



# Windows AV/TV Media Software Development Kit

## Software Programming Guide

### APPROVAL SHEET

AUTHORIZED SIGNATURE	
YUAN	CUSTOMER

Issued by:	H.P.LIN (huengpei@yuan.com.tw)
File Revision:	1.1.0.113.8
Release Date:	2012/02/20

**YUAN High-Tech Development Co., Ltd.**

18F, NO.88, Sec.2, Chung Hsiao E.Rd.,  
Taipei City, Taiwan  
TEL: 886-2-2392-1233  
FAX: 886-2-2392-1338

## About SDK:

This SDK provides a powerful list of APIs for customer to develop their own application easily. All cards from YUAN are supported by this SDK with same interface. Therefore, if you had already programmed SC300 card and you want to program other products such as SC310, SC390 ... etc., you can finish the time-consuming work with little effort.

The SDK is based on DirectShow technique to combine with our capture card driver. Before using it, you need put CoInitialize/CoUninitialize into your software to enable/disable Microsoft COM library.

Tasks supported:

### **(1) Live Data (Uncompressed Data) Handling**

If you want to capture the video live data (uncompressed data) from device, such as YUY2 or UYVY, you can refer to Chapter 1. This chapter tells you how to capture and control the video live data. Implements only AMESDK\_CREATE, AMESDK\_SET\_STANDARD, AMESDK\_SET\_FORMAT and AMESDK\_RUN, then you will get live data.

### **(2) Hardware Compression Video Stream Handling**

If you want to capture hardware compression video stream, such MPEG4 or H.264, Chapter 2 explains these functions. Implements only AMESDK\_CREATE, AMESDK\_SET\_STANDARD, AMESDK\_SET\_FORMAT and AMESDK\_RUN, then you will get hardware compression video stream.

### **(3) Audio PCM Stream Handling**

If you want to capture audio PCM stream, then please refer to Chapter 3. Implements only AMESDK\_CREATE, AMESDK\_SET\_FORMAT and AMESDK\_RUN, then you will get audio PCM stream.

### **(4) Hardware Compression Audio Stream Handling**

Chapter 4 describes the functions to capture audio hardware compression data, such G.721 or G.723. It is not opened publicly. If you have such requirement, please inform us.

The Chapter 1 ~ 4 explains how to access all functions of our hardware board. To depend on different products, we can combine these functions together. For example, SC390 capture card can capture both UYVY live data and H.264 data stream at the same time; you should reference Chapter 1 and Chapter 2. In addition, there are some example codes at the end of each chapter for you to reference. Note!! we suggest you can read the product's extra programming guide at the same time, too. More custom properties are described in the document.

### **(5) Software Codec (Encoder and Decoder) Handling**

Chapter 5 describes the functions for software codec (encoder and decoder) programming such as MPEG4 and H.264 for video and PCM, G.721, and G.723 for audio.

## (6) Network Streaming Server and Client Handling

Chapter 6 describes the functions for network streaming server/client programming. RTSP server and client can be setup to broadcast media. With the help of this chapter, media access through network is accomplished.

## (7) File Record and Playback Handling

Chapter 7 describes the functions for file record/playback programming. All the file accesses for record, such as save, copy, delete, export are all implemented in this chapter. The file accesses for playback, such as seek, step, play rate, are also included. Besides media data, GPS information is also provided.

The SDK package contains library, included file, document, and sample source code, which helps you to build an application to run our capture card. Before running application, you must install CODEC.exe firstly.

The SDK structure is shown as follows.

### SDK PACKET:

- **DOC\:** Documents
- **INC\:** Header files
- **LIB\:** LIB & DLL files for VC & NET (VB/C#/J#)
- **SAMPLES\:** Sample Source Codes
- HISTORY.TXT SDK Update History

## Contents

<b>1</b>	<b>EXPORTED FUNCTIONS FOR ANALOG VIDEO CAPTURE DEVICE (YUY2/UYVY/YV12)</b>	<b>7</b>
1.00	AMESDK_CAPTURE_DEVICE_ENUMERATION	10
1.01	AMESDK_CREATE	12
1.02	AMESDK_DESTROY	16
1.03	AMESDK_RUN	17
1.04	AMESDK_STOP	18
1.05	AMESDK_AUTO_STANDARD_DETECTION	19
1.06	AMESDK_GET_STANDARD	20
1.07	AMESDK_SET_STANDARD	21
1.08	AMESDK_GET_INPUT	22
1.09	AMESDK_SET_INPUT	23
1.10	AMESDK_GET_FORMAT	24
1.11	AMESDK_SET_FORMAT	27
1.12	AMESDK_GET_DEINTERLACE	30
1.13	AMESDK_SET_DEINTERLACE	31
1.14	AMESDK_GET_MIRROR	32
1.15	AMESDK_SET_MIRROR	33
1.16	AMESDK_GET_MODE	34
1.17	AMESDK_SET_MODE	35
1.18	AMESDK_GET_FREQUENCY	36
1.19	AMESDK_SET_FREQUENCY	37
1.20	AMESDK_GET_LOCK	38
1.21	AMESDK_GET_FPS	39
1.22	AMESDK_GET_CAMERACONTROL_PROPERTY	40
1.23	AMESDK_SET_CAMERACONTROL_PROPERTY	41
1.24	AMESDK_GET_VIDEOPROCAMP_PROPERTY	42
1.25	AMESDK_SET_VIDEOPROCAMP_PROPERTY	43
1.26	AMESDK_GET_CUSTOM_PROPERTY	44
1.27	AMESDK_SET_CUSTOM_PROPERTY	46
1.28	AMESDK_GET_CUSTOM_PROPERTY_EX	48
1.29	AMESDK_SET_CUSTOM_PROPERTY_EX	49
1.30	AMESDK_OTHER_REFRESH_DISPLAY_WINDOW	50
1.31	AMESDK_OTHER_SNAPSHOT_BMP	51
1.32	AMESDK_OTHER_SNAPSHOT_JPG	52
1.33	AMESDK_OTHER_ZOOM	53
1.34	SC100#N4 (CIF) SOFTWARE PROGRAMMING GUIDE	54
1.35	SC100#N4 (VGA) SOFTWARE PROGRAMMING GUIDE	55
1.36	SC300#N8 SOFTWARE PROGRAMMING GUIDE	56
1.37	SC300#Q16 SOFTWARE PROGRAMMING GUIDE	57
1.38	SC300#D8 SOFTWARE PROGRAMMING GUIDE	58
1.39	SC280#N4 (LIVE) SOFTWARE PROGRAMMING GUIDE	59
1.40	SC380#N16 (LIVE) SOFTWARE PROGRAMMING GUIDE	60
<b>2</b>	<b>EXPORTED FUNCTIONS FOR ANALOG VIDEO CAPTURE DEVICE (MPEG4/H.264)</b>	<b>61</b>
2.01	AMESDK_CREATE	64
2.10	AMESDK_GET_FORMAT	67
2.11	AMESDK_SET_FORMAT	70
2.26	AMESDK_GET_VIDEOCOMPRESSION_PROPERTY	73
2.27	AMESDK_SET_VIDEOCOMPRESSION_PROPERTY	76
2.36	SC380#N16 (MPEG4) SOFTWARE PROGRAMMING GUIDE	77
2.37	SC380#N16 (LIVE + MPEG4) SOFTWARE PROGRAMMING GUIDE	78
2.38	SC390#N16 (H.264) SOFTWARE PROGRAMMING GUIDE	80
2.39	SC390#N16 (LIVE + H.264) SOFTWARE PROGRAMMING GUIDE	81
<b>3</b>	<b>EXPORTED FUNCTIONS FOR ANALOG AUDIO CAPTURE DEVICE (PCM)</b>	<b>83</b>
3.01	AMESDK_CREATE	85
3.05	AMESDK_GET_INPUT	88
3.06	AMESDK_SET_INPUT	89
3.07	AMESDK_GET_FORMAT	90

3.08	AMESDK_SET_FORMAT	92
3.09	AMESDK_GET_VOLUME	94
3.10	AMESDK_SET_VOLUME	95
3.11	SC100#N4 SOFTWARE PROGRAMMING GUIDE	96
3.12	SC300#N8 SOFTWARE PROGRAMMING GUIDE	97
3.13	SC310#N8 SOFTWARE PROGRAMMING GUIDE	98
<b>4</b>	<b>EXPORTED FUNCTIONS FOR ANALOG AUDIO CAPTURE DEVICE (G.721/G.723)</b>	<b>99</b>
<b>5</b>	<b>EXPORTED FUNCTIONS FOR SOFTWARE ENCODER AND DECODER PROGRAMMING</b>	<b>101</b>
5.01	AMESDK_CREATE	103
5.13	AMESDK_GET_FORMAT	105
5.14	AMESDK_SET_FORMAT	107
5.15	AMESDK_CODEC_ENCODE	109
5.16	AMESDK_CODEC_DECODE	110
5.17	AMESDK_CODEC_DECODE_EX	111
<b>6</b>	<b>EXPORTED FUNCTIONS FOR NETWORK SERVER AND CLIENT PROGRAMMING</b>	<b>112</b>
6.01	AMESDK_CREATE	114
6.12	AMESDK_NETWORK_SET_VIDEO_STREAM_FORMAT	116
6.13	AMESDK_NETWORK_SET_AUDIO_STREAM_FORMAT	117
6.14	AMESDK_NETWORK_SET_VIDEO_STREAM_BUFFER	118
6.15	AMESDK_NETWORK_SET_AUDIO_STREAM_BUFFER	119
6.16	AMESDK_NETWORK_GET_VIDEO_STREAM_STATISTICS	120
6.17	AMESDK_NETWORK_GET_AUDIO_STREAM_STATISTICS	121
6.18	AMESDK_NETWORK_SET_STREAMING_ADAPTER	122
6.19	AMESDK_NETWORK_SET_STREAMING_PORT	123
6.20	AMESDK_NETWORK_SET_USER_ACCOUNT	124
6.21	AMESDK_NETWORK_SET_CALLBACK	125
6.22	AMESDK_NETWORK_SET_CUSTOM_PROPERTY	127
6.23	AMESDK_NETWORK_GET_CUSTOM_PROPERTY	128
6.24	AMESDK_NETWORK_CONNECT_STREAMING_SERVER	129
<b>7</b>	<b>EXPORTED FUNCTIONS FOR FILE RECORD AND PLAYBACK PROGRAMMING</b>	<b>130</b>
7.01	AMESDK_CREATE	133
7.05	AMESDK_PAUSE	135
7.06	AMESDK_STEP	136
7.17	AMESDK_FILE_GET_VIDEO_STREAM_FORMAT	137
7.18	AMESDK_FILE_SET_VIDEO_STREAM_FORMAT	139
7.19	AMESDK_FILE_GET_AUDIO_STREAM_FORMAT	141
7.20	AMESDK_FILE_SET_AUDIO_STREAM_FORMAT	142
7.21	AMESDK_FILE_SET_VIDEO_STREAM_BUFFER	143
7.22	AMESDK_FILE_SET_AUDIO_STREAM_BUFFER	145
7.23	AMESDK_FILE_GET_VIDEO_STREAM_BUFFER	146
7.24	AMESDK_FILE_GET_AUDIO_STREAM_BUFFER	147
7.25	AMESDK_FILE_GET_VIDEO_STREAM_DATA	148
7.26	AMESDK_FILE_GET_AUDIO_STREAM_DATA	149
7.27	AMESDK_FILE_GET_VIDEO_TIMESTAMP_MAP	150
7.28	AMESDK_FILE_GET_AUDIO_TIMESTAMP_MAP	151
7.29	AMESDK_FILE_GET_VIDEO_SECTION_MAP	152
7.30	AMESDK_FILE_GET_AUDIO_SECTION_MAP	153
7.31	AMESDK_FILE_GET_VIDEO_MOTION_MAP	154
7.32	AMESDK_FILE_GET_VIDEO_GPS_MAP	155
7.33	AMESDK_FILE_GET_VIDEO_POS_MAP	156
7.34	AMESDK_FILE_GET_MEDIA_LENGTH	157
7.35	AMESDK_FILE_GET_MEDIA_POSITION	158
7.36	AMESDK_FILE_SET_MEDIA_POSITION	159
7.37	AMESDK_FILE_GET_MEDIA_PLAYBACK_RATE	160
7.38	AMESDK_FILE_SET_MEDIA_PLAYBACK_RATE	161
7.39	AMESDK_FILE_GET_MEDIA_PLAYBACK_SKIP_MODE	162
7.40	AMESDK_FILE_SET_MEDIA_PLAYBACK_SKIP_MODE	163
7.41	AMESDK_FILE_SEEK	164

7.42	AMESDK_FILE_FLUSH .....	165
7.43	AMESDK_FILE_DELETE_THE_OLDEST_FILE .....	166
7.44	AMESDK_FILE_GET_FILE_INFO_LIST .....	168
7.45	AMESDK_FILE_UPDATE_FILE_INFO_LIST .....	170
7.46	AMESDK_FILE_FREE_FILE_INFO_LIST .....	172
7.47	AMESDK_FILE_EXPORT .....	173
7.48	AMESDK_FILE_SELF_REPAIR .....	175
7.49	AMESDK_FILE_REPAIR .....	176
7.50	AMESDK_FILE_DIAGNOSIS .....	177
7.51	AMESDK_FILE_STD_RESTORE .....	178
7.52	AN AVI FILE WRITER SOFTWARE PROGRAMMING GUIDES .....	179
7.53	AN AVI FILE PLAYER SOFTWARE PROGRAMMING GUIDES .....	182
<b>8</b>	<b>EXPORTED FUNCTIONS FOR GPS DEVICE .....</b>	<b>184</b>
8.01	AMESDK_CREATE .....	186
8.05	AMESDK_GPS_GET_DATA .....	187
<b>9</b>	<b>EXPORTED FUNCTIONS FOR SOFTWARE DEINTERLACER .....</b>	<b>188</b>
9.01	AMESDK_CREATE .....	190
9.07	AMESDK_DI_DEINTERLACE .....	191

## **1 Exported Functions for Analog Video Capture Device (YUY2/UYVY/YV12)**

SUPPORT DEVICE:

PD652, SC100, SC200, SC230, SC280,  
SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0,  
SC3B0, SC500, SC510, SC580, SC590,

## Exported Functions For Analog Video Capture Device

1.00	AMESDK_CAPTURE_DEVICE_ENUMERATION
1.01	AMESDK_CREATE
1.02	AMESDK_DESTROY
1.03	AMESDK_RUN
1.04	AMESDK_STOP
1.05	AMESDK_AUTO_STANDARD_DETECTION
1.06	AMESDK_GET_STANDARD
1.07	AMESDK_SET_STANDARD
1.08	AMESDK_GET_INPUT
1.09	AMESDK_SET_INPUT
1.10	AMESDK_GET_FORMAT
1.11	AMESDK_SET_FORMAT
1.12	AMESDK_GET_DEINTERLACE
1.13	AMESDK_SET_DEINTERLACE
1.14	AMESDK_GET_MIRROR
1.15	AMESDK_SET_MIRROR
1.16	AMESDK_GET_MODE
1.17	AMESDK_SET_MODE
1.18	AMESDK_GET_FREQUENCY
1.19	AMESDK_SET_FREQUENCY
1.20	AMESDK_GET_LOCK
1.21	AMESDK_GET_FPS
1.22	AMESDK_GET_CAMERACONTROL_PROPERTY
1.23	AMESDK_SET_CAMERACONTROL_PROPERTY
1.24	AMESDK_GET_VIDEOPROCAMP_PROPERTY
1.25	AMESDK_SET_VIDEOPROCAMP_PROPERTY
1.26	AMESDK_GET_CUSTOM_PROPERTY
1.27	AMESDK_SET_CUSTOM_PROPERTY
1.28	AMESDK_GET_CUSTOM_PROPERTY_EX
1.29	AMESDK_SET_CUSTOM_PROPERTY_EX
1.30	AMESDK_OTHER_REFRESH_DISPLAY_WINDOW
1.31	AMESDK_OTHER_SNAPSHOT_BMP
1.32	AMESDK_OTHER_SNAPSHOT_JPG
1.33	AMESDK_OTHER_ZOOM
1.34	SC100#N4 (CIF) Software Programming Guide
1.35	SC100#N4 (VGA) Software Programming Guide
1.36	SC300#N8 Software Programming Guide



Exported Functions For Analog Video Capture Device	
1.37	SC300#D16 Software Programming Guide
1.38	SC300#Q32 Software Programming Guide
1.39	SC280#N4 (LIVE) Software Programming Guide
1.40	SC380#N16 (LIVE) Software Programming Guide

## 1.00 AMESDK\_CAPTURE\_DEVICE\_ENUMERATION

This function is used to enumerate all capture devices on the platform.

```

    BOOL AMESDK_CAPTURE_DEVICE_ENUMERATION(  ULONGLONG * * ppVideoDeviceList,
                                              ULONG *      pVideoDeviceSize,
                                              ULONGLONG * * ppVideoEncoderDeviceList,
                                              ULONG *      pVideoEncoderDeviceSize,
                                              ULONGLONG * * ppAudioDeviceList,
                                              ULONG *      pAudioDeviceSize,
                                              ULONGLONG * * ppAudioEncoderDeviceList,
                                              ULONG *      pAudioEncoderDeviceSize
    );

```

### Parameters:

Parameter	IN/OUT	Description
ppVideoDeviceList	OUT	<b>Video Device List.</b> Pointer to the address of all video capture devices.
pVideoDeviceSize	OUT	<b>Video Device Size.</b> Pointer to a variable that stores the quantity of video capture device.
ppVideoEncoderDeviceList	OUT	<b>Video Encoder Device List.</b> Pointer to the address of all video encoder devices.
pVideoEncoderDeviceSize	OUT	<b>Video Encoder Device Size.</b> Pointer to a variable that stores the quantity of video encoder device.
ppAudioDeviceList	OUT	<b>Audio Device List.</b> Pointer to the address of all audio capture devices.
pAudioDeviceSize	OUT	<b>Audio Device Size.</b> Pointer to a variable that stores the quantity of audio capture device.
ppAudioEncoderDeviceList	OUT	<b>Audio Encoder Device List.</b> Pointer to the address of all audio encoder devices.
pAudioEncoderDeviceSize	OUT	<b>Audio Encoder Device Size.</b> Pointer to a variable that stores the quantity of audio encoder device.

### Return Values:

BOOL. Here, every item of the returned list uses 8 bytes to describe the device's information. The lower 4 bytes are the device's serial number. The upper 4 bytes are driver's product id. For instance, SC300N8 could return the value, 0x0000680212ABF308, from this function.

### Support Devices:

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

## Examples:

EX1: To enumerate all capture devices on the platform.

```
ULONGLONG * p_video_device_list = NULL;
ULONGLONG * p_video_encoder_device_list = NULL;
ULONGLONG * p_audio_device_list = NULL;
ULONGLONG * p_audio_encoder_device_list = NULL;
ULONG      n_video_device_list_size = 0;
ULONG      n_video_encoder_device_list_size = 0;
ULONG      n_audio_device_list_size = 0;
ULONG      n_audio_encoder_device_list_size = 0;

AMESDK_CAPTURE_DEVICE_ENUMERATION( &p_video_device_list,
                                     &n_video_device_list_size,
                                     &p_video_encoder_device_list,
                                     &n_video_encoder_device_list_size,
                                     &p_audio_device_list,
                                     &n_audio_device_list_size,
                                     &p_audio_encoder_device_list,
                                     &n_audio_encoder_device_list_size );
```

## 1.01 AMESDK\_CREATE

The function helps you to open an analog video capture device and also allows you to attach a preview window or register a callback function on it. The callback function will offer you to obtain and to access the whole frame buffer. The data in frame buffer is live data (uncompressed data).

```

DEVICE_HANDLE AMESDK_CREATE( LPTSTR          pszDevName,
                             UINT           iDevNum,
                             ULONG          eDevType,
                             HWND           hDisplayWindow_A,
                             PF_BUFFER_CALLBACK pBufferCB_A,
                             PVOID          pUserData_A,
                             HWND           hDisplayWindow_B = NULL,
                             PF_BUFFER_CALLBACK pBufferCB_B   = NULL,
                             PVOID          pUserData_B       = NULL,
                             HWND           hDisplayWindow_C = NULL,
                             PF_BUFFER_CALLBACK pBufferCB_C   = NULL,
                             PVOID          pUserData_C       = NULL,
                             HWND           hDisplayWindow_D = NULL,
                             PF_BUFFER_CALLBACK pBufferCB_D   = NULL,
                             PVOID          pUserData_D       = NULL
);

typedef ULONG (DEVICE_HANDLE);

typedef BOOL (* PF_BUFFER_CALLBACK)( double  dSampleTime,
                                     BYTE *   pBuffer,
                                     ULONG    nBufferLen,
                                     BOOL     bIsKeyFrame,
                                     PVOID    pUserData
);

```

### Parameters:

Parameter	IN/OUT	Description
pszDevName	IN	<b>Device Name.</b> To give a device name that is used to create specific device. Currently, we support these device names below: "AH8400 PCI", "CX2581 PCI", "CX2588 PCI", "DC1150 USB", "FH8735 PCI", "TW2809 PCI", "TW5864 PCI", "TW6802 PCI", "SA7160 PCI", "SL6010 PCI".
iDevNum	IN	<b>Device Number.</b> If there are more than one device with the same device name on platform, you can use this parameter to recognize it.
eDevType	IN	<b>Device Type.</b> Always 0 for analog capture device.
hDisplayWindow_A hDisplayWindow_B hDisplayWindow_C hDisplayWindow_D	IN	<b>Display Window.</b> Pointer to one WHND window handle. If it isn't NULL, function will automatically display video on this window. If it is NULL, function will not display video on it. Parameters B, C and D are only used to support multiple channels function, if this device can offer more than one channel input, such as SC300D16 and SC300Q32.
pBufferCB_A pBufferCB_B pBufferCB_C	IN	<b>Callback Function.</b> Pointer to one callback function. If it is NULL, function will not return frame buffer to software caller. If it isn't NULL, caller will obtain

pBufferCB_D		frame buffer from callback when every frame is arrived. Parameters B, C and D are only used to support multiple channels function, if this device can offer more than one channel input.
pUserData_A pUserData_B pUserData_C pUserData_D	IN	<b>User Data.</b> Pointer to one data pointer. The parameters will be passed through callback. Parameters B, C and D are only used to support multiple channels function, if this device can offer more than one channel input.

## Return Value:

If AMESDK\_CREATE is successful, it will return one DEVICE\_HANDLE. If it is fail, it will return error code. As one of below:

0x80000000 - Parameter, pszDevName, is wrong.  
 0x80000001 - Unknown error.  
 0x80000002 - Device queue is full already.

## Supported Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

## Examples:

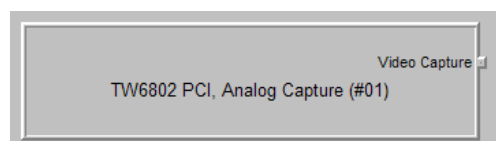
```
HWND hWnd = CreateWindowEx( ... );

BOOL bcb( double dSampleTime, BYTE * pBuffer, ULONG nBufferLen, BOOL bIsKeyFrame ,
          PVOID pUserData )
{
    ...

    return TRUE;
}
```

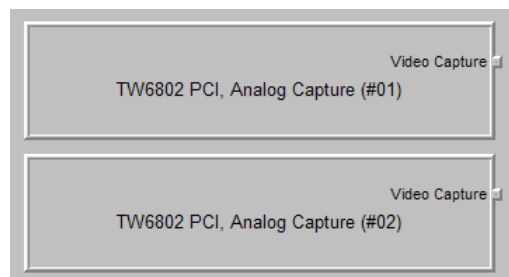
EX1: Don't need to display video and to get frame buffer from callback function.

```
hDev = AMESDK_CREATE( "TW6802 PCI", 0, 0, NULL, NULL, NULL );
```



EX2: If you have two devices on one PC, you can use parameter #2 to open 2nd device.

```
hDev0 = AMESDK_CREATE( "TW6802 PCI", 0, 0, NULL, NULL, NULL );
hDev1 = AMESDK_CREATE( "TW6802 PCI", 1, 0, NULL, NULL, NULL );
```



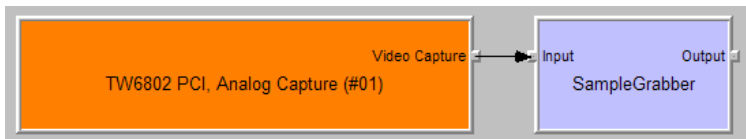
EX3: To display video on your attached window by SDK engine.

```
hDev = AMESDK_CREATE( "TW6802 PCI", 0, 0, hWnd, NULL, NULL );
```



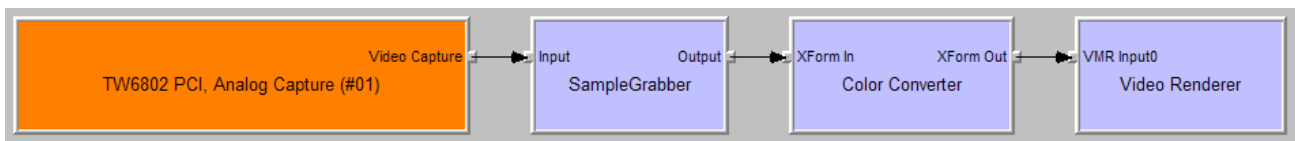
EX4: To register the callback function on the device.

```
hDev = AMESDK_CREATE( "TW6802 PCI", 0, 0, NULL, &bcb, this );
```



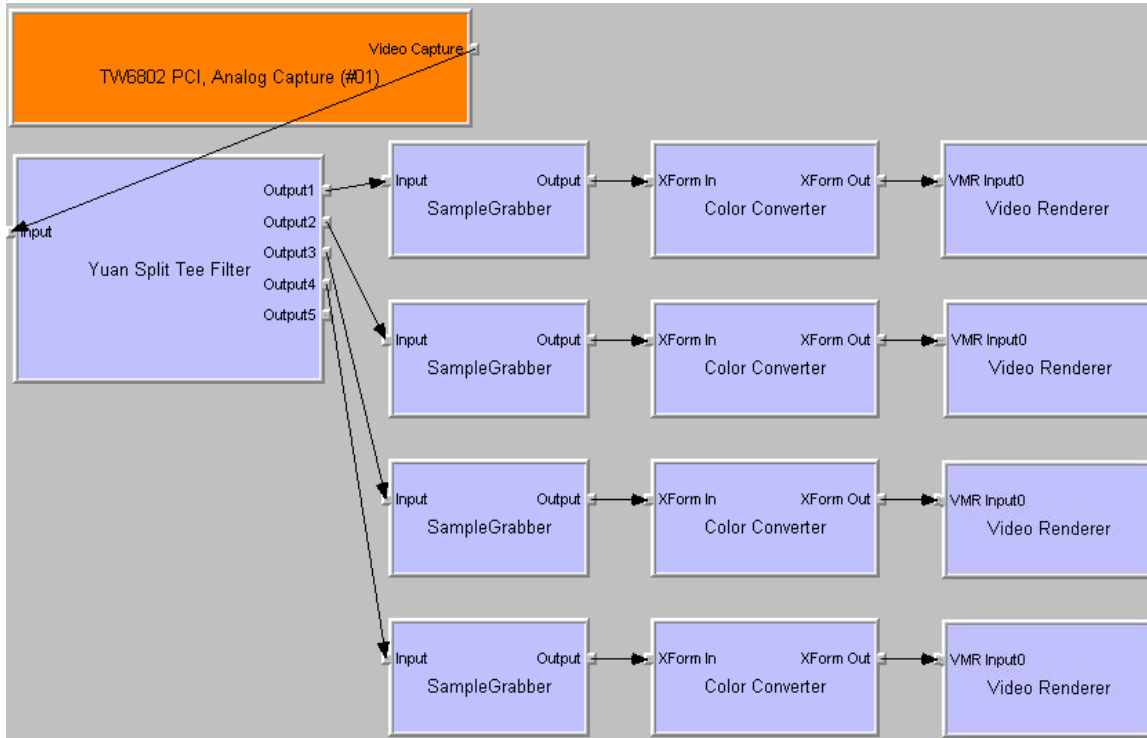
EX5: To display video on your attached window and register a callback function on the device.

```
hDev = AMESDK_CREATE( "TW6802 PCI", 0, 0, hWnd, &bcb, this );
```



EX6: To support multiple (4) channels on one device (for switching card only).

```
hDev = AMESDK_CREATE( "TW6802 PCI", 0, 0, hWnd_A, &bcB_A, NULL,
                    hWnd_B, &bcB_B, NULL,
                    hWnd_C, &bcB_C, NULL,
                    hWnd_D, &bcB_D, NULL
);
```



## Remarks:

The input string is corresponded to different product series as below:

```
"AH8400 PCI": SC290, SC390
"CX2581 PCI": SC310
"CX2588 PCI": SC340
"DC1150 USB": PD652, SC100
"FH8735 PCI": SC2A0, SC3A0, SC580
"TW2809 PCI": SC590
"TW5864 PCI": SC2B0, SC3B0
"TW6802 PCI": SC200, SC300, SC230, SC330
"SA7160 PCI": SC500, SC510
"SL6010 PCI": SC280, SC380
```

Please reference sections 1.34 ~ 1.40 to obtain more sample tutorials.

## 1.02 AMESDK\_DESTROY

To call this function will close your device and release all device resources. This function also will stop the device automatically, if it is running.

```
BOOL AMESDK_DESTROY( DEVICE_HANDLE hDevHandle );
```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device that is to be destroyed.

### Return Value:

BOOL

### Supported Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Examples:

EX1: Destroy and stop the device.

```
AMESDK_DESTROY( hDev );
```



## 1.03 AMESDK\_RUN

This function is used to change the status of device into running state.

```
BOOL AMESDK_RUN( DEVICE_HANDLE hDevHandle );
```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose status is to be changed to running status.

### Return Value:

BOOL

### Supported Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Examples:

EX1: Run the device.

```
AMESDK_RUN( hDev );
```

## 1.04 AMESDK\_STOP

This function is used to stop the device.

```
BOOL AMESDK_STOP( DEVICE_HANDLE hDevHandle );
```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device that is to be stopped.

### Return Value:

BOOL

### Supported Devices:

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Examples:

EX1: Stop the device.

```
AMESDK_STOP( hDev );
```

## 1.05 AMESDK\_AUTO\_STANDARD\_DETECTION

This function is used to detect video standard automatically. To use this function, you must make sure the video signal had already been connected with our capture card.

```

    BOOL AMESDK_EXPORT AMESDK_AUTO_STANDARD_DETECTION( DEVICE_HANDLE hDevHandle,
                                                         ULONG *          pStandard
                                                         );
    
```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose standard is to be retrieved.
pStandard	OUT	<b>Video Standard.</b> Pointer to a variable that stores the standard.

### Return Value:

BOOL

### Supported Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
       SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Examples:

EX1: Auto detect current video standard.

```

AMESDK_AUTO_STANDARD_DETECTION( hDev, &nStandard );
    
```

## 1.06 AMESDK\_GET\_STANDARD

This function is used to get current video standard.

```

    BOOL AMESDK_GET_STANDARD(  DEVICE_HANDLE  hDevHandle,
                               ULONG *        pStandard
    );

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose standard is to be retrieved.
pStandard	OUT	<b>Video Standard.</b> Pointer to a variable that stores the standard. It cannot be NULL.
		<b>SUPPORT STANDARD (60Hz):</b> KS_AnalogVideo_NTSC_M (0x00000001) KS_AnalogVideo_NTSC_M_J (0x00000002) KS_AnalogVideo_NTSC_433 (0x00000004) KS_AnalogVideo_PAL_M (0x00000200) KS_AnalogVideo_PAL_60 (0x00000800) <b>SUPPORT STANDARD (50Hz):</b> KS_AnalogVideo_PAL_B (0x00000010) KS_AnalogVideo_PAL_D (0x00000020) KS_AnalogVideo_PAL_G (0x00000040) KS_AnalogVideo_PAL_H (0x00000080) KS_AnalogVideo_PAL_I (0x00000100) KS_AnalogVideo_PAL_N (0x00000400) KS_AnalogVideo_PAL_N_COMBO (0x00100000) KS_AnalogVideo_SECAM_B (0x00001000) KS_AnalogVideo_SECAM_D (0x00002000) KS_AnalogVideo_SECAM_G (0x00004000) KS_AnalogVideo_SECAM_H (0x00008000) KS_AnalogVideo_SECAM_K (0x00010000) KS_AnalogVideo_SECAM_K1 (0x00020000) KS_AnalogVideo_SECAM_L (0x00040000) KS_AnalogVideo_SECAM_L1 (0x00080000)

### Return Value:

BOOL

### Support Devices:

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Examples:

EX1: Get the current video standard.

```
AMESDK_GET_STANDARD( hDev, &nStandard );
```

## 1.07 AMESDK\_SET\_STANDARD

This function is used to set/change video standard. It should be set or updated video standard before calling AMESDK\_RUN().

```

    BOOL AMESDK_SET_STANDARD( DEVICE_HANDLE  hDevHandle,
                               ULONG          nStandard
    );

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose standard is to be set/changed.
nStandard	IN	<b>Video Standard.</b> Specifies the standard.
		<b>SUPPORT STANDARD (60Hz):</b> KS_AnalogVideo_NTSC_M (0x00000001) KS_AnalogVideo_NTSC_M_J (0x00000002) KS_AnalogVideo_NTSC_433 (0x00000004) KS_AnalogVideo_PAL_M (0x00000200) KS_AnalogVideo_PAL_60 (0x00000800) <b>SUPPORT STANDARD (50Hz):</b> KS_AnalogVideo_PAL_B (0x00000010) KS_AnalogVideo_PAL_D (0x00000020) KS_AnalogVideo_PAL_G (0x00000040) KS_AnalogVideo_PAL_H (0x00000080) KS_AnalogVideo_PAL_I (0x00000100) KS_AnalogVideo_PAL_N (0x00000400) KS_AnalogVideo_PAL_N_COMBO (0x00100000) KS_AnalogVideo_SECAM_B (0x00001000) KS_AnalogVideo_SECAM_D (0x00002000) KS_AnalogVideo_SECAM_G (0x00004000) KS_AnalogVideo_SECAM_H (0x00008000) KS_AnalogVideo_SECAM_K (0x00010000) KS_AnalogVideo_SECAM_K1 (0x00020000) KS_AnalogVideo_SECAM_L (0x00040000) KS_AnalogVideo_SECAM_L1 (0x00080000)

### Return Value:

BOOL

### Support Devices:

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Examples:

EX1: Set the video standard to NTSC M.

```
AMESDK_SET_STANDARD( hDev, 0x00000001 ); // KS_AnalogVideo_NTSC_M
```

EX2: Set the video standard to PAL B.

```
AMESDK_SET_STANDARD( hDev, 0x00000010 ); // KS_AnalogVideo_PAL_B
```

## 1.08 AMESDK\_GET\_INPUT

This function is used to get current video input.

```

    BOOL AMESDK_GET_INPUT(  DEVICE_HANDLE  hDevHandle,
                           ULONG *         pInput
    );

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose input is to be retrieved.
pInput	OUT	<b>Video Input.</b> Pointer to a variable that stores the input. It cannot be NULL.
		<b>SUPPORT INPUTS:</b> Composite / HDMI      (0x00000000) SVideo / DVI-D      (0x00000001) Tuner / Components   (0x00000002) DVI-A                    (0x00000003) SDI                      (0x00000004)

### Return Value:

BOOL

### Support Devices:

PCTV: PD652  
 SECU: SC500, SC510, SC580, SC590

### Examples:

EX1: Get the current video input.

