

SC2B0/SC3B0 Extra Software Programming Guide

Custom Property

1. **KSPROPERTY_CUSTOM_GET_DEVICE_SERIAL_NUMBER_INFO** (0) (READ ONLY)

1. **KSPROPERTY_CUSTOM_GET_DEVICE_BUS_NUMBER_INFO** (2) (READ ONLY)

The property **KSPROPERTY_CUSTOM_GET_DEVICE_SERIAL_NUMBER_INFO** allows you to get Vendor ID (VID) and Product ID (PID) for the capture card. Vendor ID and product ID are 16-bit numbers used to identify PCI devices to a computer. The VID and PID are embedded in the capture card and communicated to the computer.

EXAMPLE#01: TO GET THE VENDER ID AND PRODUCT ID FROM THE CAPTURE CARD.

```
ULONG dwSerialNumber = 0x00000000;  
AMESDK_GET_CUSTOM_PROPERTY( hDevice, 0, &dwSerialNumber);
```

The property **KSPROPERTY_CUSTOM_GET_DEVICE_BUS_NUMBER_INFO** allows you to get current PCI bus number on the capture card. For example, the capture card on the first PCI slot, on the second PCI slot, or on the third PCI slot, etc.

EXAMPLE#02: TO GET THE BUS NUMBER ON THE CAPTURE CARD.

```
ULONG dwBusNumber = 0x00000000;  
AMESDK_GET_CUSTOM_PROPERTY( hDevice, 2, &dwBusNumber);
```

2. KSPROPERTY_CUSTOM_GET_ANALOG_VIDEO_SIGNAL_LOCK_STATUS (230) (READ ONLY)

2. KSPROPERTY_CUSTOM_GET_ANALOG_VIDEO_MACROVISION (202) (READ ONLY)

The property (230) 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_GET_CUSTOM_PROPERTY( hDev, 230, &nLock );
```

The property (202) 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#06: GET HDCP PROTECT.

```
AMESDK_GET_CUSTOM_PROPERTY( hDev, 202, &HDCP );
```

```
if( HDCP == 1 ) { RECORD_FUNCTION = DISABLE; }
```

```
if( HDCP == 0 ) { RECORD_FUNCTION = ENABLE; }
```

- 3. **KSPROPERTY_CUSTOM_GET_ANALOG_VIDEO_SINGAL_DEBUG_INFO** (271) (READ ONLY)
- 3. **KSPROPERTY_CUSTOM_GET_PREVIEW_VIDEO_STARAM_FRAME_NUMBER_INFO** (351) (READ ONLY)
- 3. **KSPROPERTY_CUSTOM_GET_PREVIEW_AUDIO_STARAM_FRAME_NUMBER_INFO** (361) (READ ONLY)
- 3. **KSPROPERTY_CUSTOM_GET_ENCODER_VIDEO_DEFAULT_FRAME_NUMBER_INFO** (430) (READ ONLY)

The property **KSPROPERTY_CUSTOM_GET_ANALOG_VIDEO_SINGAL_DEBUG_INFO** is used to get the debug information in capture card running state. The output information is 32-bit error numbers. If the number is 0, the device is working properly. You can call it in timer function to get current signal status regularly.

SUPPORT VALUE: 0: GOOD
OTHERS: ERROR BITS

EXAMPLE#01: TO GET CURRENT SINGAL DEBUG STATUS.

```
ULONG dwSingalDebugInfo = 0x00000000;
AMESDK_GET_CUSTOM_PROPERTY( hDevice, 271, &dwSingalDebugInfo);
```

The property **KSPROPERTY_CUSTOM_GET_PREVIEW_VIDEO_STARAM_FRAME_NUMBER_INFO** allows you to get the total number of frames in preview video. The property reads frame number information from video stream. You can call it in timer function to get current frame number regularly.

SUPPORT VALUE: FRAME NUMBER

EXAMPLE#02: TO GET VIDEO PREVIEW STREAM'S FRAME NUMBER.

```
ULONG dwPreviewVideoFrameNumber = 0;
AMESDK_GET_CUSTOM_PROPERTY( hDev, 351, &dwPreviewVideoFrameNumber );
```

The property **KSPROPERTY_CUSTOM_GET_PREVIEW_AUDIO_STARAM_FRAME_NUMBER_INFO** allows you to get the total number of frames in preview audio. The property reads frame number information from audio stream. You can call it in timer function to get current frame number regularly.

SUPPORT VALUE: FRAME NUMBER

EXAMPLE#03: TO GET AUDIO PREVIEW STREAM'S FRAME NUMBER.

