

SC5A0 Extra Software Programming Guide

Custom Property

- 1. **KSPROPERTY_CUSTOM_GET_DEVICE_VIDEO_CONFIG** (8)
- 1. **KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_INPUT** (201)
- 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)
- 1. **KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_RX_VGA_OFFSET_X** (221)
- 1. **KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_RX_VGA_OFFSET_Y** (222)
- 1. **KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_RX_VGA_ASPECT_RATIO** (224)
- 1. **KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_RX_VGA_HACTIVE_PIXELS** (225)
- 1. **KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_FLEXIBLE_FPS_PATCH** (218)
- 1. **KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_FLEXIBLE_RESOLUTION_PATCH** (220)
- 1. **KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_IS_SINGLE_FOMART_OUTPUT** (215)

The property **KSPROPERTY_CUSTOM_GET_DEVICE_VIDEO_CONFIG** allows you to get an OR combination of flag bits. This value shows what types of video sources you can set are supplied on one capture card.

EXAMPLE#01: TO GET THE SUPPORT INPUTS OF THE VIDEO SOURCE ON ONE CAPTURE CARD.

```
ULONG nInput = 0xFFFFFFFF;  
AMESDK_GET_CUSTOM_PROPERTY( hDevice, 8, nInput);
```

The property **KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_INPUT** allows you to get/change current video input source. We can support total 7 kinds of video input sources, HDMI, DVI-D, Components, DVI-A, SDI, COMPOSITE and SVIDEO.

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

EXAMPLE#02: SET INPUT TO HDMI.

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 201, 0 );
```

EXAMPLE#03: CHANGE TO SDI INPUT.

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 201, 4 );
```

EXAMPLE#04: GET CURRENT INPUT SOURCE.

```
AMESDK_GET_CUSTOM_PROPERTY( hDev, 201, &INPUT );
```

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#05 ENABLE THE AUTO INPUT SCAN FUNCTION

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

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 232, enable );
```

EXAMPLE#06 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	* ₁
1920×1080p@50fps	0x07800438	50 / 49.95	* ₁
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	* ₂
720×288P@50fps	0x05A00240	50 / 49.95	* ₂
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	* ₃
640×400p@60fps	0x02800190	60 / 59.94	* ₄
640×384p@60fps	0x02800180	60 / 59.94	* ₄

*₁ THE FORMAT WILL BE DOWN SPEED TO 1080P@30FPS/1080P@25FPS.

*₂ THE FORMAT IS USED BY SONY PS1/PS2 GAME MACHINE.

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

*₄ 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#07: 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

```
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)
```

EXAMPLE#08: 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;
}
```

If input is in VGA or YCbCr, these properties

```
KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_RX_VGA_OFFSET_X /
KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_RX_VGA_OFFSET_Y /
KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_RX_VGA_ASPECT_RATIO /
KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_RX_VGA_HACTIVE_PIXELS allow you to adjust
the hardware receiver's property.
```

The offset property **KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_RX_VGA_OFFSET_X**
/ **KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_RX_VGA_OFFSET_Y** 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#09: TO SET HORIZONTAL OFFSET FOR VGA.

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 221, -8 );
```

EXAMPLE#10: TO SET VERTICAL OFFSET FOR VGA.

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 222 -8 );
```

The aspect ratio property

KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_RX_VGA_ASPECT_RATIO allows you to adjust signal's aspect ratio during displaying. The boundary will be filled by black image.

SUPPORT VALUE: 0 (4 : 3), 1 (5 : 4), 3 (16 : 9), 4 (16 : 10),
5 (3 : 2), 6 (1 : 1), 7 (HACTIVE PIXELS)

EXAMPLE#11: TO SET 16:9 ASPECT RATIO FOR VGA.

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 224, 3 );
```

The horizontal active pixel property

KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_RX_VGA_HACTIVE_PIXELS allows you to set the total number of active pixels on a horizontal line. The horizontal component of timing consists of the horizontal active and horizontal blanking periods.

SUPPORT VALUE: PIXELS

EXAMPLE#12: TO SET HORIZONTAL ACTIVE PIXELS FOR VGA.

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 225, 1920 );
```

The three properties, 218, 220, 215, allows you to control the output format from one video capture filter.

The property **KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_FLEXIBLE_FPS_PATCH** 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 **KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_FLEXIBLE_RESOLUTION_PATCH** 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.

SUPPORT VALUE: 0 ~ 1 - DISABLE ~ ENABLE

EXAMPLE#13: TO ENABLE RESOLUTION SCALER.