```
AMESDK_GET_INPUT( hDev, &nInput );
```

### Remarks:

For SC500, SC510, SC580, and SC590 series, please reference their Extra Programming Guide in SDK packet in detail.

## 1.09 AMESDK\_SET\_INPUT

This function is used to set/change video input.

```

    BOOL AMESDK_SET_INPUT(  DEVICE_HANDLE hDevHandle,
                           ULONG          nInput
    );

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose input is to be set/changed.
nInput	IN	<b>Video Input.</b> Specifies the input.
		<b>SUPPORT INPUTS:</b> Composite / HDMI      (0x00000000) SVideo / DVI-D      (0x00000001) Tuner / Components   (0x00000002) DVI-A                    (0x00000003) SDI                      (0x00000004)

### Return Value:

BOOL

### Support Devices:

PCTV: PD652  
 SECU: SC500, SC510, SC580, SC590

### Examples:

EX1: Set the video input to Composite.

```
AMESDK_SET_INPUT( hDev, 0x00000000 );
```

EX2: Set the video input to SVideo.

```
AMESDK_SET_INPUT( hDev, 0x00000001 );
```

EX3: Set the video input to Tuner.

```
AMESDK_SET_INPUT( hDev, 0x00000002 );
```

### Remarks:

For SC500, SC510, SC580, and SC590 series, please reference their Extra Programming Guide in SDK packet in detail.

### 1.10 AMESDK\_GET\_FORMAT

This function is used to get current video format.

```

    BOOL AMESDK_GET_FORMAT(  DEVICE_HANDLE    hDevHandle,
                              ULONG *         pColorSpace,
                              ULONG *         pWidth,
                              ULONG *         pHeight,
                              ULONG *         pBitCount,
                              DOUBLE *        pFrameRate
    );

```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose format is to be retrieved.
pColorSpace	OUT	<b>Video Color Space.</b> Pointer to a variable that stores the color space, in MAKEFOURCC. It cannot be NULL.
pWidth	OUT	<b>Video Width.</b> Pointer to a variable that stores the width, in pixels. It cannot be NULL.
pHeight	OUT	<b>Video Height.</b> Pointer to a variable that stores the height, in pixels. It cannot be NULL.
pBitCount	OUT	<b>Video BitCount.</b> Pointer to a variable that stores the bit count. It cannot be NULL.
pFrameRate	OUT	<b>Video FrameRate.</b> Pointer to a variable that stores the frame rate, in fps. It cannot be NULL.

#### Return Value:

BOOL

#### Support Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Examples:

EX1: Get the current video format (colorspace, width, height, bit count, and frame rate).

```
AMESDK_GET_FORMAT( hDev, &nColorSpace, &nWidth, &nHeight, &nBitCount, &dFrameRate );
```



Remarks: ●: STANDARD, ○: OPTION

FORMAT (NTSC)	PD652 (YUY2)	SC100 (YUY2)	SC200 SC230 SC300 SC330 (YUY2)	SC310 SC340 (YUY2)	SC280 SC380 (UYVY)	SC290 SC390 (UYVY)	SC2A0 SC3A0 (YV12)	SC2B0 SC3B0 (YV12)
720×480×16×30.00	●	○	●	●			●	●
704×480×16×30.00	●	○	●	●	●	●	●	●
640×480×16×30.00	●	●	●	●				
720×240×16×30.00/60.00	●	○	●	●			●	●
704×240×16×30.00/60.00	●	○	●	●			●	●
640×240×16×30.00/60.00	●	○	●	●				
360×240×16×30.00/60.00	●	○	●	●				
352×240×16×30.00/60.00	●	○	●	●	●	●	●	●
320×240×16×30.00/60.00	●	○	●	●				
256×240×16×30.00/60.00	●	●						
180×120×16×30.00/60.00	●	○						
176×120×16×30.00/60.00	●	○						
160×120×16×30.00/60.00	●	○						

FORMAT (PAL)	PD652 (YUY2)	SC100 (YUY2)	SC200 SC230 SC300 SC330 (YUY2)	SC310 SC340 (YUY2)	SC280 SC380 (UYVY)	SC290 SC390 (UYVY)	SC2A0 SC3A0 (YV12)	SC2B0 SC3B0 (YV12)
720×576×16×25.00	●	○	●	●			●	●
704×576×16×25.00	●	○	●	●	●	●	●	●
640×576×16×25.00	●	●	●	●				
720×288×16×25.00/50.00	●	○	●	●			●	●
704×288×16×25.00/50.00	●	○	●	●			●	●
640×288×16×25.00/50.00	●	○	●	●				
360×288×16×25.00/50.00	●	○	●	●				
352×288×16×25.00/50.00	●	○	●	●	●	●	●	●
320×288×16×25.00/50.00	●	○	●	●				
256×288×16×25.00/50.00	●	●						
180×144×16×25.00/50.00	●	○						
176×144×16×25.00/50.00	●	○						
160×144×16×25.00/50.00	●	○						

FORMAT	SC500 (YUY2)	SC510 (YUY2)	SC580 (YV12)	SC590 (YV12)
720×480i×60.00	●	●	●	●
720×576i×50.00	●	●	●	●
720×480p×60.00	●	●	●	●
720×576p×50.00	●	●	●	●
1280×720p×60.00	●	●	●	●
1280×720p×50.00	●	●	●	●
1280×720p×30.00	●	●	●	●
1280×720p×25.00	●	●	●	●
1280×720p×24.00	●	●	●	●
1920×1080i×60.00	●	●	●	●
1920×1080i×50.00	●	●	●	●
1920×1080p×60.00	●	●	●	●
1920×1080p×50.00	●	●	●	●
1920×1080p×30.00	●	●	●	●
1920×1080p×25.00	●	●	●	●
1920×1080p×24.00	●	●	●	●
640×384p×60.00	●	●	●	●
640×400P×60.00	●	●	●	●
640×480p×60.00	●	●	●	●
800×600p×60.00	●	●	●	●
1024×768p×60.00	●	●	●	●
1280×768p×60.00	●	●	●	●
1280×800p×60.00	●	●	●	●
1280×960p×60.00	●	●	●	●
1280×1024p×60.00	●	●	●	●
1360×768p×60.00	●	●	●	●
1440×900p×60.00	●	●	●	●
720×240p×60.00	●	●	●	●
720×288p×50.00	●	●	●	●

### 1.11 AMESDK\_SET\_FORMAT

This function is used to set/change video format. It should be set or updated video format before calling AMESDK\_RUN().

```

    BOOL AMESDK_SET_FORMAT(  DEVICE_HANDLE    hDevHandle,
                              ULONG           nColorSpace,
                              ULONG           nWidth,
                              ULONG           nHeight,
                              ULONG           nBitCount,
                              DOUBLE          dFrameRate
    );

```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose format is to be set/changed.
nColorSpace	IN	<b>Video ColorSpace.</b> Specifies the colorspace, in MAKEFOURCC.
nWidth	IN	<b>Video Width.</b> Specifies the width, in pixels.
nHeight	IN	<b>Video Height.</b> Specifies the height, in pixels.
nBitCount	IN	<b>Video BitCount.</b> Specifies the bit count.
dFrameRate	IN	<b>Video FrameRate.</b> Specifies the frame rate, in fps.

#### Return Value:

BOOL

#### Support Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Examples:

EX1: Set the video format, YUY2 × 720 × 480 × 16 bits × 30.00 fps.

```
AMESDK_SET_FORMAT( hDev, MAKEFOURCC('Y', 'U', 'Y', '2'), 720, 480, 16, 30.00 );
```

EX2: Set the video format, UYVY × 352 × 288 × 16 bits × 25.00 fps.

```
AMESDK_SET_FORMAT( hDev, MAKEFOURCC('U', 'Y', 'V', 'Y'), 352, 288, 16, 25.00 );
```

Remarks: ●: STANDARD, ○: OPTION

FORMAT (NTSC)	PD652 (YUY2)	SC100 (YUY2)	SC200 SC230 SC300 SC330 (YUY2)	SC310 SC340 (YUY2)	SC280 SC380 (UYVY)	SC290 SC390 (UYVY)	SC2A0 SC3A0 (YV12)	SC2B0 SC3B0 (YV12)
720×480×30.00	●	○	●	●			●	●
704×480×30.00	●	○	●	●	●	●	●	●
640×480×30.00	●	●	●	●				
720×240×30.00/60.00	●	○	●	●			●	●
704×240×30.00/60.00	●	○	●	●			●	●
640×240×30.00/60.00	●	○	●	●				
360×240×30.00/60.00	●	○	●	●				
352×240×30.00/60.00	●	○	●	●	●	●	●	●
320×240×30.00/60.00	●	○	●	●				
256×240×30.00/60.00	●	●						
180×120×30.00/60.00	●	○						
176×120×30.00/60.00	●	○						
160×120×30.00/60.00	●	○						

FORMAT (PAL)	PD652 (YUY2)	SC100 (YUY2)	SC200 SC230 SC300 SC330 (YUY2)	SC310 SC340 (YUY2)	SC280 SC380 (UYVY)	SC290 SC390 (UYVY)	SC2A0 SC3A0 (YV12)	SC2B0 SC3B0 (YV12)
720×576×25.00	●	○	●	●			●	●
704×576×25.00	●	○	●	●	●	●	●	●
640×576×25.00	●	●	●	●				
720×288×25.00/50.00	●	○	●	●			●	●
704×288×25.00/50.00	●	○	●	●			●	●
640×288×25.00/50.00	●	○	●	●				
360×288×25.00/50.00	●	○	●	●				
352×288×25.00/50.00	●	○	●	●	●	●	●	●
320×288×25.00/50.00	●	○	●	●				
256×288×25.00/50.00	●	●						
180×144×25.00/50.00	●	○						
176×144×25.00/50.00	●	○						
160×144×25.00/50.00	●	○						

FORMAT	SC500 (YUY2)	SC510 (YUY2)	SC580 (YV12)	SC590 (YV12)
720×480i×60.00	●	●	●	●
720×576i×50.00	●	●	●	●
720×480p×60.00	●	●	●	●
720×576p×50.00	●	●	●	●
1280×720p×60.00	●	●	●	●
1280×720p×50.00	●	●	●	●
1280×720p×30.00	●	●	●	●
1280×720p×25.00	●	●	●	●
1280×720p×24.00	●	●	●	●
1920×1080i×60.00	●	●	●	●
1920×1080i×50.00	●	●	●	●
1920×1080p×60.00	●	●	●	●
1920×1080p×50.00	●	●	●	●
1920×1080p×30.00	●	●	●	●
1920×1080p×25.00	●	●	●	●
1920×1080p×24.00	●	●	●	●
640×384p×60.00	●	●	●	●
640×400P×60.00	●	●	●	●
640×480p×60.00	●	●	●	●
800×600p×60.00	●	●	●	●
1024×768p×60.00	●	●	●	●
1280×768p×60.00	●	●	●	●
1280×800p×60.00	●	●	●	●
1280×960p×60.00	●	●	●	●
1280×1024p×60.00	●	●	●	●
1360×768p×60.00	●	●	●	●
1440×900p×60.00	●	●	●	●
720×240p×60.00	●	●	●	●
720×288p×50.00	●	●	●	●

## 1.12 AMESDK\_GET\_DEINTERLACE

This function is used to check whether the deinterlace function is enabled.

```

    BOOL AMESDK_GET_DEINTERLACE(  DEVICE_HANDLE    hDevHandle,
                                   ULONG *          pDeinterlace,
                                   ULONG            nSubChannelNumber = 0
    );

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be retrieved.
pDeinterlace	OUT	<b>Deinterlace Status.</b> Pointers to a variable that stores the status of deinterlace function. It cannot be NULL. The range of value is 0,1,7 (7 Algorithms). The value 0 is to disable deinterlace function. The 7 is the best quality algorithm. For all algorithms description, please contact with us directly. It will not be opened in this document.
nSubChannelNumber	IN	<b>Sub Channel Number.</b> The range of value is from 0 to 3.

### Return Values:

BOOL

### Support Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Examples:

EX1: Get the current status of deinterlace function.

```
AMESDK_GET_DEINTERLACE( hDev, &nDeinterlace );
```

### 1.13 AMESDK\_SET\_DEINTERLACE

This function is used to set/change deinterlace function.

```

    BOOL AMESDK_SET_DEINTERLACE( DEVICE_HANDLE    hDevHandle,
                                  ULONG             nDeinterlace,
                                  ULONG             nSubChannelNumber = 0
    );

```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be changed.
nDeinterlace	IN	<b>Deinterlace Status.</b> Specifies the method of deinterlace function. The range of value is 0, 1, 7 (7 Algorithms). The value 0 is to disable deinterlace function. For the algorithm description, please contact with us directly. It will not be opened in this document.
nSubChannelNumber	IN	<b>Sub Channel Number.</b> The range of value is from 0 to 3.

#### Return Values:

BOOL

#### Support Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Examples:

EX1: Enable the deinterlace function in the best quality method.

```

BOOL AMESDK_SET_DEINTERLACE( hDev, 7 );

```

EX2: Disable the deinterlace function.

```

BOOL AMESDK_SET_DEINTERLACE( hDev, 0 );

```

### 1.14 AMESDK\_GET\_MIRROR

This function is used to check whether the mirror function is enabled.

```
BOOL AMESDK_GET_MIRROR(  DEVICE_HANDLE    hDevHandle,
                          BOOL *          pHorizontalMirror,
                          BOOL *          pVerticalMirror,
                          ULONG           nSubChannelNumber = 0
);
```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be retrieved.
pHorizontalMirror	OUT	<b>Horizontal Mirror Status.</b> Pointers to a variable that stores the status of mirror function. It cannot be NULL.
pVerticalMirror	OUT	<b>Vertical Mirror Status.</b> Pointers to a variable that stores the status of mirror function. It cannot be NULL.
nSubChannelNumber	IN	<b>Sub Channel Number.</b> The range of value is from 0 to 3.

#### Return Values:

BOOL

#### Support Devices:

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Examples:

EX1: Get the current status of mirror function.

```
AMESDK_GET_MIRROR( hDev, &bHorizontalMirror, &bVerticalMirror );
```



### 1.15 AMESDK\_SET\_MIRROR

This function is used to set/change mirror function. When mirror function is enabled, the horizontal or vertical video frame is inverted on display window.

```

BOOL AMESDK_SET_MIRROR(  DEVICE_HANDLE    hDevHandle,
                          BOOL             bHorizontalMirror,
                          BOOL             bVerticalMirror,
                          ULONG            nSubChannelNumber = 0
);

```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be changed.
bHorizontalMirror	IN	<b>Horizontal Mirror Status.</b> Specifies the method of mirror function. The value FALSE is to disable mirror function.
bVerticalMirror	IN	<b>Vertical Mirror Status.</b> Specifies the method of mirror function. The value FALSE is to disable mirror function.
nSubChannelNumber	IN	<b>Sub Channel Number.</b> The range of value is from 0 to 3.

#### Return Values:

BOOL

#### Support Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Examples:

EX1: Enable the horizontal mirror function on display window.

```

BOOL AMESDK_SET_MIRROR( hDev, TRUE, FALSE );

```

EX2: Enable the horizontal and vertical mirror functions on display window.

```

BOOL AMESDK_SET_MIRROR( hDev, TRUE, TRUE );

```

## 1.16 AMESDK\_GET\_MODE

This function is used to get current analog tuner mode.

```

    BOOL AMESDK_GET_MODE(  DEVICE_HANDLE  hDevHandle,
                           ULONG *        pMode
    );

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose tuner mode is to be retrieved.
pMode	OUT	<b>Analog Tuner Mode.</b> Pointers to a variable that stores the analog tuner mode. It cannot be NULL.
		<b>SUPPORT MODES:</b> AMTUNER_MODE_DEFAULT      (0x00000000) AMTUNER_MODE_TV        (0x00000001) AMTUNER_MODE_FM_RADIO   (0x00000002) AMTUNER_MODE_AM_RADIO   (0x00000004)

### Return Value:

BOOL

### Supported Devices:

PCTV: PD652

### Examples:

EX1: Get the device mode.

```
AMESDK_GET_MODE( hDev, &nMode );
```

## 1.17 AMESDK\_SET\_MODE

This function is used to set/change analog tuner mode.

```

    BOOL AMESDK_SET_MODE(  DEVICE_HANDLE  hDevHandle,
                           ULONG          nMode
    );

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose analog tuner mode is to be set/changed.
nMode	IN	<b>Analog Tuner Mode.</b> Specifies the analog tuner mode.
		<b>SUPPORT MODES:</b> AMTUNER_MODE_DEFAULT      (0x00000000) AMTUNER_MODE_TV        (0x00000001) AMTUNER_MODE_FM_RADIO    (0x00000002) AMTUNER_MODE_AM_RADIO   (0x00000004)

### Return Value:

BOOL

### Supported Devices:

PCTV: PD652

### Examples:

EX1: Set the device mode to analog TV.

```
AMESDK_SET_MODE( hDev, AMTUNER_MODE_TV );
```

EX2: Set the device mode to FM radio.

```
AMESDK_SET_MODE( hDev, AMTUNER_MODE_FM_RADIO );
```

EX3: Set the device mode to AM radio.

```
AMESDK_SET_MODE( hDev, AMTUNER_MODE_AM_RADIO );
```

### 1.18 AMESDK\_GET\_FREQUENCY

This function is used to get current analog tuner frequency.

```
BOOL AMESDK_GET_FREQUENCY(  DEVICE_HANDLE  hDevHandle,
                             ULONG *        nFrequency
                             );
```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose frequency is to be retrieved.
nFrequency	OUT	<b>Analog Tuner Frequency.</b> Pointer to a variable that stores the analog tuner frequency, in Hz. It cannot be NULL.

#### Return Values:

BOOL

#### Support Devices:

PCTV: PD652

#### Examples:

EX1: Get the current analog tuner frequency.

```
AMESDK_GET_FREQUENCY( hDev, &nFrequency );
```

### 1.19 AMESDK\_SET\_FREQUENCY

This function is used to set/change analog tuner frequency.

```
BOOL AMESDK_SET_FREQUENCY(  DEVICE_HANDLE  hDevHandle ,
                             ULONG          nFrequency
                             );
```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose analog tuner frequency is to be set/changed.
nFrequency	IN	<b>Analog Tuner Frequency.</b> Specifies the analog tuner frequency, in Hz.

#### Return Values:

BOOL

#### Support Devices:

PCTV: PD652

#### Examples:

EX1: Set the analog tuner to different frequencies.

```
AMESDK_SET_FREQUENCY( hDev, 473143000 );
```

```
AMESDK_SET_FREQUENCY( hDev, 545000000 );
```

```
AMESDK_SET_FREQUENCY( hDev, 666000000 );
```

## 1.20 AMESDK\_GET\_LOCK

This function is used to determine whether the signal is locked.

```

    BOOL AMESDK_GET_LOCK(  DEVICE_HANDLE  hDevHandle,
                           ULONG *        nLock
    );

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose signal status is to be retrieved.
nLock	OUT	<b>Signal Status.</b> Pointer to a variable that stores the signal status (LOCK:1/UNLOCK:0). It cannot be NULL. The status of each channel is represented by one bit of nLock. Refer to extra programming guide for details.

### Return Values:

BOOL

### Support Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Examples:

EX1: Get the current signal status.

```
AMESDK_GET_LOCK( hDev, &nLock );
```

### Remarks:

For more AMESDK\_GET\_LOCK information, please reference product's Extra Programming Guide in SDK packet. For example, SC300 user should reference this document and "SC200 & SC300 Extra Programming Guide" document both at the same time.

## 1.21 AMESDK\_GET\_FPS

This function is used to determine current fps (frames per second).

```

    BOOL AMESDK_GET_FPS( DEVICE_HANDLE    hDevHandle,
                        ULONG *           pFps,
                        ULONG             nSubChannelNumber = 0
    );

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose fps info is to be retrieved.
pFps	OUT	<b>Frames per Second.</b> Pointer to a variable that stores the fps per channel.
nSubChannelNumber	IN	<b>Sub Channel Number.</b> The range of value is from 0 to 3.

### Return Values:

BOOL

### Support Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Examples:

EX1: Get the current fps.

```
AMESDK_GET_FPS( hDev, &nFps );
```

## 1.22 AMESDK\_GET\_CAMERACONTROL\_PROPERTY

This function is used to get current Microsoft's Camera Control properties.

```

BOOL AMESDK_GET_CAMERACONTROL_PROPERTY(  DEVICE_HANDLE    hDevHandle,
                                           ULONG             nProperty,
                                           ULONG *          pValue,
                                           ULONG             nSubChannelNumber = 0
);

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be retrieved.
nProperty	IN	<b>Property.</b> Specifies the property that will be gotten from the device. As one of below: <b>SUPPORT PROPERTIES:</b> CameraControl_Pan (0x00000000) CameraControl_Tilt (0x00000001) CameraControl_Roll (0x00000002) CameraControl_Zoom (0x00000003) CameraControl_Exposure (0x00000004) CameraControl_Iris (0x00000005) CameraControl_Focus (0x00000006)
pValue	OUT	<b>Property Value.</b> Pointer to a variable that stores the specify property value. It cannot be NULL. The range of value is from 0 to 255.
nSubChannelNumber	IN	<b>Sub Channel Number.</b> The range of value is from 0 to 3.

### Return Values:

BOOL

### Support Devices:

None

### Examples:

EX1: Get the current CameraControl Zoom property.

```
AMESDK_GET_CAMERACONTROL_PROPERTY( hDev, 0x00000003, &nZoom );
```

EX2: Get the current CameraControl Focus property.

```
AMESDK_GET_CAMERACONTROL_PROPERTY( hDev, 0x00000006, &nFocus );
```



### 1.23 AMESDK\_SET\_CAMERACONTROL\_PROPERTY

This function is used to set/change Microsoft's CameraControl property.

```

BOOL AMESDK_SET_CAMERACONTROL_PROPERTY(  DEVICE_HANDLE    hDevHandle,
                                           ULONG             nProperty,
                                           ULONG             nValue,
                                           ULONG             nSubChannelNumber = 0
);

```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be changed.
nProperty	IN	<b>Property.</b> Specifies the property that is to set to the device. As one of below: <b>SUPPORT PROPERTIES:</b> CameraControl_Pan               (0x00000000) CameraControl_Tilt           (0x00000001) CameraControl_Roll           (0x00000002) CameraControl_Zoom           (0x00000003) CameraControl_Exposure       (0x00000004) CameraControl_Iris           (0x00000005) CameraControl_Focus          (0x00000006)
nValue	IN	<b>Property Value.</b> Specifies the property value. The range of value is from 0 to 255.
nSubChannelNumber	IN	<b>Sub Channel Number.</b> The range of value is from 0 to 3.

#### Return Values:

BOOL

#### Support Devices:

None

#### Examples:

EX1: Set the CameraControl Zoom property.

```
AMESDK_SET_CAMERACONTROL_PROPERTY( hDev, 0x00000003, nZoom );
```

EX2: Set the CameraControl Focus property.

```
AMESDK_SET_CAMERACONTROL_PROPERTY( hDev, 0x00000006, nFocus );
```

### 1.24 AMESDK\_GET\_VIDEOPROCAMP\_PROPERTY

This function is used to get current Microsoft's VideoProcAmp properties.

```

BOOL AMESDK_GET_VIDEOPROCAMP_PROPERTY( DEVICE_HANDLE hDevHandle,
                                         ULONG          nProperty,
                                         ULONG *        pValue,
                                         ULONG          nSubChannelNumber = 0
);

```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be retrieved.
nProperty	IN	<b>Property.</b> Specifies the property that will be gotten from the device. As one of below: <b>SUPPORT PROPERTIES:</b> VideoProcAmp_Brightness (0x00000000) VideoProcAmp_Contrast (0x00000001) VideoProcAmp_Hue (0x00000002) VideoProcAmp_Saturation (0x00000003) VideoProcAmp_Sharpness (0x00000004) VideoProcAmp_Gamma (0x00000005) VideoProcAmp_ColorEnable (0x00000006) VideoProcAmp_WhiteBalance (0x00000007) VideoProcAmp_BacklightCompensation (0x00000008) VideoProcAmp_Gain (0x00000009)
pValue	OUT	<b>Property Value.</b> Pointer to a variable that stores the specify property value. It cannot be NULL. The range of value is from 0 to 255.
nSubChannelNumber	IN	<b>Sub Channel Number.</b> The range of value is from 0 to 3.

#### Return Values:

BOOL

#### Support Devices:

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Examples:

EX1: Get the current VideoProcAmp Brightness property.

```
AMESDK_GET_VIDEOPROCAMP_PROPERTY( hDev, 0x00000000, &nBrightness );
```

EX2: Get the current VideoProcAmp Hue property.

```
AMESDK_GET_VIDEOPROCAMP_PROPERTY( hDev, 0x00000002, &nHue );
```

### 1.25 AMESDK\_SET\_VIDEOPROCAMP\_PROPERTY

This function is used to set/change Microsoft's VideoProcAmp property.

```

BOOL AMESDK_SET_VIDEOPROCAMP_PROPERTY(  DEVICE_HANDLE  hDevHandle,
                                         ULONG           nProperty,
                                         ULONG           nValue,
                                         ULONG           nSubChannelNumber = 0
);

```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be changed.
nProperty	IN	<b>Property.</b> Specifies the property that will be set to the device. As one of below: <b>SUPPORT PROPERTIES:</b> VideoProcAmp_Brightness (0x00000000) VideoProcAmp_Contrast (0x00000001) VideoProcAmp_Hue (0x00000002) VideoProcAmp_Saturation (0x00000003) VideoProcAmp_Sharpness (0x00000004) VideoProcAmp_Gamma (0x00000005) VideoProcAmp_ColorEnable (0x00000006) VideoProcAmp_WhiteBalance (0x00000007) VideoProcAmp_BacklightCompensation (0x00000008) VideoProcAmp_Gain (0x00000009)
nValue	IN	<b>Property Value.</b> Specifies the property value. The range of value is from 0 to 255.
nSubChannelNumber	IN	<b>Sub Channel Number.</b> The range of value is from 0 to 3.

#### Return Values:

BOOL

#### Support Devices:

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Examples:

EX1: Set the VideoProcAmp Brightness property.

```
AMESDK_SET_VIDEOPROCAMP_PROPERTY( hDev, 0x00000000, nBrightness );
```

EX2: Set the VideoProcAmp Contrast property.

```
AMESDK_SET_VIDEOPROCAMP_PROPERTY( hDev, 0x00000001, nContrast );
```

EX3: Set the VideoProcAmp Hue property.

```
AMESDK_SET_VIDEOPROCAMP_PROPERTY( hDev, 0x00000002, nHue );
```

## 1.26 AMESDK\_GET\_CUSTOM\_PROPERTY

This function is used to get some custom device properties.

```

    BOOL AMESDK_GET_CUSTOM_PROPERTY( DEVICE_HANDLE    hDevHandle,
                                     ULONG             nProperty,
                                     ULONG *           pValue
    );

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be retrieved.
nProperty	IN	<b>Property.</b> Specifies the property that will be gotten from the device. As one of below: <b>SUPPORT PROPERTIES:</b> CUSTOM_GET_DEVICE_SERIAL_NUMBER (000) CUSTOM_XET_PROCESSOR_TECHNOLOGY (001) CUSTOM_GET_REMOTE_CONTROL_CODE (004) CUSTOM_XET_ANALOG_VIDEO_INPUT (201) CUSTOM_GET_ANALOG_VIDEO_MACROVISION (202) CUSTOM_XET_ANALOG_VIDEO_AGC_LEVEL (204) CUSTOM_XET_ANALOG_VIDEO_SWITCH_SPEED (205) CUSTOM_XET_ANALOG_VIDEO_SWITCH_CHANNEL_TABLE (206) CUSTOM_XET_ANALOG_VIDEO_SWITCH_RESOLUTION_TABLE (207) CUSTOM_XET_ANALOG_VIDEO_SCALE_OUTPUT (209) CUSTOM_XET_ANALOG_VIDEO_FRAME_RATE (208) CUSTOM_XET_ANALOG_VIDEO_RESOLUTION (210) CUSTOM_XET_ANALOG_VIDEO_DEMISE_TYPE (212) CUSTOM_XET_ANALOG_VIDEO_QUEUE_BUFFER_SIZE (216) CUSTOM_XET_ANALOG_VIDEO_DENOISE_TYPE (217) CUSTOM_XET_ANALOG_AUDIO_STEREO_SYSTEM (250) CUSTOM_XET_ANALOG_AUDIO_VOLUME (251) CUSTOM_XET_ANALOG_PIN_TOPOLOGY (252) CUSTOM_XET_ANALOG_SAMPLE_FREQUENCY (253) CUSTOM_GET_DIGITAL_BANDWIDTH (300) CUSTOM_GET_DIGITAL_FREQUENCY (301) CUSTOM_GET_DIGITAL_SNR (302) CUSTOM_GET_DIGITAL_BER (303) CUSTOM_GET_DIGITAL_PER (304) CUSTOM_XET_PREVIEW_VIDEO_RESOLUTION (350) CUSTOM_XET_GPIO_DIRECTION (940) CUSTOM_XET_GPIO_DATA (941)
pValue	OUT	<b>Property Value.</b> Pointer to a variable that stores the specify property value. It cannot be NULL. The range of value is dependent on its property.

### Return Values:

BOOL

### Support Devices:

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

**Examples:**

EX1: Get an unique device serial number. (0x00000000 ~ 0xFFFFFFFF)

```
AMESDK_GET_CUSTOM_PROPERTY( hDev, 0, &nSerialNumber );
```

EX2: Get the latest remote control code. (0x00 ~ 0xFF)

```
AMESDK_GET_CUSTOM_PROPERTY( hDev, 4, &IrCode );
```

**Remarks:**

For more custom property programming, please reference product's Extra Programming Guide in SDK packet. For example, SC300 user should reference this document and "SC200 & SC300 Extra Programming Guide" document both at the same time.

## 1.27 AMESDK\_SET\_CUSTOM\_PROPERTY

This function is used to set/change some custom device properties.

```

    BOOL AMESDK_SET_CUSTOM_PROPERTY( DEVICE_HANDLE    hDevHandle,
                                     ULONG             nProperty,
                                     ULONG             nValue
    );

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be changed.
nProperty	IN	<b>Property.</b> Specifies the property that will be set to the device. As one of below: SUPPORT PROPERTIES: CUSTOM_XET_PROCESSOR_TECHNOLOGY (001) CUSTOM_XET_ANALOG_VIDEO_INPUT (201) CUSTOM_XET_ANALOG_VIDEO_AGC_LEVEL (204) CUSTOM_XET_ANALOG_VIDEO_SWITCH_SPEED (205) CUSTOM_XET_ANALOG_VIDEO_SWITCH_CHANNEL_TABLE (206) CUSTOM_XET_ANALOG_VIDEO_SWITCH_RESOLUTION_TABLE (207) CUSTOM_XET_ANALOG_VIDEO_SCALE_OUTPUT (209) CUSTOM_XET_ANALOG_VIDEO_FRAME_RATE (208) CUSTOM_XET_ANALOG_VIDEO_RESOLUTION (210) CUSTOM_XET_ANALOG_VIDEO_DEMISE_TYPE (212) CUSTOM_XET_ANALOG_VIDEO_QUEUE_BUFFER_SIZE (216) CUSTOM_XET_ANALOG_VIDEO_DENOISE_TYPE (217) CUSTOM_XET_ANALOG_AUDIO_STEREO_SYSTEM (250) CUSTOM_XET_ANALOG_AUDIO_VOLUME (251) CUSTOM_XET_ANALOG_PIN_TOPOLOGY (252) CUSTOM_XET_ANALOG_SAMPLE_FREQUENCY (253) CUSTOM_XET_PREVIEW_VIDEO_RESOLUTION (350) CUSTOM_SET_OSD_LINE (920) CUSTOM_SET_OSD_COLOR (929) CUSTOM_SET_SOFTWARE_WATCHDOG_RESET (930) CUSTOM_SET_SOFTWARE_WATCHDOG_DURATION (931) CUSTOM_XET_GPIO_DIRECTION (940) CUSTOM_XET_GPIO_DATA (941)
nValue	IN	<b>Property Value.</b> Specifies the property value. The range of value is dependent on its property.

### Return Values:

BOOL

### Support Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Examples:

EX1: Set the hardware video deinterlace type in driver.

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

EX2: Set the video input of device.

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

EX3: Set the AGC value. (0x000 ~ 0x1FF)

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 204, nAGC );
```

EX4: Enable and reset the software watchdog circuit. (0x00 ~ 0x01)

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 930, 0x01 );
```

EX5: Set the switching speed of algorithm. (0x00 ~ 0x01)

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 205, 0 /*SLOW*/ );
```

```
AMESDK_SET_CUSTOM_PROPERTY( hDev, 205, 1 /*FAST*/ );
```

**Remark:**

For more custom property programming, please reference product's Extra Prgoramming Guide in SDK packet. For example, SC300 user should reference this document and "SC200 & SC300 Extra Programming Guide" document both at the same time.

## 1.28 AMESDK\_GET\_CUSTOM\_PROPERTY\_EX

This function is used to get some custom device properties.

```

    BOOL AMESDK_GET_CUSTOM_PROPERTY_EX(  DEVICE_HANDLE    hDevHandle,
                                          ULONG              nProperty,
                                          BYTE *             pValue,
                                          ULONG              nBytes
    );

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be retrieved.
nProperty	IN	<b>Property.</b> Specifies the property that will be gotten from the device. As one of below: <b>SUPPORT PROPERTIES:</b> CUSTOM_XET_ANALOG_VIDEO_SWITCH_TABLE           (206) (12 BYTES) CUSTOM_XET_DIGITAL_ENCRYPT_KEY                   (310) (16 BYTES)
pValue	OUT	<b>Property Value.</b> Pointer to a variable that stores the specify property value. It cannot be NULL. The range of value is dependent on its property.
nBytes	IN	<b>Property Size.</b> Specifies the size of property value.

