S26X2016 SDK API Manual

v.0.5

May 2016

I.Capture Card SDK API

The S26X2016 Capture Card SDK API is designed for capturing camera signal from capture card. One camera signal is treated as one "Channel" in following sections.

The SDK's dll exports one interface, ***IPVCapCardSDK***, to let caller get necessary information and control capturing behaviour. It has five streams, for different format of source stream, including Raw Video, Raw Audio, Main Encoded Video, Sub Encoded Video and Encoded Audio.

These APIs are defined in IPVCapCardSDK.h

The Class ID and Interface ID is defined in IPVCardSrcFilter.h
Please note, not all types of capture card support all these streams, you can use the API in the exported interface to get related information.

II.Enumeration and Structures

PV_SOURCE_STREAM_TYPE : Specify the source stream type, corresponding to five output pins
PV_SOURCE_STREAM_RAW_VIDEO = (1 << 0), // "<<" means logical left shift
PV_SOURCE_STREAM_RAW_AUDIO = (1 << 1),
PV_SOURCE_STREAM_ENCODE_MAIN_VIDEO = (1 << 2),
PV_SOURCE_STREAM_ENCODE_SUB_VIDEO = (1 << 3),
PV_SOURCE_STREAM_ENCODE_AUDIO = (1 << 4)

PV_VIDEO_STANDARD : Video system standard type
PV_VIDEO_STD_NTSC,
PV_VIDEO_STD_PAL

PV_VIDEO_SIZE_TYPE : Enum types for selecting video size by common defined types
PV_VIDEO_SIZE_NONE = 0,
PV_VIDEO_SIZE_QCIF,
PV_VIDEO_SIZE_CIF,
PV_VIDEO_SIZE_4CIF,
PV_VIDEO_SIZE_HALF_D1,
PV_VIDEO_SIZE_D1,

PV_VIDEO_SIZE_QQHD,
PV_VIDEO_SIZE_QUARTER_HD,
PV_VIDEO_SIZE_HALF_HEIGHT_HD,
PV_VIDEO_SIZE_HALF_WIDTH_HD,
PV_VIDEO_SIZE_HD,
MAX_PV_VIDEO_SIZE

PV_PIXEL_FORMAT : The pixel format for raw video frames
PV_VIDEO_PIXEL_YUY2,
PV_VIDEO_PIXEL_YV12, //4:2:0 Planar
PV_VIDEO_PIXEL_Y41P, //4:1:1 Packet
PV_VIDEO_PIXEL_RGB555,
PV_VIDEO_PIXEL_RGB565,
MAX_PV_VIDEO_PIXEL

PV_VIDEO_PARAM : Video parameter name for video adjustment
PV_VIDEO_PARAM_BRIGHTNESS,
PV_VIDEO_PARAM_CONTRAST,
PV_VIDEO_PARAM_SATURATION,
PV_VIDEO_PARAM_HUE

PV_VIDEO_INPUT_MODE : Video system input mode
PV_VIDEO_INPUT_MODE_1CH = 0X01,
PV_VIDEO_INPUT_MODE_2CH = 0X03,
PV_VIDEO_INPUT_MODE_4CH = 0X0F,

PV_WDT_STATE : The type for watch dog state
PV_WDT_STATE_START,
PV_WDT_STATE_STOP,
PV_WDT_STATE_CLEAR

PV_NOTIFY_TYPE : The type for notify callback
PV_NOTIFY_TYPE_VIDEO_STATUS,,
MAX_PV_NOTIFY_TYPE

III.IPVCapCardSDK

These are a serious of APIs for setting and getting related configurations.

GetBoardCount

Syntax	*HRESULT* GetBoardCount(unsigned int *pCount)
Description	Get available board (capture card) amount in the system
Parameters	pCount [out]: board count

GetChannelCount

Syntax	*HRESULT* GetChannelCount(unsigned int *pCount)
Description	Get available channel amount in the system
Parameters	pCount [out]: channel count
Returned Value	Get channel amount in the system
Remarks	Make sure you call GetChannelCount before using other APIs.

