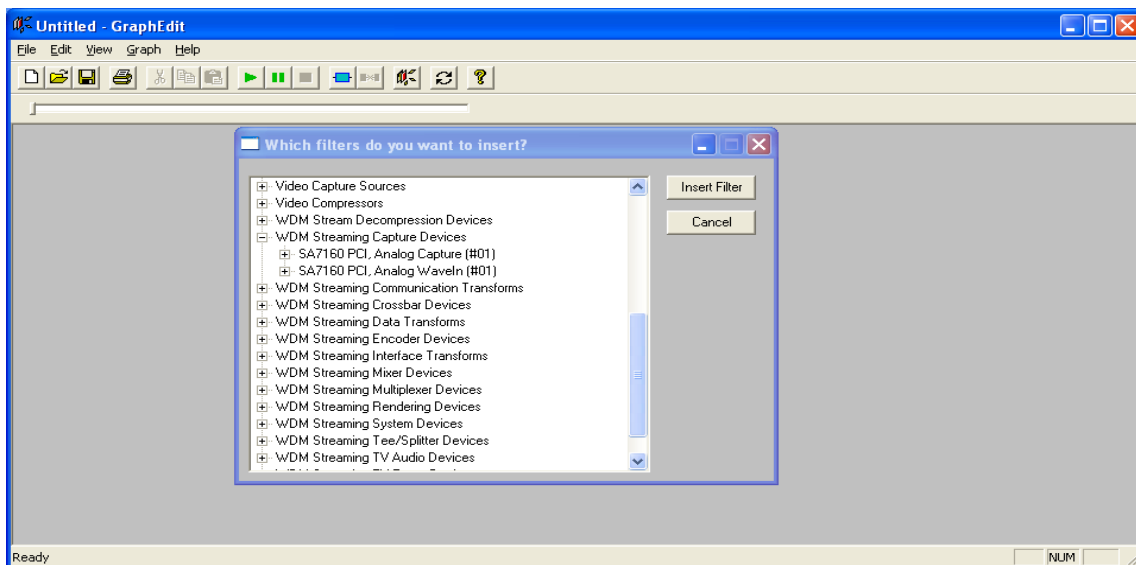


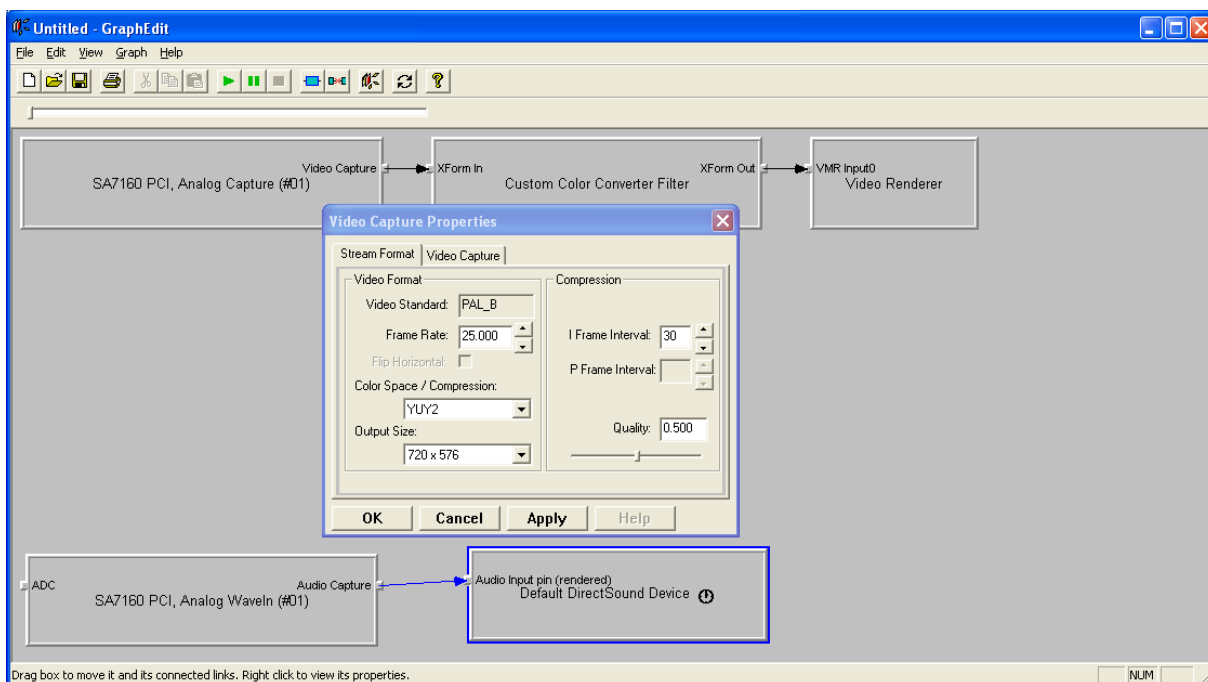
# SC500 & SC510 DirectShow Software Programming Guide

Customer uses DirectShow to develop software can bypass our SDK to access SAA7160 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.