### Return Values:

BOOL

### Support Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
       SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Examples:

EX1: Get current switch table. (12 BYTES)

```
AMESDK_GET_CUSTOM_PROPERTY_EX( hDev, 206, pSwitchTable, 12 );
```

### Remark:

For more custom property programming, please reference product's Extra Prgoramming Guide in SDK packet. For example, SC300 user should reference this document and "SC200 & SC300 Extra Programming Guide" document both at the same time.



## 1.29 AMESDK\_SET\_CUSTOM\_PROPERTY\_EX

This function is used to set/change some custom device properties.

```

    BOOL AMESDK_SET_CUSTOM_PROPERTY_EX(  DEVICE_HANDLE    hDevHandle,
                                          ULONG              nProperty,
                                          BYTE *             pValue,
                                          ULONG              nBytes
    );

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be changed.
nProperty	IN	<b>Property.</b> Specifies the property that will be set to the device. As one of below: <b>SUPPORT PROPERTIES:</b> CUSTOM_XET_ANALOG_VIDEO_SWITCH_TABLE      (206) (12 BYTES) CUSTOM_XET_DIGITAL_ENCRYPT_KEY              (310) (16 BYTES) CUSTOM_SET_OSD_TEXT_STRING                  (921) (64 BYTES)
pValue	IN	<b>Property Value.</b> Specifies the property value. The range of value is dependent on its property.
nBytes	IN	<b>Property Size.</b> Specifies the size of property value.

### Return Values:

BOOL

### Support Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
       SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Examples:

EX1: Setup the switch table. (12 BYTES)

```
AMESDK_SET_CUSTOM_PROPERTY_EX( hDev, 206, pSwitchTable, 12 );
```

### Remark:

For more custom property programming, please reference product's Extra Prgoramming Guide in SDK packet. For example, SC300 user should reference this document and "SC200 & SC300 Extra Programming Guide" document both at the same time.

### 1.30 AMESDK\_OTHER\_REFRESH\_DISPLAY\_WINDOW

This function is used to refresh display window size to fit the attached window size. If the size of attached window, which is used as the HWND parameter in AMESDK\_CREATE() function, is changed by your software, you should call this function right away.

```
BOOL AMESDK_OTHER_REFRESH_DISPLAY_WINDOW( DEVICE_HANDLE  hDevHandle,
                                           ULONG          nSubChannelNumber = 0
                                           );
```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose display window size is to be changed.
nSubChannelNumber	IN	<b>Sub Channel Number.</b> The range of value is from 0 to 3.

#### Return Values:

BOOL

#### Support Devices:

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Examples:

EX1: Refresh display video size after changing size of attached window.

```
hDev = AMESDK_CREATE( "TW6802 PCI", 0, 0, hWnd, NULL, NULL );
```

```
...
```

```
MoveWindow( hWnd, 0, 0, 1024, 768, NULL );
```

```
AMESDK_OTHER_REFRESH_DISPLAY_WINDOW( hDev );
```

### 1.31 AMESDK\_OTHER\_SNAPSHOT\_BMP

This function is used to snapshot a BITMAP picture from the device to disk.

```
BOOL AMESDK_OTHER_SNAPSHOT_BMP(  DEVICE_HANDLE    hDevHandle,  
                                CHAR *              pszFilePathName,  
                                ULONG               nSubChannelNumber = 0  
);
```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be retrieved.
pszFilePathName	IN	<b>Property.</b> Specifies the file path that is used to save the picture.
nSubChannelNumber	IN	<b>Sub Channel Number.</b> The range of value is from 0 to 3.

#### Return Values:

BOOL

#### Support Devices:

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
      SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Examples:

EX1: Snapshot one BITMAP picture from device.

```
AMESDK_OTHER_SHAPSHOT_BMP( hDev, "C:\\SC\\200805201.BMP" );
```

### 1.32 AMESDK\_OTHER\_SNAPSHOT\_JPG

This function is used to snapshot a JPEG picture from the device to disk.

```
BOOL AMESDK_OTHER_SNAPSHOT_JPG(  DEVICE_HANDLE    hDevHandle,
                                  CHAR *              pszFilePathName,
                                  ULONG                nQuality = 80,
                                  ULONG                nSubChannelNumber = 0
                                  );
```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be retrieved.
pszFilePathName	IN	<b>Property.</b> Specifies the file path that is used to save the picture.
nQuality	IN	<b>Quality.</b> Specifies the compressed quality. The range of value is from 0 to 255.
nSubChannelNumber	IN	<b>Sub Channel Number.</b> The range of value is from 0 to 3.

#### Return Values:

BOOL

#### Support Devices:

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Examples:

EX1: Snapshot one JPEG picture from device in the best quality.

```
AMESDK_OTHER_SNAPSHOT_JPG( hDev, " C:\\SC\\200805201.JPG", 100 );
```

### 1.33 AMESDK\_OTHER\_ZOOM

This function is used to zoom in/out video.

```

    BOOL AMESDK_OTHER_ZOOM( DEVICE_HANDLE    hDevHandle,
                           ULONG             nLeftPos = 0,
                           ULONG             nTopPos = 0,
                           double            dRate = 1.0,
                           ULONG             nSubChannelNumber = 0
    );

```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be changed.
nLeftPos	IN	<b>Left Position.</b> Left position of zooming
nTopPos	IN	<b>Top Position.</b> Top position of zooming
dRate	IN	<b>Zoom Rate.</b> Zooming rate 1.0 ~ 10.0
nSubChannelNumber	IN	<b>Sub Channel Number.</b> The range of value is from 0 to 3.

#### Return Values:

BOOL

#### Support Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Examples:

EX1: Set zoom rate to 1.0.

```
AMESDK_OTHER_ZOOM( hDev, 0, 0, 1.0 );
```

## 1.34 SC100#N4 (CIF) Software Programming Guide

### REFERENCE DEVICES: SC100#N4 (CIF)

```

DEVICE_HANDLE hDev[ 4 ]; // USING 4 DEVICES TO SUPPORT 4 CHANNELS

BOOL HwInitialize( ... )
{
    // CREATE DEVICES
    //
    hDev[ 0 ] = AMESDK_CREATE( "DC1150 USB", 0, 0, hWnd0, &bc0, NULL ); // CH00
    hDev[ 1 ] = AMESDK_CREATE( "DC1150 USB", 1, 0, hWnd1, &bc1, NULL ); // CH01
    hDev[ 2 ] = AMESDK_CREATE( "DC1150 USB", 2, 0, hWnd2, &bc2, NULL ); // CH02
    hDev[ 3 ] = AMESDK_CREATE( "DC1150 USB", 3, 0, hWnd3, &bc3, NULL ); // CH03

    if( (hDev[ 0 ] & 0x80000000) ||
        (hDev[ 1 ] & 0x80000000) ||
        (hDev[ 2 ] & 0x80000000) ||
        (hDev[ 3 ] & 0x80000000) ) { // CHECK ERROR FLAG

        HwUninitialize();

        return FALSE;
    }

    AMESDK_SET_STANDARD( hDev[ 0 ], KS_AnalogVideo_NTSC_M ); // SET STANDARD
    0 ... 3
    AMESDK_SET_STANDARD( hDev[ 3 ], KS_AnalogVideo_NTSC_M );

    AMESDK_SET_FORMAT( hDev[ 0 ], 256, 240, 16, 30.00 ); // SET FORMAT
    0 ... 3
    AMESDK_SET_FORMAT( hDev[ 3 ], 256, 240, 16, 30.00 );

    AMESDK_RUN( hDev[ 0 ] ); // RUN DEVICES
    0 ... 3
    AMESDK_RUN( hDev[ 3 ] );
}

BOOL HwUninitialize( ... )
{
    AMESDK_DESTROY( hDev[ 0 ] ); // STOP & CLOSE DEVICES
    AMESDK_DESTROY( hDev[ 1 ] );
    AMESDK_DESTROY( hDev[ 2 ] );
    AMESDK_DESTROY( hDev[ 3 ] );
}

```

## 1.35 SC100#N4 (VGA) Software Programming Guide

### REFERENCE DEVICES: SC100#N4 (VGA)

```

DEVICE_HANDLE hDev[ 2 ]; // USING 2 DEVICES TO SUPPORT 4 CHANNELS

BOOL HwInitialize( ... )
{
    // CREATE DEVICES
    //
    hDev[ 0 ] = AMESDK_CREATE( "DC1150 USB", 0, 0, hWnd00, &bc00, NULL, // CH00
                               hWnd01, &bc01, NULL, // CH01
    );
    hDev[ 1 ] = AMESDK_CREATE( "DC1150 USB", 2, 0, hWnd02, &bc02, NULL, // CH02
                               hWnd03, &bc03, NULL, // CH03
    );
    if( (hDev[ 0 ] & 0x80000000) ||
        (hDev[ 1 ] & 0x80000000) ) { // CHECK ERROR FLAG

        HwUninitialize();

        return FALSE;
    }

    AMESDK_SET_STANDARD( hDev[ 0 ], KS_AnalogVideo_NTSC_M ); // SET STANDARD
    AMESDK_SET_STANDARD( hDev[ 1 ], KS_AnalogVideo_NTSC_M );

    AMESDK_SET_FORMAT( hDev[ 0 ], 640, 480, 16, 30.00 ); // SET FORMAT
    AMESDK_SET_FORMAT( hDev[ 1 ], 640, 480, 16, 30.00 );

    AMESDK_RUN( hDev[ 0 ] ); // RUN DEVICES
    AMESDK_RUN( hDev[ 1 ] );
}

BOOL HwUninitialize( ... )
{
    AMESDK_DESTROY( hDev[ 0 ] ); // STOP & CLOSE DEVICES
    AMESDK_DESTROY( hDev[ 1 ] );
}

```

## 1.36 SC300#N8 Software Programming Guide

**REFERENCE DEVICES: SC300#N4/#N8**

DEVICE\_HANDLE hDev[ 8 ]; // USING 8 DEVICES TO SUPPORT 8 CHANNELS

```

BOOL HwInitialize( ... )
{
    // CREATE DEVICES
    //
    hDev[ 0 ] = AMESDK_CREATE( "TW6802 PCI", 0, 0, hWnd0, &bcb0, NULL ); // CH00
    hDev[ 1 ] = AMESDK_CREATE( "TW6802 PCI", 1, 0, hWnd1, &bcb1, NULL ); // CH01
    hDev[ 2 ] = AMESDK_CREATE( "TW6802 PCI", 2, 0, hWnd2, &bcb2, NULL ); // CH02
    hDev[ 3 ] = AMESDK_CREATE( "TW6802 PCI", 3, 0, hWnd3, &bcb3, NULL ); // CH03
    hDev[ 4 ] = AMESDK_CREATE( "TW6802 PCI", 4, 0, hWnd4, &bcb4, NULL ); // CH04
    hDev[ 5 ] = AMESDK_CREATE( "TW6802 PCI", 5, 0, hWnd5, &bcb5, NULL ); // CH05
    hDev[ 6 ] = AMESDK_CREATE( "TW6802 PCI", 6, 0, hWnd6, &bcb6, NULL ); // CH06
    hDev[ 7 ] = AMESDK_CREATE( "TW6802 PCI", 7, 0, hWnd7, &bcb7, NULL ); // CH07

    if( (hDev[ 0 ] & 0x80000000) ||
        (hDev[ 1 ] & 0x80000000) ||
        (hDev[ 2 ] & 0x80000000) ||
        (hDev[ 3 ] & 0x80000000) ||
        (hDev[ 4 ] & 0x80000000) ||
        (hDev[ 5 ] & 0x80000000) ||
        (hDev[ 6 ] & 0x80000000) ||
        (hDev[ 7 ] & 0x80000000) ) { // CHECK ERROR FLAG

        HwUninitialize();

        return FALSE;
    }

    AMESDK_SET_STANDARD( hDev[ 0 ], KS_AnalogVideo_NTSC_M ); // SET STANDARD
    0 ... 7
    AMESDK_SET_STANDARD( hDev[ 7 ], KS_AnalogVideo_NTSC_M );

    AMESDK_SET_FORMAT( hDev[ 0 ], 720, 480, 16, 30.00 ); // SET FORMAT
    0 ... 7
    AMESDK_SET_FORMAT( hDev[ 7 ], 720, 480, 16, 30.00 );

    AMESDK_RUN( hDev[ 0 ] ); // RUN DEVICES
    0 ... 7
    AMESDK_RUN( hDev[ 7 ] );
}

BOOL HwUninitialize( ... )
{
    AMESDK_DESTROY( hDev[ 0 ] ); // STOP & CLOSE DEVICES
    AMESDK_DESTROY( hDev[ 1 ] );
    AMESDK_DESTROY( hDev[ 2 ] );
    AMESDK_DESTROY( hDev[ 3 ] );
    AMESDK_DESTROY( hDev[ 4 ] );
    AMESDK_DESTROY( hDev[ 5 ] );
    AMESDK_DESTROY( hDev[ 6 ] );
    AMESDK_DESTROY( hDev[ 7 ] );
}

```



## 1.37 SC300#Q16 Software Programming Guide

REFERENCE DEVICES: SC200#Q4, SC300#Q4/#Q8/#Q16/#Q32

```

DEVICE_HANDLE hDev[ 4 ]; // USING 4 DEVICES TO SUPPORT 16 CHANNELS

BOOL HwInitialize( ... )
{
    // CREATE DEVICES
    //
    hDev[ 0 ] = AMESDK_CREATE( "TW6802 PCI", 0, 0, hWnd00, &bcb00, NULL, // CH00
                               hWnd01, &bcb01, NULL, // CH01
                               hWnd02, &bcb02, NULL, // CH02
                               hWnd03, &bcb03, NULL, // CH03
    );
    hDev[ 1 ] = AMESDK_CREATE( "TW6802 PCI", 1, 0, hWnd04, &bcb04, NULL, // CH04
                               hWnd05, &bcb05, NULL, // CH05
                               hWnd06, &bcb06, NULL, // CH06
                               hWnd07, &bcb07, NULL, // CH07
    );
    hDev[ 2 ] = AMESDK_CREATE( "TW6802 PCI", 2, 0, hWnd08, &bcb08, NULL, // CH08
                               hWnd09, &bcb09, NULL, // CH09
                               hWnd10, &bcb10, NULL, // CH10
                               hWnd11, &bcb11, NULL, // CH11
    );
    hDev[ 3 ] = AMESDK_CREATE( "TW6802 PCI", 3, 0, hWnd12, &bcb12, NULL, // CH12
                               hWnd13, &bcb13, NULL, // CH13
                               hWnd14, &bcb14, NULL, // CH14
                               hWnd15, &bcb15, NULL, // CH15
    );
    if( (hDev[ 0 ] & 0x80000000) ||
        (hDev[ 1 ] & 0x80000000) ||
        (hDev[ 2 ] & 0x80000000) ||
        (hDev[ 3 ] & 0x80000000) ) { // CHECK ERROR FLAG

        HwUninitialize();

        return FALSE;
    }

    AMESDK_SET_STANDARD( hDev[ 0 ], KS_AnalogVideo_NTSC_M ); // SET STANDARD
    0 ... 3
    AMESDK_SET_STANDARD( hDev[ 3 ], KS_AnalogVideo_NTSC_M );

    AMESDK_SET_FORMAT( hDev[ 0 ], 720, 480, 16, 30.00 ); // SET FORMAT
    0 ... 3
    AMESDK_SET_FORMAT( hDev[ 3 ], 720, 480, 16, 30.00 );

    AMESDK_RUN( hDev[ 0 ] ); // RUN DEVICES
    0 ... 3
    AMESDK_RUN( hDev[ 3 ] );
}

BOOL HwUninitialize( ... )
{
    AMESDK_DESTROY( hDev[ 0 ] ); // STOP & CLOSE DEVICES
    AMESDK_DESTROY( hDev[ 1 ] );
    AMESDK_DESTROY( hDev[ 2 ] );
    AMESDK_DESTROY( hDev[ 3 ] );
}

```

## 1.38 SC300#D8 Software Programming Guide

REFERENCE DEVICES: SC300#D4/#D8/#D16

```

DEVICE_HANDLE hDev[ 4 ]; // USING 4 DEVICES TO SUPPORT 8 CHANNELS

BOOL HwInitialize( ... )
{
    // CREATE DEVICES
    //
    hDev[ 0 ] = AMESDK_CREATE( "TW6802 PCI", 0, 0, hWnd0, &bcb0, NULL, // CH00
                               hWnd1, &bcb1, NULL, // CH01
    );
    hDev[ 1 ] = AMESDK_CREATE( "TW6802 PCI", 1, 0, hWnd2, &bcb2, NULL, // CH02
                               hWnd3, &bcb3, NULL, // CH03
    );
    hDev[ 2 ] = AMESDK_CREATE( "TW6802 PCI", 2, 0, hWnd4, &bcb4, NULL, // CH04
                               hWnd5, &bcb5, NULL, // CH05
    );
    hDev[ 3 ] = AMESDK_CREATE( "TW6802 PCI", 3, 0, hWnd6, &bcb6, NULL, // CH06
                               hWnd7, &bcb7, NULL, // CH07
    );
    if( (hDev[ 0 ] & 0x80000000) ||
        (hDev[ 1 ] & 0x80000000) ||
        (hDev[ 2 ] & 0x80000000) ||
        (hDev[ 3 ] & 0x80000000) ) { // CHECK ERROR FLAG

        HwUninitialize();

        return FALSE;
    }

    AMESDK_SET_STANDARD( hDev[ 0 ], KS_AnalogVideo_NTSC_M ); // SET STANDARD
    0 ... 3
    AMESDK_SET_STANDARD( hDev[ 3 ], KS_AnalogVideo_NTSC_M );

    AMESDK_SET_FORMAT( hDev[ 0 ], 720, 480, 16, 30.00 ); // SET FORMAT
    0 ... 3
    AMESDK_SET_FORMAT( hDev[ 3 ], 720, 480, 16, 30.00 );

    AMESDK_RUN( hDev[ 0 ] ); // RUN DEVICES
    0 ... 3
    AMESDK_RUN( hDev[ 3 ] );
}

BOOL HwUninitialize( ... )
{
    AMESDK_DESTROY( hDev[ 0 ] ); // STOP & CLOSE DEVICES
    AMESDK_DESTROY( hDev[ 1 ] );
    AMESDK_DESTROY( hDev[ 2 ] );
    AMESDK_DESTROY( hDev[ 3 ] );
}

```

## 1.39 SC280#N4 (LIVE) Software Programming Guide

### REFERENCE DEVICES: SC280#N4

```

DEVICE_HANDLE hDev[ 4 ]; // USING 4 DEVICES TO SUPPORT 4 CHANNELS

BOOL HwInitialize( ... )
{
    // CREATE DEVICES
    //
    hDev[ 0 ] = AMESDK_CREATE( "SL6010 PCI ", 0, 0, hWnd00, &bc00, NULL ); // CH00
    hDev[ 1 ] = AMESDK_CREATE( "SL6010 PCI ", 1, 0, hWnd01, &bc01, NULL ); // CH01
    hDev[ 2 ] = AMESDK_CREATE( "SL6010 PCI ", 2, 0, hWnd02, &bc02, NULL ); // CH02
    hDev[ 3 ] = AMESDK_CREATE( "SL6010 PCI ", 3, 0, hWnd03, &bc03, NULL ); // CH03

    if( (hDev[ 0 ] & 0x80000000) ||
        (hDev[ 1 ] & 0x80000000) ||
        (hDev[ 2 ] & 0x80000000) ||
        (hDev[ 3 ] & 0x80000000) ) { // CHECK ERROR FLAG

        HwUninitialize();

        return FALSE;
    }

    AMESDK_SET_STANDARD( hDev[ 0 ], KS_AnalogVideo_NTSC_M ); // SET STANDARD
    0 ... 3
    AMESDK_SET_STANDARD( hDev[ 3 ], KS_AnalogVideo_NTSC_M );

    AMESDK_SET_FORMAT( hDev[ 0 ], 704, 480, 24, 30.00 ); // SET FORMAT
    0 ... 3
    AMESDK_SET_FORMAT( hDev[ 3 ], 704, 480, 24, 30.00 );

    AMESDK_RUN( hDev[ 0 ] ); // RUN DEVICES
    0 ... 3
    AMESDK_RUN( hDev[ 3 ] );
}

BOOL HwUninitialize( ... )
{
    AMESDK_DESTROY( hDev[ 0 ] ); // STOP & CLOSE DEVICES
    AMESDK_DESTROY( hDev[ 1 ] );
    AMESDK_DESTROY( hDev[ 2 ] );
    AMESDK_DESTROY( hDev[ 3 ] );
}

```

## 1.40 SC380#N16 (LIVE) Software Programming Guide

REFERENCE DEVICES: SC380#N4/#N8/#N16

```

DEVICE_HANDLE hDev[ 16 ]; // USING 16 DEVICES TO SUPPORT 16 CHANNELS

BOOL HwInitialize( ... )
{
    // CREATE DEVICES
    //
    hDev[ 0 ] = AMESDK_CREATE( "SL6010 PCI", 0, 0, hWnd00, &bcb00, NULL ); // CH00
    hDev[ 1 ] = AMESDK_CREATE( "SL6010 PCI", 1, 0, hWnd01, &bcb01, NULL ); // CH01
    hDev[ 2 ] = AMESDK_CREATE( "SL6010 PCI", 2, 0, hWnd02, &bcb02, NULL ); // CH02
    0 ... 15
    hDev[ 15 ] = AMESDK_CREATE( "SL6010 PCI", 15, 0, hWnd15, &bcb15, NULL ); // CH15

    if( (hDev[ 0 ] & 0x80000000) ||
        (hDev[ 1 ] & 0x80000000) ||
        (hDev[ 2 ] & 0x80000000) ||
        0 ... 15
        (hDev[ 15 ] & 0x80000000) ) { // CHECK ERROR FLAG

        HwUninitialize();

        return FALSE;
    }

    AMESDK_SET_STANDARD( hDev[ 0 ], KS_AnalogVideo_NTSC_M ); // SET STANDARD
    0 ... 15
    AMESDK_SET_STANDARD( hDev[ 15 ], KS_AnalogVideo_NTSC_M );

    AMESDK_SET_FORMAT( hDev[ 0 ], 352, 240, 16, 30.00 ); // SET FORMAT
    0 ... 15
    AMESDK_SET_FORMAT( hDev[ 15 ], 352, 240, 16, 30.00 );

    AMESDK_RUN( hDev[ 0 ] ); // RUN DEVICES
    0 ... 15
    AMESDK_RUN( hDev[ 15 ] );
}

BOOL HwUninitialize( ... )
{
    AMESDK_DESTROY( hDev[ 0 ] ); // STOP & CLOSE DEVICES
    AMESDK_DESTROY( hDev[ 1 ] );
    AMESDK_DESTROY( hDev[ 2 ] );
    0 ... 15
    AMESDK_DESTROY( hDev[ 15 ] );
}

```

## **2 Exported Functions for Analog Video Capture Device (MPEG4/H.264)**

SUPPORT DEVICE:

SC280, SC290, SC2A0, SC2B0,  
SC380, SC390, SC3A0, SC3B0,  
SC580, SC590,

Export Functions for Analog Video Capture Device	
2.00	AMESDK_CAPTURE_DEVICE_ENUMERATION (SEE 1.00)
2.01	AMESDK_CREATE
2.02	AMESDK_DESTROY (SEE 1.02)
2.03	AMESDK_RUN (SEE 1.03)
2.04	AMESDK_STOP (SEE 1.04)
2.05	AMESDK_AUTO_STANDARD_DETECTION (SEE 2.05)
2.06	AMESDK_GET_STANDARD (SEE 1.06)
2.07	AMESDK_SET_STANDARD (SEE 1.07)
2.08	AMESDK_GET_INPUT (SEE 1.08)
2.09	AMESDK_SET_INPUT (SEE 1.09)
2.10	AMESDK_GET_FORMAT
2.11	AMESDK_SET_FORMAT
2.12	AMESDK_GET_DEINTERLACE (SEE 1.12)
2.13	AMESDK_SET_DEINTERLACE (SEE 1.13)
2.14	AMESDK_GET_MIRROR (SEE 1.14)
2.15	AMESDK_SET_MIRROR (SEE 1.15)
2.16	AMESDK_GET_MODE (SEE 1.16)
2.17	AMESDK_SET_MODE (SEE 1.17)
2.18	AMESDK_GET_FREQUENCY (SEE 1.18)
2.19	AMESDK_SET_FREQUENCY (SEE 1.19)
2.20	AMESDK_GET_LOCK (SEE 1.20)
2.21	AMESDK_GET_FPS (SEE 1.21)
2.22	AMESDK_GET_CAMERACONTROL_PROPERTY (SEE 1.22)
2.23	AMESDK_SET_CAMERACONTROL_PROPERTY (SEE 1.23)
2.24	AMESDK_GET_VIDEOPROCAMP_PROPERTY (SEE 1.24)
2.25	AMESDK_SET_VIDEOPROCAMP_PROPERTY (SEE 1.25)
2.26	AMESDK_GET_VIDEOCOMPRESSION_PROPERTY
2.27	AMESDK_SET_VIDEOCOMPRESSION_PROPERTY
2.28	AMESDK_GET_CUSTOM_PROPERTY (SEE 1.28)
2.29	AMESDK_SET_CUSTOM_PROPERTY (SEE 1.29)
2.30	AMESDK_GET_CUSTOM_PROPERTY_EX (SEE 1.30)
2.31	AMESDK_SET_CUSTOM_PROPERTY_EX (SEE 1.31)
2.32	AMESDK_OTHER_REFRESH_DISPLAY_WINDOW (SEE 1.32)
2.33	AMESDK_OTHER_SNAPSHOT_BMP (SEE 1.33)
2.34	AMESDK_OTHER_SNAPSHOT_JPG (SEE 1.34)
2.35	AMESDK_OTHER_ZOOM (SEE 1.35)

Export Functions for Analog Video Capture Device	
2.36	SC380#N16 (MPEG4) Software Programming Guide
2.37	SC380#N16 (LIVE & MPEG4) Software Programming Guide
2.38	SC390#N16 (H.264) Software Programming Guide
2.39	SC390#N16 (LIVE & H.264) Software Programming Guide

## 2.01 AMESDK\_CREATE

The function helps you to open an analog video hardware-compressed capture device and also allows you to attach a preview window or register a callback function on it. The callback function will offer you to obtain and to access the whole stream buffer.

```

DEVICE_HANDLE AMESDK_CREATE( LPTSTR          pszDevName,
                             UINT           iDevNum,
                             ULONG          eDevType,
                             HWND           hDisplayWindow,
                             PF_BUFFER_CALLBACK pBufferCB,
                             PVOID          pUserData,
                             );
typedef ULONG (DEVICE_HANDLE);

typedef BOOL (* PF_BUFFER_CALLBACK) ( double   dSampleTime,
                                     BYTE *    pBuffer,
                                     ULONG      nBufferLen,
                                     BOOL       bIsKeyFrame,
                                     PVOID      pUserData
                                     );

```

### Parameters:

Parameter	IN/OUT	Description
pszDevName	IN	<b>Device Name.</b> To give a device name that is used to create specific device. Currently, we support these device names below: "AH8400 PCI, Analog Encoder", "FH8735 PCI, Analog Encoder", "SL6010 PCI, Analog Encoder", "TW5864 PCI, Analog Encoder". "TW2809 PCI, Analog Encoder".
iDevNum	IN	<b>Device Number.</b> If there are more than one devices with the same device name on platform. You can use this parameter to recognize it.
eDevType	IN	<b>Device Type.</b> Always 0 for analog capture device.
hDisplayWindow	IN	<b>Display Window.</b> Pointer to one WHND window handle. If it isn't NULL, function will automatically display video on this window. If it is NULL, function will not display video on it.
pBufferCB	IN	<b>Callback Function.</b> Pointer to one callback function. If it is NULL, function will not return bit stream buffer to software caller. If it isn't NULL, caller will obtain bit stream buffer from callback when each frame is arrived.
pUserData	IN	<b>User Data.</b> Pointer to one data pointer. The parameters will be passed through callback.

### Return Value:

If AMESDK\_CREATE is successful, it will return one DEVICE\_HANDLE. If it is fail, it will return error code. As one of below:

```

0x80000000 - Parameter, pszDevName, is wrong.
0x80000001 - Unknown error.
0x80000002 - Device queue is full already.

```



## Supported Devices:

SECU: SC280, SC380, SC290, SC2A0, SC2B0, SC390, SC3A0, SC3B0, SC580, SC590

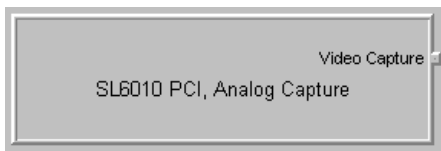
## Examples:

```
HWND hWnd = CreateWindowEx( ... );
```

```
BOOL bcb( double dSampleTime, BYTE * pBuffer, ULONG nBufferLen, BOOL bIsKeyFrame ,
          PVOID pUserData )
{
    ...
    return TRUE;
}
```

EX1: Don't need to display video and to get bit stream buffer from callback function.

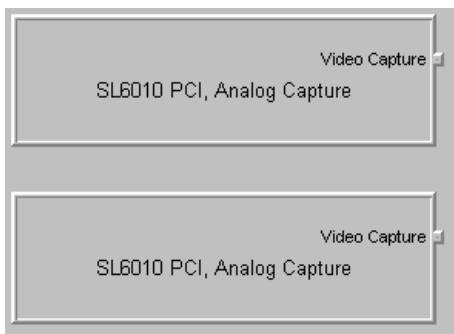
```
hDev = AMESDK_CREATE( "SL6010 PCI, Analog Encoder", 0, 0, NULL, NULL, NULL );
```



EX2: If you have two devices on one PC, you can use parameter #2 to open 2nd device.

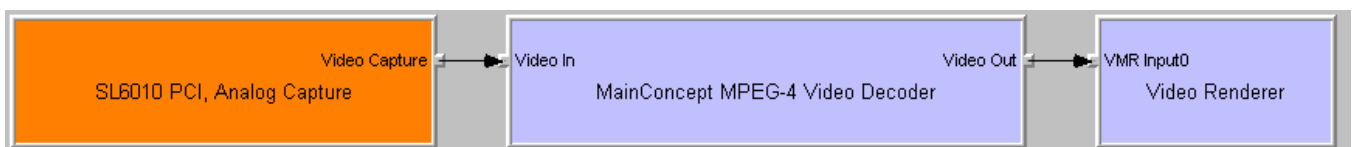
```
hDev0 = AMESDK_CREATE( "SL6010 PCI, Analog Encoder", 0, 0, NULL, NULL, NULL );
```

```
hDev1 = AMESDK_CREATE( "SL6010 PCI, Analog Encoder", 1, 0, NULL, NULL, NULL );
```



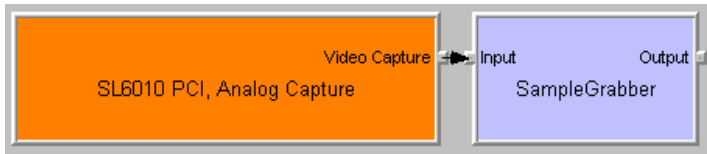
EX3: To display video (real-time playback) on your attached window by SDK engine.

```
hDev = AMESDK_CREATE( "SL6010 PCI, Analog Encoder", 0, 0, hWnd, NULL, NULL );
```



EX4: To register the callback function on the device.

```
hDev = AMESDK_CREATE( "SL6010 PCI, Analog Encoder", 0, 0, NULL, &bcb, this );
```



EX5: Both.

```
hDev = AMESDK_CREATE( "SL6010 PCI, Analog Encoder", 0, 0, hWnd, &bcb, this );
```



## Remarks:

Please reference sections 2.36 ~ 2.39 to obtain more sample tutorials.

## 2.10 AMESDK\_GET\_FORMAT

This function is used to get current video format.

```

BOOL AMESDK_GET_FORMAT(  DEVICE_HANDLE    hDevHandle,
                          ULONG *          pColorSpace,
                          ULONG *          pWidth,
                          ULONG *          pHeight,
                          ULONG *          pBitCount,
                          DOUBLE *         pFrameRate
);

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose format is to be retrieved.
pColorSpace	OUT	<b>Video Color Space.</b> Pointer to a variable that stores the color space, in MAKEFOURCC. It cannot be NULL.
pWidth	OUT	<b>Video Width.</b> Pointer to a variable that stores the width, in pixels. It cannot be NULL.
pHeight	OUT	<b>Video Height.</b> Pointer to a variable that stores the height, in pixels. It cannot be NULL.
pBitCount	OUT	<b>Video BitCount.</b> Pointer to a variable that stores the bit count. It cannot be NULL.
pFrameRate	OUT	<b>Video FrameRate.</b> Pointer to a variable that stores the frame rate, in fps. It cannot be NULL.

### Return Value:

BOOL

### Support Devices:

SECU: SC280, SC380, SC290, SC2A0, SC2B0, SC390, SC3A0, SC3B0, SC580, SC590

### Examples:

EX1: Get the current video format (colorspace, width, height, bit count, and frame rate).

```
AMESDK_GET_FORMAT( hDev, &nColorSpace, &nWidth, &nHeight, &nBitCount, &dFrameRate );
```

## Remarks:

FORMAT (NTSC)	SC280 SC380 (XVID)	SC290 SC390 (H264)	SC2A0 SC3A0 (H264 / X264)	SC2B0 SC3B0 (H264 / X264)
720×480×24×30.00		●	● / x	● / x
704×480×24×30.00	●	●	● / x	● / x
640×480×24×30.00		●		
720×240×24×30.00		●	● / x	● / x
704×240×24×30.00	●	●	● / x	● / x
640×240×24×30.00		●		
360×240×24×30.00				
352×240×24×30.00	●	●	● / ●	● / ●
320×240×24×30.00		●		
176×120×24×30.00			● / ●	● / ●

FORMAT (PAL)	SC280 SC380 (XVID)	SC290 SC390 (H264)	SC2A0 SC3A0 (H264 / X264)	SC2B0 SC3B0 (H264 / X264)
720×576×24×25.00		●	● / x	● / x
704×576×24×25.00	●	●	● / x	● / x
640×576×24×25.00		●		
720×288×24×25.00		●	● / x	● / x
704×288×24×25.00	●	●	● / x	● / x
640×288×24×25.00		●		
360×288×24×25.00				
352×288×24×25.00	●	●	● / ●	● / ●
320×288×24×25.00		●		
176×144×24×25.00			● / ●	● / ●

FORMAT	SC580 (H264 / X264)	SC590 (H264 / X264)
720×480i×60.00	●	●
720×576i×50.00	●	●
720×480p×60.00	●	●
720×576p×50.00	●	●
1280×720p×60.00	●	●
1280×720p×50.00	●	●
1280×720p×30.00	●	●
1280×720p×25.00	●	●
1280×720p×24.00	●	●
1920×1080i×60.00	●	●
1920×1080i×50.00	●	●
1920×1080p×60.00	●	●
1920×1080p×50.00	●	●
1920×1080p×30.00	●	●
1920×1080p×25.00	●	●
1920×1080p×24.00	●	●
640×384p×60.00	●	●
640×400P×60.00	●	●
640×480p×60.00	●	●
800×600p×60.00	●	●
1024×768p×60.00	●	●
1280×768p×60.00	●	●
1280×800p×60.00	●	●
1280×960p×60.00	●	●
1280×1024p×60.00	●	●
1360×768p×60.00	●	●
1440×900p×60.00	●	●
720×240p×60.00	●	●
720×288p×50.00	●	●

## Remarks:

MAKEFOURCC('H, '2, '6, '4) is used by main stream output and MAKEFOURCC('X, '2, '6, '4) is used by sub stream output.

