

# SC590 Extra Software Programming Guide

## Custom Property

- 1. **KSPROPERTY\_CUSTOM\_XET\_ANALOG\_VIDEO\_INPUT\_AUTO\_SCAN** (232)
- 1. **KSPROPERTY\_CUSTOM\_GET\_ANALOG\_VIDEO\_RESOLUTION** (210) (READ ONLY)
- 1. **KSPROPERTY\_CUSTOM\_GET\_ANALOG\_VIDEO\_INTERLEAVED** (223) (READ ONLY)
- 1. **KSPROPERTY\_CUSTOM\_GET\_ANALOG\_VIDEO\_FRAME\_RATE** (208) (READ ONLY)
- 1. **KSPROPERTY\_CUSTOM\_GET\_ANALOG\_VIDEO\_FRACTION\_1000\_1001** (241)

The **KSPROPERTY\_CUSTOM\_XET\_ANALOG\_VIDEO\_INPUT\_AUTO\_SCAN** allows you to enable or disable the automatic scan video input signal source. If this function detects the actual video input source and format on capture card, it will automatically set the correct video input source and format.

SUPPORT VALUE: 0 ~ 1 - DISABLE ~ ENABLE

EXAMPLE#01: ENABLE THE AUTO INPUT SCAN FUNCTION

```
LONG enable = 0x01;  
AMESDK_SET_CUSTOM_PROPERTY( hDev, 232, enable );
```

EXAMPLE#02: DISENABLE THE AUTO INPUT SCAN FUNCTION

```
LONG disable = 0x00;  
AMESDK_SET_CUSTOM_PROPERTY( hDev, 232, disable );
```

These properties **KSPROPERTY\_CUSTOM\_GET\_ANALOG\_VIDEO\_RESOLUTION** / **KSPROPERTY\_CUSTOM\_GET\_ANALOG\_VIDEO\_INTERLEAVED** / **KSPROPERTY\_CUSTOM\_GET\_ANALOG\_VIDEO\_FRAME\_RATE** / **KSPROPERTY\_CUSTOM\_GET\_ANALOG\_VIDEO\_FRACTION\_1000\_1001** 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 / 59.94
1920×1080p@50fps	0x07800438	50 / 49.95
1920×1080p@30fps	0x07800438	30 / 29.97
1920×1080p@25fps	0x07800438	25 / 24.97
1920×1080p@24fps	0x07800438	24 / 23.97

1920×1080i@60fps	0x0780021C	60 / 59.94	
1920×1080i@50fps	0x0780021C	50 / 49.95	
1280×720P@60fps	0x050002D0	60 / 59.94	
1280×720P@50fps	0x050002D0	50 / 49.95	
1280×720P@30fps	0x050002D0	30 / 29.97	
1280×720P@25fps	0x050002D0	25 / 24.97	
1280×720P@24fps	0x050002D0	24 / 23.97	
720×480P@60fps	0x02D001E0	60 / 59.94	
720×576P@50fps	0x02D00240	50 / 49.95	
720×480i@60fps	0x02D000F0	60 / 59.94	
720×576i@50fps	0x02D00120	50 / 49.95	
720×240P@60fps	0x05A001E0	60 / 59.94	* <sub>1</sub>
720×288P@50fps	0x05A00240	50 / 49.95	* <sub>1</sub>
1440×900p@60fps	0x05A00384	60 / 59.94	
1280×1024p@60fps	0x05000400	60 / 59.94	
1280×960p@60fps	0x050003C0	60 / 59.94	
1280×800p@60fps	0x05000320	60 / 59.94	
1280×768p@60fps	0x05000300	60 / 59.94	
1024×768p@60fps	0x04000300	60 / 59.94	
800×600p@60fps	0x03200258	60 / 59.94	
640×480p@60fps	0x028001E0	60 / 59.94	* <sub>2</sub>
640×400p@60fps	0x02800190	60 / 59.94	* <sub>3</sub>
640×384p@60fps	0x02800180	60 / 59.94	* <sub>3</sub>

\*<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).

