DVP-7020BE

16 Channel PCI-bus Surveillance Capture card

Acknowledgments
IBM and PC are trademarks of International Business Machines Corporation. MS-DOS, Windows, Microsoft Visual C++ and Visual BASIC are trade-marks of Microsoft Corporation. Intel and Pentium are trademarks of Intel Corporation. Delphi and C++ Builder are trademarks of Inprise Corporation.

CE notification
The DVP-7020BE, developed by ADVANTECH CO., LTD., has passed the CE test for environmental specifications when shielded cables are used for external wiring. We recommend the use of shielded cables. This kind of cable is available from Advantech. Please contact your local supplier for ordering information

On-line Technical Support
For technical support and service, please visit our support website at: **http://www.advantech.com/support**

Part No. 2062702010 1st Edition
Printed in Taiwan July. 2006
Rev. 1.00

CHAPTER 1

General Information

Chapter 1 General Information

DVP-7020BE is 16 channel input, PCI-bus video capture card. It supports up to 16 channel input by share-frame technology and captures up to D1 resolution at 120/100 fps frame rate. DVP-7020BE supports NTSC/PAL composite video input through BNC connectors and digitizes the data to PC through PCI bus. The DVP-7020BE is a digital video surveillance card with SDK (software develop kit).

1.1 Hardware Requirement

- Intel Pentium III 1GHz or above (CPU speed depends on video frame rate, channels and resolution)
- 256 MB RAM or above
- Free PCI slot(s)
- CD-ROM
- Hard disk with 1G free space

1.2 Software Requirement

- Microsoft Windows XP with DirectX 9.0 or above

1.3 Block Diagram

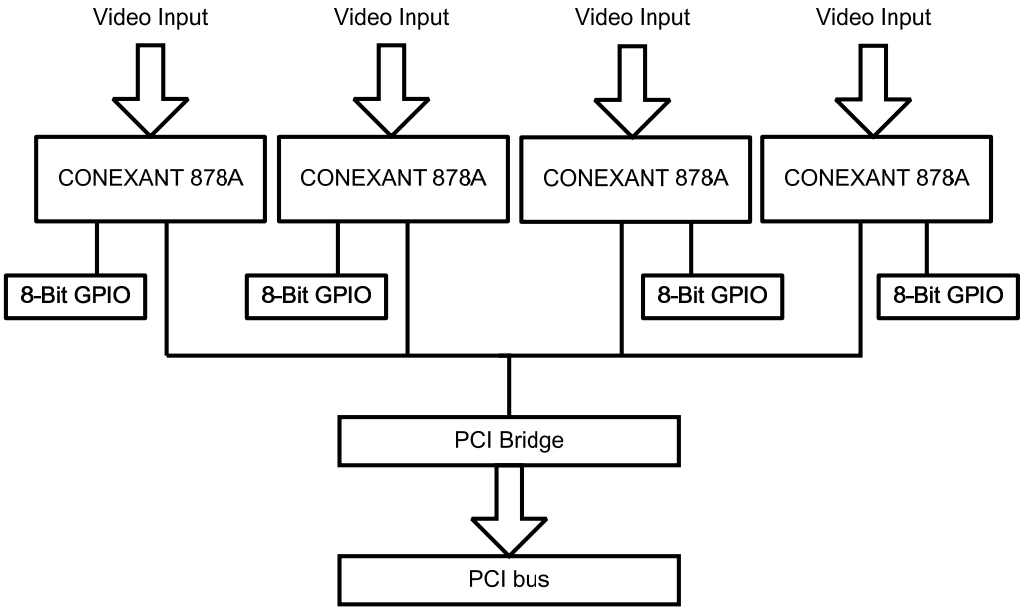


Figure 1.1 System diagram

1.4 Packing List

DVP-7020BE PCI capture card	X 1
Utility CD (Driver, Manual, SDK, Sample, Sample source code)	X 1
Connection cable for WDT	X 1
DVA-210 (4 Channel expand board)	X 3