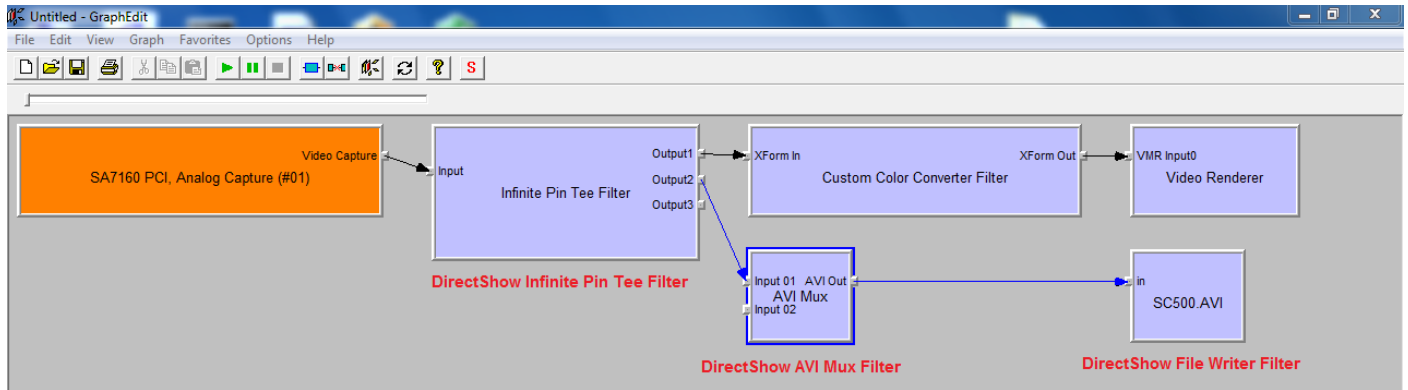
All filter names are "SA7160 PCI, Analog Capture (#XX)" for video, and "SA7160 PCI, Analog WaveIn (#XX)" for audio. They are registered at "WDM Streaming Captures Devices" category.



Here, the video format is YUY2 and audio format is PCM. The connection of filters are 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, 0x1C };
```

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_VIDEO_FLEXIBLE_FPS_PATCH = 218,  
    KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_FLEXIBLE_RESOLUTION_PATCH = 220,  
    KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_VGA_YCBCR_AUTO_PHASE = 219,  
    KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_VGA_YCBCR_OFFSET_X = 221,  
    KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_VGA_YCBCR_OFFSET_Y = 222,  
    KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_COLOR_RANGE = 231,  
    KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_INPUT_EQ = 240,  
    KSPROPERTY_CUSTOM_XET_ANALOG_AUDIO_SAMPLE_FREQUENCY = 253,  
    KSPROPERTY_CUSTOM_XET_ANALOG_AUDIO_INPUT = 255,  
    KSPROPERTY_CUSTOM_XET_GPIO_DIRECTION = 940,  
    KSPROPERTY_CUSTOM_XET_GPIO_DATA = 941,  
    KSPROPERTY_CUSTOM_XET_GPIO_SUPPORT = 942,  
} 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  
5: Composite  
6: S-Video

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	* <sub>1</sub>
1920×1080p@50fps	0x07800438	50	* <sub>1</sub>
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	* <sub>2</sub>
720×288P@50fps	0x05A00240	50	* <sub>2</sub>
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	* <sub>3</sub>
640×400p@60fps	0x02800190	60	* <sub>4</sub>
640×384p@60fps	0x02800180	60	* <sub>4</sub>

\*<sub>1</sub> THE FORMAT IS USED BY SC510 SERIES.

\*<sub>2</sub> THE FORMAT IS USED BY SONY PS1/PS2 GAME MACHINE.

\*<sub>3</sub> THE FORMAT IS USED BY MICROSOFT XBOX360 GAME MACHINE (640×480p@60fps).

\*<sub>4</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]

### 3.6. KSPROPERTY\_CUSTOM\_XET\_ANALOG\_VIDEO\_FLEXIBLE\_FPS\_PATCH (218)

### 3.6. KSPROPERTY\_CUSTOM\_XET\_ANALOG\_VIDEO\_FLEXIBLE\_RESOLUTION\_PATCH (220)

The two properties to allow you to control the output format from one video capture filter.

The property, 218, allows you to adjust the video's frame rate from driver side. If it is disabled, the output frame rate is equal to input signal's frame rate.

SUPPORT VALUE: 0 ~ 1 - DISABLE ~ ENABLE

The property, 220, allows you to adjust the video's resolution from hardware board. If it is disabled, the output resolution is equal to input signal's resolution. If it is enabled, we will enable one auto scalar to output user's customize format. For example, input resolution is 1920x1080 and capture output pin's resolution is 720x480.

Note, to enable it, you need reboot the system.

SUPPORT VALUE: 0 ~ 1 - DISABLE ~ ENABLE

EXAMPLE#01: TO ENABLE IMAGE SCALER.