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

The resolution property **KSPROPERTY\_CUSTOM\_GET\_ANALOG\_VIDEO\_RESOLUTION:**

SUPPORT VALUE: RESOLUTION = (WIDTH << 16) | (HEIGHT << 0)

The interleaved property **KSPROPERTY\_CUSTOM\_GET\_ANALOG\_VIDEO\_INTERLEAVED:**

SUPPORT VALUE: 0: PROGRESSIVE

1: INTERLACED

The frame rate property **KSPROPERTY\_CUSTOM\_GET\_ANALOG\_VIDEO\_FRAME\_RATE:**

SUPPORT VALUE: 24 / 25 / 30 / 50 / 60 FPS

EXAMPLE#03: GET CURRENT VIDEO FORMAT.

```
AMESDK_GET_CUSTOM_PROPERTY( hDev, 210, &RESOLUTION );  
AMESDK_GET_CUSTOM_PROPERTY( hDev, 223, &INTERLACED );  
AMESDK_GET_CUSTOM_PROPERTY( hDev, 208, &FRAMERATE );
```

To obtain a more precise frame rate, combined with fraction property.  
The **KSPROPERTY\_CUSTOM\_GET\_ANALOG\_VIDEO\_FRACTION\_1000\_1001**:

SUPPORT VALUE: 23.97 / 24.97 / 29.97 / 49.95 / 59.94 FPS

$$\begin{aligned} 23.97 &= 24 * (1000/1001) \\ 24.97 &= 25 * (1000/1001) \\ 29.97 &= 30 * (1000/1001) \\ 49.95 &= 50 * (1000/1001) \\ 59.94 &= 60 * (1000/1001) \end{aligned}$$

EXAMPLE#04: TO GET MORE ACCURATE VIDEO FRAME RATE.

```
DWORD dw_framerate_fraction_property = 0;  
double d_video_framerate_property = 0.0;
```

```
AMESDK_GET_CUSTOM_PROPERTY( hDev, 208, &FRAMERATE );  
AMESDK_GET_CUSTOM_PROPERTY( hDev, 241, &dw_framerate_fraction_property );
```

```
d_video_framerate_property = FRAMERATE;
```

```
if ( dw_framerate_fraction_property == 1 )  
{  
    d_video_framerate_property *= 1000;  
    d_video_framerate_property /= 1001;  
}
```

## 2. **KSPROPERTY\_CUSTOM\_GET\_DEVICE\_AUDIO\_CONFIG** (9)

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

The property **KSPROPERTY\_CUSTOM\_GET\_DEVICE\_AUDIO\_CONFIG** allows you to get an OR combination of flag bits. This value shows what types of audio sources you can set are supplied on one capture card.

EXAMPLE#01: TO GET THE SUPPORT INPUTS OF THE AUDIO SOURCE ON ONE CAPTURE CARD.  
`ULONG nInput = 0xFFFFFFFF;`  
`AMESDK_GET_CUSTOM_PROPERTY( hDevice, 9, nInput);`

The property **KSPROPERTY\_CUSTOM\_XET\_ANALOG\_AUDIO\_INPUT** 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#02: CHANGE TO EMBEDDED AUDIO INPUT.  
`AMESDK_SET_CUSTOM_PROPERTY( hDev, 255, 0 );`

EXAMPLE#03: CHANGE TO LINE-IN INPUT.  
`AMESDK_SET_CUSTOM_PROPERTY( hDev, 255, 1 );`

EXAMPLE#04: GET CURRENT AUDIO INPUT SOURCE.  
`AMESDK_GET_CUSTOM_PROPERTY( hDev, 255, &INPUT );`

### 3. KSPROPERTY\_CUSTOM\_GET\_ANALOG\_VIDEO\_SIGNAL\_LOCK\_STATUS (230) (READ ONLY)

The property is used to determine whether the signal is locked.

SUPPORT VALUE: 0 ~ 1 - UNLOCK ~ LOCK

EXAMPLE#01: TO GET THE CURRENT SIGNAL STATUS.

```
LONG nLock = 0x00;
```

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 230, &nLock );
```

## 5. KSPROPERTY\_CUSTOM\_XET\_ANALOG\_VIDEO\_SOG (234)

If your input supports SOG (Sync On Green), you can use the property to enable or disable it.

SUPPORT VALUE: 0 ~ 1 - DISABLE ~ ENABLE

EXAMPLE#01: TO ENABLE SYNC ON GREEN

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

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 234, input );
```

**6. KSPROPERTY\_CUSTOM\_XET\_ANALOG\_VIDEO\_VERTICAL\_MIRROR (244)**

**6. KSPROPERTY\_CUSTOM\_XET\_ANALOG\_VIDEO\_HORIZONTAL\_MIRROR (245)**

This property is used to set mirror function. When mirror function is enabled, the vertical or horizontal video frame is inverted on display window. Same as deinterlacing, the property is used for display engine only.