1.5 Dimensions

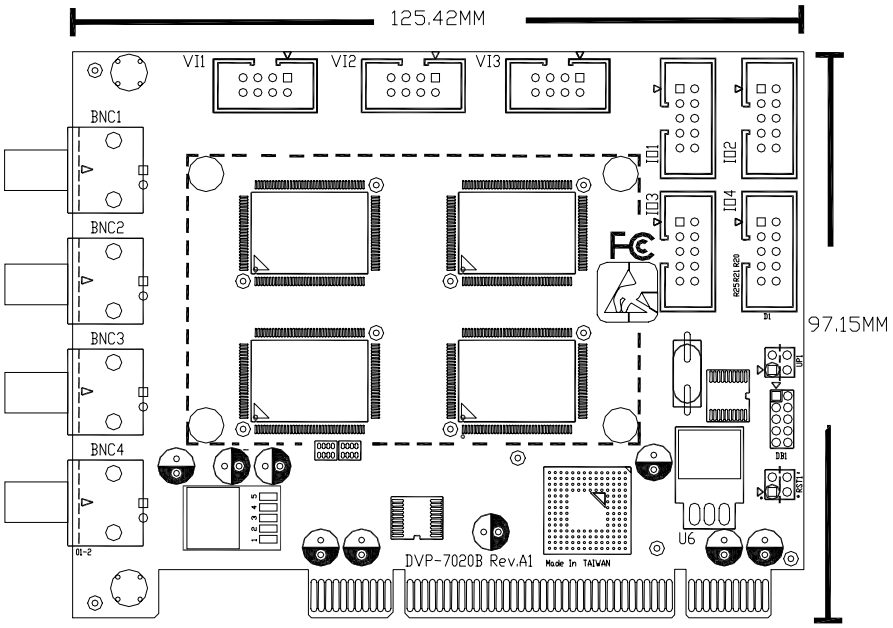


Figure 1.2 Dimensions

1.6 Connector location

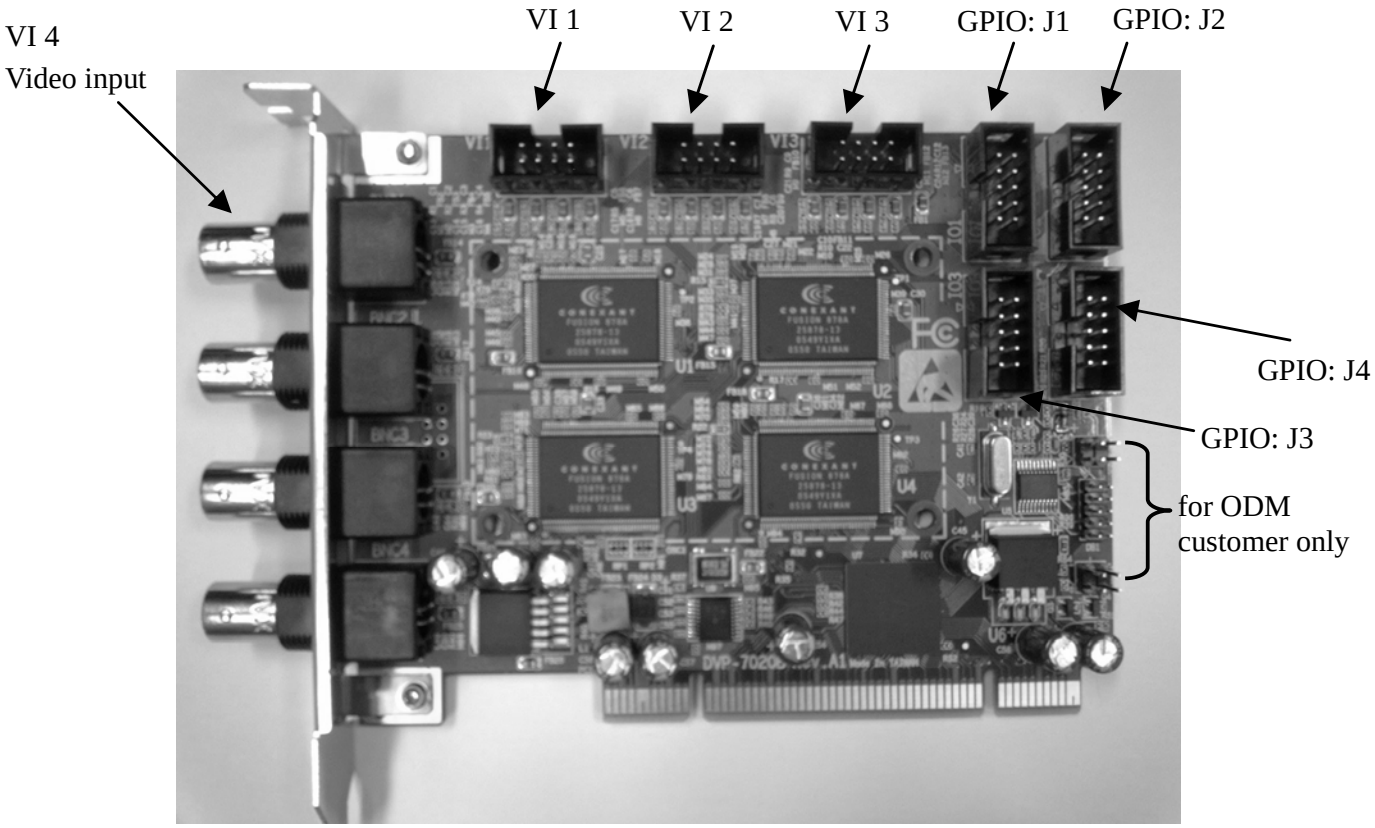


Figure 1.3 Connector location

1.7 Pin definition

1.7.1 GPIO: J1

- 8 bit TTL/CMOS level Digital I/O.

GPIO (J1) Pin define	
Pin no.	**Description**
Pin 1	OUT0
Pin 2	OUT1
Pin 3	OUT2
Pin 4	OUT3
Pin 5	IN0
Pin 6	IN1
Pin 7	IN2
Pin 8	IN3
Pin 9	VCC
Pin 10	GND

Table 1.1 GPIO J1 pin definition

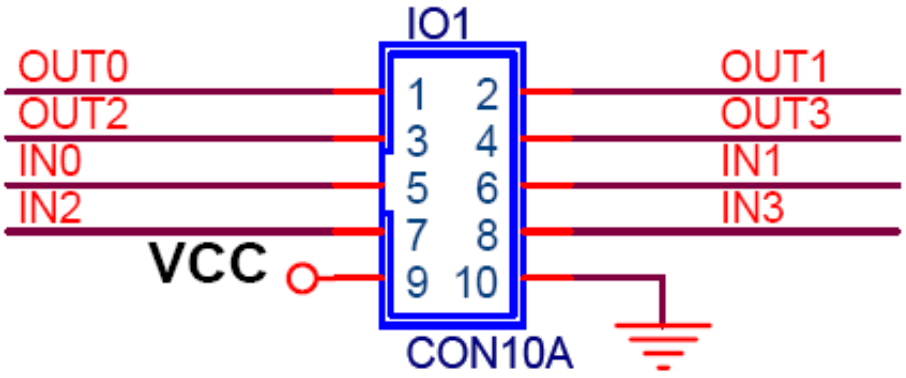


Figure 1.4 GPIO(J1) pin definition

1.7.2 GPIO: J2

- 8 bit TTL/CMOS level Digital I/O.

GPIO (J2) Pin define	
Pin no.	**Description**
Pin 1	OUT4
Pin 2	OUT5
Pin 3	OUT6
Pin 4	OUT7
Pin 5	IN4
Pin 6	IN5
Pin 7	IN6
Pin 8	IN7
Pin 9	VCC
Pin 10	GND

Table 1.2 GPIO J2 pin definition

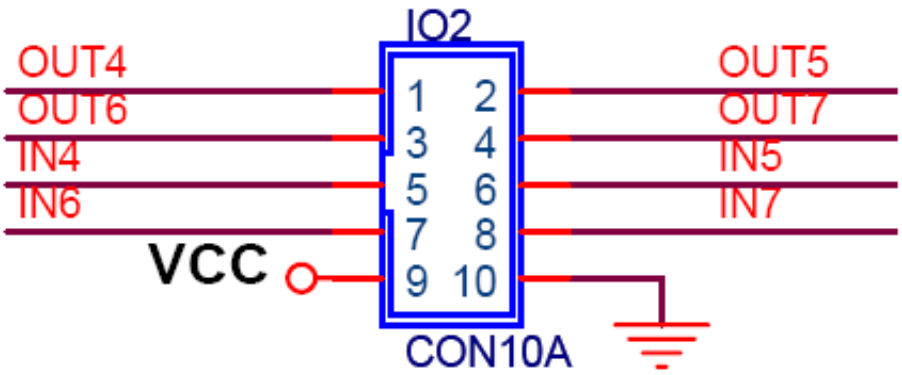


Figure 1.5 GPIO(J2) pin definition

1.7.3 GPIO: J3

- 8 bit TTL/CMOS level Digital I/O.

GPIO (J3) Pin define	
Pin no.	**Description**
Pin 1	OUT8
Pin 2	OUT9
Pin 3	OUT10
Pin 4	OUT11
Pin 5	IN8
Pin 6	IN9
Pin 7	IN10
Pin 8	IN11
Pin 9	VCC
Pin 10	GND

Table 1.3 GPIO J3 pin definition

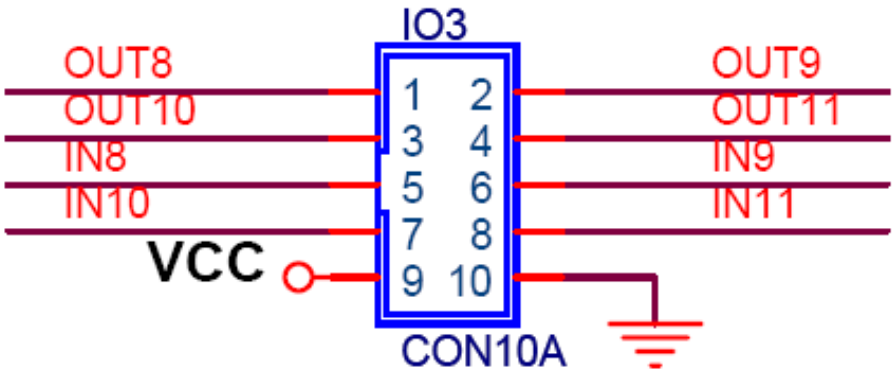


Figure 1.6 GPIO(3) pin definition

1.7.4 GPIO: J4

- 8 bit TTL/CMOS level Digital I/O.