```
ULONG dwPreviewAudioFrameNumber = 0;
AMESDK_GET_CUSTOM_PROPERTY( hDev, 361, &dwPreviewAudioFrameNumber);
```

The property **KSPROPERTY_CUSTOM_GET_ENCODER_VIDEO_DEFAULT_FRAME_NUMBER_INFO** allows you to get the total number of frames in video encoder. The property reads frame number information from compressed video stream. You can call it in timer

function to get current frame number regularly.

SUPPORT VALUE: FRAME NUMBER

EXAMPLE#04: TO GET VIDEO ENCODER STREAM STREAM'S FRAME NUMBER.

```
ULONG dwEncoderVideoFrameNumber = 0;
```

```
AMESDK_GET_CUSTOM_PROPERTY( hDev, 430, &dwEncoderVideoFrameNumber);
```

- 4. KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_DEINTERLACE_TYPE (200)
- 4. KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_VERTICAL_MIRROR (244)
- 4. KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_HORIZONTAL_MIRROR (245)

TW5864 offers one hardware-based de-interlace on chip.

SUPPORT VALUE: 0 ~ 1 - OFF ~ ON

EXAMPLE#01: 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 deinterlce function, you don't need call AMESDK_SET_DEINTERLACE again.

The two properties (244/245) are 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#02: ENABLE THE VERTICAL MIRROR FUNCTION ON DISPLAY WINDOW

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

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

EXAMPLE#03: ENABLE THE HORIZONTAL MIRROR FUNCTION ON DISPLAY WINDOW

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

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

5. 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 );
```

6. KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_INPUT_AUTO_SCAN (232)

This property 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 );
```

7. KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_QUEUE_BUFFER_SIZE (216)

The property allows 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 when you try to adjust the queue buffer size of which exceeds your system resource.

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 );
```


8. KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_FLEXIBLE_RESOLUTION_PATCH (220)

The property 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 customized format. For example, input resolution is 704x480 and capture output pin's resolution is 352x240.

SUPPORT VALUE: 0 ~ 1 - DISABLE ~ ENABLE

EXAMPLE#01: TO ENABLE RESOLUTION SCALER.

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

9. KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_FLEXIBLE_FPS_PATCH (218)

The property allows you to control the output format from one video capture filter. It 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

EXAMPLE#01: TO ENABLE FRAMERATE SCALER.

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 218, 1 );
```

10. KSPROPERTY_CUSTOM_XET_PREVIEW_VIDEO_STERAM_POST_RESOLUTION (350)

The property allows you to adjust current video resolution dynamically. The driver will re-allocate memory during changing video format on capture card running state.

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

EXAMPLE#01: TO SET PREVIEW VIDEO RESOLUTION DYNAMICALLY.

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 350, &RESOLUTION );
```

11. KSPROPERTY_CUSTOM_XET_PREVIEW_VIDEO_STREAM_POST_SKIP_FRAMERATE (246)

11. KSPROPERTY_CUSTOM_XET_PREVIEW_VIDEO_STARAM_POST_AVG_FRAMERATE (247)

The property (246) allows you to adjust current video skip frame rate dynamically. The range of the property is from 1 to 255. It is identical to the skip number of frame. For example, the value 1 will generate the preview frame rate, 15.000fps.

SUPPORT VALUE: 0: DISABLE
 1, 2, 3, 4, ... SKIP

EXAMPLE#01: TO SET PREVIEW VIDEO SKIP FRAMERATE DYNAMICALLY.

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 246, &FRAMERATE );
```

The property (247) allows you to adjust current video average frame rate dynamically. The range of the property is from 1 to 85. To enable it, our driver will follow the setting value to output one average fps. For example, 9 mean 9.00fps.

SUPPORT VALUE: 0: DISABLE
 1 ~ 85 FPS

EXAMPLE#02: TO SET PREVIEW VIDEO AVERAGE FRAMERATE DYNAMICALLY.

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 247, &FRAMERATE );
```

12. KSPROPERTY_CUSTOM_XET_GPIO_DIRECTION (940)

12. KSPROPERTY_CUSTOM_XET_GPIO_DATA (941)

12. KSPROPERTY_CUSTOM_GET_GPIO_SUPPORT (942) (READ ONLY)

The properties allow you to access TW5864'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

EXAMPLE#01: TO DEFINE GPIO AS 2 OUTPUT PINS [0:1] AND 2 INPUT PINS [2:3].