```
ULONG mode = 1;
```

```
m_pKsPropertySet->Set(  PROPSETID_AMEBDAD_CUSTOM_PROP,  
                          KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_FLEXIBLE_RESOLUTION_PATCH,  
                          NULL, 0,  
                          &mode, sizeof(ULONG) );
```

3.7. KSPROPERTY\_CUSTOM\_XET\_ANALOG\_VIDEO\_VGA\_YCBCR\_AUTO\_PHASE (219)

3.7. KSPROPERTY\_CUSTOM\_XET\_ANALOG\_VIDEO\_VGA\_YCBCR\_OFFSET\_X (221)

3.7. KSPROPERTY\_CUSTOM\_XET\_ANALOG\_VIDEO\_VGA\_YCBCR\_OFFSET\_Y (222)

If input is in VGA or YCbCr, these properties allow you to adjust the hardware receiver's property.

The auto phase property can be set as below:

SUPPORT VALUE: 0 ~ 63 - MANUAL PHASE DEGREE

SUPPORT VALUE: 0x80000000 - AUTO PHASE

The offset property allows you to adjust the horizontal and vertical offset for signal. Moreover, our driver will do auto memorize for setting value in next detection.

SUPPORT VALUE: -127 ~ +128

EXAMPLE#01: TO SET HORIZONTAL OFFSET FOR VGA.

```
LONG offset = -8;
```

```
m_pKsPropertySet->Set( PROPSETID_AMEBDAD_CUSTOM_PROP,  
                        KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_OFFSET_X, NULL, 0,  
                        &offset, sizeof(LONG) );
```

### 3.8. KSPROPERTY\_CUSTOM\_XET\_ANALOG\_VIDEO\_COLOR\_RANGE (231)

The property allows you to control each input (HDMI, COMPONENT, VGA) to different scale rang. You should choose proper mode because then you just can achieve the most accurate color.

We can use a 32-bit number (4 byte) as input value:

A 2-bit **operation code** can be set as below to specify the conversion operation:

- 0: Keep the color range unchanged. (Default)
- 1: Shrink the input from full range to limited range. (16-235 level)
- 2: Expand the input from limited range to full range. (0-255 level)

Other bit fields are used to represent as below:

- [1:0] Operation code for HDMI input when register reveals 0 "Default (depend on video format)"
- [5:4] Operation code for HDMI input when register reveals 1 "Limited range"
- [9:8] Operation code for HDMI input when register reveals 2 "Full range"
- [13:12] Operation code for Component input
- [17:16] Operation code for VGA input

NOTE: Normally it is recommended to set operation code to default. If the displayed black or white color in the video input is not enough true. You can use the mode adjustment to change the color quality for video input.

EXAMPLE#01: TO CHANGE HDMI INPUT LIMITED RANGE TO FULL RANGE

```
LONG input = 0x00020;  
m_pKsPropertySet->Set( PROPSETID_AMEBDAD_CUSTOM_PROP,  
                        KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_COLOR_RANGE, NULL, 0,  
                        &input, sizeof(LONG) );
```

EXAMPLE#02: TO CHANGE HDMI INPUT FULL RANGE TO LIMITED RANGE

```
LONG input = 0x00100;  
m_pKsPropertySet->Set( PROPSETID_AMEBDAD_CUSTOM_PROP,  
                        KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_COLOR_RANGE, NULL, 0,  
                        &input, sizeof(LONG) );
```

EXAMPLE#03: TO CHANGE ALL INPUT TO LIMITED RANGE