GPIO (J4) Pin define	
Pin no.	**Description**
Pin 1	OUT8
Pin 2	OUT9
Pin 3	OUT10
Pin 4	OUT11
Pin 5	IN8
Pin 6	IN9
Pin 7	IN10
Pin 8	IN11
Pin 9	VCC
Pin 10	GND

Table 1.4 GPIO J4 pin definition

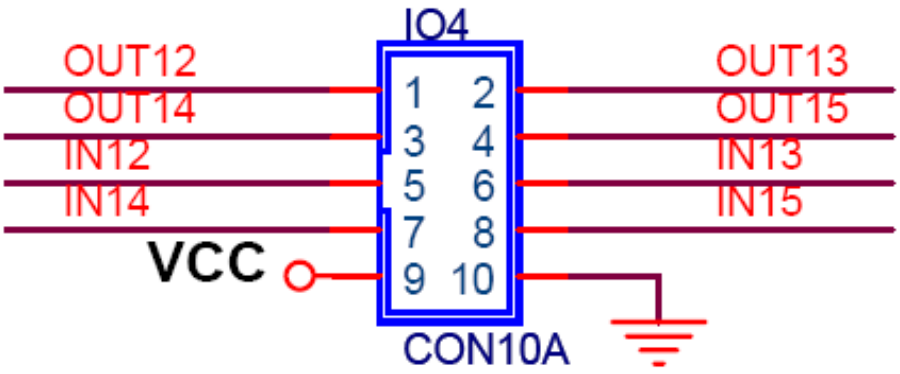


Figure 1.7 GPIO(J4) pin definition

1.7.6 External video input: VI

By share frame technology, DVP-7020BE can receive 16 channel composite inputs through 4 VIs. The description for these 4VIs are shown in Figure 1.8~1.11

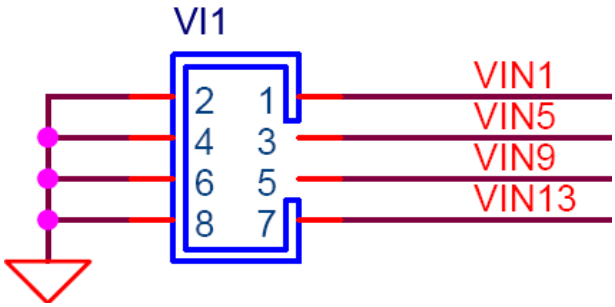


Figure 1.8 VI 1 pin definition

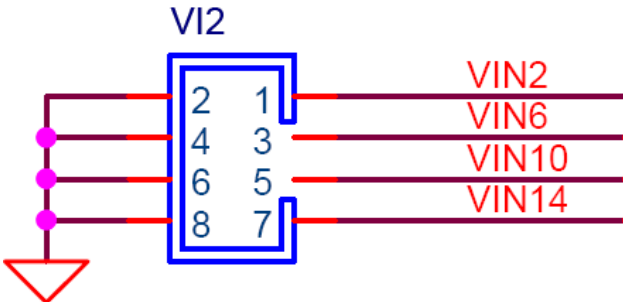


Figure 1.9 VI 2 pin definition

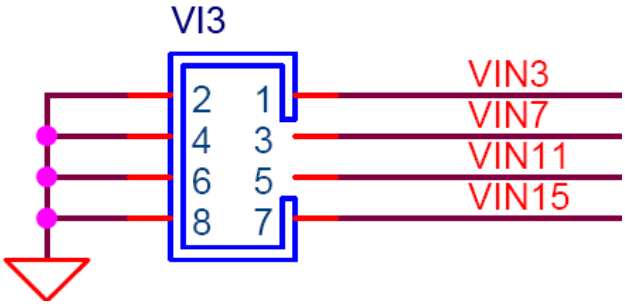


Figure 1.10 VI 3 pin definition

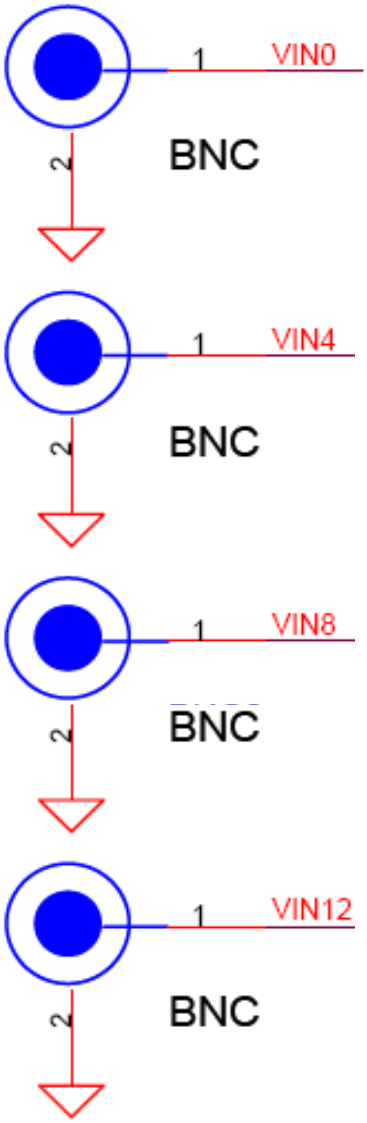


Figure 1.11 VI 4pin definition

1.8 Hardware Installation

1 Turn off your computer and unplug the power cord.

2 Remove the cover of your computer.

3 Touch the metal part on the surface of your computer to neutralize the static electricity that might be on your body.

4 Place the DVP-7020BE into Mother Board PCI slot.

5 Connect appropriate accessories (Video cable to camera. if necessary) to the DVP-7020BE.

6 Replace the cover of your computer chassis.

7 Plug in the power cord and turn on the computer.

Note: Keep the anti-static bag for future use. You might need the original bag to store the card if you have to remove the card from the PC or transport it elsewhere.