You should set or update video format before AMESDK\_RUN() call.

## 2.11 AMESDK\_SET\_FORMAT

This function is used to set/change video format.

```

    BOOL AMESDK_SET_FORMAT(  DEVICE_HANDLE    hDevHandle,
                              ULONG           nColorSpace,
                              ULONG           nWidth,
                              ULONG           nHeight,
                              ULONG           nBitCount,
                              DOUBLE          dFrameRate
    );

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose format is to be set/changed.
nColorSpace	IN	<b>Video ColorSpace.</b> Specifies the colorspace, in MAKEFOURCC.
nWidth	IN	<b>Video Width.</b> Specifies the width, in pixels.
nHeight	IN	<b>Video Height.</b> Specifies the height, in pixels.
nBitCount	IN	<b>Video BitCount.</b> Specifies the bit count.
dFrameRate	IN	<b>Video FrameRate.</b> Specifies the frame rate, in fps.

### Return Value:

BOOL

### Support Devices:

SECU: SC280, SC380, SC290, SC2A0, SC2B0, SC390, SC3A0, SC3B0, SC580, SC590

### Examples:

EX1: Set the video format, XVID × 704 × 480 × 24 bits × 30.00 fps.

```
AMESDK_SET_FORMAT( hDev, MAKEFOURCC('X', 'V', 'I', 'D'), 704, 480, 24, 30.00 );
```

EX2: Set the video format, H264 × 352 × 288 × 24 bits × 25.00 fps.

```
AMESDK_SET_FORMAT( hDev, MAKEFOURCC('H', '2', '6', '4'), 352, 288, 24, 25.00 );
```

EX2: Set the video format, X264 × 352 × 288 × 24 bits × 25.00 fps.

```
AMESDK_SET_FORMAT( hDev, MAKEFOURCC('X', '2', '6', '4'), 352, 288, 24, 25.00 );
```

## Remarks:

FORMAT (NTSC)	SC280 SC380 (XVID)	SC290 SC390 (H264)	SC2A0 SC3A0 (H264 / X264)	SC2B0 SC3B0 (H264 / X264)
720×480×24×30.00		●	● / x	● / x
704×480×24×30.00	●	●	● / x	● / x
640×480×24×30.00		●		
720×240×24×30.00		●	● / x	● / x
704×240×24×30.00	●	●	● / x	● / x
640×240×24×30.00		●		
360×240×24×30.00				
352×240×24×30.00	●	●	● / ●	● / ●
320×240×24×30.00		●		
176×120×24×30.00			● / ●	● / ●

FORMAT (PAL)	SC280 SC380 (XVID)	SC290 SC390 (H264)	SC2A0 SC3A0 (H264 / X264)	SC2B0 SC3B0 (H264 / X264)
720×576×24×25.00		●	● / x	● / x
704×576×24×25.00	●	●	● / x	● / x
640×576×24×25.00		●		
720×288×24×25.00		●	● / x	● / x
704×288×24×25.00	●	●	● / x	● / x
640×288×24×25.00		●		
360×288×24×25.00				
352×288×24×25.00	●	●	● / ●	● / ●
320×288×24×25.00		●		
176×144×24×25.00			● / ●	● / ●

FORMAT	SC580 (H264 / X264)	SC590 (H264 / X264)
720×480i×60.00	●	●
720×576i×50.00	●	●
720×480p×60.00	●	●
720×576p×50.00	●	●
1280×720p×60.00	●	●
1280×720p×50.00	●	●
1280×720p×30.00	●	●
1280×720p×25.00	●	●
1280×720p×24.00	●	●
1920×1080i×60.00	●	●
1920×1080i×50.00	●	●
1920×1080p×60.00	●	●
1920×1080p×50.00	●	●
1920×1080p×30.00	●	●
1920×1080p×25.00	●	●
1920×1080p×24.00	●	●
640×384p×60.00	●	●
640×400P×60.00	●	●
640×480p×60.00	●	●
800×600p×60.00	●	●
1024×768p×60.00	●	●
1280×768p×60.00	●	●
1280×800p×60.00	●	●
1280×960p×60.00	●	●
1280×1024p×60.00	●	●
1360×768p×60.00	●	●
1440×900p×60.00	●	●
720×240p×60.00	●	●
720×288p×50.00	●	●

## Remarks:

MAKEFOURCC('H, '2, '6, '4) is used by main stream output and MAKEFOURCC('X, '2, '6, '4) is used by sub stream output.

You should set or update video format before AMESDK\_RUN() call.



## 2.26 AMESDK\_GET\_VIDECOMPRESSION\_PROPERTY

This function is used to get some custom device properties.

```

    BOOL AMESDK_GET_VIDECOMPRESSION_PROPERTY(    DEVICE_HANDLE    hDevHandle,
                                                    ULONG        nProperty,
                                                    ULONG *      pValue
    );

```

**Parameters:**

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be retrieved.
nProperty	IN	<b>Property.</b> Specifies the property that will be gotten from the device. As one of below: <b>SUPPORT PROPERTIES:</b> VideoCompression_KeyFrameRate (0x00000000) VideoCompression_Quality (0x00000001) VideoCompression_BitRateMode (0x00000003) VideoCompression_BitRate (0x00000004) VideoCompression_QPStep (0x00000005) VideoCompression_PeakBitRate (0x00000006) VideoCompression_TroughQuality (0x00000007) VideoCompression_PostResolution (0x00000008) VideoCompression_PostSkipFrameRate (0x00000009) VideoCompression_PostAvgFrameRate (0x0000000D) VideoCompression_BFrames (0x0000000A) VideoCompression_Profile (0x0000000B) VideoCompression_AspectRatio (0x0000000C)
pValue	OUT	<b>Property Value.</b> Pointer to a variable that stores the specified property value. It cannot be NULL. The range of value is dependent on its property.

**Return Values:**

BOOL

**Support Devices:**

SECU: SC280, SC380, SC290, SC2A0, SC2B0, SC390, SC3A0, SC3B0, SC580, SC590

**Examples:**

EX1: Get the current VideoCompression Quality property.

```
AMESDK_GET_VIDECOMPRESSION_PROPERTY( hDev, 0x00000001, &nQuality );
```

**Remarks:**

The KeyFrameRate is GOP. It is renamed by Microsoft's compression interface.

For the QPStep property, it is used in CBR and HBR mode by SC290/SC390. In CBR mode, generally, you need just adjust the target bitrate, then the inner encoder will check current bitrate to meet your target value. When the QPStep is 1, the encoder will do the fps-checking during every frame. Here, the output bitrate of encoder will be very close to your target bitrate. When the QPStep is set to 30, the encoder will do the fps-checking per 30 frames. It will offer bigger tolerance on bitrate controlling, but it will cause the output bitrate is not accurate.

For the PeakBitRate and TroughQuality, they are used only in HBR mode by SC290/SC390. In HBR mode, if the target bitrates over one GOP period is above PeakBitRate, encoding mode changes from VBR to CBR. If current quality is lower than TroughQuality over a GOP period encoding mode changes from CBR to VBR.

The BFrames property is used by H.264 MainProfile solution only, such as SC2A0, SC3A0 and SC580. The range is usually from 0 to 2.

Here, there is one table to describe the allowed parameters in different BitRateMode as below:

BitRateMode	Property	Support Device
VBR (0x00000000)	KeyFrameRate, Quality,	SC280, SC380, SC290, SC390, SC2B0, SC3B0,
VBR (0x00000000)	KeyFrameRate, Quality, BFrames,	SC2A0, SC3A0, SC580,
CBR (0x00000001)	KeyFrameRate, BitRate,	SC2B0, SC3B0,
CBR (0x00000001)	KeyFrameRate, BitRate, QPStep,	SC290, SC390,
CBR (0x00000001)	KeyFrameRate, BitRate, BFrames,	SC2A0, SC3A0, SC580,
HBR (0x00000002)	KeyFrameRate, Quality, BitRate,	SC2B0, SC3B0,
HBR (0x00000002)	KeyFrameRate, Quality, BitRate, TroughQuality, PeakBitRate, QPStep,	SC290, SC390,
HBR (0x00000002)	KeyFrameRate, Quality, BitRate, BFrames,	SC2A0, SC3A0, SC580,

The PostResolution, PostSkipFrameRate, and PostAvgFramRate are dynamically used to adjust current stream format, which is set by AMESDK\_SET\_FORMAT function at initialize stage. There are two FrameRate control methods can be used by your application. The range of PostSkipFrameRate property is from 0 to 255. It is identical to the skip number of frame (or field). The value 1 will generate the recording frame rate, 15.000fps, in NTSC. The range of PostAvgFramRate property is from 0 to 60. To enable it, our driver will follow the setting value to output one average fps. For example, 9 mean 9.00fps.

For example, the PostResolution property for SC390 is described by the table as below:

Value	Resolution
0x002D01E0 / 0x002D0240	720X480 / 720X576
0x002C01E0 / 0x002C0240	704X480 / 704X576
0x002801E0 / 0x00280240	640X480 / 640X576
0x002D00F0 / 0x002D0120	720X240 / 720X288
0x002C00F0 / 0x002C0120	704X240 / 704X288
0x002800F0 / 0x00280120	640X240 / 640X288
0x001680F0 / 0x00168120	360X240 / 360X288
0x001600F0 / 0x00160120	352X240 / 352X288
0x001400F0 / 0x00140120	320X240 / 320X288

Moreover, for HBR user (SC290/SC390 user), please reference to this table as default setting:

BitRate	PeakBitRate	Quality	TroughQuality	QPStep
4,096 KB	6,144 KB	6000	6500	30
2,048 KB	3,072 KB	5000	5500	20
1,536 KB	2,048 KB	4000	4500	10
1,024 KB	1,536 KB	3000	3500	1

## 2.27 AMESDK\_SET\_VIDECOMPRESSION\_PROPERTY

This function is used to set some custom device properties.

```

    BOOL AMESDK_SET_VIDECOMPRESSION_PROPERTY(    DEVICE_HANDLE    hDevHandle,
                                                  ULONG          nProperty,
                                                  ULONG          nValue
    );

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be set.
nProperty	IN	<b>Property.</b> Specifies the property that will be set to the device. As one of below: <b>SUPPORT PROPERTIES:</b> VideoCompression_KeyFrameRate (0x00000000) VideoCompression_Quality (0x00000001) VideoCompression_OverrideKeyFrame (0x00000002) VideoCompression_BitRateMode (0x00000003) VideoCompression_BitRate (0x00000004) VideoCompression_QPStep (0x00000005) VideoCompression_PeakBitRate (0x00000006) VideoCompression_TroughQuality (0x00000007) VideoCompression_PostResolution (0x00000008) VideoCompression_PostSkipFrameRate (0x00000009) VideoCompression_PostAvgFrameRate (0x0000000D) VideoCompression_BFrames (0x0000000A) VideoCompression_Profile (0x0000000B) VideoCompression_AspectRatio (0x0000000C)
nValue	IN	<b>Property Value.</b> Pointer to a variable that stores the specified property value. It cannot be NULL. The range of value is dependent on its property.

### Return Values:

BOOL

### Support Devices:

SECU: SC280, SC380, SC290, SC2A0, SC2B0, SC390, SC3A0, SC3B0, SC580, SC590

### Examples:

EX1: Set the current VideoCompression Quality property.

```
AMESDK_SET_VIDECOMPRESSION_PROPERTY( hDev, 0x00000001, nQuality );
```

### Remarks:

The OverrideKeyFrame is used to query one key frame as next incoming frame right away. For example, developer can call it at the file changing stage.

## 2.36 SC380#N16 (MPEG4) Software Programming Guide

REFERENCE DEVICES: SC280#N4, SC380#N4/#N8/#N16

```

DEVICE_HANDLE hDev[ 16 ]; // USING 16 DEVICES TO SUPPORT 16 CHANNELS

BOOL HwInitialize( ... )
{
    // CREATE DEVICES
    //
    hDev[ 0 ] = AMESDK_CREATE( "SL6010 PCI, Analog Encoder", 0, 0, hWnd00, &bcb00, NULL ); // CH00
    hDev[ 1 ] = AMESDK_CREATE( "SL6010 PCI, Analog Encoder", 1, 0, hWnd01, &bcb01, NULL ); // CH01
    hDev[ 2 ] = AMESDK_CREATE( "SL6010 PCI, Analog Encoder", 2, 0, hWnd02, &bcb02, NULL ); // CH02
    0 ... 15
    hDev[ 15 ] = AMESDK_CREATE( "SL6010 PCI, Analog Encoder", 15, 0, hWnd15, &bcb15, NULL ); // CH15

    if( (hDev[ 0 ] & 0x80000000) ||
        (hDev[ 1 ] & 0x80000000) ||
        (hDev[ 2 ] & 0x80000000) ||
        0 ... 15
        (hDev[ 15 ] & 0x80000000) ) { // CHECK ERROR FLAG

        HwUninitialize();

        return FALSE;
    }

    AMESDK_SET_STANDARD( hDev[ 0 ], KS_AnalogVideo_NTSC_M ); // SET STANDARD
    0 ... 15
    AMESDK_SET_STANDARD( hDev[ 15 ], KS_AnalogVideo_NTSC_M );

    AMESDK_SET_FORMAT( hDev[ 0 ], 704, 480, 24, 30.00 ); // SET FORMAT
    0 ... 15
    AMESDK_SET_FORMAT( hDev[ 15 ], 704, 480, 24, 30.00 );

    AMESDK_RUN( hDev[ 0 ] ); // RUN DEVICES
    0 ... 15
    AMESDK_RUN( hDev[ 15 ] );
}

BOOL HwUninitialize( ... )
{
    AMESDK_DESTROY( hDev[ 0 ] ); // STOP & CLOSE DEVICES
    AMESDK_DESTROY( hDev[ 1 ] );
    AMESDK_DESTROY( hDev[ 2 ] );
    0 ... 15
    AMESDK_DESTROY( hDev[ 15 ] );
}

```

## 2.37 SC380#N16 (LIVE + MPEG4) Software Programming Guide

REFERENCE DEVICES: SC280#N4, SC380#N4/#N8/#N16

```

DEVICE_HANDLE hDev_Live[ 16 ]; // USING 16 DEVICES TO SUPPORT 16 CHANNELS (UYVY)

DEVICE_HANDLE hDev[ 16 ]; // USING 16 DEVICES TO SUPPORT 16 CHANNELS (MPEG4)

BOOL HwInitialize( ... )
{
    // CREATE DEVICES
    //
    // LIVE DISPLAY & GET UYVY FRAME BUFFER
    //
    hDev_Live[ 0 ] = AMESDK_CREATE( "SL6010 PCI", 0, 0, hWnd00, &bc00_Live, NULL ); // CH00
    hDev_Live[ 1 ] = AMESDK_CREATE( "SL6010 PCI", 1, 0, hWnd01, &bc01_Live, NULL ); // CH01
    hDev_Live[ 2 ] = AMESDK_CREATE( "SL6010 PCI", 2, 0, hWnd02, &bc02_Live, NULL ); // CH02
    0 ... 15
    hDev_Live[ 15 ] = AMESDK_CREATE( "SL6010 PCI", 15, 0, hWnd15, &bc15_Live, NULL ); // CH15

    // GET MPEG4 BIT STREAM BUFFER
    //
    hDev[ 0 ] = AMESDK_CREATE( "SL6010 PCI, Analog Encoder", 0, 0, NULL, &bc00, NULL ); // CH00
    hDev[ 1 ] = AMESDK_CREATE( "SL6010 PCI, Analog Encoder", 1, 0, NULL, &bc01, NULL ); // CH01
    hDev[ 2 ] = AMESDK_CREATE( "SL6010 PCI, Analog Encoder", 2, 0, NULL, &bc02, NULL ); // CH02
    0 ... 15
    hDev[ 15 ] = AMESDK_CREATE( "SL6010 PCI, Analog Encoder", 15, 0, NULL, &bc15, NULL ); // CH15

    if( (hDev_Live[ 0 ] & 0x80000000) ||
        (hDev_Live[ 1 ] & 0x80000000) ||
        (hDev_Live[ 2 ] & 0x80000000) ||
        0 ... 15
        (hDev_Live[ 15 ] & 0x80000000) ) { // CHECK ERROR FLAG

        HwUninitialize();

        return FALSE;
    }

    if( (hDev[ 0 ] & 0x80000000) ||
        (hDev[ 1 ] & 0x80000000) ||
        (hDev[ 2 ] & 0x80000000) ||
        0 ... 15
        (hDev[ 15 ] & 0x80000000) ) { // CHECK ERROR FLAG

        HwUninitialize();

        return FALSE;
    }

    AMESDK_SET_STANDARD( hDev_Live[ 0 ], KS_AnalogVideo_NTSC_M ); // SET STANDARD
    0 ... 15
    AMESDK_SET_STANDARD( hDev_Live[ 15 ], KS_AnalogVideo_NTSC_M );

    AMESDK_SET_STANDARD( hDev[ 0 ], KS_AnalogVideo_NTSC_M ); // SET STANDARD
    0 ... 15
    AMESDK_SET_STANDARD( hDev[ 15 ], KS_AnalogVideo_NTSC_M );

    AMESDK_SET_FORMAT( hDev_Live[ 0 ], 352, 240, 16, 30.00 ); // SET FORMAT
    0 ... 15
    AMESDK_SET_FORMAT( hDev_Live[ 15 ], 352, 240, 16, 30.00 );

```

```

AMESDK_SET_FORMAT( hDev[ 0 ], 704, 480, 24, 30.00 ); // SET FORMAT
0 ... 15
AMESDK_SET_FORMAT( hDev[ 15 ], 704, 480, 24, 30.00 );

AMESDK_SET_CUSTOM_PROPERTY( hDev[ 0 ], 402, 30000); // SET FRAME RATE
0 ... 15
AMESDK_SET_CUSTOM_PROPERTY( hDev[ 15 ], 402, 30000);

AMESDK_SET_CUSTOM_PROPERTY( hDev[ 0 ], 404, 8 ); //SET QUALITY
0 ... 15
AMESDK_SET_CUSTOM_PROPERTY( hDev[ 15 ], 404, 8 );

AMESDK_SET_CUSTOM_PROPERTY( hDev[ 0 ], 405, 32 ); // SET GOP
0 ... 15
AMESDK_SET_CUSTOM_PROPERTY( hDev[ 15 ], 405, 32 );

AMESDK_RUN( hDev_Live[ 0 ] ); // RUN DEVICES
0 ... 15
AMESDK_RUN( hDev_Live[ 15 ] );

AMESDK_RUN( hDev[ 0 ] ); // RUN DEVICES
0 ... 15
AMESDK_RUN( hDev[ 15 ] );
}

BOOL HwUninitialize( ... )
{
    AMESDK_DESTROY( hDev_Live[ 0 ] ); // STOP & CLOSE DEVICES
    AMESDK_DESTROY( hDev_Live[ 1 ] );
    AMESDK_DESTROY( hDev_Live[ 2 ] );
    0 ... 15
    AMESDK_DESTROY( hDev_Live[ 15 ] );

    AMESDK_DESTROY( hDev[ 0 ] ); // STOP & CLOSE DEVICES
    AMESDK_DESTROY( hDev[ 1 ] );
    AMESDK_DESTROY( hDev[ 2 ] );
    0 ... 15
    AMESDK_DESTROY( hDev[ 15 ] );
}

```

## 2.38 SC390#N16 (H.264) Software Programming Guide

REFERENCE DEVICES: SC290#N4, SC390#N4/#N8/#N16

```

DEVICE_HANDLE hDev[ 16 ]; // USING 16 DEVICES TO SUPPORT 16 CHANNELS

BOOL HwInitialize( ... )
{
    // CREATE DEVICES
    //
    hDev[ 0 ] = AMESDK_CREATE( "AH8400 PCI, Analog Encoder", 0, 0, NULL, &bc00, NULL ); // CH00
    hDev[ 1 ] = AMESDK_CREATE( "AH8400 PCI, Analog Encoder", 1, 0, NULL, &bc01, NULL ); // CH01
    hDev[ 2 ] = AMESDK_CREATE( "AH8400 PCI, Analog Encoder", 2, 0, NULL, &bc02, NULL ); // CH02
    0 ... 15
    hDev[ 15 ] = AMESDK_CREATE( "AH8400 PCI, Analog Encoder", 15, 0, NULL, &bc15, NULL ); // CH15

    if( (hDev[ 0 ] & 0x80000000) ||
        (hDev[ 1 ] & 0x80000000) ||
        (hDev[ 2 ] & 0x80000000) ||
        0 ... 15
        (hDev[ 15 ] & 0x80000000) ) { // CHECK ERROR FLAG

        HwUninitialize();

        return FALSE;
    }

    AMESDK_SET_STANDARD( hDev[ 0 ], KS_AnalogVideo_NTSC_M ); // SET STANDARD
    0 ... 15
    AMESDK_SET_STANDARD( hDev[ 15 ], KS_AnalogVideo_NTSC_M );

    AMESDK_SET_FORMAT( hDev[ 0 ], 704, 480, 24, 30.00 ); // SET FORMAT
    0 ... 15
    AMESDK_SET_FORMAT( hDev[ 15 ], 704, 480, 24, 30.00 );

    AMESDK_RUN( hDev[ 0 ] ); // RUN DEVICES
    0 ... 15
    AMESDK_RUN( hDev[ 15 ] );
}

BOOL HwUninitialize( ... )
{
    AMESDK_DESTROY( hDev[ 0 ] ); // STOP & CLOSE DEVICES
    AMESDK_DESTROY( hDev[ 1 ] );
    AMESDK_DESTROY( hDev[ 2 ] );
    0 ... 15
    AMESDK_DESTROY( hDev[ 15 ] );
}

```



## 2.39 SC390#N16 (LIVE + H.264) Software Programming Guide

REFERENCE DEVICES: SC290#N4, SC390#N4/#N8/#N16

```

DEVICE_HANDLE hDev_Live[ 16 ]; // USING 16 DEVICES TO SUPPORT 16 CHANNELS (UYVY)

DEVICE_HANDLE hDev[ 16 ]; // USING 16 DEVICES TO SUPPORT 16 CHANNELS (MPEG4)

BOOL HwInitialize( ... )
{
    // CREATE DEVICES
    //
    // LIVE DISPLAY & GET UYVY FRAME BUFFER
    //
    hDev_Live[ 0 ] = AMESDK_CREATE( "AH8400 PCI", 0, 0, hWnd00, &bc00_Live, NULL ); // CH00
    hDev_Live[ 1 ] = AMESDK_CREATE( "AH8400 PCI", 1, 0, hWnd01, &bc01_Live, NULL ); // CH01
    hDev_Live[ 2 ] = AMESDK_CREATE( "AH8400 PCI", 2, 0, hWnd02, &bc02_Live, NULL ); // CH02
    0 ... 15
    hDev_Live[ 15 ] = AMESDK_CREATE( "AH8400 PCI", 15, 0, hWnd15, &bc15_Live, NULL ); // CH15

    // GET H.264 BIT STREAM BUFFER
    //
    hDev[ 0 ] = AMESDK_CREATE( "AH8400 PCI, Analog Encoder", 0, 0, NULL, &bc00, NULL ); // CH00
    hDev[ 1 ] = AMESDK_CREATE( "AH8400 PCI, Analog Encoder", 1, 0, NULL, &bc01, NULL ); // CH01
    hDev[ 2 ] = AMESDK_CREATE( "AH8400 PCI, Analog Encoder", 2, 0, NULL, &bc02, NULL ); // CH02
    0 ... 15
    hDev[ 15 ] = AMESDK_CREATE( "AH8400 PCI, Analog Encoder", 15, 0, NULL, &bc15, NULL ); // CH15

    if( (hDev_Live[ 0 ] & 0x80000000) ||
        (hDev_Live[ 1 ] & 0x80000000) ||
        (hDev_Live[ 2 ] & 0x80000000) ||
        0 ... 15
        (hDev_Live[ 15 ] & 0x80000000) ) { // CHECK ERROR FLAG

        HwUninitialize();

        return FALSE;
    }

    if( (hDev[ 0 ] & 0x80000000) ||
        (hDev[ 1 ] & 0x80000000) ||
        (hDev[ 2 ] & 0x80000000) ||
        0 ... 15
        (hDev[ 15 ] & 0x80000000) ) { // CHECK ERROR FLAG

        HwUninitialize();

        return FALSE;
    }

    AMESDK_SET_STANDARD( hDev_Live[ 0 ], KS_AnalogVideo_NTSC_M ); // SET STANDARD
    0 ... 15
    AMESDK_SET_STANDARD( hDev_Live[ 15 ], KS_AnalogVideo_NTSC_M );

    AMESDK_SET_STANDARD( hDev[ 0 ], KS_AnalogVideo_NTSC_M ); // SET STANDARD
    0 ... 15
    AMESDK_SET_STANDARD( hDev[ 15 ], KS_AnalogVideo_NTSC_M );

    AMESDK_SET_FORMAT( hDev_Live[ 0 ], 352, 240, 16, 30.00 ); // SET FORMAT
    0 ... 15
    AMESDK_SET_FORMAT( hDev_Live[ 15 ], 352, 240, 16, 30.00 );

```

```

AMESDK_SET_FORMAT( hDev[ 0 ], 704, 480, 24, 30.00 ); // SET FORMAT
0 ... 15
AMESDK_SET_FORMAT( hDev[ 15 ], 704, 480, 24, 30.00 );

AMESDK_SET_CUSTOM_PROPERTY( hDev[ 0 ], 402, 30000); // SET FRAME RATE
0 ... 15
AMESDK_SET_CUSTOM_PROPERTY( hDev[ 15 ], 402, 30000);

AMESDK_SET_CUSTOM_PROPERTY( hDev[ 0 ], 404, 8 ); //SET QUALITY
0 ... 15
AMESDK_SET_CUSTOM_PROPERTY( hDev[ 15 ], 404, 8 );

AMESDK_SET_CUSTOM_PROPERTY( hDev[ 0 ], 405, 32 ); // SET GOP
0 ... 15
AMESDK_SET_CUSTOM_PROPERTY( hDev[ 15 ], 405, 32 );

AMESDK_RUN( hDev_Live[ 0 ] ); // RUN DEVICES
0 ... 15
AMESDK_RUN( hDev_Live[ 15 ] );

AMESDK_RUN( hDev[ 0 ] ); // RUN DEVICES
0 ... 15
AMESDK_RUN( hDev[ 15 ] );
}

BOOL HwUninitialize( ... )
{
    AMESDK_DESTROY( hDev_Live[ 0 ] ); // STOP & CLOSE DEVICES
    AMESDK_DESTROY( hDev_Live[ 1 ] );
    AMESDK_DESTROY( hDev_Live[ 2 ] );
    0 ... 15
    AMESDK_DESTROY( hDev_Live[ 15 ] );

    AMESDK_DESTROY( hDev[ 0 ] ); // STOP & CLOSE DEVICES
    AMESDK_DESTROY( hDev[ 1 ] );
    AMESDK_DESTROY( hDev[ 2 ] );
    0 ... 15
    AMESDK_DESTROY( hDev[ 15 ] );
}

```

## **3 Exported Functions for Analog Audio Capture Device (PCM)**

SUPPORT DEVICE:

PD652, SC100, SC200, SC230, SC280,  
SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0,  
SC3B0, SC500, SC510, SC580, SC590,

Exported Functions for Analog Audio Capture Device	
3.00	AMESDK_CAPTURE_DEVICE_ENUMERATION (SEE 1.00)
3.01	AMESDK_CREATE
3.02	AMESDK_DESTROY (SEE 1.02)
3.03	AMESDK_RUN (SEE 1.03)
3.04	AMESDK_STOP (SEE 1.04)
3.05	AMESDK_GET_INPUT
3.06	AMESDK_SET_INPUT
3.07	AMESDK_GET_FORMAT
3.08	AMESDK_SET_FORMAT
3.09	AMESDK_GET_VOLUME
3.10	AMESDK_SET_VOLUME
3.11	SC100#N4 Software Programming Guide
3.12	SC300#N8 Software Programming Guide
3.13	SC310#N8 Software Programming Guide

## 3.01 AMESDK\_CREATE

The function helps you to open an analog audio capture device and also allows you to attach a preview window or to register a callback function on it. The callback function will offer you to obtain and to access the whole sound buffer.

```

DEVICE_HANDLE AMESDK_CREATE( LPTSTR          pszDevName,
                             UINT           iDevNum,
                             ULONG          eDevType,
                             HWND           hDisplayWindow,
                             PF_BUFFER_CALLBACK pBufferCB,
                             PVOID          pUserData
);

typedef ULONG (DEVICE_HANDLE);

typedef BOOL (* PF_BUFFER_CALLBACK)( double    dSampleTime,
                                     BYTE *    pBuffer,
                                     ULONG     nBufferLen,
                                     BOOL      bIsKeyFrame,
                                     PVOID     pUserData );
    
```

### Parameters:

Parameter	IN/OUT	Description
pszDevName	IN	<b>Device Name.</b> To give a device name that is used to create specific device. Currently, we support these device names below: "AH8400 PCI, Analog WaveIn", "CX2581 PCI, Analog WaveIn", "CX2588 PCI, Analog WaveIn", "DC1150 USB, Analog WaveIn", "FH8735 PCI, Analog WaveIn", "TW2809 PCI, Analog WaveIn", "TW5804 PCI, Analog WaveIn", "TW6802 PCI, Analog WaveIn", "SA7160 PCI, Analog WaveIn", "SL6010 PCI, Analog WaveIn".
iDevNum	IN	<b>Device Number.</b> If there are more than one devices with the same device name on platform. You can use this parameter to recognize it.
eDevType	IN	<b>Device Type.</b> Always 0 for analog capture device.
hDisplayWindow	IN	<b>Display Window.</b> Pointer to one WHND window handle. If it isn't NULL, function will automatically play sound on this window. If it is NULL, function will mute on it.
pBufferCB	IN	<b>Callback Function.</b> Pointer to one callback function. If it is NULL, function will not return sound buffer to software caller. If it isn't NULL, caller will obtain sound buffer from callback when each frame is coming.
pUserData	IN	<b>User Data.</b> Pointer to one data pointer. The parameters will be passed through callback.

## Return Value:

If AMESDK\_CREATE is successful, it will return one DEVICE\_HANDLE. If it is fail, it will show the error code. As one of below:

0x80000000 - Parameter, pszDevName, is wrong.  
 0x80000001 - Unknown error.  
 0x80000002 - Device queue is full already.

## Supported Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

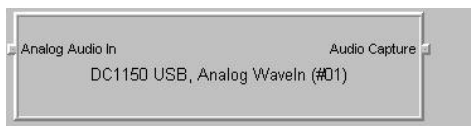
## Examples:

```
HWND wnd = CreateWindowEx( ... );
```

```
BOOL bcb( double dSampleTime, BYTE * pBuffer, ULONG nBufferLen , BOOL bIsKeyFrame,
          PVOID pUserData )
{
    ...
    return TRUE;
}
```

EX1: Don't need to display video and to get sound buffer from callback function.

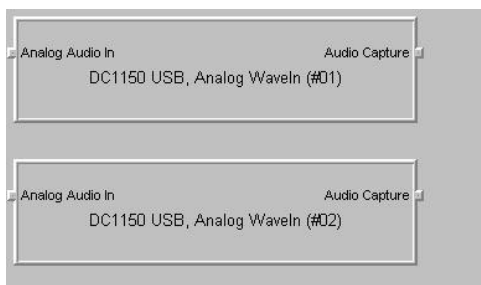
```
hDev = AMESDK_CREATE( "DC1150 USB, Analog WaveIn", 0, 0, NULL, NULL, NULL );
```



EX2: If you have two devices on one PC, you can use parameter #2 to open 2nd device.

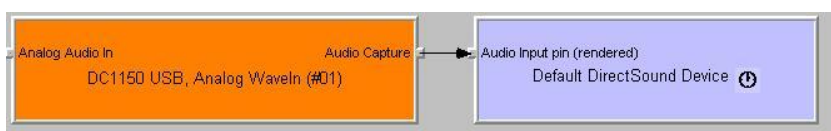
```
hDev0 = AMESDK_CREATE( "DC1150 USB, Analog WaveIn", 0, 0, NULL, NULL, NULL );
```

```
hDev1 = AMESDK_CREATE( "DC1150 USB, Analog WaveIn", 1, 0, NULL, NULL, NULL );
```



EX3: To play sound on your attached window by SDK engine.

```
hDev = AMESDK_CREATE( "DC1150 USB, Analog WaveIn", 0, 0, hWnd, NULL, NULL );
```



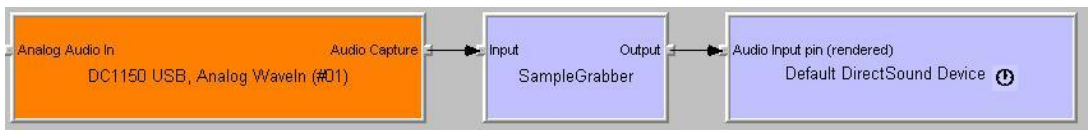
EX4: To register the callback function on the device.

```
hDev = AMESDK_CREATE( "DC1150 USB, Analog WaveIn", 0, 0, NULL, &bcb, this );
```



EX5: Both.

```
hDev = AMESDK_CREATE( "DC1150 USB, Analog WaveIn", 0, 0, hWnd, &bcb, this );
```



## Remarks:

Please reference sections 3.9 ~ 3.11 to obtain more sample tutorials.