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

To configure a capture filter's output format, the property

KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_IS_SINGLE_FOMART_OUTPUT can expose all range of output formats or expose only single format. If it is disabled, the capture filter expose all output formats. If it is enabled, we can only expose one format on the video capture filter output.

SUPPORT VALUE: 0: EXPOSE ALL FORMATS

1: SINGLE FORMAT

We can combine these three properties to remove image scale function.

EXAMPLE#14: TO REMOVE IMAGE SCALER.

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 215, 0 );
```

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 218, 0 );
```

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 220, 0 );
```

Note, to enable them (218, 220, 215), you need reboot the system.

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_XET_ANALOG_VIDEO_DEINTERLACE_TYPE (200)

QP0203 offers one hardware-based deinterlacer on chip. The property will allow you to access it. You can call the function, AMESDK_SET_CUSTOM_PROPERTY, to enable/disable this function.

SUPPORT VALUE: 0 ~ 1 - DISABLE ~ ENABLE

EXAMPLE#01: TO TURN OFF HARDWARE DEINTERLACE FUNCTION.

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 200, 0 );
```

EXAMPLE#02: TO TURN ON HARDWARE DEINTERLACE FUNCTION.

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 200, 1 );
```

Note!! The function, AMESDK_SET_DEINTERLACE, is used for software-based deinterlacer only. If you enable the hardware-based deinterlacer function, you don't need call AMESDK_SET_DEINTERLACE again.

4. 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; }
```

5. 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 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 SIZE TO 16 FRAMES

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

6. 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 );
```

7. 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 it makes you 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;  
AMESDK_SET_CUSTOM_PROPERTY( hDev, 231, input );
```

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

```
LONG input = 0x00100;  
AMESDK_SET_CUSTOM_PROPERTY( hDev, 231, input );
```

EXAMPLE#03: TO CHANGE ALL INPUT TO LIMITED RANGE

```
LONG input = 0x11100;  
AMESDK_SET_CUSTOM_PROPERTY( hDev, 231, input );
```

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

```
LONG input = 0x22222;  
AMESDK_SET_CUSTOM_PROPERTY( hDev, 231, input );
```

8. KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_CUSTOMIZED_OUTPUT_RESOLUTION (233)

The property allows user to set customized output resolution on the capture card. The driver will perform a check to ensure the resolution you have added is safe. If it is not, the return value of the property will be FALSE.

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

EXAMPLE#01: TO SETUP ONE CUSTOMIZED OUTPUT RESOLUTION.

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 233, &RESOLUTION );
```

9. 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 );
```

10. KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_DVI_YCBCR (235)

The property allows you to use the DVI-I connector for component incoming signals.

SUPPORT VALUE: 0 ~ 1 - DISABLE ~ ENABLE

EXAMPLE#01: TO ENABLE THE FUNCTION

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

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 235, input );
```

11. 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;
```

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 240, input );
```

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

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

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 240, input );
```

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

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

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 240, input );
```


12. KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_VERTICAL_MIRROR (244)

12. 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);
```

13. 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 );
```

14. 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 );
```

15. 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 );
```

16. KSPROPERTY_CUSTOM_GET_ANALOG_VIDEO_SIGNAL_COLORIMETRY (370) (READ ONLY)

16. KSPROPERTY_CUSTOM_GET_ANALOG_VIDEO_SIGNAL_COLORRANGE (371) (READ ONLY)

The property allows you to get the YCbCr transfer matrix and RGB color range for incoming signal. For example, suppose the device converts from RGB to YCbCr. If it gets the YCbCr matrix is 2 and the RGB color range is 1, the device converts full range RGB to ITU-R BT.709 YCbCr.

SUPPORT VALUE: 0: UNKNOWN

1: BT601

2: BT709

EXAMPLE#01: GET CURRENT YCbCr TRANSFER MATRIX.

```
ULONG nYCbCrMatrix = 0;
```

```
AMESDK_GET_CUSTOM_PROPERTY( hDev, 370, &nYCbCrMatrix );
```

SUPPORT VALUE: 0: UNKNOWN

1: FULL RANGE (0 ~ 255)

2: LIMITED RANGE (16 ~ 235)

EXAMPLE#02: GET CURRENT RGB COLOR RANGE.