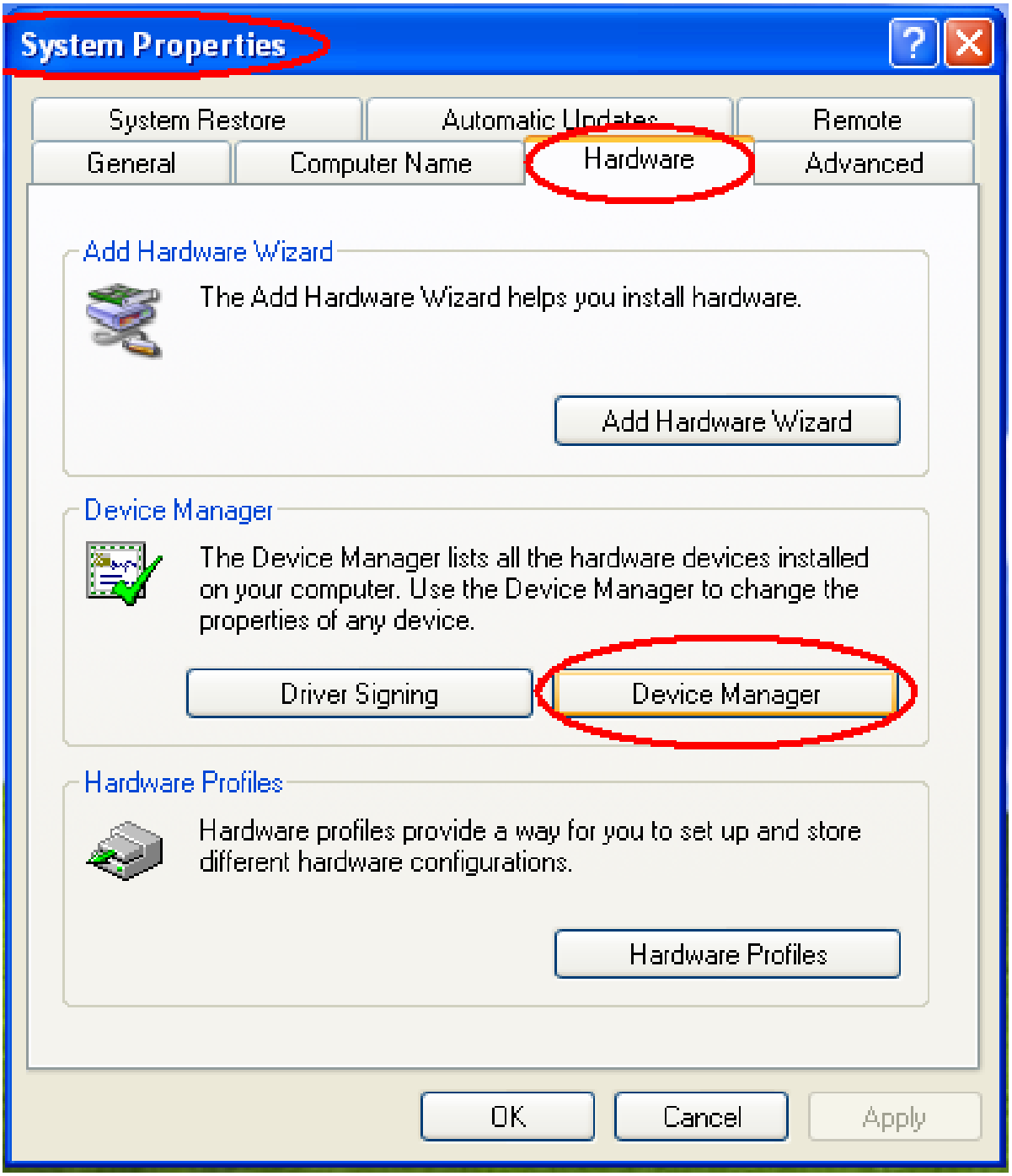
1.9 Software / Driver Installation

Before you begin

To facilitate the installation of the enhanced display device drivers and utility software, you should read the instructions in this chapter carefully before you attempt installation. The device drivers for the DVP-7020BE board are located on the software installation CD. The auto-run function of the driver CD will guide and link you to the utilities and device drivers under Windows system. Before you begin, it is important to note that most display drivers need to have the relevant software application already installed in the system prior to installing the enhanced display drivers. In addition, many of the installation procedures assume that you are familiar with both the relevant software applications and operating system commands. Review the relevant operating system commands and the pertinent sections of your application software user's manual before performing the installation.

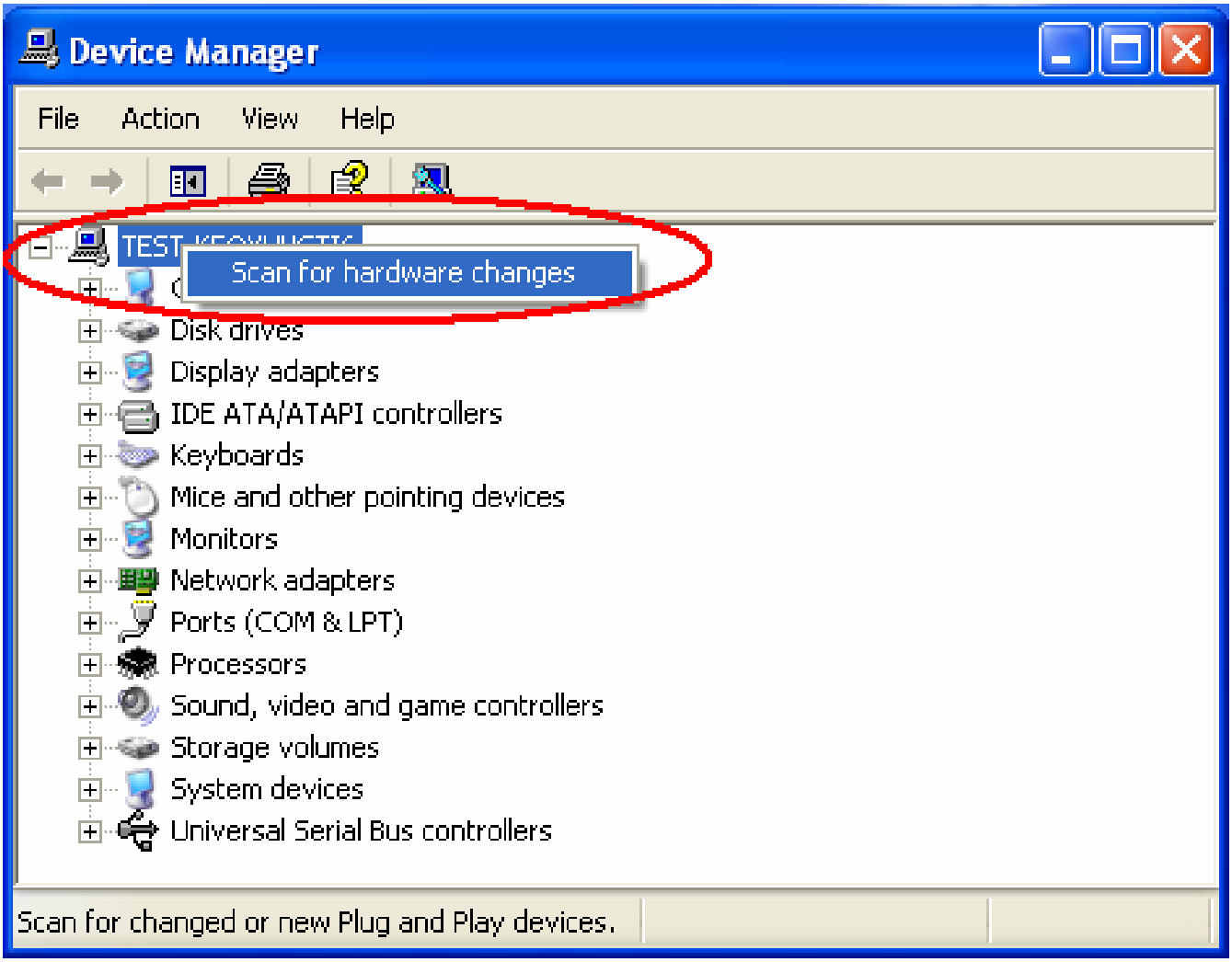
Installation

Step 1: Pop-up the “System Properties” window, choose the “Hardware” page, and press the “Device Manager” bottom.