### 3.05 AMESDK\_GET\_INPUT

This function is used to get current audio input.

```
BOOL AMESDK_GET_INPUT( DEVICE_HANDLE  hDevHandle,
                       ULONG *        pInput
);
```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose input is to be retrieved.
pInput	OUT	<b>Video Input.</b> Pointer to a variable that stores the input. It cannot be NULL.
		<b>SUPPORT INPUTS:</b> Embedded Audio      (0x00000000) Line In              (0x00000001)

#### Return Value:

BOOL

#### Support Devices:

PCTV: PD652  
SECU: SC500, SC510, SC580, SC590

#### Examples:

EX1: Get the current audio input.

```
AMESDK_GET_INPUT( hDev, &nInput );
```

#### Remarks:

For SC500, SC510, SC580, and SC590 series, please reference their Extra Programming Guide in SDK packet in detail.



### 3.06 AMESDK\_SET\_INPUT

This function is used to set/change audio input.

```
BOOL AMESDK_SET_INPUT( DEVICE_HANDLE hDevHandle,
                       ULONG          nInput
                       );
```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose input is to be set/changed.
nInput	IN	<b>Video Input.</b> Specifies the input.
		<b>SUPPORT INPUTS:</b> Embedded Audio    (0x00000000) Line In            (0x00000001)

#### Return Value:

BOOL

#### Support Devices:

PCTV: PD652  
SECU: SC500, SC510, SC580, SC590

#### Examples:

EX1: Set the audio input to Embedded Audio.

```
AMESDK_SET_INPUT( hDev, 0x00000000 );
```

EX2: Set the audio input to Line In.

```
AMESDK_SET_INPUT( hDev, 0x00000001 );
```

#### Remarks:

For SC500, SC510, SC580, and SC590 series, please reference their Extra Programming Guide in SDK packet in detail.

## 3.07 AMESDK\_GET\_FORMAT

This function is used to get current audio format.

```

BOOL AMESDK_GET_FORMAT( DEVICE_HANDLE hDevHandle,
                        ULONG *        pChannels,
                        ULONG *        pBitsPerSample,
                        ULONG *        pSamplesPerSec );
    
```

**Parameters:**

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be get.
pChannels	OUT	<b>Audio Channels.</b> Pointer to a variable that stores the audio number of channels. It cannot be NULL.
pBitsPerSample	OUT	<b>Audio Bits Per Sample.</b> Pointer to a variable that stores the audio bits per sample. It cannot be NULL.
pSamplesPerSec	OUT	<b>Audio Samples Per Sec.</b> Pointer to a variable that stores the audio sample rate in samples per second. It cannot be NULL.
		<b>SUPPORT FORMAT (MC/PDXXX):</b> 2 × 16 × 48000Hz <b>SUPPORT FORMAT (SC100):</b> 1 × 8 × 8000Hz <b>SUPPORT FORMAT (SC200, SC230, SC300, SC330):</b> 1 × 8 × 8000Hz 1 × 8 × 16000Hz 1 × 8 × 24000Hz 1 × 8 × 32000Hz 1 × 8 × 40000Hz 1 × 8 × 48000Hz 1 × 16 × 8000Hz 1 × 16 × 16000Hz 1 × 16 × 24000Hz 1 × 16 × 32000Hz 1 × 16 × 40000Hz 1 × 16 × 48000Hz <b>SUPPORT FORMAT (SC310, SC340):</b> 1 × 16 × 8000Hz 1 × 16 × 16000Hz 1 × 16 × 24000Hz 1 × 16 × 32000Hz 1 × 16 × 40000Hz 1 × 16 × 48000Hz <b>SUPPORT FORMAT (SC280, SC380):</b> 1 × 16 × 8000Hz <b>SUPPORT FORMAT (SC290, SC390):</b> 1 × 16 × 8000Hz <b>SUPPORT FORMAT (SC3A0):</b> 1 × 16 × 16000Hz <b>SUPPORT FORMAT (SC2B0, SC3B0):</b> 1 × 16 × 8000Hz <b>SUPPORT FORMAT (SC500, SC510, SC580, SC590):</b> 2 × 16 × 32000Hz 2 × 16 × 44100Hz 2 × 16 × 48000Hz

**Return Values:**

BOOL

**Support Devices:**

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590

**Examples:**

EX1: Get the current audio format (channels, bits per sample, and samples per sec).

```
AMESDK_GET_FORMAT( hDev, &nChannels, &nBitsPerSample, &nSamplesPerSec );
```

## 3.08 AMESDK\_SET\_FORMAT

This function is used to set/change audio format.

```

BOOL AMESDK_SET_FORMAT(  DEVICE_HANDLE hDevHandle,
                          ULONG          nChannels,
                          ULONG          nBitsPerSample,
                          ULONG          nSamplesPerSec );
    
```

**Parameters:**

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be set.
nChannels	IN	<b>Audio Channels.</b> Specifies the number of channels in the waveform data. Mono data uses 1 channel and stereo data uses 2 channels.
nBitsPerSample	IN	<b>Audio Bits Per Sample.</b> Specifies the bits per sample in the waveform data.
nSamplesPerSec	IN	<b>Audio Samples Per Sec.</b> Specifies the sample rate in samples per second.
		<b>SUPPORT FORMAT (MC/PDXXX):</b> 2 × 16 × 48000Hz <b>SUPPORT FORMAT (SC100):</b> 1 × 8 × 8000Hz <b>SUPPORT FORMAT (SC200, SC230, SC300, SC330):</b> 1 × 8 × 8000Hz 1 × 8 × 16000Hz 1 × 8 × 24000Hz 1 × 8 × 32000Hz 1 × 8 × 40000Hz 1 × 8 × 48000Hz 1 × 16 × 8000Hz 1 × 16 × 16000Hz 1 × 16 × 24000Hz 1 × 16 × 32000Hz 1 × 16 × 40000Hz 1 × 16 × 48000Hz <b>SUPPORT FORMAT (SC310, SC340):</b> 1 × 16 × 8000Hz 1 × 16 × 16000Hz 1 × 16 × 24000Hz 1 × 16 × 32000Hz 1 × 16 × 40000Hz 1 × 16 × 48000Hz <b>SUPPORT FORMAT (SC280, SC380):</b> 1 × 16 × 8000Hz <b>SUPPORT FORMAT (SC290, SC390):</b> 1 × 16 × 8000Hz <b>SUPPORT FORMAT (SC3A0):</b> 1 × 16 × 16000Hz <b>SUPPORT FORMAT (SC2B0, SC3B0):</b> 1 × 16 × 8000Hz <b>SUPPORT FORMAT (SC500, SC510, SC580, SC590):</b> 2 × 16 × 32000Hz 2 × 16 × 44100Hz 2 × 16 × 48000Hz

**Return Values:**

BOOL

**Support Devices:**

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

**Examples:**

EX1: Set the audio format (channels, bits per sample, and samples per sec).

```
AMESDK_SET_FORMAT( hDev, 1, 16, 8000 );
```

## 3.09 AMESDK\_GET\_VOLUME

This function is used to get the volume (amplitude) of the sound card renderer.

```
BOOL AMESDK_GET_VOLUME( DEVICE_HANDLE hDevHandle, ULONG * pVolume );
```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be get.
pVolume	OUT	<b>Audio Volume Amplitude.</b> Pointer to a variable that stores the audio amplitude. It cannot be NULL.

### Return Values:

BOOL

### Support Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Examples:

EX1: Get the value of audio volume.

```
AMESDK_GET_VOLUME( hDev, &nVolume );
```

### Remarks:

If you need control the hardware input amplitude on the capture card, you need to use AMESDK\_GET\_CUSTOM\_PROPERTY to access it.

## 3.10 AMESDK\_SET\_VOLUME

This function is used to set the volume (amplitude) of the sound card renderer.

```
BOOL AMESDK_SET_VOLUME( DEVICE_HANDLE hDevHandle, ULONG nVolume );
```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be set.
nVolume	IN	<b>Audio Volume Amplitude.</b> A variable that sets the audio signal amplitude. Support Ranges: 0 (Mute) ~ 100 (Full).

### Return Values:

BOOL

### Support Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Examples:

EX1: Set the audio volume amplitude.

```
AMESDK_SET_VOLUME( hDev, 128 );
```

### Remarks:

If you need control the hardware input amplitude on the capture card, you need to use AMESDK\_SET\_CUSTOM\_PROPERTY to access it.

## 3.11 SC100#N4 Software Programming Guide

### REFERENCE DEVICES: SC100#N4

```

DEVICE_HANDLE hDev[ 4 ]; // USING 4 DEVICES TO SUPPORT 4 CHANNELS

BOOL HwInitialize( ... )
{
    // CREATE DEVICES
    //
    hDev[ 0 ] = AMESDK_CREATE( "DC1150 USB, Analog Wave In", 0, 0, hWnd0, &bc0, NULL ); // CH00
    hDev[ 1 ] = AMESDK_CREATE( "DC1150 USB, Analog Wave In", 1, 0, hWnd1, &bc1, NULL ); // CH01
    hDev[ 2 ] = AMESDK_CREATE( "DC1150 USB, Analog Wave In", 2, 0, hWnd2, &bc2, NULL ); // CH02
    hDev[ 3 ] = AMESDK_CREATE( "DC1150 USB, Analog Wave In", 3, 0, hWnd3, &bc3, NULL ); // CH03

    if( (hDev[ 0 ] & 0x80000000) ||
        (hDev[ 1 ] & 0x80000000) ||
        (hDev[ 2 ] & 0x80000000) ||
        (hDev[ 3 ] & 0x80000000) ) { // CHECK ERROR FLAG

        HwUninitialize();

        return FALSE;
    }

    AMESDK_SET_VOLUME( hDev[ 0 ], 0 /*MUTE*/ ); // SET VOLUME
    0 ... 3
    AMESDK_SET_FORMAT( hDev[ 3 ], 0 /*MUTE*/ );

    AMESDK_RUN( hDev[ 0 ] ); // RUN DEVICES
    0 ... 3
    AMESDK_RUN( hDev[ 3 ] );
}

BOOL HwUninitialize( ... )
{
    AMESDK_DESTROY( hDev[ 0 ] ); // STOP & CLOSE DEVICES
    AMESDK_DESTROY( hDev[ 1 ] );
    AMESDK_DESTROY( hDev[ 2 ] );
    AMESDK_DESTROY( hDev[ 3 ] );
}

```



## 3.12 SC300#N8 Software Programming Guide

REFERENCE DEVICES: SC200#ALL, SC300#ALL

```

DEVICE_HANDLE hDev[ 8 ]; // USING 8 DEVICES TO SUPPORT 8 CHANNELS

BOOL HwInitialize( ... )
{
    // CREATE DEVICES
    //
    hDev[ 0 ] = AMESDK_CREATE( "TW6802 PCI, Analog Wave In", 0, 0, hWnd0, &bc0, NULL ); // CH00
    hDev[ 1 ] = AMESDK_CREATE( "TW6802 PCI, Analog Wave In", 1, 0, hWnd1, &bc1, NULL ); // CH01
    hDev[ 2 ] = AMESDK_CREATE( "TW6802 PCI, Analog Wave In", 2, 0, hWnd2, &bc2, NULL ); // CH02
    hDev[ 3 ] = AMESDK_CREATE( "TW6802 PCI, Analog Wave In", 3, 0, hWnd3, &bc3, NULL ); // CH03
    hDev[ 4 ] = AMESDK_CREATE( "TW6802 PCI, Analog Wave In", 4, 0, hWnd4, &bc4, NULL ); // CH04
    hDev[ 5 ] = AMESDK_CREATE( "TW6802 PCI, Analog Wave In", 5, 0, hWnd5, &bc5, NULL ); // CH05
    hDev[ 6 ] = AMESDK_CREATE( "TW6802 PCI, Analog Wave In", 6, 0, hWnd6, &bc6, NULL ); // CH06
    hDev[ 7 ] = AMESDK_CREATE( "TW6802 PCI, Analog Wave In", 7, 0, hWnd7, &bc7, NULL ); // CH07

    if( (hDev[ 0 ] & 0x80000000) ||
        (hDev[ 1 ] & 0x80000000) ||
        (hDev[ 2 ] & 0x80000000) ||
        (hDev[ 3 ] & 0x80000000) ||
        (hDev[ 4 ] & 0x80000000) ||
        (hDev[ 5 ] & 0x80000000) ||
        (hDev[ 6 ] & 0x80000000) ||
        (hDev[ 7 ] & 0x80000000) ) { // CHECK ERROR FLAG

        HwUninitialize();

        return FALSE;
    }

    AMESDK_SET_VOLUME( hDev[ 0 ], 0 /*MUTE*/ ); // SET VOLUME
    0 ... 7
    AMESDK_SET_VOLUME( hDev[ 7 ], 0 /*MUTE*/ );

    AMESDK_RUN( hDev[ 0 ] ); // RUN DEVICES
    0 ... 7
    AMESDK_RUN( hDev[ 7 ] );
}

BOOL HwUninitialize( ... )
{
    AMESDK_DESTROY( hDev[ 0 ] ); // STOP & CLOSE DEVICES
    AMESDK_DESTROY( hDev[ 1 ] );
    AMESDK_DESTROY( hDev[ 2 ] );
    AMESDK_DESTROY( hDev[ 3 ] );
    AMESDK_DESTROY( hDev[ 4 ] );
    AMESDK_DESTROY( hDev[ 5 ] );
    AMESDK_DESTROY( hDev[ 6 ] );
    AMESDK_DESTROY( hDev[ 7 ] );
}

```

## 3.13 SC310#N8 Software Programming Guide

REFERENCE DEVICES: SC310#ALL

DEVICE\_HANDLE hDev[ 8 ]; // USING 8 DEVICES TO SUPPORT 8 CHANNELS

BOOL HwInitialize( ... )

```
{
    // CREATE DEVICES
    //
    hDev[ 0 ] = AMESDK_CREATE( "CX2581 PCI, Analog Wave In", 0, 0, hWnd0, &bc0, NULL ); // CH00
    hDev[ 1 ] = AMESDK_CREATE( "CX2581 PCI, Analog Wave In", 1, 0, hWnd1, &bc1, NULL ); // CH01
    hDev[ 2 ] = AMESDK_CREATE( "CX2581 PCI, Analog Wave In", 2, 0, hWnd2, &bc2, NULL ); // CH02
    hDev[ 3 ] = AMESDK_CREATE( "CX2581 PCI, Analog Wave In", 3, 0, hWnd3, &bc3, NULL ); // CH03
    hDev[ 4 ] = AMESDK_CREATE( "CX2581 PCI, Analog Wave In", 4, 0, hWnd4, &bc4, NULL ); // CH04
    hDev[ 5 ] = AMESDK_CREATE( "CX2581 PCI, Analog Wave In", 5, 0, hWnd5, &bc5, NULL ); // CH05
    hDev[ 6 ] = AMESDK_CREATE( "CX2581 PCI, Analog Wave In", 6, 0, hWnd6, &bc6, NULL ); // CH06
    hDev[ 7 ] = AMESDK_CREATE( "CX2581 PCI, Analog Wave In", 7, 0, hWnd7, &bc7, NULL ); // CH07

    if( (hDev[ 0 ] & 0x80000000) ||
        (hDev[ 1 ] & 0x80000000) ||
        (hDev[ 2 ] & 0x80000000) ||
        (hDev[ 3 ] & 0x80000000) ||
        (hDev[ 4 ] & 0x80000000) ||
        (hDev[ 5 ] & 0x80000000) ||
        (hDev[ 6 ] & 0x80000000) ||
        (hDev[ 7 ] & 0x80000000) ) { // CHECK ERROR FLAG

        HwUninitialize();

        return FALSE;
    }

    AMESDK_SET_VOLUME( hDev[ 0 ], 0 /*MUTE*/ ); // SET VOLUME
    0 ... 7
    AMESDK_SET_VOLUME( hDev[ 7 ], 0 /*MUTE*/ );

    AMESDK_RUN( hDev[ 0 ] ); // RUN DEVICES
    0 ... 7
    AMESDK_RUN( hDev[ 7 ] );
}
```

BOOL HwUninitialize( ... )

```
{
    AMESDK_DESTROY( hDev[ 0 ] ); // STOP & CLOSE DEVICES
    AMESDK_DESTROY( hDev[ 1 ] );
    AMESDK_DESTROY( hDev[ 2 ] );
    AMESDK_DESTROY( hDev[ 3 ] );
    AMESDK_DESTROY( hDev[ 4 ] );
    AMESDK_DESTROY( hDev[ 5 ] );
    AMESDK_DESTROY( hDev[ 6 ] );
    AMESDK_DESTROY( hDev[ 7 ] );
}
```

## **4 Exported Functions for Analog Audio Capture Device (G.721/G.723)**

SUPPORT DEVICE:

NONE

Exported Functions for Analog Audio Capture Device	
4.00	AMESDK_CAPTURE_DEVICE_ENUMERATION (SEE 1.00)
4.01	AMESDK_CREATE
4.02	AMESDK_DESTROY (SEE 1.02)
4.03	AMESDK_RUN (SEE 1.03)
4.04	AMESDK_STOP (SEE 1.04)
4.05	AMESDK_GET_INPUT (SEE 3.05)
4.06	AMESDK_SET_INPUT (SEE 3.06)
4.07	AMESDK_GET_FORMAT (SEE 3.07)
4.08	AMESDK_SET_FORMAT (SEE 3.08)
4.09	AMESDK_GET_VOLUME (SEE 3.09)
4.10	AMESDK_SET_VOLUME (SEE 3.10)

## **5 Exported Functions for Software Encoder and Decoder Programming**

SUPPORT DEVICE:

PD652, SC100, SC200, SC230, SC280,  
SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0,  
SC3B0, SC500, SC510, SC580, SC590,

Exported Functions for Software Encoder and Decoder Programming	
5.01	AMESDK_CREATE (ENCODER/DECODER)
5.02	AMESDK_DESTROY (ENCODER/DECODER) (SEE 1.02)
5.03	AMESDK_RUN (ENCODER/DECODER) (SEE 1.03)
5.04	AMESDK_STOP (ENCODER/DECODER) (SEE 1.04)
5.05	AMESDK_GET_DEINTERLACE (DECODER) (SEE 1.12)
5.06	AMESDK_SET_DEINTERLACE (DECODER) (SEE 1.13)
5.07	AMESDK_GET_MIRROR (DECODER) (SEE 1.14)
5.08	AMESDK_SET_MIRROR (DECODER) (SEE 1.15)
5.09	AMESDK_OTHER_REFRESH_DISPLAY_WINDOW (DECODER) (SEE 1.30)
5.10	AMESDK_OTHER_SNAPSHOT_BMP (DECODER) (SEE 1.31)
5.11	AMESDK_OTHER_SNAPSHOT_JPG (DECODER) (SEE 1.32)
5.12	AMESDK_OTHER_ZOOM (DECODER) (SEE 1.33)
5.13	AMESDK_GET_FORMAT (ENCODER/DECODER)
5.14	AMESDK_SET_FORMAT (ENCODER/DECODER)
5.15	AMESDK_CODEC_ENCODE (ENCODER)
5.16	AMESDK_CODEC_DECODE (DECODER)
5.17	AMESDK_CODEC_DECODE_EX (DECODER)
5.18	A MPEG4 Video Encoder Software Programming Guide
5.19	A H.264 Video Encoder Software Programming Guide
5.20	A MPEG4 Video Decoder Software Programming Guide
5.21	A H.264 Video Decoder Software Programming Guide
5.22	A PCM Audio Decoder Software Programming Guide

### 5.01 AMESDK\_CREATE

The function helps you to open a video codec device and also allows you to attach a playback window or register a callback function on it. The callback function will offer you to obtain and to access the whole frame buffer.

```

DEVICE_HANDLE AMESDK_CREATE( LPTSTR          pszDevName,
                             UINT           iDevNum,
                             ULONG          eDevType,
                             HWND           hDisplayWindow,
                             PF_BUFFER_CALLBACK pBufferCB,
                             PVOID          pUserData
);

typedef ULONG (DEVICE_HANDLE);

typedef BOOL (* PF_BUFFER_CALLBACK)( double   dSampleTime,
                                     BYTE *   pBuffer,
                                     ULONG    nBufferLen,
                                     BOOL     bIsKeyFrame,
                                     PVOID    pUserData
);

```

#### Parameters:

Parameter	IN/OUT	Description
pszDevName	IN	<b>Device Name.</b> To give a device name that is used to create specific device. Currently, we support these codec names below: "Common Analog Encoder (MPEG4)", "Common Analog Decoder (MPEG4)", "Common Analog Encoder (H.264)", "Common Analog Decoder (H.264)", "Common Analog Intel Encoder (H.264)", "Common Analog Intel Decoder (H.264)", "Common Analog Decoder (PCM)"
iDevNum	IN	<b>Device Number.</b> Here, SDK supports you to create 64 Common Analog Decoders and 64 Common Analog Encoders at the same time.
eDevType	IN	<b>Device Type.</b> Number 6 for Common Analog Decoder. Number 7 for Common Analog Encoder.
hDisplayWindow	IN	<b>Display Window.</b> Pointer to one WHND window handle. If it isn't NULL, function will automatically display video on this window. If it is NULL, function will not display video on it. For encoder, it is always NULL.
pBufferCB	IN	<b>Callback Function.</b> Pointer to one callback function. If it is NULL, function will not return frame buffer to software caller. If it isn't NULL, caller will obtain frame buffer from callback when every frame is received. For encoder, it is always NULL.
pUserData	IN	<b>User Data.</b> Pointer to one data pointer. The parameters will be passed through callback. For encoder, it is always NULL.

## Return Value:

If AMESDK\_CREATE is successful, it will return one DEVICE\_HANDLE. If it is fail, it will return error code. As one of below:

0x80000000 - Parameter, pszDevName, is wrong.  
0x80000001 - Unknown error.  
0x80000002 - Device queue is full already.

## Supported Devices:

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

## Examples:

```
HWND hWnd = CreateWindowEx( ... );
```

```
BOOL bcb( double dSampleTime, BYTE * pBuffer, ULONG nBufferLen, BOOL bIsKeyFrame,  
          PVOID pUserData )  
{  
    ...  
    return TRUE;  
}
```

EX1: Create one H.264 Video Decoder Device.

```
hDev = AMESDK_CREATE( "Common Analog Decoder (H.264)", 0, 6, hWnd, bcb, param );
```

EX2: Create four MPEG4 Video Encoder Devices.

```
hDev0 = AMESDK_CREATE( "Common Analog Encoder (MPEG4)", 0, 7, NULL, NULL, NULL );  
hDev1 = AMESDK_CREATE( "Common Analog Encoder (MPEG4)", 1, 7, NULL, NULL, NULL );  
hDev2 = AMESDK_CREATE( "Common Analog Encoder (MPEG4)", 2, 7, NULL, NULL, NULL );  
hDev3 = AMESDK_CREATE( "Common Analog Encoder (MPEG4)", 3, 7, NULL, NULL, NULL );
```

EX3: Create two PCM Audio Decoder Devices.

```
hDev0 = AMESDK_CREATE( "Common Analog Decoder (PCM)", 0, 6, hWnd[ 0 ], bcb[ 0 ], param[ 0 ] );  
hDev1 = AMESDK_CREATE( "Common Analog Decoder (PCM)", 1, 6, hWnd[ 1 ], bcb[ 1 ], param[ 1 ] );
```

## Remarks:

If software developer wants to use Intel H.264 encoder/decoder in your project, you need use the library from "VC.INTEL" or "NET.INTEL". Moreover, you need download and install Intel Media SDK, too.



## 5.13 AMESDK\_GET\_FORMAT

This function is used to get current video/audio stream format.

```

BOOL AMESDK_GET_FORMAT(  DEVICE_HANDLE    hDevHandle,
                          ULONG *          pColorSpaceType,
                          ULONG *          pWidth,
                          ULONG *          pHeight,
                          ULONG *          pBitCount,
                          double *         pFrameRate,
                          ULONG *          pRecordMoe,
                          ULONG *          pBitRate,
                          ULONG *          pQuality,
                          ULONG *          pGop,
                          ULONG *          pInterlaceMode,
                          ULONG *          pAspectRatio,
                          ULONG *          pPerformanceLevel = NULL );

BOOL AMESDK_GET_FORMAT(  DEVICE_HANDLE    hDevHandle,
                          ULONG *          pChannels,
                          ULONG *          pBitsPerSample,
                          ULONG *          pSamplesPerSec
                          );

```

**Parameters:**

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose format is to be retrieved.
pColorSpace	OUT	<b>Color Space.</b> Pointer to a variable that stores the color space, in MAKEFOURCC. It cannot be NULL.
pWidth	OUT	<b>Width.</b> Pointer to a variable that stores the width, in pixels. It cannot be NULL.
pHeight	OUT	<b>Height.</b> Pointer to a variable that stores the height, in pixels. It cannot be NULL.
pBitCount	OUT	<b>Bit Count.</b> Pointer to a variable that stores the bit count. It cannot be NULL.
pFrameRate	OUT	<b>Frame Rate.</b> Pointer to a variable that stores the frame rate, in fps. It cannot be NULL.
pRecordMoe	OUT	<b>Record Mode.</b> Pointer to a variable that stores the encoder's mode. It is NULL for decoder.
pBitRate	OUT	<b>Bit Rate.</b> Pointer to a variable that stores the encoder's target bit rate. It is NULL for decoder.
pQuality	OUT	<b>Quality.</b> Pointer to a variable that stores the encoder's quality. It is NULL for decoder.
pGop	OUT	<b>GOP.</b> Pointer to a variable that stores the encoder's GOP value. It is NULL for decoder.
pInterlaceMode	OUT	<b>Interlace Mode.</b> Pointer to a variable that stores the encoder's interlace mode. Interleave mode is 1 and progressive mode is 0.
pAspectRatio	OUT	<b>Aspect Ratio.</b> Pointer to a variable that stores the encoder's aspect ratio (x:y). The 16 MSBs are used for x, and the 16 LSBs are for y.

pPerformanceLevel	OUT	<b>Performance.</b> Pointer to a variable that stores the encoder's performance parameter. It is NULL for decoder.
pChannels	OUT	<b>Audio Channels.</b> Pointer to a variable that stores the audio channels. It cannot be NULL.
pBitsPerSample	OUT	<b>Bits Per Sample.</b> Pointer to a variable that stores the number of bits per sample. It cannot be NULL.
pSamplesPerSec	OUT	<b>Sample Per Second.</b> Pointer to a variable that stores the number of sample per second. It cannot be NULL.

## Return Value:

BOOL

## Support Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

## Examples:

EX1: Get the current video encoder's format.

```
AMESDK_GET_FORMAT( hDev, &nColorSpace, &nWidth, &nHeight, &nBitCount, &dFrameRate,
                  &nRecordMoe, &nBitRate, &nQuality, &nGOP, &InterlaceMode,
                  &AspectRatio, &nPerformance );
```

EX2: Get the current audio encoder's format.

```
AMESDK_GET_FORMAT( hDev, &nChannels, &nBitsPerSample, &nSamplesPerSec );
```

## 5.14 AMESDK\_SET\_FORMAT

This function is used to configure the encoder.

```

    BOOL AMESDK_SET_FORMAT( DEVICE_HANDLE    hDevHandle,
                           ULONG             nColorSpaceType,
                           ULONG             nWidth,
                           ULONG             nHeight,
                           ULONG             nBitCount,
                           double            dFrameRate,
                           ULONG             nRecordMoe,
                           ULONG             nBitRate,
                           ULONG             nQuality,
                           ULONG             nGop,
                           ULONG             nInterlaceMode,
                           ULONG             nAspectRatio,
                           ULONG             nPerformanceLevel = 0
    );
    BOOL AMESDK_SET_FORMAT( DEVICE_HANDLE    hDevHandle,
                           ULONG             nChannels,
                           ULONG             nBitsPerSample,
                           ULONG             nSamplesPerSec
    );

```

## Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose format is to be updated.
nColorSpaceType	IN	<b>Video ColorSpace.</b> Specifies the colorspace, in MAKEFOURCC.
nWidth	IN	<b>Video Width.</b> Specifies the width, in pixels.
nHeight	IN	<b>Video Height.</b> Specifies the height, in pixels.
nBitCount	IN	<b>Video BitCount.</b> Specifies the bit count.
dFrameRate	IN	<b>Video FrameRate.</b> Specifies the frame rate, in fps.
nRecordMoe	IN	<b>Record Mode.</b> Specifies the recoder mode (VBR 0, CBR 1).
nBitRate	IN	<b>Bit Rate.</b> Specifies the bit rate in bps (0~800000000).
nQuality	IN	<b>Quality.</b> Specifies the quality (QP value, 0~10000).
nGop	IN	<b>GOP.</b> Specifies the GOP (0~255).
nInterlaceMode	IN	<b>Interlace Mode.</b> Specifies the interlace mode. Interleave mode is 1 and progressive mode is 0.
nAspectRatio	IN	<b>Aspect Ratio.</b> Specifies the aspect ratio (x:y). The 16 MSBs are used for x, and the 16 LSBs are for y. nAspectRatio = ((x << 16) & 0xFFFF0000)   ((y << 0) & 0x0000FFFF).
nPerformanceLevel	IN	<b>Performance.</b> Specifies the performance speed level (0~15). 0 is highest speed. Low speed makes better quality.

**Return Value:**

BOOL

**Support Devices:**

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

**Examples:**

EX1: Set the current video encoder's format.

```
AMESDK_SET_FORMAT( hDev,  
                   MAKEFOURCC('H', '2', '6', '4'),  
                   720, 480, 24, 29.970,  
                   0,  
                   8000000,  
                   5000,  
                   30,  
                   0,  
                   0,  
                   0  
);
```

### 5.15 AMESDK\_CODEC\_ENCODE

This function is used to encode one raw data buffer.

```

BOOL AMESDK_CODEC_ENCODE( DEVICE_HANDLE hDevHandle,
                           BYTE *        pSrcFrameBuffer,
                           ULONG         nSrcFrameColorSpaceType,
                           ULONG         nSrcFrameWidth,
                           ULONG         nSrcFrameHeight,
                           BYTE * *      ppDstStreamBuffer,
                           ULONG *       pDstStreamBufferSize,
                           BOOL *        pIsKeyFrame
);

```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be updated.
pSrcFrameBuffer	IN	<b>Frame Buffer.</b> Pointer to the address of uncompressed video source.
nSrcFrameColorSpaceType	IN	<b>Video ColorSpace.</b> Specifies the colorspace of source, in MAKEFOURCC.
nSrcFrameWidth	IN	<b>Frame Width.</b> Specifies the width of source, in pixels.
nSrcFrameHeight	IN	<b>Frame Height.</b> Specifies the height of source, in pixels.
ppDstStreamBuffer	OUT	<b>Pointer</b> of destination of compressed video.
pDstStreamBufferSize	OUT	<b>Size</b> of destination to store compressed video.
pIsKeyFrame	OUT	<b>Key Frame.</b> I frame or not.

#### Return Value:

BOOL

#### Support Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Examples:

EX1: Set the current video encoder's configure.

```

AMESDK_CODEC_ENCODE( hDev,
                     pSrcFrameBuffer,
                     nSrcFrameColorSpaceType,
                     nSrcFrameWidth,
                     nSrcFrameHeight,
                     &pDstStreamBuffer,
                     &nDstStreamBufferSize,
                     &nIsKeyFrame
);

```

### 5.16 AMESDK\_CODEC\_DECODE

This function is used to decode the stream buffer.

```
BOOL AMESDK_CODEC_DECODE( DEVICE_HANDLE hDevHandle,
                           BYTE *        pSrcStreamBuffer,
                           ULONG         nSrcStreamBufferSize,
                           BOOL          bIsKeyFrame
                           );
```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be updated.
pSrcStreamBuffer	IN	<b>Pointer</b> of address of stream buffer.
nSrcStreamBufferSize	IN	<b>Size</b> of stream buffer.
bIsKeyFrame	IN	<b>Key Frame.</b> I frame or not.

#### Return Value:

BOOL

#### Support Devices:

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Examples:

EX1: To decode one video frame.

```
AMESDK_CODEC_DECODE( hDev,
                     pSrcStreamBuffer,
                     nSrcStreamBufferSize,
                     bIsKeyFrame
                     );
```

### 5.17 AMESDK\_CODEC\_DECODE\_EX

This function is used to decode the stream buffer and to return original frame buffer right away.

```

BOOL AMESDK_CODEC_DECODE_EX( DEVICE_HANDLE  hDevHandle,
                              BYTE *         pSrcStreamBuffer,
                              ULONG          nSrcStreamBufferSize,
                              BOOL           bIsKeyFrame,
                              BYTE *         pDstFrameBuffer,
                              LONG *         pDstFrameBufferSize );

```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be updated.
pSrcStreamBuffer	IN	<b>Pointer</b> of address of stream buffer.
nSrcStreamBufferSize	IN	<b>Size</b> of stream buffer.
bIsKeyFrame	IN	<b>Key Frame.</b> I frame or not.
pDstFrameBuffer	OUT	<b>Pointer</b> of frame buffer of destination.
pDstFrameBufferSize	OUT	<b>Size</b> of frame buffer of destination.