SetVideoStandard

Syntax	*HRESULT* SetVideoStandard (PV_VIDEO_STANDARD iVideoStandard)
Description	Set video standard. Call this function if you connect camera to the system after it boots up, or you change video standard of camera
Parameters	iVideoStandard [in]: video standard, please refer to PV_VIDEO_STANDARD for value
Returned Value	S_OK: succeeded E_NOTIMPL: Not implemented (TW2809)

	E_FAIL: failed
Remarks	After calling this function, all the parameters concerning with the video such as brightness, contrast, saturation and hue will become default setup.

GetVideoStandard

Syntax	*HRESULT* GetVideoStandard (PV_VIDEO_STANDARD *pVideoStandard)
Description	Get input video standard
Parameters	pVideoStandard [out] video standard, please refer to PV_VIDEO_STANDARD for value
Returned Value	S_OK: succeeded E_FAIL: failed
Remarks	

SetVideoInputMode

Syntax	*HRESULT* SetVideoInputMode (PV_VIDEO_INPUT_MODE iVideoInputMode)
Description	Set video input mode. Call this function if you plug non-real time board into the system before it boots up
Parameters	iVideoInputMode [in]: video input mode, please refer to PV_VIDEO_INPUT_MODE for value
Returned Value	S_OK: succeeded E_NOTIMPL: Only implemented for TW68XX E_FAIL: failed
Remarks	After calling this function, all the product must re-start

GetSupportSourceStream

Syntax	*HRESULT* GetSupportSourceStream (unsigned int

	uChannelNum, unsigned int *pSupportFunctions)
Description	Query available streams for capture
Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount pSupportFunctions [out]: Bit OR value of PV_SOURCE_STREAM_TYPE
Returned Value	S_OK: succeeded E_FAIL: failed

GetSupportVideoSize

Syntax	*HRESULT* GetSupportVideoSize (unsigned int uChannelNum, PV_SOURCE_STREAM_TYPE srcStream, unsigned int *pSupportVideoSize)
Description	Query available video size for specific stream in a channel
Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount srcStream [in]: stream type, please refer to PV_SOURCE_STREAM_TYPE for value pSupportVideoSize [out]: Bit OR value of PV_VIDEO_SIZE_TYPE
Returned Value	S_OK: succeeded E_FAIL: failed
Remarks	The caller can use a bit shifted mask to check the returned capability value. For example, the result of bit AND operation between the return caps and (1 << PV_VIDEO_SIZE_QCIF) is non-zero if the stream of this channel support QCIF size.

GetSupportRawVideoPixelFormat

Syntax	*HRESULT* GetSupportRawVideoPixelFormat (unsigned int

	uChannelNum, unsigned int *pSupportVideoPixelFormat)
Description	Query available pixel format for raw video stream
Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount pSupportVideoPixelFormat [out]: Bit OR value of PV_PIXEL_FORMAT
Returned Value	S_OK: succeeded E_FAIL: failed
Remarks	The caller can use a bit shifted mask to check the returned capability value. For example, the result of bit AND operation between the return caps and (1 << PV_VIDEO_PIXEL_YUY2) is non-zero if the stream of this channel support YUY2 format.

GetSupportOSD

Syntax	*HRESULT* GetSupportOSD (unsigned int uChannelNum, bool *pSupportOSD)
Description	Get if OSD function is supported or not in a channel
Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount pSupportOSD [out]: TRUE：support OSD , FALSE：not support
Returned Value	S_OK: succeeded E_FAIL: failed

GetSupportPreviewOSD

Syntax	*HRESULT* GetSupportPreviewOSD (unsigned int uChannelNum, bool *pSupportText, bool* pSupportMask, bool* pSupportLogo)
Description	Get if OSD (On Screen Display Text) function is supported or not in a channel's video preview
Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount pSupportText [out]:

TRUE：support Preview OSD , FALSE：not support
pSupportMask [out]: Currently not used
pSupportLogo [out]: Currently not used

Returned Value S_OK: succeeded
E_FAIL: failed

GetSupportRateControlType

Syntax *HRESULT* GetSupportRateControlType (unsigned int uChannelNum, unsigned int * pSupportRateControlType)

Description Query available rate control type in a channel

Parameters uChannelNum [in]:
channel index, must be smaller than available channel amount
pSupportRateControlType [out]:
Bit OR value of PV_RATE_CONTROL_TYPE