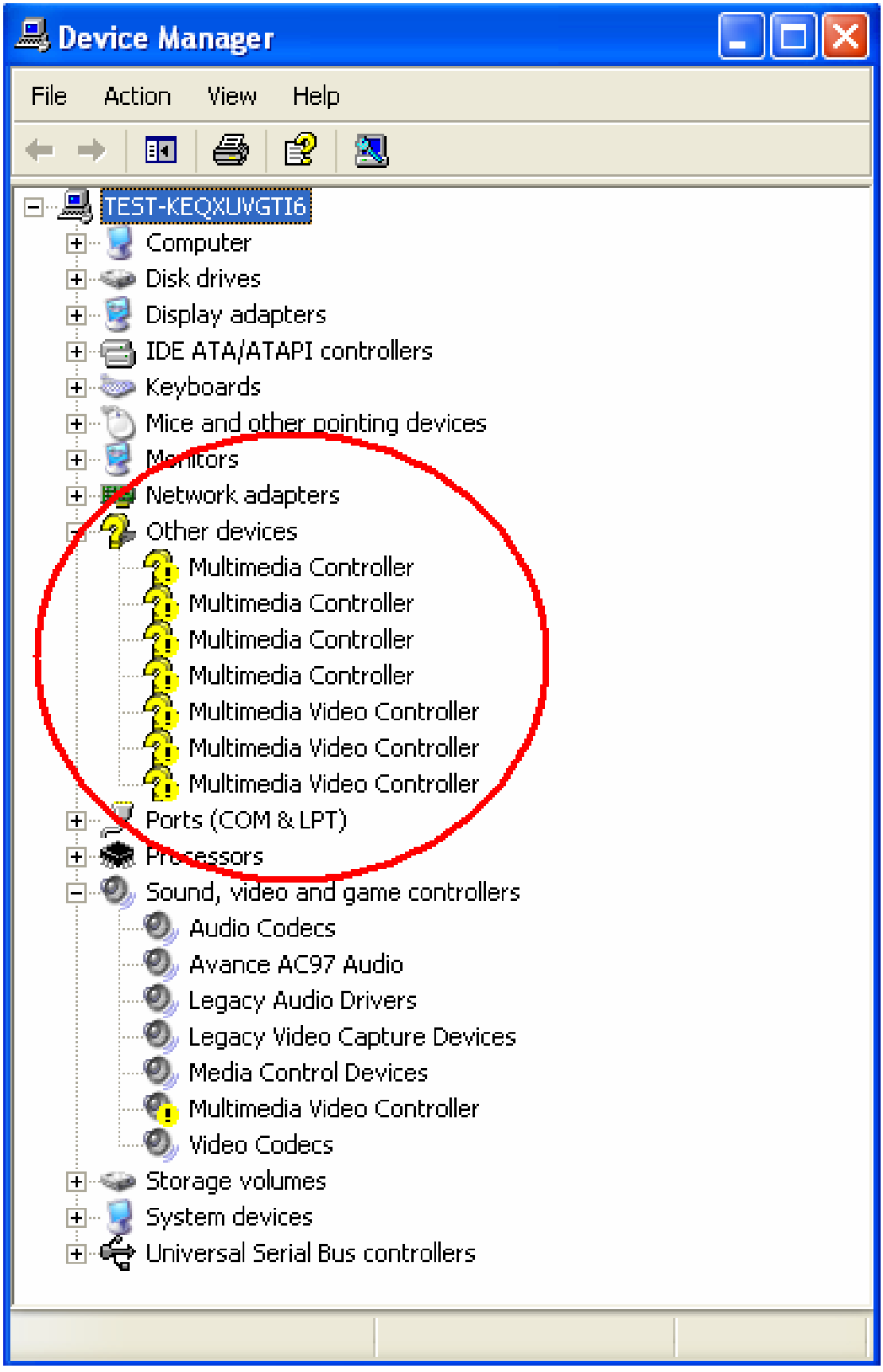


Step 2: Click the PC icon and press the left bottom of the mouse.

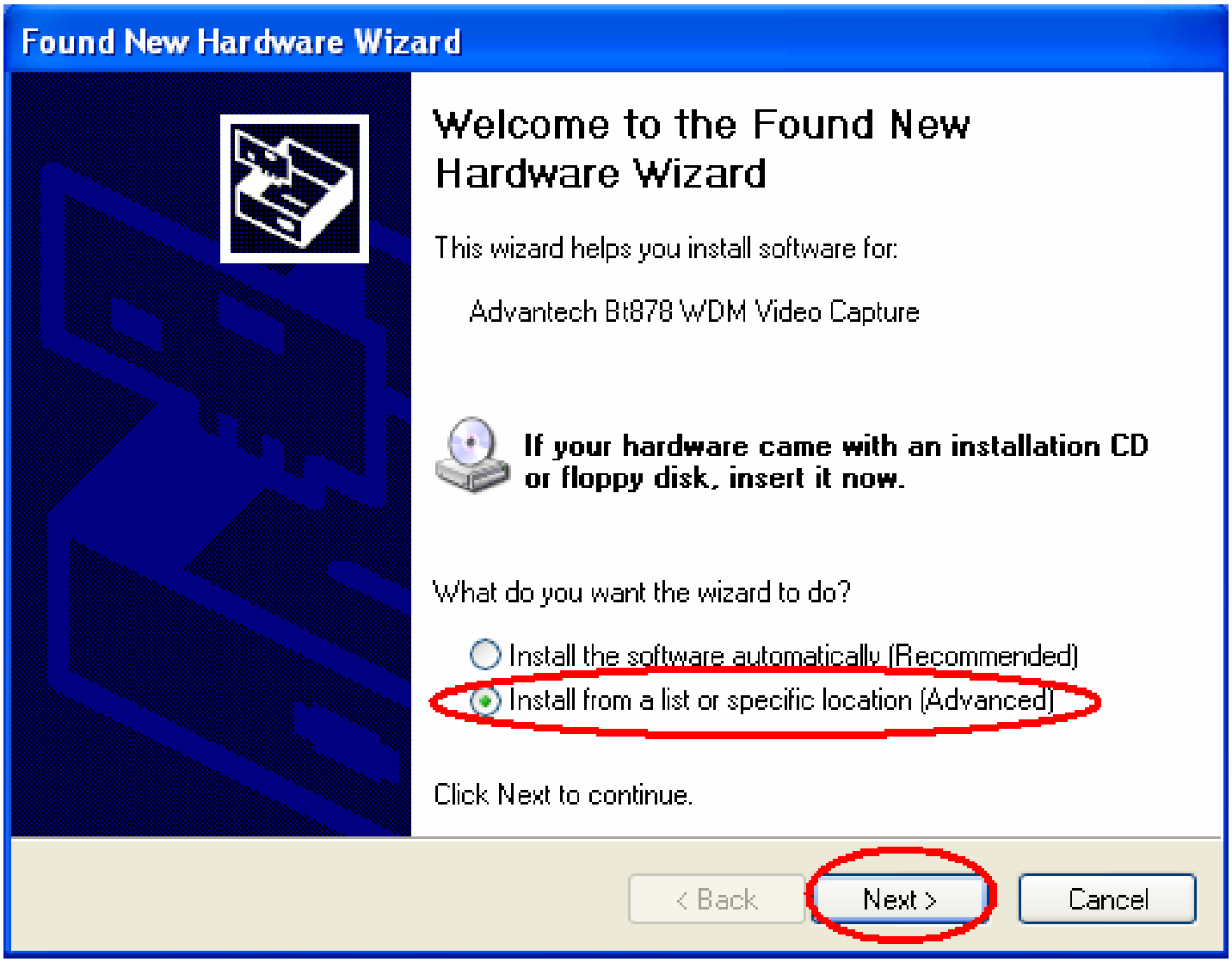
Press the “Scan for hardware changes”.



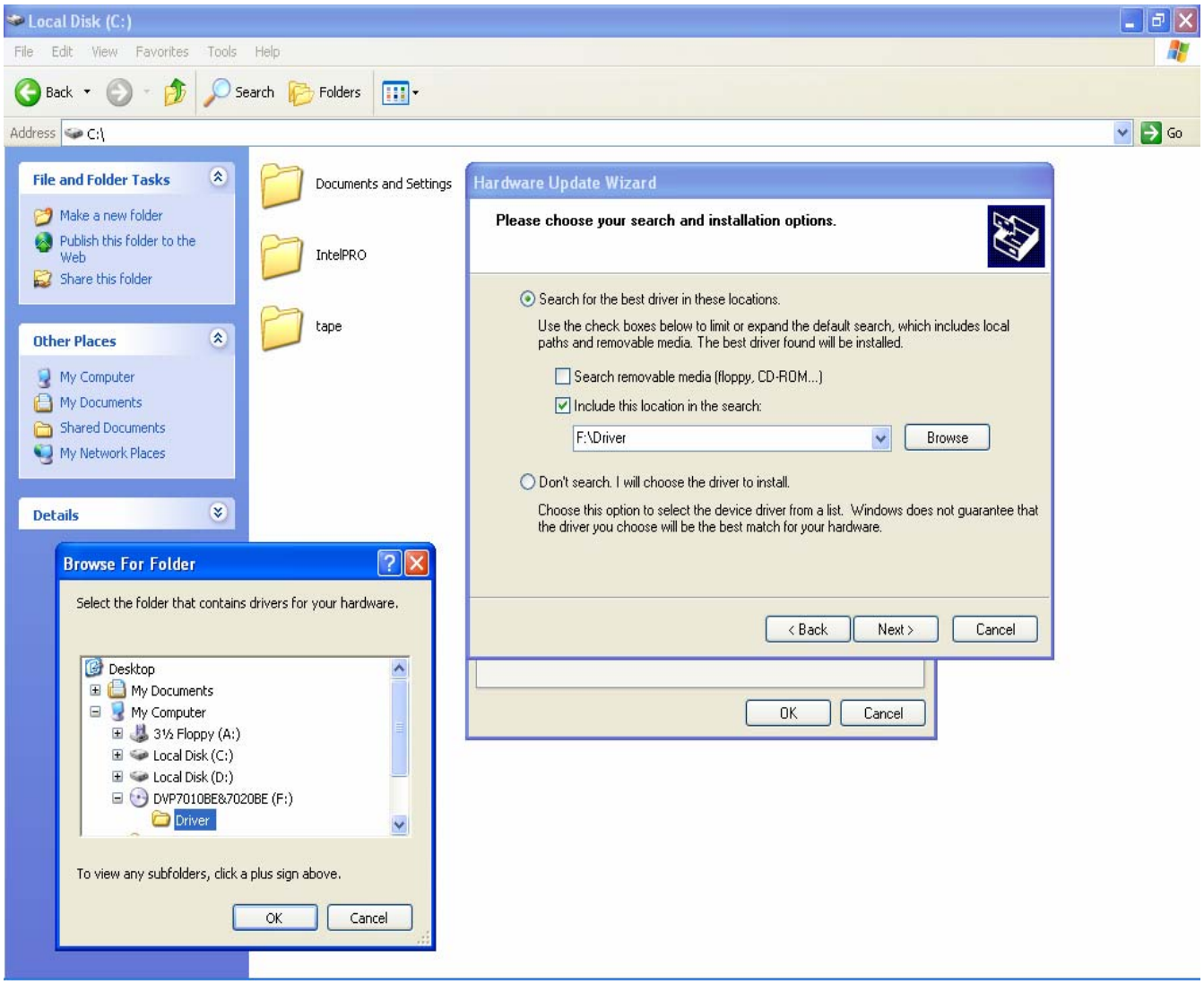
Step 3: The system will show the un-known devices like below window.