#### Return Value:

BOOL

#### Support Devices:

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Examples:

EX1: To decode one video frame.

```

AMESDK_CODEC_DECODE_EX( hDev,
                        pSrcFrameBuffer,
                        nSrcStreamBufferSize,
                        bIsKeyFrame,
                        pDstFrameBuffer,
                        &nDstFrameBufferSize
);

```

## **6 Exported Functions for Network Server and Client Programming**

SUPPORT DEVICE:

PD652, SC100, SC200, SC230, SC280,  
SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0,  
SC3B0, SC500, SC510, SC580, SC590,



Exported Functions for Network Server and Client Programming	
6.01	AMESDK_CREATE (SOURCE/RENDERER)
6.02	AMESDK_DESTROY (SOURCE/RENDERER) (SEE 1.02)
6.03	AMESDK_RUN (SOURCE/RENDERER) (SEE 1.03)
6.04	AMESDK_STOP (SOURCE/RENDERER) (SEE 1.04)
6.05	AMESDK_GET_DEINTERLACE (SOURCE) (SEE 1.12)
6.06	AMESDK_SET_DEINTERLACE (SOURCE) (SEE 1.13)
6.07	AMESDK_GET_MIRROR (SOURCE) (SEE 1.14)
6.08	AMESDK_SET_MIRROR (SOURCE) (SEE 1.15)
6.09	AMESDK_OTHER_REFRESH_DISPLAY_WINDOW (SOURCE) (SEE 1.30)
6.10	AMESDK_OTHER_SNAPSHOT_BMP (SOURCE) (SEE 1.31)
6.11	AMESDK_OTHER_SNAPSHOT_JPG (SOURCE) (SEE 1.32)
6.12	AMESDK_OTHER_ZOOM (SOURCE) (SEE 1.33)
6.12	AMESDK_NETWORK_SET_VIDEO_STREAM_FORMAT (RENDERER)
6.13	AMESDK_NETWORK_SET_AUDIO_STREAM_FORMAT (RENDERER)
6.14	AMESDK_NETWORK_SET_VIDEO_STREAM_BUFFER (RENDERER)
6.15	AMESDK_NETWORK_SET_AUDIO_STREAM_BUFFER (RENDERER)
6.16	AMESDK_NETWORK_GET_VIDEO_STREAM_STATISTICS (RENDERER)
6.17	AMESDK_NETWORK_GET_AUDIO_STREAM_STATISTICS (RENDERER)
6.18	AMESDK_NETWORK_SET_STREAMING_ADAPTER (RENDERER)
6.19	AMESDK_NETWORK_SET_STREAMING_PORT (RENDERER)
6.20	AMESDK_NETWORK_SET_USER_ACCOUNT (RENDERER)
6.21	AMESDK_NETWORK_SET_CALLBACK (RENDERER)
6.22	AMESDK_NETWORK_SET_CUSTOM_PROPERTY (SOURCE)
6.23	AMESDK_NETWORK_GET_CUSTOM_PROPERTY (SOURCE)
6.24	AMESDK_NETWORK_CONNECT_STREAMING_SERVER (SOURCE)
6.25	A Network Server Software Programming Guide
6.26	A Network Client Software Programming Guide

## 6.01 AMESDK\_CREATE

The function helps you to create a network streaming server and client. In SDK, we regard the streaming server as one network renderer device and it can output video and audio streams to internet. On the other hand, the network source device is one streaming client. Your software can use it to receive video and audio streams from specific network streaming server. For a network source device, it also allows you to attach a preview window on it. If the parameter is not NULL, our SDK can help you to display video and audio on this window. Of course, we also allow you to register a callback function on it. The callback function will offer you to obtain and to access the whole frame buffer.

The network function is based on standard RTSP protocol to transfer data between internets. Currently, we support RTSP over UDP and RTSP over TCP both solutions. Developer can use VLC to test network streaming server.

```

DEVICE_HANDLE AMESDK_CREATE( LPTSTR          pszDevName,
                             UINT           iDevNum,
                             ULONG          eDevType,
                             HWND           hDisplayWindow_Video,
                             PF_BUFFER_CALLBACK pBufferCB_Video,
                             PVOID          pUserData_Video,
                             HWND           hDisplayWindow_Audio,
                             PF_BUFFER_CALLBACK pBufferCB_Audio,
                             PVOID          pUserData_Audio
);
typedef ULONG (DEVICE_HANDLE);
    
```

### Parameters:

Parameter	IN/OUT	Description
pszDevName	IN	<b>Device Name.</b> To give a device name that is used to create specific device. Currently, we support these device names below: "Common Analog Network Source %s". "Common Analog Network Renderer %s". Here, %s is your URL path and initial parameters for device initialization. Please reference subsection, <b>Examples</b> , to obtain more introductions.
iDevNum	IN	<b>Device Number.</b> Here, SDK supports you to create 64 Common Analog Network Sources and 64 Common Analog Network Renderers at the same time.
eDevType	IN	<b>Device Type.</b> Number 4 for Common Analog Network Source. Number 5 for Common Analog Network Renderer.
hDisplayWindow	IN	<b>Display Window.</b> Pointer to one WHND window handle. If it isn't NULL, function will automatically play video and sound on this window. If it is NULL, function will hide on it. For network renderer, it is always NULL.
pBufferCB	IN	<b>Callback Function.</b> Pointer to one callback function. If it is NULL, function will not return frame buffer to software caller. If it isn't NULL, caller will obtain frame buffer from callback when every frame is coming. For network renderer, it is always NULL.
pUserData	IN	<b>User Data.</b> Pointer to one data pointer. The parameters will be passed through callback. For network renderer, it is always NULL.

## Return Value:

If AMESDK\_CREATE is successful, it will return one DEVICE\_HANDLE. If it is fail, it will show the error code. As one of below:

0x80000000 - Parameter, pszDevName, is wrong.  
0x80000001 - Unknown error.  
0x80000002 - Device queue is full already.

## Support Capture Devices:

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

## Support Network Devices:

SOURCE, RENDERER

## Examples:

EX1: Create a network streaming server to support 4 media streams at the same time. Here, every media stream can support one video and one audio substreams.

```
hDev = AMESDK_CREATE( "Common Analog Network Renderer streams=4",  
                    0,  
                    5,  
                    NULL,  
                    NULL,  
                    NULL );
```

EX2: Open a client proxy to receive media stream from server without authorization. Developer can use the URL on VLC player, too.

```
hDev = AMESDK_CREATE( "Common Analog Network Source url=rtsp://10.10.10.1/session0.mpg",  
                    0,  
                    4,  
                    hWnd,  
                    NULL,  
                    NULL );
```

EX3: Open a client proxy to receive media stream from server with authorization. Developer can use the URL on VLC player, too.

```
hDev = AMESDK_CREATE( "Common Analog Network Source url=rtsp://root:root@10.10.10.1/session0.mpg",  
                    0,  
                    4,  
                    hWnd,  
                    NULL,  
                    NULL );
```

## 6.12 AMESDK\_NETWORK\_SET\_VIDEO\_STREAM\_FORMAT

This function is to set video stream format.

```

BOOL AMESDK_NETWORK_SET_VIDEO_STREAM_FORMAT(
    DEVICE_HANDLE    hDevHandle,
    ULONG            nStreamNumber,
    ULONG            nColorSpaceType,
    ULONG            nWidth,
    ULONG            nHeight,
    ULONG            nBitCount,
    double           dFrameRate,
    ULONG            nBandwidth
);

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be updated.
nStreamNumber	IN	<b>Stream Number.</b> Specifies which channel stream to set.
nColorSpaceType	IN	<b>Color Space.</b> Specifies the color space type of the video.
nWidth	IN	<b>Width.</b> Specifies the width of the video.
nHeight	IN	<b>Height.</b> Specifies height of the video.
nBitCount	IN	<b>Bit Count.</b> Specifies the bit count of the video.
dFrameRate	IN	<b>Frame Rate.</b> Specifies the frame rate of the video.
nBandwidth	IN	<b>Bandwidth.</b> Specifies the bandwidth for the video

### Return Values:

BOOL

### Support Capture Devices:

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support Network Devices:

RENDERER

### Examples:

EX1: Set video stream format to NTSC.

```
AMESDK_NETWORK_SET_VIDEO_STREAM_FORMAT( hDev, 0, 0x34363248, 704, 480, 24, 29.97, 4000000 );
```

EX2: Set video stream format to PAL.

```
AMESDK_NETWORK_SET_VIDEO_STREAM_FORMAT( hDev, 0, 0x34363248, 704, 576, 24, 25.00, 4000000 );
```

### 6.13 AMESDK\_NETWORK\_SET\_AUDIO\_STREAM\_FORMAT

This function is to set audio stream format.

```

    BOOL AMESDK_NETWORK_SET_AUDIO_STREAM_FORMAT(
        DEVICE_HANDLE    hDevHandle,
        ULONG             nStreamNumber,
        ULONG             nChannels,
        ULONG             nBitsPerSample,
        ULONG             nSamplesPerSec,
        ULONG             nBandwidth
    );

```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be updated.
nStreamNumber	IN	<b>Stream Number.</b> Specifies which channel stream to set.
nChannels	IN	<b>Channel Number.</b> Specifies the audio channel of the audio stream. (1:Mono, 2:Stereo)
nBitsPerSample	IN	<b>Bits Per Sample.</b> Specifies the bit count of the sampling rate.
nSamplesPerSec	IN	<b>Samples Per Second.</b> Specifies the number of sample per second.
nBandwidth	IN	<b>Bandwidth.</b> Specifies the bandwidth for the audio.

#### Return Values:

BOOL

#### Support Capture Devices:

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Support Network Devices:

RENDERER

#### Examples:

Ex1: Set audio stream format to Mono.

```
AMESDK_NETWORK_SET_AUDIO_STREAM_FORMAT( hDev, 0, 1, 16, 8000, 128000 );
```

Ex1: Set audio stream format to Stereo.

```
AMESDK_NETWORK_SET_AUDIO_STREAM_FORMAT( hDev, 0, 2, 16, 48000, 1536000 );
```

## 6.14 AMESDK\_NETWORK\_SET\_VIDEO\_STREAM\_BUFFER

This function is to set stream buffer for video stream.

```

BOOL AMESDK_NETWORK_SET_VIDEO_STREAM_BUFFER(
    DEVICE_HANDLE    hDevHandle,
    ULONG            nStreamNumber,
    BYTE *           pStreamBuffer,
    ULONG            nStreamBufferSize,
    BOOL             bIsKeyFrame,
    ULONGLONG        nTimeStamp = 0x0000000000000000
);

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be updated.
nStreamNumber	IN	<b>Stream Number.</b> Specifies which channel stream to set.
pStreamBuffer	IN	<b>Stream Buffer.</b> Specifies the start address of video stream buffer.
nStreamBufferSize	IN	<b>Stream Buffer Size.</b> Specifies the buffer size of the video stream.
bIsKeyFrame	IN	<b>Key Frame.</b> Specifies I frame or not.
nTimeStamp	IN	<b>Time Stamp.</b> Specifies the time stamp.

### Return Values:

BOOL

### Support Capture Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support Network Devices:

RENDERER

### Examples:

Ex1: To set stream buffer for video stream.

```
AMESDK_NETWORK_SET_VIDEO_STREAM_BUFFER( hDev, 0, pStreamBuffer, 0x00096000, TRUE );
```

### 6.15 AMESDK\_NETWORK\_SET\_AUDIO\_STREAM\_BUFFER

This function is to set stream buffer for audio stream.

```

BOOL AMESDK_NETWORK_SET_AUDIO_STREAM_BUFFER(
    DEVICE_HANDLE    hDevHandle,
    ULONG            nStreamNumber,
    BYTE *           pStreamBuffer,
    ULONG            nStreamBufferSize,
    ULONGLONG        nTimeStamp = 0x0000000000000000
);

```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be updated.
nStreamNumber	IN	<b>Stream Number.</b> Specifies which channel stream to set.
pStreamBuffer	IN	<b>Stream Buffer.</b> Specifies the start address of audio stream buffer.
nStreamBufferSize	IN	<b>Stream Buffer Size.</b> Specifies the buffer size of the audio stream.
nTimeStamp	IN	<b>Time Stamp.</b> Specifies the time stamp.

#### Return Values:

BOOL

#### Support Capture Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Support Network Devices:

RENDERER

#### Examples:

Ex1: To set stream buffer for audio stream.

```
AMESDK_NETWORK_SET_AUDIO_STREAM_BUFFER( hDev, 0, pStreamBuffer, 320 );
```

## 6.16 AMESDK\_NETWORK\_GET\_VIDEO\_STREAM\_STATISTICS

This function is to check the current status of network connection. Programmer can use this function to design QOS function in the streaming server.

```

BOOL AMESDK_NETWORK_GET_VIDEO_STREAM_STATISTICS (
    DEVICE_HANDLE    hDevHandle,
    ULONG            nStreamNumber,
    ULONG *          pBufferQueueSize,
    ULONG *          pFrameQueueSize,
    ULONG *          pClients
);

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be retrieved.
nStreamNumber	IN	<b>Stream Number.</b> Specifies which channel stream to get.
pBufferQueueSize	OUT	<b>Buffer Queue Size.</b> Specifies queue size in byte.
pFrameQueueSize	OUT	<b>Frame Queue Size.</b> Specifies queue size in frame.
pClients	OUT	<b>Number Of Client.</b> Specifies the quantity of clients.

### Return Values:

BOOL

### Support Capture Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support Network Devices:

RENDERER

### Examples:

EX1: To obtain current bandwidth status of network connection.

```

AMESDK_NETWORK_GET_VIDEO_STREAM_STATISTICS( hDev,
    0,
    &nBufferQueueSize,
    &nFrameQueueSize,
    &nClients );

```



### 6.17 AMESDK\_NETWORK\_GET\_AUDIO\_STREAM\_STATISTICS

This function is to check current bandwidth status of network connection. Programmer can use this function to design QOS function in the streaming server.

```

BOOL AMESDK_NETWORK_GET_AUDIO_STREAM_STATISTICS (
    DEVICE_HANDLE    hDevHandle,
    ULONG            nStreamNumber,
    ULONG *          pBufferQueueSize,
    ULONG *          pFrameQueueSize,
    ULONG *          pClients
);

```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be retrieved.
nStreamNumber	IN	<b>Stream Number.</b> Specifies which channel stream to get.
pBufferQueueSize	OUT	<b>Buffer Queue Size.</b> Specifies queue size in byte.
pFrameQueueSize	OUT	<b>Frame Queue Size.</b> Specifies queue size in frame.
pClients	OUT	<b>Number Of Client.</b> Specifies the quantity of clients.

#### Return Values:

BOOL

#### Support Capture Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Support Network Devices:

RENDERER

#### Examples:

EX1: To obtain current bandwidth status of network connection.

```

AMESDK_NETWORK_GET_AUDIO_STREAM_STATISTICS( hDev,
    0,
    &nBufferQueueSize,
    &nFrameQueueSize,
    &nClients );

```

## 6.18 AMESDK\_NETWORK\_SET\_STREAMING\_ADAPTER

This function is to select outputted network adapter.

```
BOOL AMESDK_NETWORK_SET_STREAMING_ADAPTER(  
    DEVICE_HANDLE hDevHandle,  
    CHAR *        pszOutgoingNetworkInterface  
);
```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be updated.
pszOutgoingNetworkInterface	IN	<b>Network Adapter.</b> Specifies the adapter's address.

### Return Values:

BOOL

### Support Capture Devices:

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support Network Devices:

RENDERER

### Examples:

EX1: To setup local network adapter as output adapter.

```
AMESDK_NETWORK_SET_STREAMING_ADAPTER( hDev, "127.0.0.1" );
```

## 6.19 AMESDK\_NETWORK\_SET\_STREAMING\_PORT

This function is to set port for streaming.

```
BOOL AMESDK_NETWORK_SET_STREAMING_PORT( DEVICE_HANDLE hDevHandle,  
                                         USHORT        wPort  
                                         );
```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be updated.
wPort	IN	<b>Port number.</b> Specifies the port number.

### Return Values:

BOOL

### Support Capture Devices:

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support Network Devices:

RENDERER

### Examples:

EX1: To setup streaming port.

```
AMESDK_NETWORK_SET_STREAMING_PORT( hDev, 554 );
```

## 6.20 AMESDK\_NETWORK\_SET\_USER\_ACCOUNT

This function is to set user account and password.

```
BOOL AMESDK_NETWORK_SET_USER_ACCOUNT( DEVICE_HANDLE hDevHandle,  
                                       CHAR *         pszUserName,  
                                       CHAR *         pUserPassword  
);
```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be updated.
pszUserName	IN	<b>User Name.</b> Specifies the user name.
pUserPassword	IN	<b>Password.</b> Specifies the pass word.

### Return Values:

BOOL

### Support Capture Devices:

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support Network Devices:

RENDERER

### Examples:

EX1: To setup user name and password.

```
AMESDK_NETWORK_SET_USER_ACCOUNT( hDev, "root", "root" );
```

## 6.21 AMESDK\_NETWORK\_SET\_CALLBACK

This function is to register some callback functions on server side. All 4 callback functions must be implemented before using AMESDK\_NETWORK\_GET\_CUSTOM\_PROPERTY() and AMESDK\_NETWORK\_SET\_CUSTOM\_PROPERTY().

```

BOOL AMESDK_NETWORK_SET_CALLBACK(

    DEVICE_HANDLE                hDevHandle,
    PF_NETWORK_JOIN_SESSION_CALLBACK pJoinSessionCB,
    PF_NETWORK_LEAVE_SESSION_CALLBACK pLeaveSessionCB,
    PF_NETWORK_SET_CUSTOM_PROPERTY_CALLBACK pSetCustomPropertyCB,
    PF_NETWORK_GET_CUSTOM_PROPERTY_CALLBACK pGetCustomPropertyCB,
    PVOID                        pUserData

);

typedef BOOL (AMESDK_EXPORT *PF_NETWORK_JOIN_SESSION_CALLBACK) (
    ULONG nStreamNumber, CHAR * pszUserName, ULONG dwClientAddress, PVOID pUserData );

typedef BOOL (AMESDK_EXPORT *PF_NETWORK_LEAVE_SESSION_CALLBACK) (
    ULONG nStreamNumber, CHAR * pszUserName, ULONG dwClientAddress, PVOID pUserData );

typedef BOOL (AMESDK_EXPORT *PF_NETWORK_SET_CUSTOM_PROPERTY_CALLBACK) (
    ULONG nStreamNumber, CHAR * pszProperty, CHAR * pszValue, PVOID pUserData );

typedef CHAR * (AMESDK_EXPORT *PF_NETWORK_GET_CUSTOM_PROPERTY_CALLBACK) (
    ULONG nStreamNumber, CHAR * pszProperty, PVOID pUserData );

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be updated.
pJoinSessionCB	IN	<b>Pointer</b> of join session callback function.
pLeaveSessionCB	IN	<b>Pointer</b> of leave session callback function.
pSetCustomPropertyCB	IN	<b>Pointer</b> of set custom property callback function.
pGetCustomPropertyCB	IN	<b>Pointer</b> of get custom property callback function.
pUserData	IN	<b>User data.</b> Pointer to the address of user data.

### Return Values:

BOOL

## Support Capture Devices:

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

## Support File Devices:

RENDERER

## Examples:

```
BOOL on_join_session( ULONG nStreamNumber,
                     CHAR * pszUserName,
                     ULONG dwClientAddress,
                     PVOID pUserData )
{
    ...
}

BOOL on_leave_session( ULONG nStreamNumber,
                     CHAR * pszUserName,
                     ULONG dwClientAddress,
                     PVOID pUserData )
{
    ...
}

BOOL on_set_custom_property( ULONG nStreamNumber,
                           CHAR * pszProperty,
                           CHAR * pszValue,
                           PVOID pUserData )
{
    // RECEIVE COMMAND OF AMESDK_NETWORK_SET_CUSTOM_PROPERTY ISSUED BY CLIENT.
}

CHAR * on_get_custom_property( ULONG nStreamNumber,
                             CHAR * pszProperty,
                             PVOID pUserData )
{
    // RECEIVE COMMAND OF AMESDK_NETWORK_GET_CUSTOM_PROPERTY ISSUED BY CLIENT.
}

AMESDK_NETWORK_SET_CALLBACK( hDevHandle,
                           on_join_session,
                           on_leave_session,
                           on_set_custom_property,
                           on_get_custom_property );
```

## 6.22 AMESDK\_NETWORK\_SET\_CUSTOM\_PROPERTY

This function is used to set one custom property to server by client. At server side, it is necessary to design AMESDK\_NETWORK\_SET\_CALLBACK() before using this function.

```
BOOL AMESDK_NETWORK_SET_CUSTOM_PROPERTY( DEVICE_HANDLE hDevHandle,
                                          CHAR *        pszProperty,
                                          CHAR *        pszValue,
                                          ULONG         nBytes
);
```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be updated.
pszProperty	IN	<b>Property.</b> Specifies the property that will be set to the server.
pszValue	IN	<b>Property Value.</b> Specifies the property value. The range of value is dependent on its property.
nBytes	IN	<b>Size.</b> Size of pszValue

### Return Values:

BOOL

### Support Capture Devices:

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support Network Devices:

SOURCE

### Examples:

EX1: To set/change custom property to server.

```
AMESDK_NETWORK_SET_CUSTOM_PROPERTY( hDev, "ptz", p_cmd, sizeof(p_cmd) );
```

### 6.23 AMESDK\_NETWORK\_GET\_CUSTOM\_PROPERTY

This function is used to get one custom property from server by client. At server side, it is necessary to design AMESDK\_NETWORK\_SET\_CALLBACK() before using this function.

```

    BOOL AMESDK_NETWORK_GET_CUSTOM_PROPERTY( DEVICE_HANDLE  hDevHandle,
                                              CHAR *         pszProperty,
                                              CHAR *         pszValue,
                                              ULONG          nBytes
    );

```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be updated.
pszProperty	IN	<b>Property.</b> Specifies the property that will be gotten from the server.
pszValue	IN	<b>Property Value.</b> Specifies the property value. The range of value is dependent on its property.
nBytes	IN	<b>Size.</b> Size of pszValue

#### Return Values:

BOOL

#### Support Capture Devices:

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Support Network Devices:

SOURCE

#### Examples:

EX1: To get custom property from server.

```
AMESDK_NETWORK_GET_CUSTOM_PROPERTY( hDev, "gps", p_gps, sizeof(p_gps) );
```



## 6.24 AMESDK\_NETWORK\_CONNECT\_STREAMING\_SERVER

This function is to connect to streaming server with UDP/TCP protocol.

```
BOOL AMESDK_NETWORK_CONNECT_STREAMING_SERVER(  
    DEVICE_HANDLE hDevHandle,  
    ULONG nProtocol = 0x00000000  
);
```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be updated.
nProtocol	IN	<b>Protocol.</b> UDP is 0, TCP is 1.

### Return Values:

BOOL

### Support Capture Devices:

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support Network Devices:

SOURCE

### Examples:

EX1: Connect to streaming server with UDP protocol.

```
AMESDK_NETWORK_CONNECT_STREAMING_SERVER( hDev, 0x00000000 ); // UDP
```

EX2: Connect to streaming server with TCP protocol.

```
AMESDK_NETWORK_CONNECT_STREAMING_SERVER( hDev, 0x00000001 ); // TCP
```

## **7 Exported Functions for File Record and Playback Programming**

SUPPORT DEVICE:

PD652, SC100, SC200, SC230, SC280,  
SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0,  
SC3B0, SC500, SC510, SC580, SC590,

Exported Functions for File Record and Playback Programming	
7.01	AMESDK_CREATE (SOURCE/RENDERER)
7.02	AMESDK_DESTROY (SOURCE/RENDERER) (SEE 1.02)
7.03	AMESDK_RUN (SOURCE) (SEE 1.03)
7.04	AMESDK_STOP (SOURCE) (SEE 1.04)
7.05	AMESDK_PAUSE (SOURCE)
7.06	AMESDK_STEP (SOURCE)
7.07	AMESDK_GET_DEINTERLACE (SOURCE) (SEE 1.12)
7.08	AMESDK_SET_DEINTERLACE (SOURCE) (SEE 1.13)
7.09	AMESDK_GET_MIRROR (SOURCE) (SEE 1.14)
7.10	AMESDK_SET_MIRROR (SOURCE) (SEE 1.15)
7.11	AMESDK_OTHER_REFRESH_DISPLAY_WINDOW (SOURCE) (SEE 1.30)
7.12	AMESDK_OTHER_SNAPSHOT_BMP (SOURCE) (SEE 1.31)
7.13	AMESDK_OTHER_SNAPSHOT_JPG (SOURCE) (SEE 1.32)
7.14	AMESDK_OTHER_ZOOM (SOURCE) (SEE 1.33)
7.15	AMESDK_GET_VOLUME (SOURCE) (SEE 3.07)
7.16	AMESDK_SET_VOLUME (SOURCE) (SEE 3.08)
7.17	AMESDK_FILE_GET_VIDEO_STREAM_FORMAT (SOURCE/RENDERER)
7.18	AMESDK_FILE_SET_VIDEO_STREAM_FORMAT (RENDERER)
7.19	AMESDK_FILE_GET_AUDIO_STREAM_FORMAT (SOURCE/RENDERER)
7.20	AMESDK_FILE_SET_AUDIO_STREAM_FORMAT (RENDERER)
7.21	AMESDK_FILE_SET_VIDEO_STREAM_BUFFER (SOURCE/RENDERER)
7.22	AMESDK_FILE_SET_AUDIO_STREAM_BUFFER (RENDERER)
7.23	AMESDK_FILE_GET_VIDEO_STREAM_BUFFER (SOURCE/RENDERER)
7.24	AMESDK_FILE_GET_AUDIO_STREAM_BUFFER (RENDERER)
7.25	AMESDK_FILE_GET_VIDEO_STREAM_DATA (SOURCE)
7.26	AMESDK_FILE_GET_AUDIO_STREAM_DATA (SOURCE)
7.27	AMESDK_FILE_GET_VIDEO_TIMESTAMP_MAP (SOURCE)
7.28	AMESDK_FILE_GET_AUDIO_TIMESTAMP_MAP (SOURCE)
7.29	AMESDK_FILE_GET_VIDEO_SECTION_MAP (SOURCE)
7.30	AMESDK_FILE_GET_AUDIO_SECTION_MAP (SOURCE)
7.31	AMESDK_FILE_GET_VIDEO_MOTION_MAP (SOURCE)
7.32	AMESDK_FILE_GET_VIDEO_GPS_MAP (SOURCE)
7.33	AMESDK_FILE_GET_VIDEO_POS_MAP (SOURCE)
7.34	AMESDK_FILE_GET_MEDIA_LENGTH (SOURCE)
7.35	AMESDK_FILE_GET_MEDIA_POSITION (SOURCE)
7.36	AMESDK_FILE_SET_MEDIA_POSITION (SOURCE)
7.37	AMESDK_FILE_GET_MEDIA_PLAYBACK_RATE (SOURCE)
7.38	AMESDK_FILE_SET_MEDIA_PLAYBACK_RATE (SOURCE)
7.39	AMESDK_FILE_GET_MEDIA_PLAYBACK_SKIP_MODE (SOURCE)

Exported Functions for File Record and Playback Programming	
7.40	AMESDK_FILE_SET_MEDIA_PLAYBACK_SKIP_MODE (SOURCE)
7.41	AMESDK_FILE_SEEK (SOURCE)
7.42	AMESDK_FILE_FLUSH (RENDERER)
7.43	AMESDK_FILE_DELETE_THE_OLDEST_FILE (SOURCE/RENDERER)
7.44	AMESDK_FILE_GET_FILE_INFO_LIST (SOURCE/RENDERER)
7.45	AMESDK_FILE_UPDATE_FILE_INFO_LIST (SOURCE/RENDERER)
7.46	AMESDK_FILE_FREE_FILE_INFO_LIST (SOURCE/RENDERER)
7.47	AMESDK_FILE_EXPORT (SOURCE/RENDERER)
7.48	AMESDK_FILE_SELF_REPAIR (SOURCE/RENDERER)
7.49	AMESDK_FILE_REPAIR (SOURCE/RENDERER)
7.50	AMESDK_FILE_DIAGNOSIS (SOURCE/RENDERER)
7.51	AMESDK_FILE_STD_RESTORE (SOURCE/RENDERER)
7.52	An AVI File Writer Software Programming Guide (RENDERER)
7.53	An AVI File Player Software Programming Guide (SOURCE)

## 7.01 AMESDK\_CREATE

The function helps you to access a media file. In SDK, we see the media file as one file source device and it can output video and audio streams to your software. On the other hand, the file renderer device is one file writer. Your software can use it to record video and audio streams into specific media file. For a file source device, it also allows you to attach a preview window on it. If the parameter is not NULL, our SDK can help you to display video and audio on this window.

```

DEVICE_HANDLE AMESDK_CREATE( LPTSTR          pszDevName,
                             UINT           iDevNum,
                             ULONG          eDevType,
                             HWND           hDisplayWindow,
                             PF_BUFFER_CALLBACK pBufferCB,
                             PVOID          pUserData
);
typedef ULONG (DEVICE_HANDLE);

```

### Parameters:

Parameter	IN/OUT	Description
pszDevName	IN	<b>Device Name.</b> To give a device name that is used to create specific device. Currently, we support these device names below: "Common Analog File Source %s". "Common Analog File Renderer %s". Here, %s is your file path of the media file.
iDevNum	IN	<b>Device Number.</b> Here, SDK supports you to create 64 Common Analog File Sources and 64 Common Analog File Renderers at the same time.
eDevType	IN	<b>Device Type.</b> Number 2 for Common Analog File Source. Number 3 for Common Analog File Renderer.
hDisplayWindow	IN	<b>Display Window.</b> Pointer to one WHND window handle. If it isn't NULL, function will automatically play video and sound on this window. If it is NULL, function will hide on it. For file renderer, it is always NULL.
pBufferCB	IN	<b>Callback Function.</b> It is always NULL.
pUserData	IN	<b>User Data.</b> It is always NULL.

### Return Value:

If AMESDK\_CREATE is successful, it will return one DEVICE\_HANDLE. If it is fail, it will show the error code. As one of below:

```

0x80000000 - Parameter, pszDevName, is wrong.
0x80000001 - Unknown error.
0x80000002 - Device queue is full already.

```

### Support Capture Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

## Support File Devices:

SOURCE, RENDERER

## Examples:

EX1: Open a media file then play it on an attached window.

```
hDev = AMESDK_CREATE( "Common Analog File Source C:\\200812050000.AVI",  
                      0,  
                      2,  
                      hWnd,  
                      NULL,  
                      NULL );
```

EX2: Create a media file writer to record video and audio streams.

```
hDev = AMESDK_CREATE( "Common Analog File Renderer C:\\200812050000.AVI",  
                      0,  
                      3,  
                      NULL,  
                      NULL,  
                      NULL );
```

EX3: Create two media file writers at the same time.

```
hDev = AMESDK_CREATE( "Common Analog File Renderer C:\\200812050000.AVI",  
                      0,  
                      3,  
                      NULL,  
                      NULL,  
                      NULL );
```

```
hDev = AMESDK_CREATE( "Common Analog File Renderer C:\\200812060000.AVI",  
                      1,  
                      3,  
                      NULL,  
                      NULL,  
                      NULL );
```

## 7.05 AMESDK\_PAUSE

This function is used to pause the device.

```
BOOL AMESDK_PAUSE( DEVICE_HANDLE hDevHandle );
```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device that is to be paused.

### Return Value:

BOOL

### Support Capture Devices:

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support File Devices:

SOURCE

### Examples:

EX1: Pause the device.

```
AMESDK_PAUSE( hDev );
```

## 7.06 AMESDK\_STEP

This function is used to step forward by the specified number of frames.

```
BOOL AMESDK_STEP( DEVICE_HANDLE hDevHandle,
                  ULONG          nSteps );
```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device that is to be stepped.
nSteps	IN	<b>Steps.</b> Specifies the number of frames to skip. If nSteps is 1, the file source steps forward one frame. If nSteps is a number n greater than 1, the graph skips n - 1 frames and shows the n-th frame.

### Return Values:

BOOL

### Support Capture Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support File Devices:

SOURCE

### Examples:

EX1: Step to next frame.

```
AMESDK_STEP( hDev, 1 );
```

EX2: Skip 5 frames and show the next 6th frame.

```
AMESDK_STEP( hDev, 6 );
```



**7.17 AMESDK\_FILE\_GET\_VIDEO\_STREAM\_FORMAT**

This function is used to get the video stream format from one media file.

```

BOOL AMESDK_FILE_GET_VIDEO_STREAM_FORMAT( DEVICE_HANDLE  hDevHandle,
                                           ULONG *        pColorSpaceType,
                                           ULONG *        pWidth,
                                           ULONG *        pHeight,
                                           ULONG *        pBitCount,
                                           DOUBLE *       pFrameRate,
                                           DWORD *        pCustomFlags );

```

**Parameters:**

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose video stream format is to be retrieved.
pColorSpaceType	OUT	<b>Color Space.</b> Pointer to a variable that stores the color space type of the video.
pWidth	OUT	<b>Width.</b> Pointer to a variable that stores the width of the video.
pHeight	OUT	<b>Height.</b> Pointer to a variable that stores the height of the video.
pBitCount	OUT	<b>Bit Count.</b> Pointer to a variable that stores the bit count of the video.
pFrameRate	OUT	<b>Frame Rate.</b> Pointer to a variable that stores the frame rate of the video.
pCustomFlags	OUT	<b>Custom Flags.</b> Pointer to a variable that stores the custom flags of the video stream. The following flags are defined: <b>AMESDK_FILE_CUSTOMFLAG_ISINTERLEAVED</b> Indicates this video stream is interleaved. It need do deinterlace during playback. <b>AMESDK_FILE_CUSTOMFLAG_ISSYNCHRONIZATION</b> Indicates this video stream need enable one post-sync algorithm to re-calculate correct playback frame rate. It is suggested to be used by all live data cards, such as SC100, SC200, SC230, SC300, SC310, SC330, SC340 and SC500. <b>AMESDK_FILE_CUSTOMFLAG_NONINDEX</b> Indicates timestamp information will not be inserted into recording file. Note!! Non-index recording file cannot support file export function.

**Return Values:**

BOOL

**Support Capture Devices:**

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310, SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

## Support File Devices:

SOURCE, RENDERER

## Examples:

EX1: Get the video stream format from file.

```
AMESDK_FILE_GET_VIDEO_STREAM_FORMAT( hDev,
                                     &nColorSpaceType,
                                     &nWidth,
                                     &nHeight,
                                     &nBitCount,
                                     &nFrameRate
                                     &nCustomFlags );

if( nCustomFlags & AMESDK_FILE_CUSTOMFLAG_ISINTERLEAVED ) {
    AMESDK_SET_DEINTERLACE( hDev, 0x00000007 );
}
```

**7.18 AMESDK\_FILE\_SET\_VIDEO\_STREAM\_FORMAT**

This function is used to set/change the video stream format.

```

BOOL AMESDK_FILE_SET_VIDEO_STREAM_FORMAT( DEVICE_HANDLE  hDevHandle,
                                           ULONG          nColorSpaceType,
                                           ULONG          nWidth,
                                           ULONG          nHeight,
                                           ULONG          nBitCount,
                                           DOUBLE         nFrameRate,
                                           DWORD          dwCustomFlags = 0 );

```

**Parameters:**

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose video stream format is to be set.
nColorSpaceType	IN	<b>Color Space.</b> Specifies the color space type of the video.
nWidth	IN	<b>Width.</b> Specifies the width of the video.
nHeight	IN	<b>Height.</b> Specifies height of the video.
nBitCount	IN	<b>Bit Count.</b> Specifies the bit count of the video.
nFrameRate	IN	<b>Frame Rate.</b> Specifies the frame rate of the video.
dwCustomFlags	IN	<b>Custom Flags.</b> Custom flags for the video stream. The following flags are defined: <b>AMESDK_FILE_CUSTOMFLAG_ISINTERLEAVED</b> Indicates this video stream is interleaved. It need do deinterlace during playback. <b>AMESDK_FILE_CUSTOMFLAG_ISSYNCHRONIZATION</b> Indicates this video stream need enable one post-sync algorithm to re-calculate playback frame rate. It is suggested to be used by all live data cards, such as SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310, SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590. <b>AMESDK_FILE_CUSTOMFLAG_NONINDEX</b> Indicates timestamp information will not be inserted into recording file. Note!! Non-index recording file cannot support file export function.