Returned Value S_OK: succeeded
E_FAIL: failed

OpenChannel

Syntax *HRESULT* OpenChannel (unsigned int uChannelNum)

Description Open channel

Parameters uChannelNum [in]:
channel index, must be smaller than available channel amount

Returned Value S_OK: succeeded
E_FAIL: failed
PVSDK_E_CH_ALREADY_OPENED
PVSDK_E_INVALID_CHANNEL

CloseChannel

Syntax *HRESULT* CloseChannel (unsigned int uChannelNum)

Description Close channel

Parameters uChannelNum [in]:
channel index, must be smaller than available channel amount

Returned Value	S_OK: succeeded S_FALSE: channel is not opened PVSDK_E_INVALID_CHANNEL PVSDK_E_INVALID_VIDEO_SIZE

GetChannelReference

Syntax	*HRESULT Get*ChannelReference (unsigned int uChannelNum, unsigned int *pChannelReference)
Description	Get channel reference information
Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount pChannelReference [out]: channel reference, UINT_MAX: mean the channel is not sub and system under real time mode. When the reference value is the same as the channel number, meaning the channel is main and system under switch mode. Other sub channels' reference value will be set to main channel number.
Returned Value	S_OK: succeeded PVSDK_E_INVALID_CHANNEL:

RegGlobalCallback

Syntax	*HRESULT* RegGlobalCallback (PV_SOURCE_STREAM_TYPE srcStream, CaptureDataCallback pCallback, void *pContext)
Description	Register global callback
Parameters	srcStream [in]: stream type, please refer to PV_SOURCE_STREAM_TYPE for value pCallback [in]: typedef HRESULT (CALLBACK* CaptureDataCallback)(unsigned int nChannel, PV_CALLBACK_DATA *pRawData, void *pContext); nChannel [in]: channel index, must be smaller than available channel amount pRawData [out]: channel type, please refer to PV_CHANNEL_TYPE for value pContext [out]: context passed back to caller

Returned Value S_OK: succeeded
PVSDK_E_NO_CHANNEL:

Remarks

RegChannelCallback

Syntax *HRESULT* RegChannelCallback (unsigned int uChannelNum,
PV_SOURCE_STREAM_TYPE srcStream,
CaptureDataCallback pCallback,
void *pContext)

Description Register callback by channel

Parameters uChannelNum [in]:
channel index, must be smaller than available channel amount
srcStream [in]:
stream type, please refer to PV_SOURCE_STREAM_TYPE for value
pCallback [in]:
typedef HRESULT (CALLBACK* CaptureDataCallback)(unsigned int nChannel,
PV_CALLBACK_DATA *pRawData, void *pContext);
nChannel [in]:
channel index, must be smaller than available channel amount
pRawData [out]:
channel type, please refer to PV_CHANNEL_TYPE for value
pContext [out]: context passed back to caller of this function

Returned Value S_OK: succeeded
PVSDK_E_INVALID_CHANNEL:

RegNotifyCallback

Syntax *HRESULT* RegNotifyCallback (unsigned int uChannelNum, NotifyDataCallback pCallback, void *pContext)

Description Register notify callback by channel

Parameters uChannelNum [in]:
channel index, must be smaller than available channel amount
pCallback [in]:
typedef HRESULT (CALLBACK* NotifyDataCallback)(unsigned int nChannel,
PV_NOTIFY_DATA *pNotifyData, void *pContext);
nChannel [in]:
channel index, must be smaller than available channel amount
pRawData [out]:
channel type, please refer to PV_CHANNEL_TYPE for value
pContext [out]: context passed back to caller of this function

Returned Value S_OK: succeeded
PVSDK_E_INVALID_
CHANNEL:

UseDefaultParam

Syntax *HRESULT* UseDefaultParam (unsigned int uChannelNum)

Description Use default parameters by channel

Parameters uChannelNum [in]:
channel index, must be smaller than available channel amount

Returned Value S_OK: succeeded

GetRawVideoSize

Syntax *HRESULT* GetRawVideoSize (unsigned int uChannelNum, PV_VIDEO_SIZE_TYPE *pVideoSize)