Step 4: Click the below icon to specify the driver location.

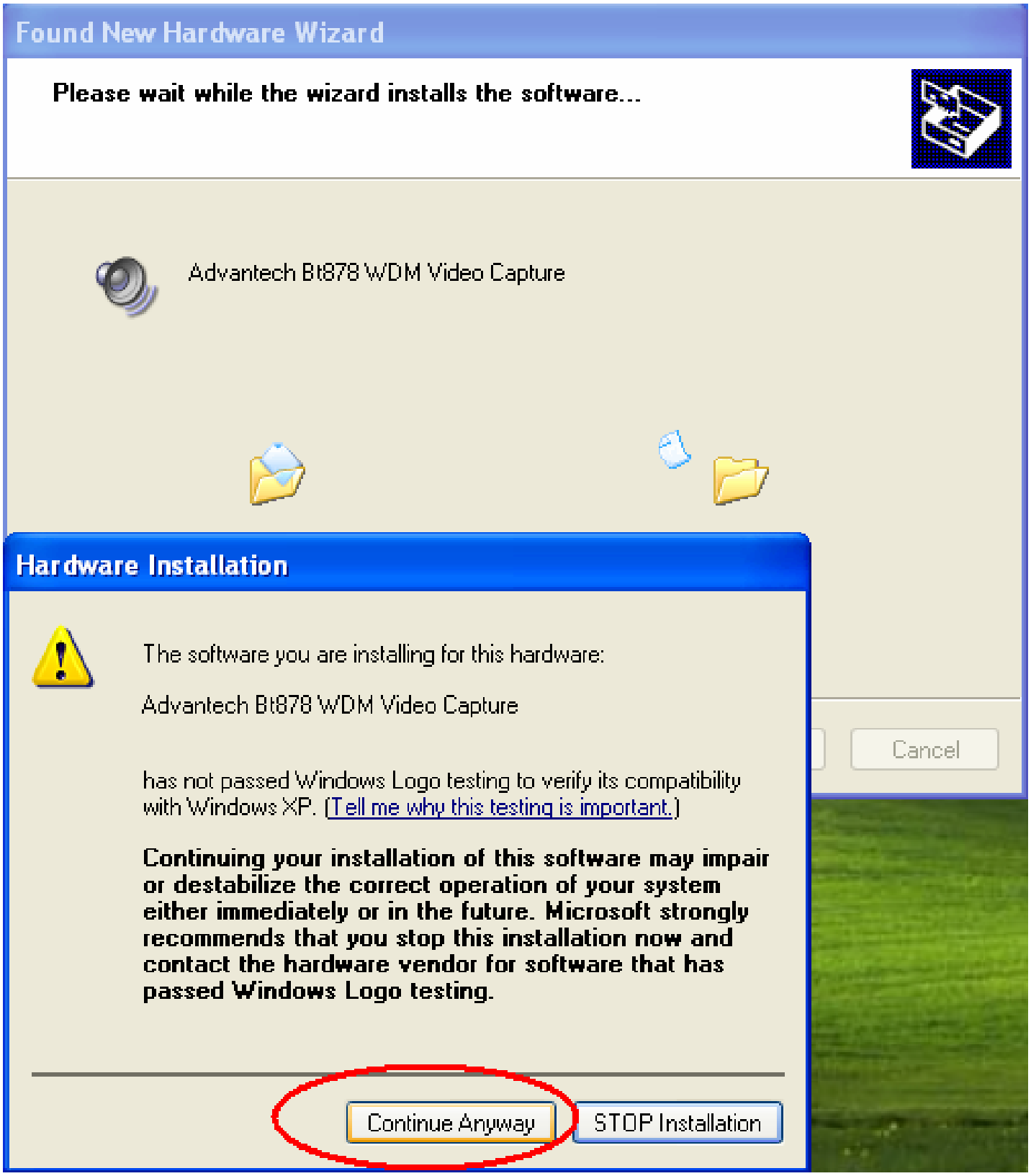


Step 5: Specify the driver under the CD_ROM\driver

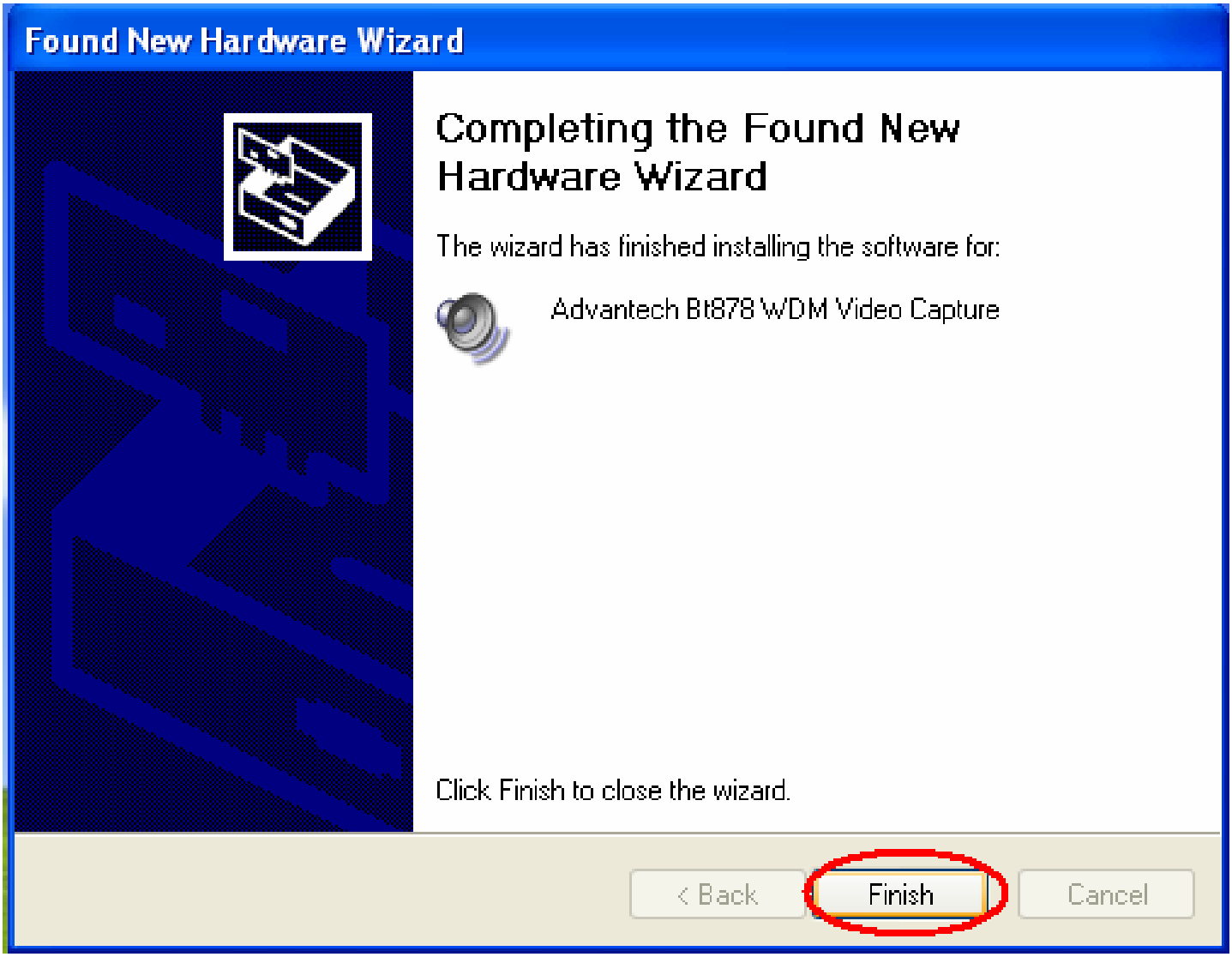


Step 6: Push the “Next” bottom to process the installation.

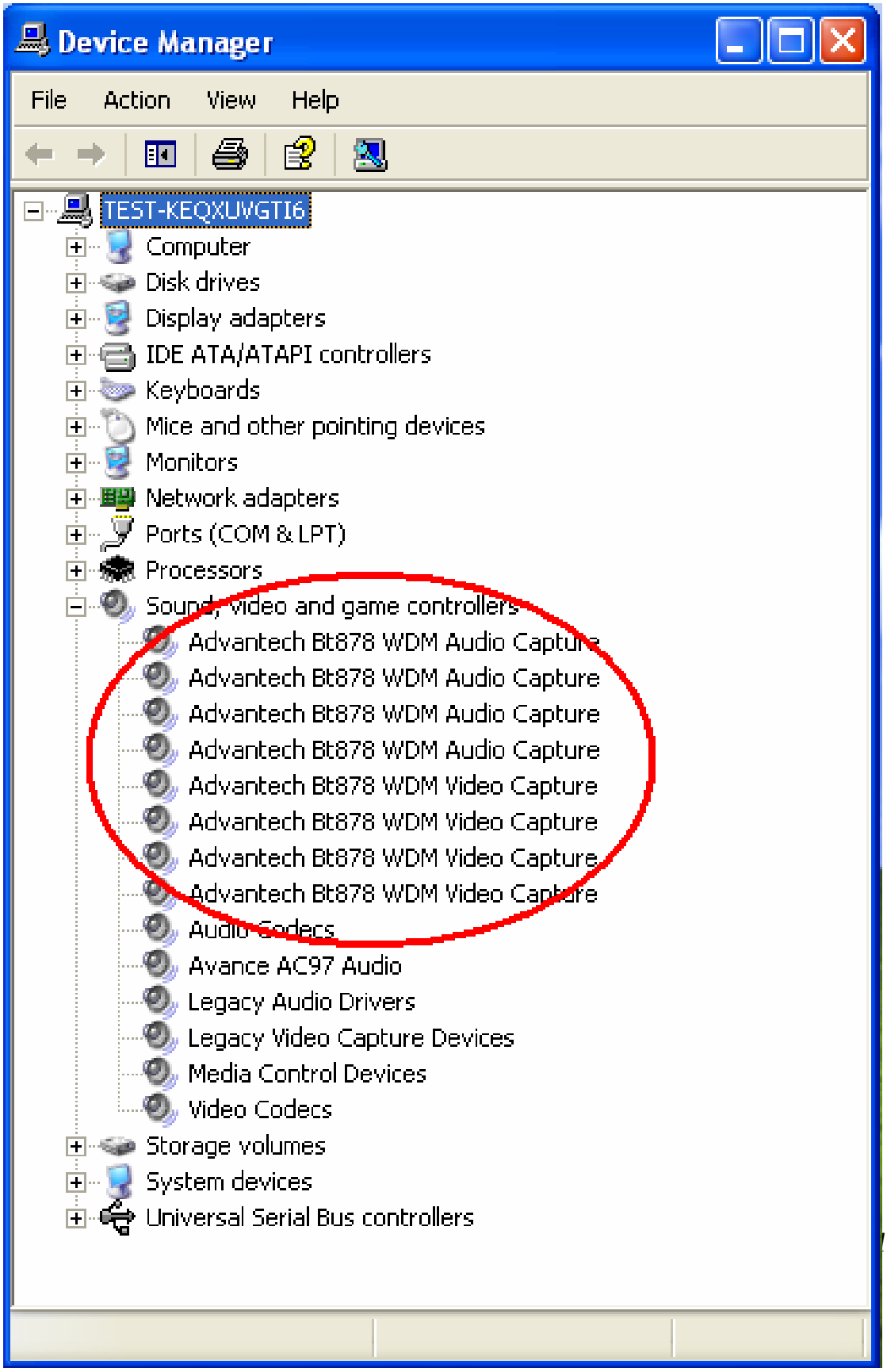