SUPPORT VALUE: 0 ~ 1 - DISABLE ~ ENABLE

EXAMPLE#01: ENABLE THE VERTICAL MIRROR FUNCTION ON DISPLAY WINDOW

```
LONG enable = 0x01;
```

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 244, enable);
```

EXAMPLE#02: ENABLE THE HORIZONTAL MIRROR FUNCTION ON DISPLAY WINDOW

```
LONG enable = 0x01;
```

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 245, enable);
```

## 8. KSPROPERTY\_CUSTOM\_XET\_ANALOG\_AUDIO\_VOLUME (251)

The property is used to control the current audio ADC's volume on the capture card.

SUPPORT VALUE: 0 (Mute): ~ 255 (Full)

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

EXAMPLE#01: TO SET THE AUDIO VOLUME AMPLITUDE.

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 251, 128 );
```

EXAMPLE#02: TO GET THE AUDIO VOLUME AMPLITUDE.

```
AMESDK_GET_CUSTOM_PROPERTY( hDev, 251, &VOLUME );
```



## 9. 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.

```
AMESDK_GET_CUSTOM_PROPERTY( hDev, 202, &HDCP );  
IF( HDCP == 1 ) { RECORD_FUNCTION = DISABLE; }  
IF( HDCP == 0 ) { RECORD_FUNCTION = ENABLE; }
```

### 3. Dynamically Change of Preview Resolution

Our driver can allow you to dynamically change preview resolution size. Because SC590N4 is designed by PCI-Ex1 bandwidth, the max preview configuration can output 1CH 1920×1080@30fps and 3CHs x 960x540@30fps. The helper property can help you obtain one good quality preview video on full-screen mode of your software:

EXAMPLE#01: TO SET VIDEO RESOLUTION FORM 960X540 TO 1920X1080.

```
hDev = AMESDK_CREATE( hDev, ... );
AMESDK_SET_FORMAT( hDev, MAKEFOURCC('Y', 'U', 'Y', '2'), 960, 540, 29.970 );
AMESDK_RUN( hDev );
...
AMESDK_STOP( hDev );
AMESDK_SET_FORMAT( hDev, MAKEFOURCC('Y', 'U', 'Y', '2'), 1920, 1080, 29.970 );
AMESDK_RUN( hDev );
```

#### 4. `KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_QUEUE_BUFFER_SIZE` (216)

The property allow you to specify the number of the rendered video frame in the queue buffer for a preview or hardware-encoded (main, sub) stream. By the default, the queue size of the corresponding a preview and hardware-encoded stream is set 10 and 16. Here we recommended use the size by default because this is implicated in many resource issues. For example, the unexpected signal error may occur if the total buffer sizes you want to set exceed the system capabilities.

Note: Setting queue buffer size will involve in dynamically allocated memory.

EXAMPLE#01: TO SET THE PREVIEW QUEUE SIZE TO 10 FRAMES

```
LONG nBfferSize = 10;  
AMESDK_SET_CUSTOM_PROPERTY( hPreviewDevice, 216, nBfferSize );
```

EXAMPLE#02: TO SET THE HARDWARE-ENCODED QUEUE(MAIN) SIZE TO 16 FRAMES

```
LONG nBfferSize = 16;  
AMESDK_SET_CUSTOM_PROPERTY( hMainDevice, 216, nBfferSize );
```

EXAMPLE#03: TO SET THE HARDWARE-ENCODED QUEUE(SUB) SIZE TO 16 FRAMES

```
LONG nBfferSize = 16;  
AMESDK_SET_CUSTOM_PROPERTY( hSubDevice, 216, nBfferSize );
```

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

## 5. 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.

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 220, 1 );
```

## 6. KSPROPERTY\_CUSTOM\_XET\_ANALOG\_VIDEO\_INPUT\_BANDWIDTH (248)

The property allows you to get/set current video input bandwidth for the HDMI or DVI input. We can support total 6 kinds of video input bandwidth, 50%, 75%, 100%, 125%, 150%, and 200%. By default, the bandwidth is 75%.

SUPPORT HDMI/DVI BANDWIDTH: 0: 50%  
1: 75%  
2: 100%  
3: 125%  
4: 150%  
5: 200%

EXAMPLE#01: SET THE INPUT BANDWIDTH PERCENTAGE TO 50 PERCENT.

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 248, 0 );
```

EXAMPLE#02: SET THE INPUT BANDWIDTH PERCENTAGE TO 100 PERCENT.

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 248, 2 );
```

EXAMPLE#03: SET THE INPUT BANDWIDTH PERCENTAGE TO 200 PERCENT.