Description Get raw video size

Parameters uChannelNum [in]:
channel index, must be smaller than available channel amount
pVideoSize [out]:
video size, please refer to PV_VIDEO_SIZE_TYPE for value

Returned Value S_OK: succeeded
PVSDK_E_INVALID_CHANNEL:

GetVideoSize

Syntax	*HRESULT* GetVideoSize (unsigned int uChannelNum, PV_SOURCE_STREAM_TYPE srcStream, unsigned int *pWidth, unsigned int *pHeight)
Description	Get video size for different stream type
Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount srcStream [in]: stream type, please refer to PV_SOURCE_STREAM_TYPE for value Take a attention, it only support video stream part pWidth [out]: video width depending on source stream type pHeight [out]: video height depending on source stream type
Returned Value	S_OK: succeeded PVSDK_E_INVALID_CHANNEL:

SetRawVideoSize

Syntax	*HRESULT* SetRawVideoSize (unsigned int uChannelNum, PV_VIDEO_SIZE_TYPE iVideoSize)
Description	Set Raw video size
Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount iVideoSize [in]: video size, please refer to PV_VIDEO_SIZE_TYPE for value
Returned Value	S_OK: succeeded E_FAIL: failed PVSDK_E_CH_NOT_OPENED: PVSDK_E_INVALID_VIDEO_SIZE:

GetRawVideoFrameRate

Syntax	*HRESULT* GetRawVideoFrameRate (unsigned int uChannelNum, unsigned int *pFrameRate)
Description	Get frame rate for raw video
Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount

	pFrameRate [out]: Frame rate
Returned Value	S_OK: succeeded PVSDK_E_INVALID_CHANNEL:

SetRawVideoFrameRate

Syntax	*HRESULT* SetRawVideoFrameRate (unsigned int uChannelNum, unsigned int uFrameRate)
Description	Set frame rate for raw video
Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount uFramerate [in]: frame rate
Returned Value	S_OK: succeeded E_FAIL: failed E_INVALIDARG: Invalid parameter value E_NOTIMPL: No implement for enabling switch mode PVSDK_E_CH_NOT_OPENED PVSDK_E_INVALID_FRAME_RATE

GetRawVideoPixelFormat

Syntax	*HRESULT* GetRawVideoPixelFormat (unsigned int uChannelNum, PV_PIXEL_FORMAT *pPixelFormat)
Description	Get pixel format for raw video
Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount pPixelFormat [out]: pixel format, please refer to PV_PIXEL_FORMAT for value
Returned Value	S_OK: succeeded PVSDK_E_INVALID_CHANNEL:

SetRawVideoPixelFormat

Syntax	*HRESULT* SetRawVideoPixelFormat (unsigned int uChannelNum, PV_PIXEL_FORMAT iPixelFormat)
Description	Set pixel format for raw video
Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount iPixelFormat [in]: pixel format, please refer to PV_VIDEO_SIZE_TYPE for value
Returned Value	S_OK: succeeded E_FAIL: failed E_INVALIDARG: Invalid parameter value PVSDK_E_CH_NOT_OPENED:

GetVideoParam

Syntax	*HRESULT* GetVideoParam(unsigned int uChannelNum, PV_VIDEO_PARAM iVideoParam, unsigned int *pValue)
Description	Get value for specifying video parameter
Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount iVideoParam [in]: specify video parameter, please refer to PV_VIDEO_PARAM for value pValue [out]: Current value for specifying video parameter
Returned Value	S_OK: succeeded E_FAIL: failed E_NOTIMPL: Not implemented (TW2809) PVSDK_E_CH_NOT_OPENED:

SetVideoParam

Syntax	*HRESULT* SetVideoParam(unsigned int uChannelNum, PV_VIDEO_PARAM iVideoParam, UINT uValue)
Description	Set value for specifying video parameter
Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount

	iVideoParam [in]: specify video parameter, please refer to PV_VIDEO_PARAM for value uValue [in]: value for specifying video parameter
Returned Value	S_OK: succeeded E_FAIL: failed E_NOTIMPL: Not implemented (TW2809) E_INVALIDARG: One or more arguments are not valid PVSDK_E_CH_NOT_OPENED