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 940, 0x00000003 );
```

EXAMPLE#02: TO DEFINE GPIO AS 4 OUTPUT PINS [0:3] THEN PULL HIGH FOR ALL.

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 940, 0x0000000F );
```

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 941, 0x0000000F );
```

EXAMPLE#03: TO DEFINE GPIO AS 4 INPUT PINS [0:3] THEN READ DATA FROM IT.

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 940, 0x00000000 );
```

```
AMESDK_GET_CUSTOM_PROPERTY( hDev, 941, &GPIO );
```

NOTE!! For DirectShow developer, please use IKsPropertySet to access the two properties. The property size is sizeof(ULONG) always.

13. KSPROPERTY_CUSTOM_SET_OSD_LINE (920) (WRITE ONLY)

13. KSPROPERTY_CUSTOM_SET_OSD_TEXT_STRING (921) (WRITE ONLY)

The properties allow you to change TW5864'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.

SUPPORT VALUE: 0 ~ 7 - LINE#0 ~ LINE#7

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

```
CHAR string[] = "0000.00.00 00:00:00:";
```

```
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 );
```

- 14. KSPROPERTY_CUSTOM_GET_MOTION_MASK_STATUS (900) (READ ONLY)
- 14. KSPROPERTY_CUSTOM_XET_MOTION_TEMPORAL_SENSITIVITY (910)
- 14. KSPROPERTY_CUSTOM_XET_MOTION_SPATIAL_SENSITIVITY (911)
- 14. KSPROPERTY_CUSTOM_XET_MOTION_LEVEL_SENSITIVITY (912)
- 14. KSPROPERTY_CUSTOM_XET_MOTION_SPEED (913)

TW5864 can offer hardware-based motion detection function. The property set allows you to access it. The properties *XET_MOTION_*_SENSITIVITY and *XET_MOTION_SPEED help you to control the motion detection's quality in different application.

SENSITIVITY SUPPORT VALUE: 0 ~ 15 - MORE ~ LESS SENSITIVE

SPEED SUPPORT VALUE: 0 ~ 63 - 1 FIELD ~ 64 FIELD INTERVALS

The property *GET_MOTION_MASK_STATUS returns the result of motion detection. TW5864 uses 192 (16x12) detection cells in full screen for motion detection. Each detection cell is composed of 44 pixels and 20 lines for NTSC and 24 lines for PAL. The property *GET_MOTION_MASK_STATUS offer 24 bytes (192 bits) to describe all 192 cells' result.

| | | | | | | | | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 001 | 002 | 003 | 004 | 005 | 006 | 007 | 008 | 009 | 010 | 011 | 012 | 013 | 014 | 015 | 016 |
| 017 | 018 | 019 | 020 | 021 | 022 | 023 | 024 | 025 | 026 | 027 | 028 | 029 | 030 | 031 | 032 |
| 033 | 034 | 035 | 036 | 037 | 038 | 039 | 040 | 041 | 042 | 043 | 044 | 045 | 046 | 047 | 048 |
| 049 | 050 | 051 | 052 | 053 | 054 | 055 | 056 | 057 | 058 | 059 | 060 | 061 | 062 | 063 | 064 |
| 065 | 066 | 067 | 068 | 069 | 070 | 071 | 072 | 073 | 074 | 075 | 076 | 077 | 078 | 079 | 080 |
| 081 | 082 | 083 | 084 | 085 | 086 | 087 | 088 | 089 | 090 | 091 | 092 | 093 | 094 | 095 | 096 |
| 097 | 098 | 099 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 |
| 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 | 128 |
| 129 | 130 | 131 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 | 144 |
| 145 | 146 | 147 | 148 | 149 | 150 | 151 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 | 160 |
| 161 | 162 | 163 | 164 | 165 | 166 | 167 | 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 | 176 |
| 177 | 178 | 179 | 180 | 181 | 182 | 183 | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 | 192 |

EXAMPLE#01: TO GET MOTION DETECTION RESULT.

```
BYTE status[ 24 ];
```

```
AMESDK_SET_CUSTOM_PROPERTY_EX( hDev, 900, status, sizeof(status) );
```