Step 7: Continuing the installation.



Step 8: Press the “Finish” bottom to finish the first circle installation. Then repeat the installation step 1~8 until all the un-known devices are all installed.



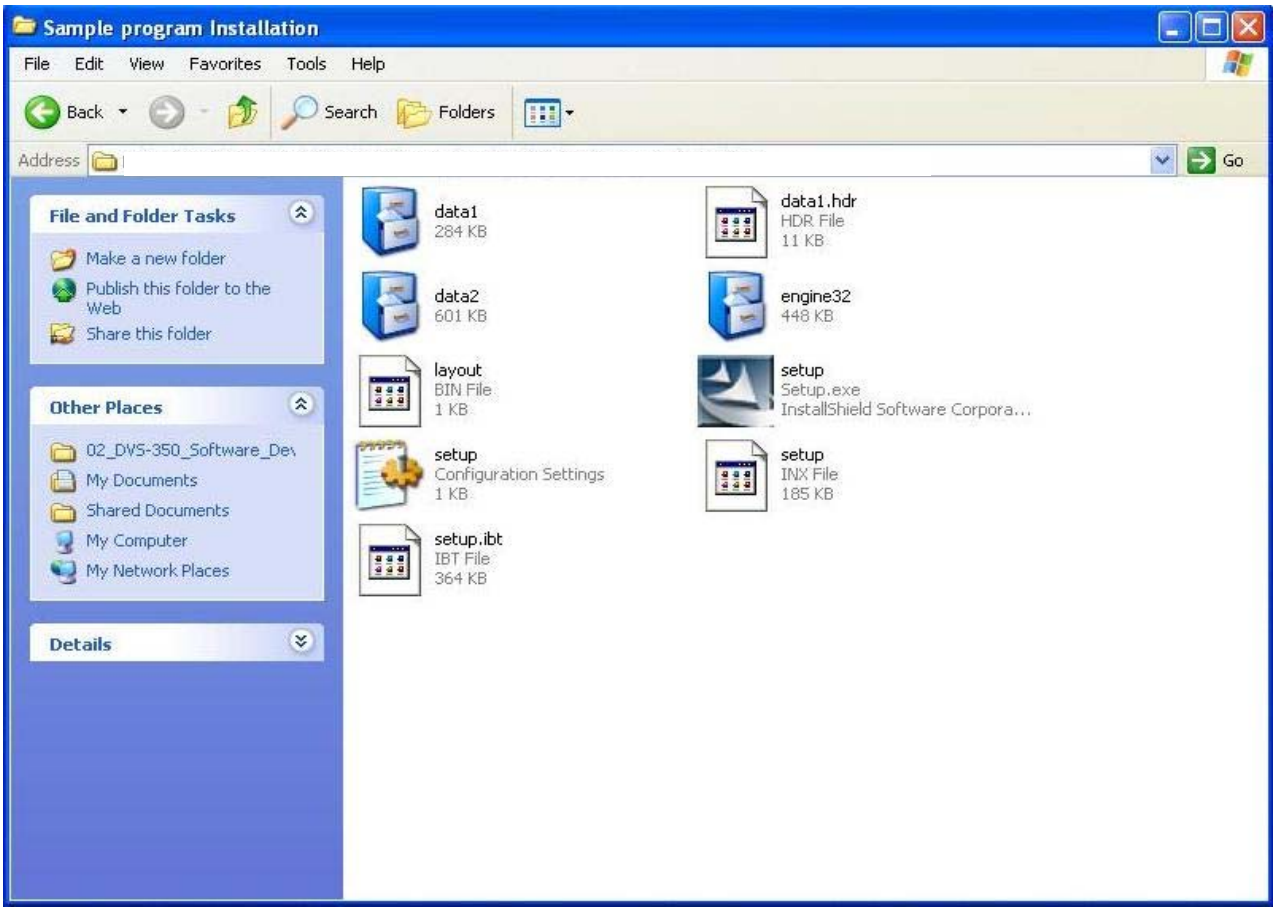
Step 9: From below window, we know there are 8 new items are installed.