GetVideoParamRange

Syntax	*HRESULT* GetVideoParamRange (unsigned int uChannelNum, PV_VIDEO_PARAM iVideoParam, UINT *pMaxValue, UINT *pMinValue)
Description	Get available range for specifying video parameter
Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount iVideoParam [in]: specify video parameter, please refer to PV_VIDEO_PARAM for value pMaxValue [out]: maximum value for specifying video parameter pMinValue [out]: minimum value for specifying video parameter
Returned Value	S_OK: succeeded E_FAIL: failed E_NOTIMPL: Not implemented (TW2809) PVSDK_E_CH_NOT_OPENED:

SetDefaultVideoParam

Syntax	*HRESULT* SetDefaultVideoParam(unsigned int uChannelNum)
Description	Set value for specifying video parameter
Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount

Returned Value	S_OK: succeeded E_FAIL: failed E_NOTIMPL: Not implemented (TW2809) PVSDK_E_CH_NOT_OPENED:

GetEnableOSD

Syntax	*HRESULT* GetEnableOSD(unsigned int uChannelNum, bool * pbEnable)
Description	Get OSD status
Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount pbEnable [out]: TRUE：enable OSD overlay, FALSE：disable OSD overlay
Returned Value	S_OK: succeeded PVSDK_E_INVALID_CHANNEL:

SetEnableOSD

Syntax	*HRESULT* SetEnableOSD (unsigned int uChannelNum, boo bEnable)
Description	Set OSD display. It can overlay current system time (Such year, month, date, second) and text in the video and this can be transparent.
Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount bEnable [in]: TRUE：enable OSD overlay, FALSE：disable OSD overlay
Returned Value	S_OK: succeeded S_FALSE: TW68xx not support E_FAIL: failed PVSDK_E_CH_NOT_OPENED

GetPreviewOSD

Syntax *HRESULT* GetPreviewOSD(unsigned int uChannelNum,
bool * pbEnableText,
bool * pbEnableMask,
bool * pbEnableLogo)

Description Get text preview OSD status

Parameters uChannelNum [in]:
channel index, must be smaller than available channel amount
pbEnableText [out]:
TRUE：enable text preview OSD overlay,
FALSE：disable text preview OSD
pbEnableMask [out]: Currently not used
pbEnableLogo [out]: Currently not used

Returned Value S_OK: succeeded
E_FAIL: failed

SetPreviewOSD

Syntax *HRESULT* SetPreviewOSD(unsigned int uChannelNum,
bool pbEnableText,
bool pbEnableMask,
bool pbEnableLogo)

Description Set text preview OSD status

Parameters uChannelNum [in]:
channel index, must be smaller than available channel amount
pbEnableText [in]:
TRUE：enable text preview OSD
FALSE： disable text preview OSD
pbEnableMask [in]: Currently not used
pbEnableLogo [in]: Currently not used

Returned Value S_OK: succeeded
E_FAIL: failed

GetOSDText

Syntax *HRESULT* GetOSDText(unsigned int uChannelNum,
int iLne, char* text, int* x, int* y,
PV_VIDEO_SIZE_TYPE videoType,

int* iFontSize, bool* bEnable)

Description Get text OSD text string content

Parameters
uChannelNum [in]:
channel index, must be smaller than available channel amount
iLine [in]: currently must be 1
text [out]: retrieved text content(Must have enough space to hold the text)
x [out]: x position on screen to display text OSD
y [out]: y position on screen to display text OSD
videoType [in]: video size
iFontSize [out]:
1：font size 12, 2 : font size 16, 3 : font size 24,
4 : font size 32, 5 : font size 40, 6 : font size 48
bEnable [out]: encode streaming text osd enable status.

Returned Value
S_OK: succeeded
E_FAIL: failed

SetOSDText

Syntax *HRESULT* SetOSDText(unsigned int uChannelNum,
int iLne, char* text, int x, int y,
PV_VIDEO_SIZE_TYPE videoType,
int iFontSize, bool bEnable)

Description Set text OSD text string content

Parameters
uChannelNum [in]:
channel index, must be smaller than available channel amount
iLine [in]: currently must be 1
text [in]: the text content to display as osd
x [in]: x position on screen to display text OSD
y [in]: y position on screen to display text OSD
videoType [in]: video size
iFontSize [in]:
1：font size 12, 2 : font size 16, 3 : font size 24,
4 : font size 32, 5 : font size 40, 6 : font size 48
bEnable [in]: enable/disable encode streaming text osd.