- 15. **KSPROPERTY_CUSTOM_XET_ENCODER_VIDEO_STERAM_RECORD_MODE** (407)
- 15. **KSPROPERTY_CUSTOM_XET_ENCODER_VIDEO_STERAM_RECORD_QUALITY** (404)
- 15. **KSPROPERTY_CUSTOM_XET_ENCODER_VIDEO_STERAM_RECORD_BITRATE** (403)
- 15. **KSPROPERTY_CUSTOM_XET_ENCODER_VIDEO_STERAM_RECORD_MAX_BITRATE** (409)
- 15. **KSPROPERTY_CUSTOM_XET_ENCODER_VIDEO_STERAM_RECORD_MIN_BITRATE** (410)

The property **KSPROPERTY_CUSTOM_XET_ENCODER_VIDEO_STERAM_RECORD_MODE** allows you to get/set record mode on hardware-compressed capture device. There are 3 kinds of encoder mode: variable bitrate (VBR), constant bitrate (CBR) and average bitrate (ABR).

SUPPORT VALUE: 0: VBR (FQP)
1: CBR
2: ABR

In the VBR mode, you choose the desired quality going from 0 (lowest quality) to 1000 (highest quality). The encoder tries to maintain the given quality for your video file. The main advantage is that you are able to specify the quality level that you want to reach, but the disadvantage is that the video size is unpredictable.

The property **KSPROPERTY_CUSTOM_XET_ENCODER_VIDEO_STERAM_RECORD_QUALITY** allows you to set a suitable quality in VBR mode.

SUPPORT VALUE: 0 ~ 10000

EXAMPLE#01: TO SET VIDEO ENCODER QUALITY.

```
AMESDK_SET_CUSTOM_PROPERTY( hEncoderDev, 407, 0 );  
AMESDK_SET_CUSTOM_PROPERTY( hEncoderDev, 404, 8000 );
```

In the CBR mode, the bitrate will be the same for the whole video file. The quality of your video is variable. The main advantage is that final video size can be accurately predicted, but the disadvantage is that the complex video parts will be a lower quality.

The property **KSPROPERTY_CUSTOM_XET_ENCODER_VIDEO_STERAM_RECORD_BITRATE** allows you to set a suitable bitrate in CBR mode.

SUPPORT VALUE: 0 ~ 60000000 BPS

EXAMPLE#02: TO SET VIDEO ENCODER BITRATE.

```
AMESDK_SET_CUSTOM_PROPERTY( hEncoderDev, 407, 1 );  
AMESDK_SET_CUSTOM_PROPERTY( hEncoderDev, 403, 12 * 1024 * 1024 );
```


In the ABR mode, you choose a target bitrate and the encoder will try to constantly maintain an average bitrate while using higher bitrate for the parts of your video that need more bits. The result will be of higher quality than CBR encoding while the average file size will remain predictable.

SUPPORT VALUE: 0 ~ 600000000 BPS

EXAMPLE#03: TO SET VIDEO ENCODER BITRATE.

```
AMESDK_SET_CUSTOM_PROPERTY( hEncoderDev, 407, 2 );  
AMESDK_SET_CUSTOM_PROPERTY( hEncoderDev, 403, 12 * 1024 * 1024 );
```

The two properties **KSPROPERTY_CUSTOM_XET_ENCODER_VIDEO_STERAM_RECORD_MAX_BITRATE** / **KSPROPERTY_CUSTOM_XET_ENCODER_VIDEO_STERAM_RECORD_MIN_BITRATE** allow you to set a suitable through/peak bitrate in ABR mode.

SUPPORT VALUE: 0 ~ 600000000 BPS

EXAMPLE#04: TO SET VIDEO ENCODER PEAK BITRATE.

```
AMESDK_SET_CUSTOM_PROPERTY( hEncoderDev, 407, 2 );  
AMESDK_SET_CUSTOM_PROPERTY( hEncoderDev, 409, 12 * 1024 * 1024 );
```

EXAMPLE#05: TO SET VIDEO ENCODER THROUGH BITRATE.

```
AMESDK_SET_CUSTOM_PROPERTY( hEncoderDev, 407, 2 );  
AMESDK_SET_CUSTOM_PROPERTY( hEncoderDev, 410, 1 * 1024 * 1024 );
```

16. KSPROPERTY_CUSTOM_XET_ENCODER_VIDEO_STREAM_POST_SKIP_FRAMERATE (402)

16. KSPROPERTY_CUSTOM_XET_ENCODER_VIDEO_STARAM_POST_AVG_FRAMERATE (422)

The property (402) allows you to adjust encoding skip frame rate dynamically. The range of the property is from 1 to 255. It is identical to the skip number of frame. For example, the value 1 will generate the encoding frame rate, 15.000fps.