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 248, 5 );
```

## 7. KSPROPERTY\_CUSTOM\_GET\_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.

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

EXAMPLE#01: GET CURRENT AUDIO SAMPLE FREQUENCY.

```
AMESDK_GET_CUSTOM_PROPERTY( hDev, 253, &FREQUENCY );
```

1. **KSPROPERTY\_CUSTOM\_XET\_GPIO\_DIRECTION (940)**
1. **KSPROPERTY\_CUSTOM\_XET\_GPIO\_DATA (941)**
1. **KSPROPERTY\_CUSTOM\_XET\_GPIO\_SUPPORT (942) (READ ONLY)**

The property allows you to access TW2809'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-SUPPOR

EXAMPLE#01: TO DEFINE GPIO AS 8 OUTPUT PINS [0:7] AND 8 INPUT PINS [8:15].  
`AMESDK_SET_CUSTOM_PROPERTY( hDev, 940, 0x00FF );`

EXAMPLE#02: TO DEFINE GPIO AS 16 OUTPUT PINS [0:15] THEN PULL HIGH FOR ALL.  
`AMESDK_SET_CUSTOM_PROPERTY( hDev, 940, 0xFFFF );`  
`AMESDK_SET_CUSTOM_PROPERTY( hDev, 941, 0xFFFF );`

EXAMPLE#03: TO DEFINE GPIO AS 16 INPUT PINS [0:15] THEN READ DATA FROM IT.  
`AMESDK_SET_CUSTOM_PROPERTY( hDev, 940, 0x0000 );`  
`AMESDK_GET_CUSTOM_PROPERTY( hDev, 941, &GPIO );`

## **2. KSPROPERTY\_CUSTOM\_SET\_OSD\_LINE (920) (WRITE ONLY)**

## **2. KSPROPERTY\_CUSTOM\_SET\_OSD\_TEXT\_STRING (921) (WRITE ONLY)**

The properties allow you to change TW2809's OSD context. The properties \*SET\_OSD\_LINE and \*SET\_OSD\_TEXT\_STRING both help you to change string context. Note!! When you set the custom string into device, our driver will auto disable default time OSD. The max string length is 16 characters.

SUPPORT VALUE: 0 ~ 2 - LINE#0 ~ LINE#2

EXAMPLE#01: TO CHANGE LINE#0'S STRING.

```
CHAR string[] = "1234567890ABCDEF";  
AMESDK_SET_CUSTOM_PROPERTY( hDev, 920, 0x00000000 );  
AMESDK_SET_CUSTOM_PROPERTY_EX( hDev, 921, (BYTE *) (string), strlen(string) + 1 );
```

EXAMPLE#02: TO CHANGE LINE#1'S STRING.

```
CHAR string[] = "CH00";  
AMESDK_SET_CUSTOM_PROPERTY( hDev, 920, 0x00000001 );  
AMESDK_SET_CUSTOM_PROPERTY_EX( hDev, 921, (BYTE *) (string), strlen(string) + 1 );
```



## 8. Access Encoder Property

Developer can use the AMESDK\_G/SET\_VIDEOCOMPRESSION\_PROPERTY function to access all TW2809's video encoder properties. These properties as describe as the table below:

PROPERTY	RANGE
VideoCompression_KeyFrameRate	0 ~ 255
VideoCompression_OverrideKeyFrame	1 (WRITE ONLY)
VideoCompression_Quality	0 ~ 10,000
VideoCompression_BitRateMode	0, 1
VideoCompression_BitRate	128,000 ~ 12,000,000
VideoCompression_PostResolution	(cx << 12) + (cy << 0)
VideoCompression_PostSkipFrameRate	0 ~ 255
VideoCompression_PostAvgFrameRate	0 ~ 60

## 9. Access Custom Property for DirectShow Developer

Customer uses DirectShow to develop software can bypass our SDK to access TW2809 directly. The interface can be queried from our capture source filter.

At Section 11.1, 11.2, 11.3, 11.4 and 11.5, you can use IKsPropertySet to access all.

### 9.1 Device Serial Number Property:

```
#define KSPROPERTY_CUSTOM_GET_DEVICE_SERIAL_NUMBER 0 (READ ONLY) (ULONG)
```

### 9.2 GPIO Property:

```
#define KSPROPERTY_CUSTOM_XET_GPIO_DIRECTION 940 (ULONG)
```

```
#define KSPROPERTY_CUSTOM_XET_GPIO_DATA 941 (ULONG)
```

```
#define KSPROPERTY_CUSTOM_XET_GPIO_SUPPORT 942 (ULONG)
```

### 9.3 GPIO Property:

```
#define KSPROPERTY_CUSTOM_SET_OSD_TEXT_STRING_1 921 (WRITE ONLY) (16 BYTES)
```

```
#define KSPROPERTY_CUSTOM_SET_OSD_TEXT_STRING_2 922 (WRITE ONLY) (16 BYTES)
```

```
#define KSPROPERTY_CUSTOM_SET_OSD_TEXT_STRING_3 923 (WRITE ONLY) (16 BYTES)
```

```
#define KSPROPERTY_CUSTOM_SET_OSD_TEXT_STRING_4 924 (WRITE ONLY) (16 BYTES)
```

```
#define KSPROPERTY_CUSTOM_SET_OSD_COLOR 929 (WRITE ONLY) (ULONG)
```

The property \*SET\_OSD\_TEXT\_STRING\_1 accesses the first 16 characters.

The property \*SET\_OSD\_TEXT\_STRING\_2 accesses the 17 ~ 32 characters.

The property \*SET\_OSD\_TEXT\_STRING\_3 accesses the 33 ~ 48 characters.

The property \*SET\_OSD\_TEXT\_STRING\_4 accesses the 48 ~ 64 characters.

### 9.4 Video & Audio Property:

```
#define KSPROPERTY_CUSTOM_GET_DEVICE_VIDEO_CONFIG 8 (READ ONLY) (ULONG)
```

```
#define KSPROPERTY_CUSTOM_GET_DEVICE_AUDIO_CONFIG 9 (READ ONLY) (ULONG)
```

```
#define KSPROPERTY_CUSTOM_GET_ANALOG_VIDEO_MACROVISION 202 (READ ONLY) (ULONG)
```

```
#define KSPROPERTY_CUSTOM_GET_ANALOG_VIDEO_FRAME_RATE 208 (READ ONLY) (ULONG)
```

```
#define KSPROPERTY_CUSTOM_GET_ANALOG_VIDEO_RESOLUTION 210 (READ ONLY) (ULONG)
```

```
#define KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_QUEUE_BUFFER_SIZE 216 (ULONG)
#define KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_FLEXIBLE_FPS_PATCH 218 (ULONG)
#define KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_FLEXIBLE_RESOLUTION_PATCH 220 (ULONG)
#define KSPROPERTY_CUSTOM_GET_ANALOG_VIDEO_INTERLEAVED 223 (READ ONLY) (ULONG)
#define KSPROPERTY_CUSTOM_GET_ANALOG_VIDEO_SIGNAL_LOCK_STATUS 230 (READ ONLY)
#define KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_INPUT_AUTO_SCAN 232 (ULONG)
#define KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_RX_VGA_SOG 234 (ULONG)
#define KSPROPERTY_CUSTOM_GET_ANALOG_VIDEO_FRACTION_1000_1001 241 (READ ONLY) (ULONG)
#define KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_VERTICAL_MIRROR 244 (ULONG)
#define KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_HORIZONTAL_MIRROR 245 (ULONG)
#define KSPROPERTY_CUSTOM_XET_PREVIEW_VIDEO_STREAM_POST_SKIP_FRAMERATE 246 (ULONG)
#define KSPROPERTY_CUSTOM_XET_PREVIEW_VIDEO_STREAM_POST_AVG_FRAMERATE 247 (ULONG)
#define KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_INPUT_BANDWIDTH 248 (ULONG)
#define KSPROPERTY_CUSTOM_XET_ANALOG_AUDIO_VOLUME 251 (ULONG)
#define KSPROPERTY_CUSTOM_GET_ANALOG_AUDIO_SAMPLE_FREQUENCY 253 (ULONG)
#define KSPROPERTY_CUSTOM_XET_ANALOG_AUDIO_INPUT 255 (ULONG)
#define KSPROPERTY_CUSTOM_XET_PREVIEW_VIDEO_STREAM_POST_RESOLUTION 350 (ULONG)
```

## 9.5 Video Encoder Property:

Please reference the two functions to get/set all video encoder's parameters.