Returned Value	S_OK: succeeded E_FAIL: failed
Note	"font" folder containing the fonts' files must be in the application program's folder

StartRawData

Syntax	*HRESULT* StartRawData (unsigned int uChannelNum, bool bAudioEnable)
Description	Start Raw video data
Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount bAudioEnable [in]: TRUE：enable audio, FALSE：disable audio
Returned Value	S_OK: succeeded E_FAIL: failed PVSDK_E_CH_NOT_OPENED

StopRawData

Syntax	*HRESULT* StopRawData (unsigned int uChannelNum)
Description	Stop Raw video data
Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount
Returned Value	S_OK: succeeded E_FAIL: failed PVSDK_E_CH_NOT_OPENED

GetBitrateControlRange

Syntax	*HRESULT* GetBitrateControlRange(unsigned int uChannelNum, unsigned long *pMaxKbps , unsigned long *pMinKbps, bool bSubStream=false)
Description	Get BitRate Range

Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount pMaxKbps [out]: maximum bitrate value for specifying pMinKbps [out]: minimum bitrate value for specifying bSubStream [in]: encode stream type, true: Main stream, false: Sub stream
Returned Value	S_OK: succeeded E_FAIL: failed E_NOTIMPL: Not implemented (TW68XX) PVSDK_E_CH_NOT_OPENED

GetVideoEncodeParam

Syntax	*HRESULT* GetVideoEncodeParam(unsigned int uChannelNum, PV_VIDEO_ENCODE_PARAM * pEncodeParam, bool bSubStream=false)
Description	Get video encode parameter
Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount pEncodeParam [out]: pixel format, please refer to PV_VIDEO_ENCODE_PARAM for value bSubStream [in]: encode stream type, true: Main stream, false: Sub stream
Returned Value	S_OK: succeeded E_FAIL: failed PVSDK_E_INVALID_CHANNEL:

SetVideoEncodeParam

Syntax *HRESULT* SetVideoEncodeParam(unsigned int uChannelNum, PV_VIDEO_ENCODE_PARAM EncodeParam, bool bSubStream=false)

Description Set video encode parameter

Parameters uChannelNum [in]:
channel index, must be smaller than available channel amount
EncodeParam [in]:
pixel format, please refer to PV_VIDEO_ENCODE_PARAM for value
bSubStream [in]:
encode stream type, true: Main stream, false: Sub stream

Returned Value S_OK: succeeded
E_FAIL: failed
E_NOTIMPL: Not implemented (TW68XX)
PVSDK_E_CH_NOT_OPENED
PVSDK_E_INVALID_VIDEO_SIZE
PVSDK_E_INVALID_RATE_CTRL_TYPE

SetEncodeAudioFlag

Syntax *HRESULT* SetEncodeAudioFlag(unsigned int uChannelNum, bool bEnable)

Description Enable encode audio parameter

Parameters uChannelNum [in]:
channel index, must be smaller than available channel amount
bEnable [in]:
enable audio, true: enable audio, false: disable audio

Returned Value S_OK: succeeded
E_FAIL: failed
E_NOTIMPL: Not implemented (TW68XX)
PVSDK_E_CH_NOT_OPENED

StartEncodeData

Syntax *HRESULT* StartEncodeData(unsigned int uChannelNum, bool

	bSubStream=false)
Description	Start encode data
Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount bSubStream [in]: encode stream type, true: Main stream, false: Sub stream
Returned Value	S_OK: succeeded E_FAIL: failed E_NOTIMPL: Not implemented (TW68XX) PVSDK_E_CH_NOT_OPENED

StopEncodeData

Syntax	*HRESULT* StopEncodeData(unsigned int uChannelNum, bool bSubStream=false)
Description	Stop encode data
Parameters	uChannelNum [in]: channel index, must be smaller than available channel amount bSubStream [in]: encode stream type, true: Main stream, false: Sub stream
Returned Value	S_OK: succeeded E_FAIL: failed E_NOTIMPL: Not implemented (TW68XX) PVSDK_E_CH_NOT_OPENED