SUPPORT VALUE: 0: DISABLE
 1, 2, 3, 4, ... SKIP

EXAMPLE#01: TO SET VIDEO ENCODER SKIP FRAMERATE DYNAMICALLY.

```
AMESDK_SET_CUSTOM_PROPERTY( hEncoderDev, 402, 1 );
```

The property (422) allows you to adjust encoding average frame rate dynamically. The range of the property is from 1 to 85. To enable it, our driver will follow the setting value to output one average fps. For example, 9 mean 9.00fps.

SUPPORT VALUE: 0: DISABLE
 1 ~ 85 FPS

EXAMPLE#02: TO SET VIDEO ENCODER AVERAGE FRAMERATE DYNAMICALLY.

```
AMESDK_SET_CUSTOM_PROPERTY( hEncoderDev, 422, 9 );
```

17. KSPROPERTY_CUSTOM_XET_ENCODER_VIDEO_STERAM_POST_RESOLUTION (401)

The property allows you to adjust video encoding resolution dynamically. The driver will re-allocate memory during changing video format on capture card running state.

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

EXAMPLE#01: TO SET VIDEO ENCODER RESOLUTION DYNAMICALLY.

```
AMESDK_SET_CUSTOM_PROPERTY( hEncoderDev, 401, &RESOLUTION );
```

18. KSPROPERTY_CUSTOM_XET_ENCODER_VIDEO_STERAM_PROFILE (412)

The property allows you to adjust video encoder profile on hardware-compressed capture device. There is only one kind of profile value.

SUPPORT VALUE: 0: DEFAULT (BASELINE)
 1: BASELINE

EXAMPLE#01: TO SET VIDEO ENCODER PROFILE.

```
AMESDK_SET_CUSTOM_PROPERTY( hEncoderDev, 412, 1 );
```

19. KSPROPERTY_CUSTOM_XET_ENCODER_VIDEO_STERAM_ENTROPY (415)

The property allows you to adjust video encoder entropy on hardware-compressed capture device. There is only one kind of entropy value.

SUPPORT VALUE: 0: DEFAULT (CAVLC)
 1: CAVLC

EXAMPLE#01: TO SET VIDEO ENCODER ENTROPY.

```
AMESDK_SET_CUSTOM_PROPERTY( hEncoderDev, 412, 1 );
```

20. KSPROPERTY_CUSTOM_XET_ENCODER_VIDEO_STERAM_GOP (405)

The property allows you to set the maximum number of frames between each key frame on hardware-compressed capture device. For example, a value of 100 will create a key frame every 100 frames. A smaller GOP value will increase the size of your video file, but it will allow more precise playback in most players.

The GOP set to higher value can increase the compression ratio, but that would not be free to jump to any time point in playback.

SUPPORT VALUE: 0 ~ 255

EXAMPLE#01: TO SET VIDEO ENCODER GOP.

```
AMESDK_SET_CUSTOM_PROPERTY( hEncoderDev, 405, 30 );
```

21. KSPROPERTY_CUSTOM_SET_ENCODER_VIDEO_STERAM_FORCE_KEY_FRAME (406)

The property allows you to set video encoder force key frame on hardware-compressed capture device. The property puts key frame on the next frame or forcing a key frame at specified timestamp.

SUPPORT VALUE: 1: FROCE KEYFRAME

EXAMPLE#01: TO SET VIDEO ENCODER FORCE KEYFRAME.

```
AMESDK_SET_CUSTOM_PROPERTY( hEncoderDev, 406, 1 );
```

22. KSPROPERTY_CUSTOM_XET_ENCODER_VIDEO_STERAM_FRAME_QUEUE_LENGTH (424)

The property allows you to specify the number of the rendered video frame in the queue buffer for video encoded stream. By the default, the queue size of the corresponding video encoded stream is set 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 VIDEO ENCODER QUEUE SIZE TO 16 FRAMES

```
LONG nBfferSize = 16;
```

```
AMESDK_SET_CUSTOM_PROPERTY(hEncoderDev, 424, nBfferSize );
```