**Return Values:**

BOOL

**Support Capture Devices:**

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310, SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

**Support File Devices:**

RENDERER

## Examples:

EX1: Set video stream format.

```
AMESDK_FILE_SET_VIDEO_STREAM_FORMAT( hDev, 0x44495658 /*XVID*/, 704, 480, 24, 29.970 );
AMESDK_FILE_SET_VIDEO_STREAM_FORMAT( hDev, 0x58564944 /*DIVX*/, 704, 480, 24, 29.970 );
AMESDK_FILE_SET_VIDEO_STREAM_FORMAT( hDev, 0x34363248 /*H264*/, 704, 480, 24, 29.970 );
AMESDK_FILE_SET_VIDEO_STREAM_FORMAT( hDev, 0x44495658 /*XVID*/, 704, 576, 24, 25.000 );
AMESDK_FILE_SET_VIDEO_STREAM_FORMAT( hDev, 0x58564944 /*DIVX*/, 704, 576, 24, 25.000 );
AMESDK_FILE_SET_VIDEO_STREAM_FORMAT( hDev, 0x34363248 /*H264*/, 704, 576, 24, 25.000 );
```

EX2: Set video stream format with custom flags, CUSTOMFLAG\_ISINTERLEAVED.

```
AMESDK_FILE_SET_VIDEO_STREAM_FORMAT( hDev, 0x34363248 /*H264*/,
                                     704, 480, 24, 29.970,
                                     AMESDK_FILE_CUSTOMFLAG_ISINTERLEAVED );

AMESDK_FILE_SET_VIDEO_STREAM_FORMAT( hDev, 0x44495658 /*XVID*/,
                                     704, 240, 24, 29.970,
                                     0x00000000 );

AMESDK_FILE_SET_VIDEO_STREAM_FORMAT( hDev, 0x44495658 /*XVID*/,
                                     352, 240, 24, 29.970,
                                     0x00000000 );
```

EX3: Set video stream format with custom flags, CUSTOMFLAG\_ISSYNCHRONIZATION.

```
AMESDK_FILE_SET_VIDEO_STREAM_FORMAT( hDev, 0x44495658 /*XVID*/,
                                     704, 480, 24, 29.970,
                                     AMESDK_FILE_CUSTOMFLAG_ISSYNCHRONIZATION |
                                     AMESDK_FILE_CUSTOMFLAG_ISINTERLEAVED );
```

### 7.19 AMESDK\_FILE\_GET\_AUDIO\_STREAM\_FORMAT

This function is used to get the audio stream format from one media file.

```
BOOL AMESDK_FILE_GET_AUDIO_STREAM_FORMAT( DEVICE_HANDLE hDevHandle,  
                                           ULONG *        pChannels,  
                                           ULONG *        pBitsPerSample,  
                                           ULONG *        pSamplesPerSec );
```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose audio format is to be retrieved.
pChannels	OUT	<b>Channel number.</b> Pointer to a variable that stores the number of the audio channel of the audio stream. (1:Mono, 2:Stereo)
pBitsPerSample	OUT	<b>Bits Per Sample.</b> Pointer to a variable that stores the bit count of the sampling rate.
pSamplesPerSec	OUT	<b>Samples Per Second.</b> Pointer to a variable that stores the number of sample per second.

#### Return Values:

BOOL

#### Support Capture Devices:

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Support File Devices:

RENDERER

#### Examples:

EX1: Get the audio stream format.

```
AMESDK_FILE_GET_AUDIO_STREAM_FORMAT( hDev,  
                                     &nChannels,  
                                     &nBitsPerSample,  
                                     &nSamplesPerSec );
```

**7.20 AMESDK\_FILE\_SET\_AUDIO\_STREAM\_FORMAT**

This function is used to set/change the audio stream format.

```

    BOOL AMESDK_FILE_SET_AUDIO_STREAM_FORMAT( DEVICE_HANDLE hDevHandle,
                                                ULONG          nChannels,
                                                ULONG          nBitsPerSample,
                                                ULONG          nSamplesPerSec );

```

**Parameters:**

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose audio format is to be set.
pChannels	IN	<b>Channel number.</b> Specifies the audio channel of the audio stream. (1:Mono, 2:Stereo)
pBitsPerSample	IN	<b>Bits Per Sample.</b> Specifies the bit count of the sampling rate.
pSamplesPerSec	IN	<b>Samples Per Second.</b> Specifies the number of sample per second.

**Return Values:**

BOOL

**Support Capture Devices:**

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

**Support File Devices:**

RENDERER

**Examples:**

EX1: Set the audio stream format.

```
AMESDK_FILE_SET_AUDIO_STREAM_FORMAT( hDev, 1, 16, 48000 );
```

```
AMESDK_FILE_SET_AUDIO_STREAM_FORMAT( hDev, 2, 16, 8000 );
```

**7.21 AMESDK\_FILE\_SET\_VIDEO\_STREAM\_BUFFER**

This function is used to push one video stream buffer into one media file.

```

BOOL AMESDK_FILE_SET_VIDEO_STREAM_BUFFER(

    DEVICE_HANDLE          hDevHandle,
    BYTE *                 pStreamBuffer,
    ULONG                  nStreamBufferSize,
    BOOL                   bIsKeyFrame,
    ULONGLONG              nTimeStamp = 0x0000000000000000,
    AMESDK_SECTION_INFO *  pSectionData = NULL,
    AMESDK_MOTION_INFO *   pMotionData = NULL,
    AMESDK_GPS_INFO *      pGpsData = NULL,
    AMESDK_POS_INFO *      pPosData = NULL
);

struct AMESDK_SECTION_INFO {

    ULONG          m_nStartSample;
    ULONG          m_nStopSample;
    ULONGLONG      m_nStartTimeStamp;
    ULONGLONG      m_nStopTimeStamp;
    ULONG          m_nSectionNumber;
};

struct AMESDK_MOTION_INFO {

    ULONG          m_nStartSample;
    ULONG          m_nStopSample;
    ULONGLONG      m_nStartTimeStamp;
    ULONGLONG      m_nStopTimeStamp;
    ULONG          m_nStrength;
};

struct AMESDK_GPS_INFO {

    ULONG          m_nStartSample;
    ULONG          m_nStopSample;
    ULONGLONG      m_nStartTimeStamp;
    ULONGLONG      m_nStopTimeStamp;
    float          m_fLongitude;
    float          m_fLatitude;
    float          m_fSpeed;
    float          m_fAngle;
    BOOL           m_bIsVaild;
};

struct AMESDK_POS_INFO {

    ULONG          m_nStartSample;
    ULONG          m_nStopSample;
    ULONGLONG      m_nStartTimeStamp;
    ULONGLONG      m_nStopTimeStamp;
};

```

## Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose video buffer is to be set.
pStreamBuffer	IN	<b>Stream Buffer.</b> Specifies the start address of video stream buffer.
nStreamBufferSize	IN	<b>Stream Buffer Size.</b> Specifies the buffer size of the video stream.
bIsKeyFrame	IN	<b>Key Frame.</b> Specifies whether the frame is I frame or not.
nTimeStamp	IN	<b>Time Stamp.</b> Specifies the frame's timestamp information.
pSectionData	IN	<b>Section Data.</b> Specifies the frame's section information.
pMotionData	IN	<b>Motion Data.</b> Specifies the frame's motion information.
pGpsData	IN	<b>GPS Data.</b> Specifies the frame's GPS information.
pPosData	IN	<b>POS Data.</b> Specifies the frame's POS information.

## Return Values:

BOOL

## Support Capture Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

## Support File Devices:

RENDERER

## Examples:

EX1: Set the video stream buffer.

```
AMESDK_FILE_SET_VIDEO_STREAM_BUFFER( hDev, po, 0x00080000, TRUE );
```

EX2: Set the video stream buffer with motion data into renderer.

```
AMESDK_GPS_INFO s_motion_info = { -1, -1, -1, -1, 0xFF };
AMESDK_FILE_SET_VIDEO_STREAM_BUFFER( hDev, po, 0x00080000, FALSE,
                                     NULL, &s_motion_info, NULL, NULL );
```

EX3: Set the video stream buffer with gps data into renderer.

```
AMESDK_GPS_INFO s_gps_info = { -1, -1, -1, -1, 100.000, 20.000, 120.000, 250.000, TRUE };
AMESDK_FILE_SET_VIDEO_STREAM_BUFFER( hDev, po, 0x00080000, FALSE,
                                     NULL, NULL, &s_gps_info, NULL );
```



## 7.22 AMESDK\_FILE\_SET\_AUDIO\_STREAM\_BUFFER

This function is used to push one audio stream buffer into one media file.

```

BOOL AMESDK_FILE_SET_AUDIO_STREAM_BUFFER(

    DEVICE_HANDLE          hDevHandle,
    BYTE *                 pStreamBuffer,
    ULONG                  nStreamBufferSize,
    ULONGLONG              nTimeStamp = 0x0000000000000000,
    AMESDK_SECTION_INFO *  pSectionData = NULL
);
struct AMESDK_SECTION_INFO {

    ULONG          m_nStartSample;
    ULONG          m_nStopSample;
    ULONGLONG      m_nStartTimeStamp;
    ULONGLONG      m_nStopTimeStamp;
    ULONG          m_nSectionNumber;
};

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose audio buffer is to be set.
pStreamBuffer	IN	<b>Stream Buffer.</b> Specifies the start address of audio stream buffer.
nStreamBufferSize	IN	<b>Stream Buffer Size.</b> Specifies the buffer size of the audio stream.
nTimeStamp	IN	<b>Time Stamp.</b> Specifies the frame's timestamp information.
pSectionData	IN	<b>Section Data,</b> Specifies the frame's section information.

### Return Values:

BOOL

### Support Capture Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support File Devices:

RENDERER

### Examples:

EX1: Set the audio stream buffer.

```
AMESDK_FILE_SET_AUDIO_STREAM_BUFFER( hDev, po, 1920 );
```

### 7.23 AMESDK\_FILE\_GET\_VIDEO\_STREAM\_BUFFER

This function is used to get one video stream buffer from one media file.

```

    BOOL AMESDK_FILE_GET_VIDEO_STREAM_BUFFER(  DEVICE_HANDLE  hDevHandle,
                                                ULONG          nSample,
                                                BYTE *         pStreamBuffer,
                                                ULONG *         pStreamBufferSize
    );

```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose video buffer is to be set.
nSample	IN	<b>Sample.</b> Specifies which sample's stream buffer is to be retrieved.
pStreamBuffer	OUT	<b>Stream Buffer.</b> Specifies the start address of video stream buffer.
pStreamBufferSize	OUT	<b>Stream Buffer Size.</b> Specifies the buffer size of the video stream.

#### Return Values:

BOOL

#### Support Capture Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Support File Devices:

SOURCE, RENDERER

#### Examples:

EX1: Get the first video stream buffer.

```
AMESDK_FILE_GET_VIDEO_STREAM_BUFFER( hDev, 0, pStreamBuffer, pStreamBufferSize );
```

## 7.24 AMESDK\_FILE\_GET\_AUDIO\_STREAM\_BUFFER

This function is used to get one audio stream buffer from one media file.

```

    BOOL AMESDK_FILE_GET_AUDIO_STREAM_BUFFER( DEVICE_HANDLE  hDevHandle,
                                              ULONG           nSample,
                                              BYTE *         pStreamBuffer,
                                              ULONG           nStreamBufferSize
    );

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose audio buffer is to be set.
nSample	IN	<b>Sample.</b> Specifies which sample's stream buffer is to be retrieved.
pStreamBuffer	OUT	<b>Stream Buffer.</b> Specifies the start address of audio stream buffer.
pStreamBufferSize	OUT	<b>Stream Buffer Size.</b> Specifies the buffer size of the audio stream.

### Return Values:

BOOL

### Support Capture Devices:

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support File Devices:

SOURCE, RENDERER

### Examples:

EX1: Get the first audio stream buffer.

```
AMESDK_FILE_GET_AUDIO_STREAM_BUFFER( hDev, 0, pStreamBuffer, pStreamBufferSize );
```

## 7.25 AMESDK\_FILE\_GET\_VIDEO\_STREAM\_DATA

This function is used to get one video sample's private stream data. The timestamp parameter is recorded one absolute system local time for this sample. For example, the sample is recorded on 2009.07.25.12:06:30,000,000,0 ns. User can use Microsoft's Win32 API, `FileTimeToSystemTime` to exchange it into `SYSTEMTIME` structure.

```

BOOL AMESDK_FILE_GET_VIDEO_STREAM_DATA( DEVICE_HANDLE    hDevHandle,
                                         ULONG            nSample,
                                         ULONGLONG *      pTimeStamp,
                                         BOOL *           pIsKeyFrame );

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose information is to be retrieved.
nSample	IN	<b>Sample.</b> Specifies which sample's information is to be retrieved.
pTimeStamp	OUT	<b>Time Stamp, in 100ns units.</b> Pointer to a variable that receives the time stamp of the sample.
pIsKeyFrame	OUT	<b>Key Frame.</b> Pointer to a variable that indicates whether the frame is key frame or not.

### Return Values:

BOOL

### Support Capture Devices:

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support File Devices:

SOURCE

### Examples:

EX1: Get the first video sample's private data.

```
AMESDK_FILE_GET_VIDEO_STREAM_DATA( hDev, 0, &nTimeStamp, &bIsKeyFrame );
```

EX2: Get the second video sample's private data.

```
AMESDK_FILE_GET_VIDEO_STREAM_DATA( hDev, 1, &nTimeStamp, &bIsKeyFrame );
```

## 7.26 AMESDK\_FILE\_GET\_AUDIO\_STREAM\_DATA

This function is used to get the audio's private stream data. The timestamp parameter is recorded one absolute system local time for this sample. For example, the sample is recorded on 2009.07.25.12:06:30,000,000,0 ns. User can use Microsoft's Win32 API, `FileTimeToSystemTime` to exchange it into `SYSTEMTIME` structure.

```
BOOL AMESDK_FILE_GET_AUDIO_STREAM_DATA( DEVICE_HANDLE    hDevHandle,
                                         ULONG             nSample,
                                         ULONGLONG *       pTimeStamp );
```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose information is to be retrieved.
nSample	IN	<b>Sample.</b> Specifies which sample's information is to be retrieved.
pTimeStamp	OUT	<b>Time Stamp, in 100ns units.</b> Pointer to a variable that receives the time stamp of the sample.

### Return Values:

BOOL

### Support Capture Devices:

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support File Devices:

SOURCE

### Examples:

EX1: Get the first audio sample's private data.

```
AMESDK_FILE_GET_AUDIO_STREAM_DATA( hDev, 0, &nTimeStamp );
```

EX2: Get the second audio sample's private data.

```
AMESDK_FILE_GET_AUDIO_STREAM_DATA( hDev, 1, &nTimeStamp );
```

## 7.27 AMESDK\_FILE\_GET\_VIDEO\_TIMESTAMP\_MAP

This function is used to get complete timestamp map from video stream. Please refer to function AMESDK\_FILE\_SET\_VIDEO\_STREAM\_BUFFER.

```
BOOL AMESDK_FILE_GET_VIDEO_TIMESTAMP_MAP( DEVICE_HANDLE  hDevHandle,
                                           ULONGLONG **   ppTimeStampMap,
                                           ULONG *        pTimeStampMapSize );
```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose information is to be retrieved.
ppTimeStampMap	OUT	<b>TimeStamp Map.</b> Pointer to a pointer that receives the complete data buffer of timestamp information. System will automatically release this buffer memory.
pTimeStampMapSize	OUT	<b>TimeStamp Map Size.</b> Pointer to a variable that receives the size of timestamp map.

### Return Values:

BOOL

### Support Capture Devices:

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support File Devices:

SOURCE

### Examples:

EX1: Get the timestamp map information.

```
AMESDK_FILE_GET_VIDEO_TIMESTAMP_MAP( hDev, &pTimeStampMap, &nTimeStampMapSize );
```

## 7.28 AMESDK\_FILE\_GET\_AUDIO\_TIMESTAMP\_MAP

This function is used to get complete timestamp map from audio stream. Please refer to function AMESDK\_FILE\_SET\_AUDIO\_STREAM\_BUFFER.

```
BOOL AMESDK_FILE_GET_AUDIO_TIMESTAMP_MAP( DEVICE_HANDLE hDevHandle,
                                           ULONGLONG ** ppTimeStampMap,
                                           ULONG *      pTimeStampMapSize );
```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose information is to be retrieved.
ppTimeStampMap	OUT	<b>TimeStamp Map.</b> Pointer to a pointer that receives the complete data buffer of timestamp information. System will automatically release this buffer memory.
pTimeStampMapSize	OUT	<b>TimeStamp Map Size.</b> Pointer to a variable that receives the size of timestamp map.

### Return Values:

BOOL

### Support Capture Devices:

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support File Devices:

SOURCE

### Examples:

EX1: Get the timestamp map information.

```
AMESDK_FILE_GET_AUDIO_TIMESTAMP_MAP( hDev, &pTimeStampMap, &nTimeStampMapSize );
```

## 7.29 AMESDK\_FILE\_GET\_VIDEO\_SECTION\_MAP

This function is used to get complete section map from video stream. Please refer to function AMESDK\_FILE\_SET\_VIDEO\_STREAM\_BUFFER.

```
BOOL AMESDK_FILE_GET_VIDEO_SECTION_MAP( DEVICE_HANDLE      hDevHandle,
                                         AMESDK_SECTION_INFO ** ppSectionMap,
                                         ULONG *              pSectionMapSize );

struct AMESDK_SECTION_INFO {

    ULONG      m_nStartSample;
    ULONG      m_nStopSample;
    ULONGLONG  m_nStartTimeStamp;
    ULONGLONG  m_nStopTimeStamp;
    ULONG      m_nSectionNumber;
};
```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose information is to be retrieved.
ppSectionMap	OUT	<b>Section Map.</b> Pointer to a pointer that receives the complete data buffer of section information. System will automatically release this buffer memory.
pSectionMapSize	OUT	<b>Section Map Size.</b> Pointer to a variable that receives the size of section map.

### Return Values:

BOOL

### Support Capture Devices:

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support File Devices:

SOURCE

### Examples:

EX1: Get the section map infomation.

```
AMESDK_FILE_GET_VIDEO_SECTION_MAP( hDev, &pSectionMap, &nSectionMapSize );
```



### 7.30 AMESDK\_FILE\_GET\_AUDIO\_SECTION\_MAP

This function is used to get complete section map from audio stream. Please refer to function AMESDK\_FILE\_SET\_AUDIO\_STREAM\_BUFFER.

```

BOOL AMESDK_FILE_GET_AUDIO_SECTION_MAP( DEVICE_HANDLE      hDevHandle,
                                         AMESDK_SECTION_INFO ** ppSectionMap
                                         ULONG *              pSectionMapSize );

struct AMESDK_SECTION_INFO {
    ULONG      m_nStartSample;
    ULONG      m_nStopSample;
    ULONGLONG  m_nStartTimeStamp;
    ULONGLONG  m_nStopTimeStamp;
    ULONG      m_nSectionNumber;
};

```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose information is to be retrieved.
ppSectionMap	OUT	<b>Section Map.</b> Pointer to a pointer that receives the complete data buffer of section information. System will automatically release this buffer memory.
pSectionMapSize	OUT	<b>Section Map Size.</b> Pointer to a variable that receives the size of section map.

#### Return Values:

BOOL

#### Support Capture Devices:

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Support File Devices:

SOURCE

#### Examples:

EX1: Get the section map infomation.

```
AMESDK_FILE_GET_AUDIO_SECTION_MAP( hDev, &pSectionMap, &nSectionMapSize );
```

### 7.31 AMESDK\_FILE\_GET\_VIDEO\_MOTION\_MAP

This function is used to get complete motion map from video stream. Please refer to function AMESDK\_FILE\_SET\_VIDEO\_STREAM\_BUFFER.

```

BOOL AMESDK_FILE_GET_VIDEO_MOTION_MAP( DEVICE_HANDLE      hDevHandle,
                                         AMESDK_MOTION_INFO ** ppMotionMap,
                                         ULONG *              pMotionMapSize );

struct AMESDK_MOTION_INFO {
    ULONG      m_nStartSample;
    ULONG      m_nStopSample;
    ULONGLONG  m_nStartTimeStamp;
    ULONGLONG  m_nStopTimeStamp;
    BYTE       m_nStrength;
};

```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose information is to be retrieved.
ppMotionMap	OUT	<b>Motion Map.</b> Pointer to a pointer that receives the complete data buffer of motion information. System will automatically release this buffer memory.
pMotionMapSize	OUT	<b>Motion Map Size.</b> Pointer to a variable that receives the size of motion map.

#### Return Values:

BOOL

#### Support Capture Devices:

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Support File Devices:

SOURCE

#### Examples:

EX1: Get the motion map information.

```
AMESDK_FILE_GET_VIDEO_MOTION_MAP( hDev, &pMotionMap, &nMotionMapSize );
```

### 7.32 AMESDK\_FILE\_GET\_VIDEO\_GPS\_MAP

This function is used to get complete GPS map from video stream. Please reference function AMESDK\_FILE\_SET\_VIDEO\_STREAM\_BUFFER.

```

BOOL AMESDK_FILE_GET_VIDEO_GPS_MAP(  DEVICE_HANDLE      hDevHandle,
                                     AMESDK_GPS_INFO **  ppGpsMap,
                                     ULONG *              pGpsMapSize );

struct AMESDK_GPS_INFO {

    ULONG      m_nStartSample;
    ULONG      m_nStopSample;
    ULONGLONG  m_nStartTimeStamp;
    ULONGLONG  m_nStopTimeStamp;
    float      m_fLongitude;
    float      m_fLatitude;
    float      m_fSpeed;
    float      m_fAngle;
    BOOL       m_bIsVaild;
};

```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose information is to be retrieved.
ppGpsMap	OUT	<b>GPS Map.</b> Pointer to a pointer that receives the complete data buffer of GPS information. System will automatically release this buffer memory.
pGpsMapSize	OUT	<b>GPS Map Size.</b> Pointer to a variable that receives the size of GPS map.

#### Return Values:

BOOL

#### Support Capture Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Support File Devices:

SOURCE

#### Examples:

EX1: Get the GPS map infomation.

```
AMESDK_FILE_GET_VIDEO_GPS_MAP( hDev, &pGpsMap, &nGpsMapSize );
```

### 7.33 AMESDK\_FILE\_GET\_VIDEO\_POS\_MAP

This function is used to get complete POS map from video stream. Please refer to function AMESDK\_FILE\_SET\_VIDEO\_STREAM\_BUFFER.

```

BOOL AMESDK_FILE_GET_VIDEO_POS_MAP(  DEVICE_HANDLE      hDevHandle,
                                     AMESDK_POS_INFO **  ppPosMap,
                                     ULONG *              pPosMapSize );

struct AMESDK_POS_INFO {
    ULONG      m_nStartSample;
    ULONG      m_nStopSample;
    ULONGLONG  m_nStartTimeStamp;
    ULONGLONG  m_nStopTimeStamp;
};

```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose information is to be retrieved.
ppPosMap	OUT	<b>POS Map.</b> Pointer to a pointer that receives the complete data buffer of POS information. System will automatically release this buffer memory.
pPosMapSize	OUT	<b>POS Map Size.</b> Pointer to a variable that receives the size of POS map.

#### Return Values:

BOOL

#### Support Capture Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Support File Devices:

SOURCE

#### Examples:

EX1: Get the POS map infomation.

```
AMESDK_FILE_GET_VIDEO_POS_MAP( hDev, &pPosMap, &nPosMapSize );
```

## 7.34 AMESDK\_FILE\_GET\_MEDIA\_LENGTH

This function is used to get the total length of a media file. The unit can be specified by you.

```

    BOOL AMESDK_FILE_GET_MEDIA_LENGTH(  DEVICE_HANDLE    hDevHandle,
                                         LONGLONG *      pLength,
                                         DWORD           dwTimeUnits );

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose information is to be retrieved.
pLength	OUT	<b>Length.</b> Pointer to a variable that receives the length, in units of the current time format.
dwTimeUnits	IN	<b>Time Units.</b> Specifies the time format.
		<b>SUPPORT TIME FORMAT:</b> TIME_FORMAT_MEDIA_TIME    0x00000000, in 100ns units. TIME_FORMAT_FRAME    0x00000001, in frame units.

### Return Values:

BOOL

### Support Capture Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
       SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support File Devices:

SOURCE

### Examples:

EX1: Get the media length, in 100ns units.

```
AMESDK_FILE_GET_MEDIA_LENGTH( hDev, &nMediaTimeLength, 0x00000000 );
```

EX2: Get the media length, in video frame units.

```
AMESDK_FILE_GET_MEDIA_LENGTH( hDev, &nVideoFrameLength, 0x00000001 );
```

### 7.35 AMESDK\_FILE\_GET\_MEDIA\_POSITION

This function is used to get the current playback position of a media file. The unit can be specified by you.

```
BOOL AMESDK_FILE_GET_MEDIA_POSITION( DEVICE_HANDLE  hDevHandle,
                                     LONGLONG *      pPosition,
                                     DWORD           dwTimeUnits );
```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose media position is to be retrieved.
pPosition	OUT	<b>Position.</b> Pointer to a variable that receives the position, in units of the current time format.
dwTimeUnits	IN	<b>Time Units.</b> Specifies the time format.
		<b>SUPPORT TIME FORMAT:</b> TIME_FORMAT_MEDIA_TIME    0x00000000, in 100ns units. TIME_FORMAT_FRAME        0x00000001, in frame units.

#### Return Values:

BOOL

#### Support Capture Devices:

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
      SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Support File Devices:

SOURCE

#### Examples:

EX1: Get the current playback position of one media file, in 100ns units.

```
AMESDK_FILE_GET_MEDIA_POSITION( hDev, &nMediaTimePosition, 0x00000000 );
```

EX2: Get the current playback position of one media file, in video frame units.

```
AMESDK_FILE_GET_MEDIA_POSITION( hDev, &nVideoFramePosition, 0x00000001 );
```

### 7.36 AMESDK\_FILE\_SET\_MEDIA\_POSITION

This function is used to set the playback position of a media file. The unit can be specified by you.

```

BOOL AMESDK_FILE_SET_MEDIA_POSITION( DEVICE_HANDLE  hDevHandle,
                                     LONGLONG        nPosition
                                     DWORD           dwTimeUnits );

```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose media position is to be set.
pPosition	IN	<b>Position.</b> Specifies the position of one media file, in units of the current time format.
dwTimeUnits	IN	<b>Time Units.</b> Specifies the time format.
		<b>SUPPORT TIME FORMAT:</b> TIME_FORMAT_MEDIA_TIME    0x00000000, in 100ns units. TIME_FORMAT_FRAME        0x00000001, in frame units.

#### Return Values:

BOOL

#### Support Capture Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
       SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Support File Devices:

SOURCE

#### Examples:

EX1: Set the playback position of one media file, in 100ns units.

```
AMESDK_FILE_SET_MEDIA_POSITION( hDev, nMediaTimePosition, 0x00000000 );
```

EX2: Set the playback position of one media file, in video frame units.

```
AMESDK_FILE_SET_MEDIA_POSITION( hDev, nVideoFramePosition, 0x00000001 );
```

## 7.37 AMESDK\_FILE\_GET\_MEDIA\_PLAYBACK\_RATE

This function is used to get current playback rate.

```
BOOL AMESDK_FILE_GET_MEDIA_PLAYBACK_RATE( DEVICE_HANDLE hDevHandle,
                                           DOUBLE *      pRate );
```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose playback rate is to be retrieved.
pRate	OUT	<b>Rate.</b> Pointer to a variable that receives current playback rate. Here, 1.0 is normal playback speed, 0.5 is half speed, and 2.0 is twice speed.

### Return Values:

BOOL

### Support Capture Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support File Devices:

SOURCE

### Examples:

EX1: Get current playback rate.

```
AMESDK_FILE_GET_MEDIA_PLAYBACK_RATE( hDev, &dRate );
```



### 7.38 AMESDK\_FILE\_SET\_MEDIA\_PLAYBACK\_RATE

This function is used to set the playback rate.

```
BOOL AMESDK_FILE_SET_MEDIA_PLAYBACK_RATE( DEVICE_HANDLE hDevHandle,  
                                           DOUBLE          dRate );
```

#### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose playback rate is to be set.
dRate	IN	<b>Rate.</b> Specifies the playback rate. Here, 1.0 is normal playback speed, 0.5 is half speed, and 2.0 is twice speed.

#### Return Values:

BOOL

#### Support Capture Devices:

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Support File Devices:

SOURCE

#### Examples:

EX1: Set normal playback speed.

```
AMESDK_FILE_SET_MEDIA_PLAYBACK_RATE( hDev, 1.000 );
```

EX2: Set twice playback rate.

```
AMESDK_FILE_SET_MEDIA_PLAYBACK_RATE( hDev, 2.000 );
```

EX3: Set half playback rate.

```
AMESDK_FILE_SET_MEDIA_PLAYBACK_RATE( hDev, 0.500 );
```

**7.39 AMESDK\_FILE\_GET\_MEDIA\_PLAYBACK\_SKIP\_MODE**

This function is used to get current playback skip mode.

```
BOOL AMESDK_FILE_GET_MEDIA_PLAYBACK_SKIP_MODE( DEVICE_HANDLE hDevHandle,  
                                                ULONG * pSkipMode );
```

**Parameters:**

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose playback rate is to be retrieved.
pSkipMode	OUT	<b>Skip Mode.</b> Pointer to a variable that receives current playback skip mode. Here, 0 will decode all frames. 1 will decode I frame only. 2 will decode I / P frames and skip B frame.

**Return Values:**

BOOL

**Support Capture Devices:**

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

**Support File Devices:**

SOURCE

**Examples:**

EX1: Get current playback skip mode.

```
AMESDK_FILE_GET_MEDIA_PLAYBACK_SKIP_MODE( hDev, &nSkipMode );
```

#### 7.40 AMESDK\_FILE\_SET\_MEDIA\_PLAYBACK\_SKIP\_MODE

This function is used to set the playback skip mode.

```
BOOL AMESDK_FILE_SET_MEDIA_PLAYBACK_SKIP_MODE( DEVICE_HANDLE hDevHandle,
                                                ULONG          nSkipMode );
```

##### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose playback rate is to be set.
nSkipMode	IN	<b>Rate.</b> Specifies the playback skip mode. Here, 0 will decode all frames. 1 will decode I frame only. 2 will decode I / P frames and skip B frame.

##### Return Values:

BOOL

##### Support Capture Devices:

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

##### Support File Devices:

SOURCE

##### Examples:

EX1: Decode all frames.

```
AMESDK_FILE_SET_MEDIA_PLAYBACK_SKIP_MODE( hDev, 0 );
```

EX2: Decode I frame only.

```
AMESDK_FILE_SET_MEDIA_PLAYBACK_SKIP_MODE( hDev, 1 );
```

EX3: Decode I / P frames and skip B frame.

```
AMESDK_FILE_SET_MEDIA_PLAYBACK_SKIP_MODE( hDev, 2 );
```

## 7.41 AMESDK\_FILE\_SEEK

This function is used to do forward/rewind of key frame seeking.

```
BOOL AMESDK_FILE_SEEK( DEVICE_HANDLE hDevHandle, LONG nSeeks );
```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose media position is to be set.
nSeeks	IN	<b>Seeks.</b> Specifies the number of key frames to skip. If nSteps is 1, the file source steps forward to next key frame. If nSeeks is a number n greater than 1, the graph skips n - 1 key frames and shows the nth key frame. If nSeeks is a number n lower than 0, the graph rewind to last n key frames. The value should not be 0.

### Return Values:

BOOL

### Support Capture Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support File Devices:

SOURCE

### Examples:

EX1: Seek to next key frame.

```
AMESDK_FILE_SEEK( hDev, 1 );
```

EX2: Seek to last key frame.

```
AMESDK_FILE_SEEK( hDev, -1 );
```

EX3: Seek to next 5 key frames.

```
AMESDK_FILE_SEEK( hDev, 5 );
```

## 7.42 AMESDK\_FILE\_FLUSH

This helper function allows you to flush the system's cache memory into hard disk right away.

```
BOOL AMESDK_FILE_FLUSH( DEVICE_HANDLE hDevHandle );
```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose cache memory is to be flushed.

### Return Values:

BOOL

### Support Capture Devices:

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support File Devices:

RENDERER

### Examples:

EX1: Flush the cache memory into hard disk right away.

```
AMESDK_FILE_FLUSH( hDev );
```

### 7.43 AMESDK\_FILE\_DELETE\_THE\_OLDEST\_FILE

This function is used to delete the oldest file in this list you specify. If the oldest file is blocking, and it will automatically jump to next file. If there is no file can be removed from disk, it will return FALSE.

```
BOOL AMESDK_FILE_DELETE_THE_OLDEST_FILE(  
  
    CArray<AMESDK_FILE_INFO, AMESDK_FILE_INFO> * pFileInfoList,  
);  
struct AMESDK_FILE_INFO {  
  
    CHAR            m_pszFileName[ MAX_PATH ];  
    ULONGLONG      m_nFileSize;  
    ULONGLONG      m_nFileStartTime;  
    ULONGLONG      m_nFileStopTime;  
    ULONGLONG      m_nVideoStreamStartTime;  
    ULONGLONG      m_nVideoStreamStopTime;  
    ULONGLONG      m_nAudioStreamStartTime;  
    ULONGLONG      m_nAudioStreamStopTime;  
    ULONG          m_nVideoFrameNumber;  
    ULONG          m_nVideoColorSpaceType;  
    ULONG          m_nVideoWidth;  
    ULONG          m_nVideoHeight;  
    ULONG          m_nVideoBitCount;  
    double         m_dVideoFrameRate;  
    DWORD          m_dwVideoCustomFlags;  
    ULONG          m_nVideoMotionRatio;  
    ULONG          m_nAudioFrameNumber;  
    ULONG          m_nAudioChannels;  
    ULONG          m_nAudioBitsPerSample;  
    ULONG          m_nAudioSamplesPerSec;  
    ULONG          m_nAudioSamplesPerStreamBuffer;  
    DWORD          m_dwReserved;  
};
```

#### Parameters:

Parameter	IN/OUT	Description
pFileInfoList	IN/OUT	<b>File Info List.</b> Pointer to a CArray list that stores the current file information. The helper function will delete the oldest file in this list.