DisableNoisySignalColorKiller

Syntax	*HRESULT* DisableNoisySignalColorKiller()
Description	Disable color killer function
Parameters	N/A
Returned Value	S_OK: succeeded E_FAIL: failed E_NOTIMPL: Only implemented for TW68XX

GetIoInfo

Syntax	*HRESULT* GetIoInfo(PV_IO_INFO *pIoInfo)
Description	Receive IO board information under system

Parameters	pIoInfo [out]: I/O information, please refer to PV_IO_INFO for value
Returned Value	S_OK: succeeded

GetIoData

Syntax	*HRESULT* GetIoData(unsigned int uIoBoardNum, DWORD *dwData)
Description	Receive GPIO input and output Data
Parameters	uIoBoardNum [in]: I/O board index dwData [out]: Get GPIO status, bit 0-7: Input pin, bit 8-15: output pin, 1: High, 0: Low
Returned Value	S_OK: succeeded E_FAIL: failed

SetIoData

Syntax	*HRESULT* SetIoData(unsigned int uIoBoardNum, DWORD dwData)
Description	Set GPIO output Data
Parameters	uIoBoardNum [in]: I/O board index dwData [out]: Set GPIO output status, bit 0-7: Input pin(don't care), bit 8-15: output pin, 1: High, 0: Low
Returned Value	S_OK: succeeded E_FAIL: failed
Remarks	Set high or low status only for output pins.

GetProtectId

Syntax	*HRESULT* GetProtectId(unsigned int uBoardNum, BYTE *byData)
Description	Receive protect ID
Parameters	uBoardNum [in]: board index byData [out]:

	ID comes from board's chip, it is fixed value.
Returned Value	S_OK: succeeded E_FAIL: failed
Remarks	The caller can use the ID to protect self's software.

SetWDT

Syntax	*HRESULT* SetWDT(unsigned int uBoardNum, PV_WDT_STATUS WDTState)
Description	Set H/W watch dog status
Parameters	uBoardNum [in]: board index WDTState [in]: Watch dog status, please refer to PV_WDT_STATUS for value
Returned Value	S_OK: succeeded E_FAIL: failed
Remarks	When the system freeze, the H/W watch dog will restart your system if you enable WDT function.

IV. Source Code Sample

The subfolder **S26XDemoVS2010** has all MS VisualStudio2010Pro projects and C++ source and include files(some include files are also under the "include" folder in parallel with **S26XDemoVS2010** folder) for creating executable program Demo.exe, which can be run from its **S26XDemoVS2010\bin\x86** subfolder where all necessary supporting DLL and font files are copied to already.

Before running the executable sample program Demo.exe, device drivers must be installed from folder

Windows\DirectShow_V0.38\driver_x86 or **driver_x64** depending on the running MS Windows is 32-bit or 64-bit.

Also the regfilter_x86.bat or regfilter_x64.bat video codec filter registration batch file must be run for the appropriate MS Windows so that recorded video files can be played back by VideoLan or other MediaPlayer software.

The VisualC++ redistribution program vcredist_x86.exe or vcredist_x64.exe should also be run once on the MS Windows for executable program Demo.exe or other user created programs to run under appropriate Windows.

The 32-Bit application program Demo.exe can be run on both 32-Bit and 64-Bit MS Windows: on 64-Bit Windows it runs as 32-Bit application, therefore the only different requirement for it to run under 64-Bit Windows is to install 64-Bit device drivers.

V. Demo.exe Recording

Click the “Bare stream, write to file” CheckBox inside the “Encode” Dialog under “Set Parameters” Window to record, clear this CheckBox will stop recording to files.

VI. Bandwidth Consideration

4-Channel:
Raw (D1) : 720*480*30fps*3/2 ~= 14.83 MB / sec

Main: 2Mb/ sec = 0.25 MB / sec
Sub: 1Mb/ sec = 0.13 MB / sec

8-Channel & 16-Channel:

Raw (D1) : 720*480*30fps*3/2 ~=14.83MB/ sec
Main: 2Mb/ sec = 0.25 MB / sec
Sub: 1Mb/ sec = 0.13 MB / sec

Note Audio data is taken into consideration in counting bandwidth.