```
ULONG nColorRange = 0;
```

```
AMESDK_GET_CUSTOM_PROPERTY( hDev, 371, &nColorRange );
```

17. Access Encoder Property

Developer can use the AMESDK_G/SET_VIDEOCOMPRESSION_PROPERTY function to access all QP0203'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, 2
VideoCompression_BitRate	128,000 ~ 64,000,000
VideoCompression_PostResolution	(cx << 12) + (cy << 0)
VideoCompression_PostSkipFrameRate	0 ~ 255
VideoCompression_PostAvgFrameRate	0 ~ 60
VideoCompression_BFrames	0, 1, 2
VideoCompression_Profile	0 (HIGH DEFAULT), 1 (BASELINE), 2 (MAIN), 3 (HIGH)
VideoCompression_AspectRatio	(cx << 12) + (cy << 0)
VideoCompression_Level	A UNSIGNED INTEGER VALUE
VideoCompression_Entropy	0 (CABAC), 1 (CAVLC), 2 (CABAC)

17. Access Custom Property for DirectShow Developer

Customer uses DirectShow to develop software can bypass our SDK to access QP0203 directly. The interface can be queried from our capture source filter.

At Section 17.1, 17.2 and 17.3, you can use IKsPropertySet to access all.

17.1 Device Serial Number Property:

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

17.2 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_XET_ANALOG_VIDEO_DEINTERLACE_TYPE 200 (ULONG)
#define KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_INPUT 201 (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_IS_SINGLE_FOMART_OUTPUT 215 (ULONG)
#define KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_QUEUE_BUFFER_SIZE 216 (ULONG)
#define KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_RX_VGA_PHASE 219 (ULONG)
#define KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_RX_VGA_OFFSET_X 221 (ULONG)
#define KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_RX_VGA_OFFSET_Y 222 (ULONG)
#define KSPROPERTY_CUSTOM_GET_ANALOG_VIDEO_INTERLEAVED 223 (READ ONLY) (ULONG)
#define KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_RX_VGA_ASPECT_RATIO 224 (ULONG)
#define KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_RX_VGA_HACTIVE_PIXELS 225 (ULONG)
#define KSPROPERTY_CUSTOM_GET_ANALOG_VIDEO_SIGNAL_LOCK_STATUS 230 (READ ONLY) (ULONG)
#define KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_COLOR_RANGE 231 (ULONG)
#define KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_INPUT_AUTO_SCAN 232 (ULONG)
#define KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_CUSTOMIZED_OUTPUT_RESOLUTION 233 (ULONG)
#define KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_SOG 234 (ULONG)
#define KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_DVI_YCBCR 235 (ULONG)
#define KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_INPUT_EQ 240 (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 (READ ONLY) (ULONG)
#define KSPROPERTY_CUSTOM_XET_ANALOG_AUDIO_INPUT 255 (ULONG)
#define KSPROPERTY_CUSTOM_XET_PREVIEW_VIDEO_STERAM_POST_RESOLUTION 350 (ULONG)
```

```
#define KSPROPERTY_CUSTOM_XET_PREVIEW_AUDIO_SAMPLE_POST_FREQUENCY 360 (ULONG)
#define KSPROPERTY_CUSTOM_GET_ANALOG_VIDEO_SIGNAL_COLORIMETRY 370 (READ ONLY) (ULONG)
#define KSPROPERTY_CUSTOM_GET_ANALOG_VIDEO_SIGNAL_COLORRANGE 371 (READ ONLY) (ULONG)
```