#### Return Values:

BOOL

#### Support Capture Devices:

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

#### Support File Devices:

SOURCE, RENDERER

## Examples:

EX1: To delete the oldest file in this list.

```
AMESDK_FILE_GET_DELETE_THE_OLDEST_FILE( &sFileInfoList );
```

## 7.44 AMESDK\_FILE\_GET\_FILE\_INFO\_LIST

This function is used to get one file info list within the time range you specify. Note!! If the bAutoRepair parameter is enabled, the function will enable the self-repair function. When there are damaged files which are found out in this source directory, they will be fixed right away. For self-repair, please reference AMESD\_FILE\_REPAIR.

```
CArray<AMESDK_FILE_INFO, AMESDK_FILE_INFO> *
AMESDK_FILE_GET_FILE_INFO_LIST( CHAR *      pszSrcDirectory,
                                ULONGLONG   nStartSearchSystemTime,
                                ULONGLONG   nStopSearchSystemTime,
                                BOOL         bAutoRepair = FALSE,
                                volatile ULONG * pSearchProgress = NULL,
                                volatile ULONG * pRepairProgress = NULL
);

struct AMESDK_FILE_INFO {

    CHAR          m_pszFileName[ MAX_PATH ];
    ULONGLONG     m_nFileSize;
    ULONGLONG     m_nFileStartTime;
    ULONGLONG     m_nFileStopTime;
    ULONGLONG     m_nVideoStreamStartTime;
    ULONGLONG     m_nVideoStreamStopTime;
    ULONGLONG     m_nAudioStreamStartTime;
    ULONGLONG     m_nAudioStreamStopTime;
    ULONG         m_nVideoFrameNumber;
    ULONG         m_nVideoColorSpaceType;
    ULONG         m_nVideoWidth;
    ULONG         m_nVideoHeight;
    ULONG         m_nVideoBitCount;
    double        m_dVideoFrameRate;
    DWORD         m_dwVideoCustomFlags;
    ULONG         m_nVideoMotionRatio;
    ULONG         m_nAudioFrameNumber;
    ULONG         m_nAudioChannels;
    ULONG         m_nAudioBitsPerSample;
    ULONG         m_nAudioSamplesPerSec;
    ULONG         m_nAudioSamplesPerStreamBuffer;
    DWORD         m_dwReserved;
};
```

## Parameters:

Parameter	IN/OUT	Description
pszSrcDirectory	IN	<b>Search Directory.</b> Specifies the directory where you want to search.
nStartSearchSystemTime	IN	<b>Start Search Time.</b> Specifies the start search time.
nStopSearchSystemTime	IN	<b>End Search Time.</b> Specifies the end search time.
dwSearchOps	IN	<b>Search Operation.</b> Specifies the search operation.
		<b>Support Operations:</b> FILE_INFO_LIST_RESET      0x00000000 FILE_INFO_LIST_UPDATE    0x00000001
bAutoRepair	IN	<b>Auto Repair.</b> Specifies whether to enable the auto repair function.



pSearchProgress	OUT	<b>Search Progress, %</b> . Pointer to a variable that stores the current search progress. User can use it to design one progress bar control.
pRepairProgress	OUT	<b>Repair Progress, %</b> . Pointer to a variable that stores the current repair progress. User can use it to design one progress bar control.

**Return Values:**

CArray<AMESDK\_FILE\_INFO, AMESDK\_FILE\_INFO> \*. If AMESDK\_FILE\_GET\_FILE\_INFO\_LIST is successful, it will return one file info list pointer. If it is fail, it will return NULL. User should use AMESDK\_FILE\_FREE\_FILE\_INFO\_LIST to release all memory resource.

**Support Capture Devices:**

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

**Support File Devices:**

SOURCE, RENDERER

**Examples:**

EX1: Get the file information list within specified search time range.

```
pFileInfoList = AMESDK_FILE_GET_FILE_INFO_LIST( "D:\\\\SC\\",  
                                                nStartSearchSystemTime,  
                                                nStopSearchSystemTime );
```

**Remarks:**

The caller must free the returned list's memory block. You can use the helper function AMESDK\_FILE\_FREE\_FILE\_INFO\_LIST.

## 7.45 AMESDK\_FILE\_UPDATE\_FILE\_INFO\_LIST

This function is used to update one file info list within the time range you specify. Note!! If the bAutoRepair parameter is enabled, the function will enable the self-repair function. When there are damaged files which are found out in this source directory, they will be fixed right away. For self-repair, please refer to AMESD\_FILE\_REPAIR.

```

BOOL AMESDK_FILE_UPDATE_FILE_INFO_LIST(
    CArray<AMESDK_FILE_INFO, AMESDK_FILE_INFO> * pFileInfoList,
        CHAR * pszSrcDirectory,
        ULONGLONG nStartSearchSystemTime,
        ULONGLONG nStopSearchSystemTime,
        BOOL bAutoRepair = FALSE,
    volatile ULONG * pSearchProgress = NULL,
    volatile ULONG * pRepairProgress = NULL
);

struct AMESDK_FILE_INFO {

    CHAR m_pszFileName[ MAX_PATH ];
    ULONGLONG m_nFileSize;
    ULONGLONG m_nFileStartTime;
    ULONGLONG m_nFileStopTime;
    ULONGLONG m_nVideoStreamStartTime;
    ULONGLONG m_nVideoStreamStopTime;
    ULONGLONG m_nAudioStreamStartTime;
    ULONGLONG m_nAudioStreamStopTime;
    ULONG m_nVideoFrameNumber;
    ULONG m_nVideoColorSpaceType;
    ULONG m_nVideoWidth;
    ULONG m_nVideoHeight;
    ULONG m_nVideoBitCount;
    double m_dVideoFrameRate;
    DWORD m_dwVideoCustomFlags;
    ULONG m_nVideoMotionRatio;
    ULONG m_nAudioFrameNumber;
    ULONG m_nAudioChannels;
    ULONG m_nAudioBitsPerSample;
    ULONG m_nAudioSamplesPerSec;
    ULONG m_nAudioSamplesPerStreamBuffer;
    DWORD m_dwReserved;
};

```

## Parameters:

Parameter	IN/OUT	Description
pFileInfoList	IN/OUT	<b>File Info List.</b> Pointer to a CArray list.
pszSrcDirectory	IN	<b>Search Directory.</b> Specifies the directory where you want to search.
nStartSearchSystemTime	IN	<b>Start Search Time.</b> Specifies the start search time.
nStopSearchSystemTime	IN	<b>End Search Time.</b> Specifies the end search time.
bAutoRepair	IN	<b>Auto Repair.</b> Specifies whether to enable the auto repair function.
pSearchProgress	OUT	<b>Search Progress, %.</b> Pointer to a variable that stores the current search progress. User can use it to design one progress bar control.

pRepairProgress	OUT	<b>Repair Progress, %.</b> Pointer to a variable that stores the current repair progress. User can use it to design one progress bar control.
-----------------	-----	---

## Return Values:

BOOL

## Support Capture Devices:

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

## Support File Devices:

SOURCE, RENDERER

## Examples:

EX1: Update the file information list within specified search time range. The update operation will help you to save more cpu loading than the get operation. Here, the pFileInfoList should not be NULL.

```
AMESDK_FILE_UPDATE_FILE_INFO_LIST( pFileInfoList,
                                     "D:\\SC\\",
                                     nStartSearchSystemTime,
                                     nStopSearchSystemTime );
```

**7.46 AMESDK\_FILE\_FREE\_FILE\_INFO\_LIST**

This function is used to free the list's memory block. The list pointer is required from AMESDK\_FILE\_GET\_FILE\_INFO\_LIST function.

```
BOOL AMESDK_FILE_FREE_FILE_INFO_LIST(  
    CArray<AMESDK_FILE_INFO, AMESDK_FILE_INFO> * pFileInfoList  
);
```

**Parameters:**

Parameter	IN/OUT	Description
pFileInfoList	IN/OUT	<b>File Info List.</b> Pointer to a CArray list.

**Return Values:**

BOOL

**Support Capture Devices:**

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

**Support File Devices:**

SOURCE, RENDERER

**Examples:**

EX1: Free the list's memory block.

```
AMESDK_FILE_FREE_FILE_INFO_LIST( pFileInfoList );
```

### 7.47 AMESDK\_FILE\_EXPORT

This group functions are used to backup data within the time range you specify. They can export one new media file from the source directory, which could contain many small media files, to destination directory. The parameter, `nMaxExportFileSize`, allows you to control the max size of the exported media file, too. When the backup size is overflow you specify. The function will automatically generate next media file to backup remainder files. The COPY function allows you to support the real-time export function from one memory file. You can use STOP function to end the export.

```
PVOID AMESDK_FILE_EXPORT_START( CHAR *      pszSrcDirectory,
                                CHAR *      pszDstDirectory,
                                ULONGLONG   nStartExportSystemTime,
                                ULONGLONG   nStopExportSystemTime,
                                ULONGLONG   nMaxExportFileSize = 1,073,741,824,
                                volatile ULONG * pExportProgress = NULL
);
BOOL AMESDK_FILE_EXPORT_STOP(  PVOID *      pDstFileRenderer );

BOOL AMESDK_FILE_EXPORT_COPY(  PVOID *      pDstFileRenderer,
                                DEVICE_HANDLE hSrcFileRenderer,
                                ULONGLONG     nStartExportSystemTime,
                                ULONGLONG     nStopExportSystemTime,
                                BOOL          bFlush = FALSE );
```

#### Parameters:

Parameter	IN/OUT	Description
<code>pszSrcDirectory</code>	IN	<b>Source Directory.</b> Specifies the source directory or the source file path that you want to export.
<code>pszDstDirectory</code>	IN	<b>Destination Directory.</b> Specifies the destination directory or the destination file path where you want to export to.
<code>nStartExportSystemTime</code>	IN	<b>Start Export Time.</b> Specifies the start time of the data to export.
<code>nStopExportSystemTime</code>	IN	<b>End Export Time.</b> Specifies the end time of the data to export.
<code>nMaxExportFileSize</code>	IN	<b>Max Export File Size.</b> Specifies max export file size. Default size is 1GB for single file.
<code>pExportProgress</code>	OUT	<b>Export Progress, %.</b> Pointer to a variable that stores the current export progress. User can use it to design one progress bar control.
<code>pDstFileRenderer</code>	IN	<b>Destination File Renderer.</b> Specifies one destination file renderer pointer that is returned by <code>AMESDK_FILE_EXPORT_START</code> .
<code>pSrcFileRenderer</code>	IN	<b>Source File Renderer.</b> Specifies one source file renderer pointer.
<code>nStartExportSystemTime</code>	IN	<b>Start Export Time.</b> Specifies the start time of the data to export.
<code>nStopExportSystemTime</code>	IN	<b>End Export Time.</b> Specifies the end time of the data to export.
<code>bFlush</code>	IN	<b>Flush.</b> See <code>AMESDK_FILE_FLUSH</code> section.

## Return Values:

PVOID. If AMESDK\_FILE\_EXPORT\_START is successful, it will return one file renderer pointer. If it is fail, it will return NULL. User should use AMESDK\_FILE\_EXPORT\_STOP to release all memory resource.

## Support Capture Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

## Support File Devices:

SOURCE, RENDERER

## Examples:

EX1: Backup data to "E:\BACKUP\" within the time range you specify.

```
PVOID pe = AMESDK_FILE_EXPORT_START( "D:\\SC\\",
                                     "E:\\BACKUP\\",
                                     nStartExportSystemTime,
                                     nStopExportSystemTime,
                                     1024 * 1024 * 1024,
                                     &nExportProgress );
```

```
AMESDK_FILE_EXPORT_STOP( pe );
```

EX2: Backup data to "E:\BACKUP\" and the single file's max size is 256MB.

```
PVOID pe = AMESDK_FILE_EXPORT_START( "D:\\SC\\",
                                     "E:\\BACKUP\\",
                                     nStartExportSystemTime,
                                     nStopExportSystemTime,
                                     256 * 1024 * 1024,
                                     &nExportProgress );
```

```
AMESDK_FILE_EXPORT_STOP( pe );
```

EX3: Backup data to "E:\BACKUP\" and the single buackup file's max size is 50GB.

```
PVOID pe = AMESDK_FILE_EXPORT_START( "D:\\SC\\200910150000.AVI",
                                     "E:\\BACKUP\\200910150000.AVI",
                                     nStartExportSystemTime,
                                     nStopExportSystemTime,
                                     50 * 1024 * 1024 * 1024,
                                     &nExportProgress );
```

```
AMESDK_FILE_EXPORT_STOP( pe );
```

## Remarks:

The AMESDK\_FILE\_EXPORT\_COPY function allows you to export current recording file (memory file) into the destination directory. About AMESDK\_FILE\_EXPORT\_COPY extra programming, please reference sample soucecode to obtain more tutorials.

## 7.48 AMESDK\_FILE\_SELF\_REPAIR

This function is used to repair one bad record file. When your system suffered from one unknown shutdown, the recording files could be damaged, because they cannot be closed normally. The helper function can help you to repair them.

```
BOOL AMESDK_FILE_SELF_REPAIR( CHAR *      pszSrcFileName,  
                               volatile ULONG * pRepairProgress = NULL  
);
```

### Parameters:

Parameter	IN/OUT	Description
pszSrcFileName	IN	<b>Source File.</b> Specifies the source file that you want to repair.
pRepairProgress	OUT	<b>Repair Progress, %.</b> Pointer to a variable that stores the current repair progress. User can use it to design one progress bar control.

### Return Values:

BOOL.

### Support Capture Devices:

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support File Devices:

SOURCE, RENDERER

### Examples:

EX1: Repair one damaged file into new target file.

```
AMESDK_FILE_SELF_REPAIR ( "C:\\20091001000000.AVI", &nRepairProgress );
```

## 7.49 AMESDK\_FILE\_REPAIR

This function is used to repair one bad record file. When your system suffered from one unknown shutdown, the recording files could be damaged, because they cannot be closed normally. The helper function can help you to repair them.

```
BOOL AMESDK_FILE_REPAIR( CHAR *   pszSrcFileName,
                        CHAR *   pszDstFileName = NULL,
                        volatile ULONG * pRepairProgress = NULL
    );
```

### Parameters:

Parameter	IN/OUT	Description
pszSrcFileName	IN	<b>Source File.</b> Specifies the source file that you want to repair.
pszDstFileName	IN	<b>Destination File.</b> Specifies the destination file where you want to rebuild to. Note!! If the parameter is NULL, SDK will enable self-repair function to overwrite the source file.
pRepairProgress	OUT	<b>Repair Progress, %.</b> Pointer to a variable that stores the current repair progress. User can use it to design one progress bar control.

### Return Values:

BOOL.

### Support Capture Devices:

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support File Devices:

SOURCE, RENDERER

### Examples:

EX1: Repair one damaged file into new target file.

```
AMESDK_FILE_REPAIR( "C:\\20091001000000.AVI", "C:\\0000.AVI" );
```

EX2: Self-Repair.

```
AMESDK_FILE_REPAIR( "C:\\20091001000000.AVI", "C:\\20091001000000.AVI" ); or
```

```
AMESDK_FILE_REPAIR( "C:\\20091001000000.AVI" );
```

EX3: Get the repair progress during repairing.

```
AMESDK_FILE_REPAIR( "C:\\20091001000000.AVI", NULL, &nRepairProgress );
```



## 7.50 AMESDK\_FILE\_DIAGNOSIS

This function is to check the fidelity of the source file. It could be checked if additional data can be retrieved from source file.

```
BOOL AMESDK_FILE_DIAGNOSIS( CHAR * pszFileName,  
                             BOOL * pIsHealthy,  
                             BOOL * pHasIndex  
);
```

### Parameters:

Parameter	IN/OUT	Description
pszFileName	IN	<b>Source File.</b> Specifies the source file that you want to repair.
pIsHealthy	OUT	IsHealthy. Specifies the fidelity of the source file.
pHasIndex	OUT	Index. Specifies if additional data is included in source file.

### Return Values:

BOOL.

### Support Capture Devices:

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support File Devices:

SOURCE, RENDERER

### Examples:

```
AMESDK_FILE_DIAGNOSIS( "C:\\00000000.AVI", &bIsHealthy, &bHasIndex );
```

## 7.51 AMESDK\_FILE\_STD\_RESTORE

This function is to strip off the additional data from source file, the destination file will be a pure AVI file.

```
BOOL AMESDK_FILE_STD_RESTORE( CHAR *          pszSrcFileName,  
                              CHAR *          pszDstFileName,  
                              volatile ULONG * pRestoreProgress = NULL  
);
```

### Parameters:

Parameter	IN/OUT	Description
pszSrcFileName	IN	<b>Source File.</b> Specifies the source file that you want to repair.
pszDstFileName	IN	<b>Destination File.</b> Specifies the destination file where you want to rebuild to. Note!! If the parameter is NULL, SDK will enable self-repair function to overwrite the source file.
pRestoreProgress	OUT	<b>Repair Progress, %.</b> Pointer to a variable that stores the current repair progress. User can use it to design one progress bar control.

### Return Values:

BOOL.

### Support Capture Devices:

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Support File Devices:

SOURCE, RENDERER

### Examples:

Ex1: convert original file to AVI file

```
AMESDK_FILE_STD_RESTORE("C:\\20091001000000.AVI", "C:\\20091001000000.AVI")
```

## 7.52 An AVI File Writer Software Programming Guides

DEMO DEVICES: SC290#N4

```

DEVICE_HANDLE hVideoDev[ 4 ]; // VIDEO H.264 STREAM CAPTURE DEVICE

DEVICE_HANDLE hAudioDev[ 4 ]; // AUDIO PCM STREAM CAPTURE DEVICE

DEVICE_HANDLE hFileDev[ 4 ]; // FILE RENDERER DEVICE

HWND wnd[ 8 ] = { NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL };

PF_BUFFER_CALLBACK bcb[ 8 ] = { NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL };

PVOID params[ 8 ] = { this, this, this, this, this, this, this, this };

BOOL HwInitializeDevice()
{
    bcb[ 0 ] = on_process_video_encoder_buffer_CH01;
    bcb[ 1 ] = on_process_video_encoder_buffer_CH02;
    bcb[ 2 ] = on_process_video_encoder_buffer_CH03;
    bcb[ 3 ] = on_process_video_encoder_buffer_CH04;
    bcb[ 4 ] = on_process_audio_buffer_CH01;
    bcb[ 5 ] = on_process_audio_buffer_CH02;
    bcb[ 6 ] = on_process_audio_buffer_CH03;
    bcb[ 7 ] = on_process_audio_buffer_CH04;

    // INITIALIZE FILE RESOURCE
    //
    hFileDev[ 0 ] = AMESDK_CREATE( "Common Analog File Renderer C:\\CH01.AVI", 0, 3, NULL, NULL, NULL );
    hFileDev[ 1 ] = AMESDK_CREATE( "Common Analog File Renderer C:\\CH02.AVI", 1, 3, NULL, NULL, NULL );
    hFileDev[ 2 ] = AMESDK_CREATE( "Common Analog File Renderer C:\\CH03.AVI", 2, 3, NULL, NULL, NULL );
    hFileDev[ 3 ] = AMESDK_CREATE( "Common Analog File Renderer C:\\CH04.AVI", 3, 3, NULL, NULL, NULL );

    for( ULONG i = 0 ; i < 4 ; i++ ) {

        // SET VIDEO FORMAT: RESOLUTION / FRAMERATE
        //
        AMESDK_FILE_SET_VIDEO_STREAM_FORMAT( hFileDev[ i ],
                                              MAKEFOURCC('H', '2', '6', '4'),
                                              720, 480, 24, 29.970
                                              AMESDK_FILE_CUSTOMFLAG_ISINTERLEAVED );

        // SET AUDIO FORMAT: MONO / 16BITS / 8000HZ
        //
        AMESDK_FILE_SET_AUDIO_STREAM_FORMAT( hFileDev[ i ], 1, 16, 8000 );
    }

    // INITIALIZE DEVICE RESOURCE
    //
    hVideoDev[ 0 ] = AMESDK_CREATE( "AH8400 PCI, Analog Encoder", 0, 0, wnd[ 0 ], bcb[ 0 ], params[ 0 ] );
    hVideoDev[ 1 ] = AMESDK_CREATE( "AH8400 PCI, Analog Encoder", 1, 0, wnd[ 1 ], bcb[ 1 ], params[ 1 ] );
    hVideoDev[ 2 ] = AMESDK_CREATE( "AH8400 PCI, Analog Encoder", 2, 0, wnd[ 2 ], bcb[ 2 ], params[ 2 ] );
    hVideoDev[ 3 ] = AMESDK_CREATE( "AH8400 PCI, Analog Encoder", 3, 0, wnd[ 3 ], bcb[ 3 ], params[ 3 ] );

    hAudioDev[ 0 ] = AMESDK_CREATE( "AH8400 PCI, Analog WaveIn", 0, 0, wnd[ 4 ], bcb[ 4 ], params[ 4 ] );
    hAudioDev[ 1 ] = AMESDK_CREATE( "AH8400 PCI, Analog WaveIn", 1, 0, wnd[ 5 ], bcb[ 5 ], params[ 5 ] );
    hAudioDev[ 2 ] = AMESDK_CREATE( "AH8400 PCI, Analog WaveIn", 2, 0, wnd[ 6 ], bcb[ 6 ], params[ 6 ] );
    hAudioDev[ 3 ] = AMESDK_CREATE( "AH8400 PCI, Analog WaveIn", 3, 0, wnd[ 7 ], bcb[ 7 ], params[ 7 ] );
    
```

```
// SETUP VIDEO PATH (H.264) PROPERTIES
//
for( i = 0 ; i < 4 ; i++ ) {

    AMESDK_SET_STANDARD( hVideoDev[ i ], 0x00000001 );

    AMESDK_SET_FORMAT( hVideoDev[ i ], MAKEFOURCC('H', '2', '6', '4'), 720, 480, 24, 29.970 );

    AMESDK_SET_VIDEOCOMPRESSION_PROPERTY( hVideoDev[ i ], 0x00000003, 0 );

    AMESDK_SET_VIDEOCOMPRESSION_PROPERTY( hVideoDev[ i ], 0x00000001, 6000 );

    AMESDK_SET_VIDEOCOMPRESSION_PROPERTY( hVideoDev[ i ], 0x00000000, 30 );

    AMESDK_SET_DEINTERLACE( m_hVideoDev[ i ], 7 );

}

// SETUP AUDIO PATH (PCM) PROPERTIES
//
for( i = 0 ; i < 4 ; i++ ) {

    AMESDK_SET_FORMAT( hAudioDev[ i ], 1, 16, 8000 ); // MONO / 16BITS / 8000HZ

}

// START
//
AMESDK_RUN( hVideoDev[ 0 ] );
AMESDK_RUN( hVideoDev[ 1 ] );
AMESDK_RUN( hVideoDev[ 2 ] );
AMESDK_RUN( hVideoDev[ 3 ] );

AMESDK_RUN( hAudioDev[ 0 ] );
AMESDK_RUN( hAudioDev[ 1 ] );
AMESDK_RUN( hAudioDev[ 2 ] );
AMESDK_RUN( hAudioDev[ 3 ] );

}

// RECORD VIDEO DATA
//
BOOL on_process_video_encoder_buffer_CH0X( double dSampleTime,
                                           BYTE * pBuffer,
                                           ULONG nBufferLen,
                                           BOOL bIsKeyFrame,
                                           PVOID pUserData )
{
    ULONG i = nChannelNumber;

    CMyDlg * pMyDialog = (CMyDlg *) (pUserData);

    BOOL bIsRecord = pMyDialog->bIsRecord[ i ];

    if( bIsRecord ) {

        AMESDK_FILE_SET_VIDEO_STREAM_BUFFER( hFileDev[ i ],
                                             pBuffer,
                                             nBufferLen,
                                             bIsKeyFrame );

    }

}
}
```

```
// RECORD AUDIO DATA
//
BOOL on_process_audio_buffer_CH0X( double dSampleTime,
                                   BYTE * pBuffer,
                                   ULONG nBufferLen,
                                   BOOL bIsKeyFrame,
                                   PVOID pUserData )
{
    ULONG i = nChannelNumber;

    CMyDlg * pMyDialog = (CMyDlg *) (pUserData);

    BOOL bIsRecord = pMyDialog->bIsRecord[ i ];

    if( bIsRecord ) {
        AMESDK_FILE_SET_AUDIO_STREAM_BUFFER( hFileDev[ i ], pBuffer, nBufferLen );
    }
}

BOOL HwUnInitializeDevice()
{
    // STOP RECORDING
    //
    bIsRecord[ 0 ] = FALSE
    bIsRecord[ 1 ] = FALSE
    bIsRecord[ 2 ] = FALSE
    bIsRecord[ 3 ] = FALSE

    // UNINITIALIZE DEVICE RESOURCE
    //
    for( ULONG i = 0 ; i < 4 ) {
        AMESDK_DESTROY( hVideoDev[ i ] );

        AMESDK_DESTROY( hAudioDev[ i ] );
    }

    // UNINITIALIZE FILE RESOURCE
    //
    for( ULONG i = 0 ; i < 4 ; i ++ ) {
        AMESDK_DESTROY( hFileDev[ i ] );
    }
}
```

## 7.53 An AVI File Player Software Programming Guides

DEMO DEVICES: ALL

```

DEVICE_HANDLE hFileDev = 0xFFFFFFFF; // FILE SOURCE DEVICE

BOOL CMyDlg::OnInitDialog()
{
    hFileDev = AMESDK_CREATE( "Common Analog File Source C:\\20100501083000.AVI",
                             0,
                             2,
                             m_hWnd,
                             NULL,
                             NULL,
    );

    // CHECK ERROR FLAG
    //
    if( hFileDev & 0x80000000 ) {

        hFileDev = 0xFFFFFFFF;

        return FALSE;
    }

    // GET THE VIDEO STREAM FORMAT : WIDTH, HEIGHT, BIT COUNT, FRAME RATE
    //
    ULONG cx = 0; ULONG cy = 0; ULONG cz = 0; double fps = 0.0; DWORD flags = 0;

    AMESDK_FILE_GET_VIDEO_STREAM_FORMAT( hFileDev, NULL, &cx, &cy, &cz, &fps, &flags );

    // GET THE TOTAL LENGTH OF A MEDIA FILE (TIME UNITS)
    //
    LONGLONG length = 0;

    AMESDK_FILE_GET_MEDIA_LENGTH( hFileDev, &length, 0x00000000 );

    // ENABLE THE DEINTERLACE FUNCTION IN THE BEST QUALITY METHOD
    //
    if( flags & AMESDK_FILE_CUSTOMFLAG_ISINTERLEAVED ) {

        AMESDK_SET_DEINTERLACE( hFileDev, 0x00000007 );
    }
}

void CMyDlg::OnPlay()
{
    AMESDK_RUN( hFileDev );
}

void CMyDlg::OnPause()
{
    AMESDK_PAUSE( hFileDev );
}

void CMyDlg::OnStep()
{
    AMESDK_STEP( hFileDev, 1 ); // GOTO NEXT FRAME
}

```

```
void CMyDlg::OnStop()
{
    AMESDK_STOP( hFileDev );

    // JUMP TO THE FIRST FRAME (TIME UNITS)
    //
    AMESDK_FILE_SET_MEDIA_POSITION( hFileDev, 0, 0x00000000 );

    AMESDK_PAUSE( hFileDev );
}

void CMyDlg::OnDestroy()
{
    AMESDK_STOP( hFileDev );

    AMESDK_DESTROY( hFileDev );
}
```

## 8 Exported Functions for GPS Device

SUPPORT DEVICE:

PD652, SC100, SC200, SC230, SC280,  
SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0,  
SC3B0, SC500, SC510, SC580, SC590,



Exported Functions for GPS Device Programming	
8.01	AMESDK_CREATE
8.02	AMESDK_DESTROY (SEE 1.02)
8.03	AMESDK_RUN (SEE 1.03)
8.04	AMESDK_STOP (SEE 1.04)
8.05	AMESDK_GPS_GET_DATA

## 8.01 AMESDK\_CREATE

The function helps you to easily access one GPS device. It is suit to all GPS devices in mobile market. Actually, all GPS devices use the virtual COM port driver as their access interface. By the pszDevName parameter, you can setup the COM's port number and baud rate.

```

DEVICE_HANDLE AMESDK_CREATE( LPTSTR           pszDevName,
                             UINT            iDevNum,
                             ULONG           eDevType,
                             HWND            hDisplayWindow,
                             PF_BUFFER_CALLBACK pBufferCB,
                             PVOID           pUserData
);
typedef ULONG (DEVICE_HANDLE);
    
```

### Parameters:

Parameter	IN/OUT	Description
pszDevName	IN	<b>Device Name.</b> To give a device name that is used to create specific device. Currently, we support these device names below: "Common Analog GPS Source %s". Here, %s is your port and baud rate setting. Please reference subsection, <b>Examples</b> , to obtain more introductions.
iDevNum	IN	<b>Device Number.</b> Here, SDK supports you to create 4 Common Analog GPS Sources at the same time.
eDevType	IN	<b>Device Type.</b> Number 8 for Common Analog GPS Source.
hDisplayWindow	IN	<b>Display Window.</b> It is always NULL.
pBufferCB	IN	<b>Callback Function.</b> It is always NULL.
pUserData	IN	<b>User Data.</b> It is always NULL.

### Return Value:

If AMESDK\_CREATE is successful, it will return one DEVICE\_HANDLE. If it is fail, it will show the error code. As one of below:

```

0x80000000 - Parameter, pszDevName, is wrong.
0x80000001 - Unknown error.
0x80000002 - Device queue is full already.
    
```

### Support Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
       SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Examples:

EX1: Open a media file then play it on an attached window.

```

hDev = AMESDK_CREATE( "Common Analog GPS Source port=com1 baudrate=9600",
                     0,
                     8,
                     NULL, NULL, NULL );
    
```

## 8.05 AMESDK\_GPS\_GET\_DATA

This function is to retrieve the GPS data from your GPS device.

```

    BOOL AMESDK_GPS_GET_DATA( DEVICE_HANDLE hDevHandle,
                               double *      pLongitude,
                               double *      pLatitude,
                               double *      pSpeed,
                               double *      pAngle,
                               BOOL *        pIsValid
    );

```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be retrieved.
pLongitude	OUT	<b>Longitude.</b> Specifies the longitude of the position.
pLatitude	OUT	<b>Latitude.</b> Specifies the latitude of the position.
pSpeed	OUT	<b>Speed.</b> Specifies the speed of motion of the device.
pAngle	OUT	<b>Angle.</b> Specifies the angle of the moving track.
pIsValid	OUT	<b>Valid.</b> Specifies if these data are valid.

### Return Values:

BOOL

### Support Devices:

PCTV: PD652

SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Examples:

EX1: Read GPS data from SDK.

```

AMESDK_GPS_GET_DATA( hDev, &dLongitude, &dLatitude, &dSpeed, &dAngle, &bIsValid );

```

## 9 Exported Functions for Software Deinterlacer

SUPPORT DEVICE:

PD652, SC100, SC200, SC230, SC280,  
SC290, SC2A0, SC2B0, SC300, SC310,  
SC330, SC340, SC380, SC390, SC3A0,  
SC3B0, SC500, SC510, SC580, SC590,

Exported Functions for Software Deinterlacer Programming	
9.01	AMESDK_CREATE
9.02	AMESDK_DESTROY (SEE 1.02)
9.03	AMESDK_RUN (SEE 1.03)
9.04	AMESDK_STOP (SEE 1.04)
9.05	AMESDK_GET_FORMAT (SEE 1.10)
9.06	AMESDK_SET_FORMAT (SEE 1.11)
9.07	AMESDK_DI_DEINTERLACE

## 9.01 AMESDK\_CREATE

For some customers, we can offer independent deinterlace function. The function helps you to deinterlace one interleaved video frame buffer.

```

DEVICE_HANDLE AMESDK_CREATE( LPTSTR          pszDevName,
                             UINT           iDevNum,
                             ULONG          eDevType,
                             HWND          hDisplayWindow,
                             PF_BUFFER_CALLBACK pBufferCB,
                             PVOID         pUserData
);
typedef ULONG (DEVICE_HANDLE);

```

### Parameters:

Parameter	IN/OUT	Description
pszDevName	IN	<b>Device Name.</b> To give a device name that is used to create specific device. Currently, we support these device names below: "Common Analog Deinterlacer".
iDevNum	IN	<b>Device Number.</b> Here, SDK supports you to create 64 Common Analog Deinterlacer at the same time.
eDevType	IN	<b>Device Type.</b> Number 16 for Common Analog Deinterlacer.
hDisplayWindow	IN	<b>Display Window.</b> It is always NULL.
pBufferCB	IN	<b>Callback Function.</b> It is always NULL.
pUserData	IN	<b>User Data.</b> It is always NULL.

### Return Value:

If AMESDK\_CREATE is successful, it will return one DEVICE\_HANDLE. If it is fail, it will show the error code. As one of below:

```

0x80000000 - Parameter, pszDevName, is wrong.
0x80000001 - Unknown error.
0x80000002 - Device queue is full already.

```

### Support Devices:

PCTV: PD652  
 SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
 SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Examples:

EX1: Open a media file then play it on an attached window.

```
hDev = AMESDK_CREATE( "Common Analog Deinterlacer", 0, 16, NULL, NULL, NULL );
```

## 9.07 AMESDK\_DI\_DEINTERLACE

This function is used to do deinterlacing for incoming video frame buffer. The incoming video frame buffer will become one progressive frame.

```
BOOL AMESDK_DI_DEINTERLACE( DEVICE_HANDLE    hDevHandle,  
                             BYTE *          pFrameBuffer,  
                             ULONG           nFrameBufferSize );
```

### Parameters:

Parameter	IN/OUT	Description
hDevHandle	IN	<b>Device Handle.</b> Handle to the device whose property is to be retrieved.
pFrameBuffer	IN/OUT	<b>Pinter</b> of video frame buffer.
nFrameBufferSize	IN	<b>Size</b> of video frame buffer.

### Return Values:

BOOL

### Support Devices:

PCTV: PD652  
SECU: SC100, SC200, SC230, SC280, SC290, SC2A0, SC2B0, SC300, SC310,  
      SC330, SC340, SC380, SC390, SC3A0, SC3B0, SC500, SC510, SC580, SC590,

### Examples:

EX1: To do deinterlacing for incoming video frame buffer.

```
AMESDK_DI_DEINTERLACE( hDev, pFrameBuffer, nFrameBufferSize );
```