```
static const GUID GUID_KPS_TW2809 = { 0xD1E5209F, 0x68FD, 0x4529, 0xBE, 0xE0, 0x5E, 0x7A, 0x1F, 0x47, 0x92, 0x1E };
```

```
BOOL OnGetVideoCompressionProperty( ULONG nProperty, ULONG * pValue )
{
    if( NULL == m_pAMVideoCompression ) { FALSE; }

    if( NULL == m_pKsPropertySet ) { FALSE; }

    if( nProperty == 0x00000000 ) { // KEY.FRAME.RATE (GOP)
        if( S_OK != m_pAMVideoCompression->get_KeyFrameRate( (LONG *) (pValue) ) ) { return FALSE; }
    }
    if( nProperty == 0x00000001 ) { // QUALITY
        double fQuality = 0.0f;

        if( S_OK != m_pAMVideoCompression->get_Quality( &fQuality ) ) { return FALSE; }

        *pValue = (ULONG) (fQuality * 10000.0f);
    }
    if( nProperty == 0x00000003 ) { // BIT.RATE.MODE
        if( S_OK != m_pKsPropertySet->Get( GUID_KPS_TW2809, 407, NULL, 0, pValue, sizeof(ULONG), &cbBytes ) ) {
            return FALSE;
        }
    }
    if( nProperty == 0x00000004 ) { // BIT.RATE
        if( S_OK != m_pKsPropertySet->Get( GUID_KPS_TW2809, 403, NULL, 0, pValue, sizeof(ULONG), &cbBytes ) ) {
            return FALSE;
        }
    }
    if( nProperty == 0x00000008 ) { // POST.RESOLUTION
        if( S_OK != m_pKsPropertySet->Get( GUID_KPS_TW2809, 401, NULL, 0, pValue, sizeof(ULONG), &cbBytes ) ) {
            return FALSE;
        }
    }
    if( nProperty == 0x00000009 ) { // POST.SKIP.FRAME.RATE
        if( S_OK != m_pKsPropertySet->Get( GUID_KPS_TW2809, 402, NULL, 0, pValue, sizeof(ULONG), &cbBytes ) ) {
            return FALSE;
        }
    }
    if( nProperty == 0x0000000D ) { // POST.AVG.FRAME.RATE
        if( S_OK != m_pKsPropertySet->Get( GUID_KPS_TW2809, 422, NULL, 0, pValue, sizeof(ULONG), &cbBytes ) ) {
            return FALSE;
        }
    }
    return TRUE;
}
```

```

BOOL OnSetVideoCompressionProperty( ULONG nProperty, ULONG nValue )
{
    if( NULL == m_pAMVideoCompression ) { return FALSE; }

    if( NULL == m_pKsPropertySet ) { return FALSE; }

    if( nProperty == 0x00000000 ) { // KEY.FRAME.RATE (GOP)
        if( S_OK != m_pAMVideoCompression->put_KeyFrameRate( nValue ) ) { return FALSE; }
    }
    if( nProperty == 0x00000001 ) { // QUALITY
        double fQuality = nValue;

        fQuality /= 10000.0f;

        if( S_OK != m_pAMVideoCompression->put_Quality( fQuality ) ) { return FALSE; }
    }
    if( nProperty == 0x00000002 ) { // OVERRIDE.KEY.FRAME
        if( S_OK != m_pAMVideoCompression->OverrideKeyFrame( nValue ) ) { return FALSE; }
    }
    if( nProperty == 0x00000003 ) { // BIT.RATE.MODE
        if( S_OK != m_pKsPropertySet->Set( GUID_KPS_TW2809, 407, NULL, 0, &nValue, sizeof(ULONG) ) ) {
            return FALSE;
        }
    }
    if( nProperty == 0x00000004 ) { // BIT.RATE
        if( S_OK != m_pKsPropertySet->Set( GUID_KPS_TW2809, 403, NULL, 0, &nValue, sizeof(ULONG) ) ) {
            return FALSE;
        }
    }
    if( nProperty == 0x00000008 ) { // POST.RESOLUTION
        if( S_OK != m_pKsPropertySet->Set( GUID_KPS_TW2809, 401, NULL, 0, &nValue, sizeof(ULONG) ) ) {
            return FALSE;
        }
    }
    if( nProperty == 0x00000009 ) { // POST.SKIP.FRAME.RATE
        if( S_OK != m_pKsPropertySet->Set( GUID_KPS_TW2809, 402, NULL, 0, &nValue, sizeof(ULONG) ) ) {
            return FALSE;
        }
    }
    if( nProperty == 0x0000000D ) { // POST.AVG.FRAME.RATE
        if( S_OK != m_pKsPropertySet->Set( GUID_KPS_TW2809, 422, NULL, 0, &nValue, sizeof(ULONG) ) ) {
            return FALSE;
        }
    }
    return TRUE;
}

```

## **10. 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.