```
LONG input = 0x11100;
```

```
m_pKsPropertySet->Set( PROPSETID_AMEBDAD_CUSTOM_PROP,  
                        KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_COLOR_RANGE, NULL, 0,  
                        &input, sizeof(LONG) );
```

EXAMPLE#04: TO EXPAND ALL INPUT COOR RANGE NO MATTER WHAT

```
LONG input = 0x22222;
```

```
m_pKsPropertySet->Set( PROPSETID_AMEBDAD_CUSTOM_PROP,  
                        KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_COLOR_RANGE, NULL, 0,  
                        &input, sizeof(LONG) );
```

### 3.9. KSPROPERTY\_CUSTOM\_XET\_ANALOG\_VIDEO\_INPUT\_EQ (240)

The property allows you to set a suitable distance in meter when using the DVI and HDMI signal. Basically, the quality of signal can vary widely based on the cable's materials, but here can adjust the settings through the property.

SUPPORT VALUE: 0 ~ 2 - **2m, 10m, 10~15m (METER)**

EXAMPLE#01: TO SET THE CABLE LENGTH IN 2 METER

```
LONG input = 0x00;
```

```
m_pKsPropertySet->Set( PROPSETID_AMEBDAD_CUSTOM_PROP,  
                        KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_INPUT_EQ, NULL, 0,  
                        &input, sizeof(LONG) );
```

EXAMPLE#02: TO SET THE CABLE LENGTH IN 10 METER

```
LONG input = 0x01;
```

```
m_pKsPropertySet->Set( PROPSETID_AMEBDAD_CUSTOM_PROP,  
                        KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_INPUT_EQ, NULL, 0,  
                        &input, sizeof(LONG) );
```

EXAMPLE#03: TO SET THE CABLE LENGTH IN 10~15 METER

```
LONG input = 0x02;
```

```
m_pKsPropertySet->Set( PROPSETID_AMEBDAD_CUSTOM_PROP,  
                        KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_INPUT_EQ, NULL, 0,  
                        &input, sizeof(LONG) );
```



### 3.10. KSPROPERTY\_CUSTOM\_XET\_GPIO\_DIRECTION (940)

### 3.10. KSPROPERTY\_CUSTOM\_XET\_GPIO\_DATA (941)

### 3.10. KSPROPERTY\_CUSTOM\_GET\_GPIO\_SUPPORT (942) (READ ONLY)

The property allows you to access SA7160's GPIO interface. The property KSPROPERTY\_CUSTOM\_XET\_GPIO\_DIRECTION allows you to control its direction. Here, writing 1 to bit enables this pin as output pin. Usually, the GPIO is controlled by the first chipset in one board.

SUPPORT VALUE: 0 ~ 1 - INPUT ~ OUTPUT

The property KSPROPERTY\_CUSTOM\_XET\_GPIO\_DATA allows you to access GPIO's data.

SUPPORT VALUE: 0 ~ 1 - LOW ~ HIGH

The property KSPROPERTY\_CUSTOM\_XET\_GPIO\_SUPPORT allows you to obtain GPIO's information (pin size) on hardware board. Developer can use it to check if the device can support GPIO access.

SUPPORT VALUE: 0 IS NON-SUPPORT

EXAMPLE#01: TO DEFINE GPIO AS 8 OUTPUT PINS [0:7] AND 8 INPUT PINS [8:15].

```
ULONG input = 0x00FF;
m_pKsPropertySet->Set( PROPSETID_AMEBDAD_CUSTOM_PROP,
                        KSPROPERTY_CUSTOM_XET_GPIO_DIRECTION, NULL, 0,
                        &input, sizeof(ULONG) );
```

EXAMPLE#02: TO DEFINE GPIO AS 16 OUTPUT PINS [0:15] THEN PULL HIGH FOR ALL.

```
ULONG input = 0xFFFF;
ULONG data = 0xFFFF;
m_pKsPropertySet->Set( PROPSETID_AMEBDAD_CUSTOM_PROP,
                        KSPROPERTY_CUSTOM_XET_GPIO_DIRECTION, NULL, 0,
                        &input, sizeof(ULONG) );
m_pKsPropertySet->Set( PROPSETID_AMEBDAD_CUSTOM_PROP,
                        KSPROPERTY_CUSTOM_XET_GPIO_DATA, NULL, 0,
                        &data, sizeof(ULONG) );
```



#### **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.