23. Access Video Encoder Property

Developer can use the AMESDK_G/SET_VIDEOCOMPRESSION_PROPERTY function to access all TW5864's video encoder properties. These properties as describe as the table below:

| PROPERTY | RANGE |
|------------------------------------|------------------------------------|
| VideoCompression_PostResolution | (cx << 12) + (cy << 0) |
| VideoCompression_PostSkipFrameRate | 0 ~ 255 |
| VideoCompression_PostAvgFrameRate | 0 ~ 85 |
| VideoCompression_Profile | 0 (BASELINE DEFAULT), 1 (BASELINE) |
| VideoCompression_Entropy | 0 (CAVLC DEFAULT), 1 (CAVLC) |
| VideoCompression_RecordMode | 0 (VBR), 1 (CBR), 2 (ABR) |
| VideoCompression_RecordQuality | 0 ~ 10000 |
| VideoCompression_BitRate | 0 ~ 60000000 BPS |
| VideoCompression_MaxBitRate | 0 ~ 60000000 BPS |
| VideoCompression_MinBitRate | 0 ~ 60000000 BPS |
| VideoCompression_KeyFrameRate | 0 ~ 255 |
| VideoCompression_OverrideKeyFrame | 1 (WRITE ONLY) |

24. Access Custom Property for DirectShow Developer

Customer uses DirectShow to develop surveillance software can bypass our SDK to access TW5864 directly. The interface can be queried from our capture source filter.

You can use IKsPropertySet to access all.

24.1 Device Serial Number Property:

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

24.2 GPIO Property:

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

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

```
#define KSPROPERTY_CUSTOM_XET_GPIO_CONFIG_INFO 942 (ULONG)
```

24.3 OSD Property:

```
#define KSPROPERTY_CUSTOM_SET_OSD_LINE 920 (WRITE ONLY) (ULONG)
```

```
#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)
```

The property *SET_OSD_TEXT_STRING_1 accesses the first 16 characters.

The property *SET_OSD_TEXT_STRING_2 accesses the 17 ~ 32 characters.

To disable the default time OSD on LINE#0, you can follow these steps as below:

```
ULONG nValue = 0x00000000; // LINE#0
CHAR psz[ 16 ] = " "; // SPACE
if( S_OK != m_pKsPropertySet->Set( GUID_KPS_TW5864, 920, NULL, 0, &nValue, sizeof(ULONG) ) ) {
    return FALSE;
}
if( S_OK != m_pKsPropertySet->Set( GUID_KPS_TW5864, 921, NULL, 0, (BYTE *) (psz), 16 ) ) {
    return FALSE;
}
```

24.4 Video Encoder Property:

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

```

static const GUID GUID_KPS_TW5864 = { 0xD1E5209F, 0x68FD, 0x4529, 0xBE, 0xE0, 0x5E, 0x7A, 0x1F, 0x47, 0x92, 0x1D };

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_TW5864, 407, NULL, 0, pValue, sizeof(ULONG), &cbBytes ) ) {

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

        if( S_OK != m_pKsPropertySet->Get( GUID_KPS_TW5864, 403, NULL, 0, pValue, sizeof(ULONG), &cbBytes ) ) {

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

        if( S_OK != m_pKsPropertySet->Get( GUID_KPS_TW5864, 401, NULL, 0, pValue, sizeof(ULONG), &cbBytes ) ) {

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

        if( S_OK != m_pKsPropertySet->Get( GUID_KPS_TW5864, 402, 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_TW5864, 407, NULL, 0, &nValue, sizeof(ULONG) ) ) {
            return FALSE;
        }
    }
    if( nProperty == 0x00000004 ) { // BIT.RATE
        if( S_OK != m_pKsPropertySet->Set( GUID_KPS_TW5864, 403, NULL, 0, &nValue, sizeof(ULONG) ) ) {
            return FALSE;
        }
    }
    if( nProperty == 0x00000008 ) { // POST.RESOLUTION
        if( S_OK != m_pKsPropertySet->Set( GUID_KPS_TW5864, 401, NULL, 0, &nValue, sizeof(ULONG) ) ) {
            return FALSE;
        }
    }
    if( nProperty == 0x00000009 ) { // POST.SKIP.FRAME.RATE
        if( S_OK != m_pKsPropertySet->Set( GUID_KPS_TW5864, 402, NULL, 0, &nValue, sizeof(ULONG) ) ) {
            return FALSE;
        }
    }
    return TRUE;
}

```

25. 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.