17.3 Video Encoder Property:

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

```
static const GUID GUID_KPS_QP0203 = { 0xD1E5209F, 0x68FD, 0x4529, 0xBE, 0xE0, 0x5E, 0x7A, 0x1F, 0x47, 0x92, 0x21 };
```

```
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_QP0203, 407, NULL, 0, pValue, sizeof(ULONG), &cbBytes ) ) {

            return FALSE;
        }
    }
    if( nProperty == 0x00000004 ) { // BIT.RATE

        if( S_OK != m_pKsPropertySet->Get( GUID_KPS_QP0203, 403, NULL, 0, pValue, sizeof(ULONG), &cbBytes ) ) {

            return FALSE;
        }
    }
    if( nProperty == 0x00000008 ) { // POST.RESOLUTION

        if( S_OK != m_pKsPropertySet->Get( GUID_KPS_QP0203, 401, NULL, 0, pValue, sizeof(ULONG), &cbBytes ) ) {

            return FALSE;
        }
    }
    if( nProperty == 0x00000009 ) { // POST.SKIP.FRAME.RATE

        if( S_OK != m_pKsPropertySet->Get( GUID_KPS_QP0203, 402, NULL, 0, pValue, sizeof(ULONG), &cbBytes ) ) {

            return FALSE;
        }
    }
    if( nProperty == 0x0000000D ) { // POST.AVG.FRAME.RATE

        if( S_OK != m_pKsPropertySet->Get( GUID_KPS_QP0203, 422, NULL, 0, pValue, sizeof(ULONG), &cbBytes ) ) {

            return FALSE;
        }
    }
    if( nProperty == 0x0000000A ) { // B.FRAME

        if( S_OK != m_pKsPropertySet->Get( GUID_KPS_QP0203, 411, NULL, 0, pValue, sizeof(ULONG), &cbBytes ) ) {

            return FALSE;
        }
    }
    if( nProperty == 0x0000000B ) { // PROFILE

        if( S_OK != m_pKsPropertySet->Get( GUID_KPS_QP0203, 412, NULL, 0, pValue, sizeof(ULONG), &cbBytes ) ) {

            return FALSE;
        }
    }
    if( nProperty == 0x0000000C ) { // ASPECT.RATIO

        if( S_OK != m_pKsPropertySet->Get( GUID_KPS_QP0203, 413, NULL, 0, pValue, sizeof(ULONG), &cbBytes ) ) {

            return FALSE;
        }
    }
    if( nProperty == 0x0000000E ) { // LEVEL

        if( S_OK != m_pKsPropertySet->Get( GUID_KPS_QP0203, 414, NULL, 0, pValue, sizeof(ULONG), &cbBytes ) ) {

            return FALSE;
        }
    }
}
```

```
if( nProperty == 0x0000000F ) { // ENTROPY
    if( S_OK != m_pKsPropertySet->Get( GUID_KPS_QP0203, 415, 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_QP0203, 407, NULL, 0, &nValue, sizeof(ULONG) ) ) {
            return FALSE;
        }
    }
    if( nProperty == 0x00000004 ) { // BIT.RATE
        if( S_OK != m_pKsPropertySet->Set( GUID_KPS_QP0203, 403, NULL, 0, &nValue, sizeof(ULONG) ) ) {
            return FALSE;
        }
    }
    if( nProperty == 0x00000008 ) { // POST.RESOLUTION
        if( S_OK != m_pKsPropertySet->Set( GUID_KPS_QP0203, 401, NULL, 0, &nValue, sizeof(ULONG) ) ) {
            return FALSE;
        }
    }
    if( nProperty == 0x00000009 ) { // POST.SKIP.FRAME.RATE
        if( S_OK != m_pKsPropertySet->Set( GUID_KPS_QP0203, 402, NULL, 0, &nValue, sizeof(ULONG) ) ) {
            return FALSE;
        }
    }
    if( nProperty == 0x0000000D ) { // POST.AVG.FRAME.RATE
        if( S_OK != m_pKsPropertySet->Set( GUID_KPS_QP0203, 422, NULL, 0, &nValue, sizeof(ULONG) ) ) {
            return FALSE;
        }
    }
    if( nProperty == 0x0000000A ) { // B.FRAME
        if( S_OK != m_pKsPropertySet->Set( GUID_KPS_QP0203, 411, NULL, 0, &nValue, sizeof(ULONG) ) ) {
            return FALSE;
        }
    }
    if( nProperty == 0x0000000B ) { // PROFILE
        if( S_OK != m_pKsPropertySet->Set( GUID_KPS_QP0203, 412, NULL, 0, &nValue, sizeof(ULONG) ) ) {
            return FALSE;
        }
    }
    if( nProperty == 0x0000000C ) { // ASPECT.RATIO
        if( S_OK != m_pKsPropertySet->Set( GUID_KPS_QP0203, 413, NULL, 0, &nValue, sizeof(ULONG) ) ) {
            return FALSE;
        }
    }
    if( nProperty == 0x0000000E ) { // LEVEL
        if( S_OK != m_pKsPropertySet->Set( GUID_KPS_QP0203, 414, NULL, 0, &nValue, sizeof(ULONG) ) ) {
            return FALSE;
        }
    }
    if( nProperty == 0x0000000F ) { // ENTROPY
        if( S_OK != m_pKsPropertySet->Set( GUID_KPS_QP0203, 415, NULL, 0, &nValue, sizeof(ULONG) ) ) {
            return FALSE;
        }
    }
    return TRUE;
}

```

18. 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.