

SC520 Extra Software Programming Guide

Custom Property

- 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 SAA7160'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].
`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_GET_ANALOG_VIDEO_RESOLUTION (210) (READ ONLY)

2. 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 format's resolution and frame rate. All supported formats are described in the table:

FORMAT	RESOLUTION	FRAME RATE
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
720×480P@60fps	0x02D001E0	60
720×576P@50fps	0x02D00240	50
720×480i@60fps	0x02D000F0	60
720×576i@50fps	0x02D00120	50

Here, the resolution property can be described as below:

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

EXAMPLE#01: GET CURRENT VIDEO FORMAT.

```
AMESDK_GET_CUSTOM_PROPERTY( hDev, 210, &RESOLUTION );
```

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

3. KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_SWITCH_CHANNEL_TABLE (206)

In default setting, our switching algorithm uses one averaged channel table to control the channel switching sequence. The table size is 12 items length. Every item can be 0, 1, 2 or 3 to correspond to its sub-channels. For example, the split number is 4. The default switching channel table will be as { 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3 }.

Now, you can control the switching table dynamically by our SDK. For example, the table can be updated to { 0, 0, 1, 2, 0, 0, 1, 2, 0, 0, 1, 2 }. The total 20fps for every sub-channel will be changed as below:

CH#01: 10fps,
CH#02: 5fps,
CH#03: 5fps, and
CH#04: 0fps.

For another example, the table is { 0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3 }. The result simulates one channel jumping effect.

Moreover, the table also can support single channel switching such as { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 }. When the table is set, the switching mode will auto be returned to real-time mode. So, by this table, the CH#02's fps will be up to 30fps.

EXAMPLE#01: DISABLE CH#03.

```
BYTE TABLE[ 12 ] = { 0, 1, 3, 0, 1, 3, 0, 1, 3, 0, 1, 3 };  
AMESDK_SET_CUSTOM_PROPERTY_EX( hDev, 206, TABLE, 12 );
```

EXAMPLE#02: CHANNEL JUMPING.

```
BYTE TABLE[ 12 ] = { 0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3 };  
AMESDK_SET_CUSTOM_PROPERTY_EX( hDev, 206, TABLE, 12 );
```

EXAMPLE#03: GET CURRENT SWITCH CHANNEL TABLE.

```
BYTE TABLE[ 12 ];  
AMESDK_GET_CUSTOM_PROPERTY_EX( hDev, 206, TABLE, 12 );
```

EXAMPLE#04: SINGLE CHANNEL OUTPUT.

```
BYTE TABLE[ 12 ] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };  
AMESDK_SET_CUSTOM_PROPERTY_EX( hDev, 206, TABLE, 12 );
```

8. Application Note for AMESDK_GET_LOCK()

Customer to use AMESDK_GET_LOCK, please notes it. If your card is N series, the return value is described by 1 bit only. High is signal lock, and low is unlock. If your card is D series, the return value will use 2 bits to describe both sub-channels' status. If card is Q series, we will use 4 bits to describe all sub-channels.

EXAMPLE#01: GET SC520N4 SIGNAL STATUS.

```
AMESDK_GET_LOCK( hDev[ 0 ], &status[ 0 ] ); // GET CH01 STATUS
AMESDK_GET_LOCK( hDev[ 1 ], &status[ 1 ] ); // GET CH02 STATUS
AMESDK_GET_LOCK( hDev[ 2 ], &status[ 2 ] ); // GET CH03 STATUS
AMESDK_GET_LOCK( hDev[ 3 ], &status[ 3 ] ); // GET CH04 STATUS
```

EXAMPLE#02: GET SC520D8 SIGNAL STATUS.

```
AMESDK_GET_LOCK( hDev[ 0 ], &status[ 0 ] ); // GET CH01 ~ CH02 STATUS
AMESDK_GET_LOCK( hDev[ 1 ], &status[ 1 ] ); // GET CH03 ~ CH04 STATUS
AMESDK_GET_LOCK( hDev[ 2 ], &status[ 2 ] ); // GET CH05 ~ CH06 STATUS
AMESDK_GET_LOCK( hDev[ 3 ], &status[ 3 ] ); // GET CH07 ~ CH08 STATUS
```

EXAMPLE#03: GET SC520Q16 SIGNAL STATUS.

```
AMESDK_GET_LOCK( hDev[ 0 ], &status[ 0 ] ); // GET CH01 ~ CH04 STATUS
AMESDK_GET_LOCK( hDev[ 1 ], &status[ 1 ] ); // GET CH05 ~ CH08 STATUS
AMESDK_GET_LOCK( hDev[ 2 ], &status[ 2 ] ); // GET CH09 ~ CH12 STATUS
AMESDK_GET_LOCK( hDev[ 3 ], &status[ 3 ] ); // GET CH13 ~ CH16 STATUS
```

5. Access Custom Property for DirectShow Developer

Customer uses DirectShow to develop software can bypass our SDK to access SAA7160 directly. All custom properties are implemented by IKsPropertySet interface. The interface can be queried from our capture source filter.

EXAMPLE#01: GET CURRENT RESOLUTION AND FRAMERATE FOR INPUT.

```
static const GUID GUID_KPS_SAA7160 = { 0xD1E5209F, 0x68FD, 0x4529, 0xBE, 0xE0, 0x5E, 0x7A, 0x1F, 0x47, 0x92, 0x1C };  
m_pKsPropertySet->Set( GUID_KPS_SAA7160, 210, NULL, 0, &dwResolution, sizeof(ULONG) );  
m_pKsPropertySet->Set( GUID_KPS_SAA7160, 208, NULL, 0, &dwFrameRate, sizeof(ULONG) );
```

6. 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.