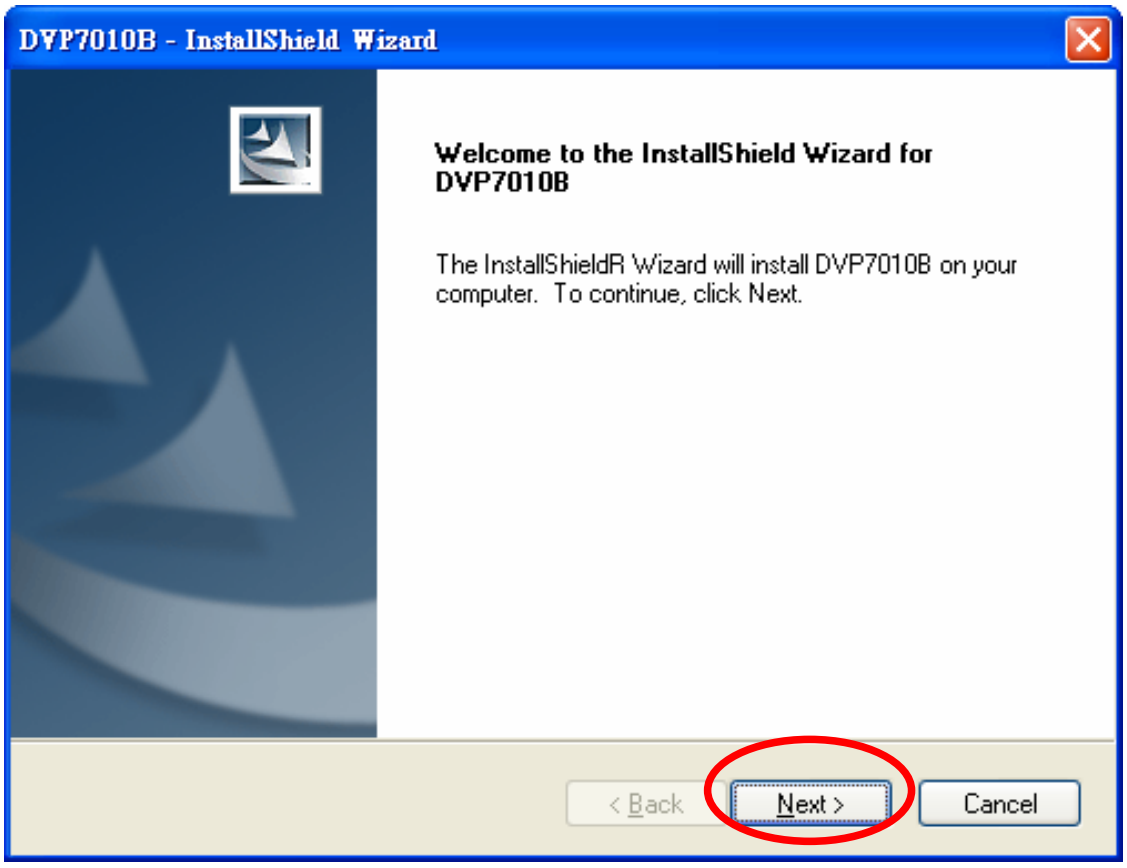
Installation of DVP-7020BE Demo Program

Step 1: Install the DVP-7020BE demo program. The executive file is in the path:

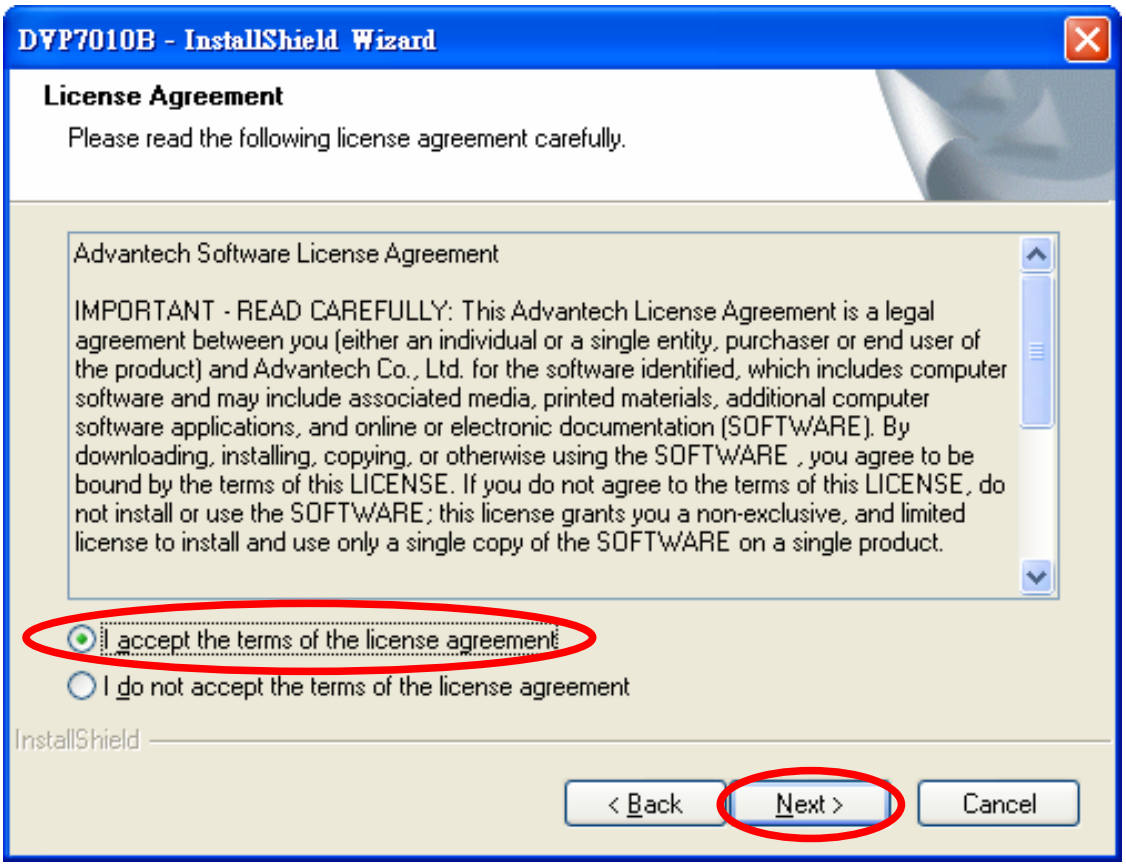
CD_ROM\DVP-7010B & 7020B SDK\DVP-7010B & 7020B Sample Installation



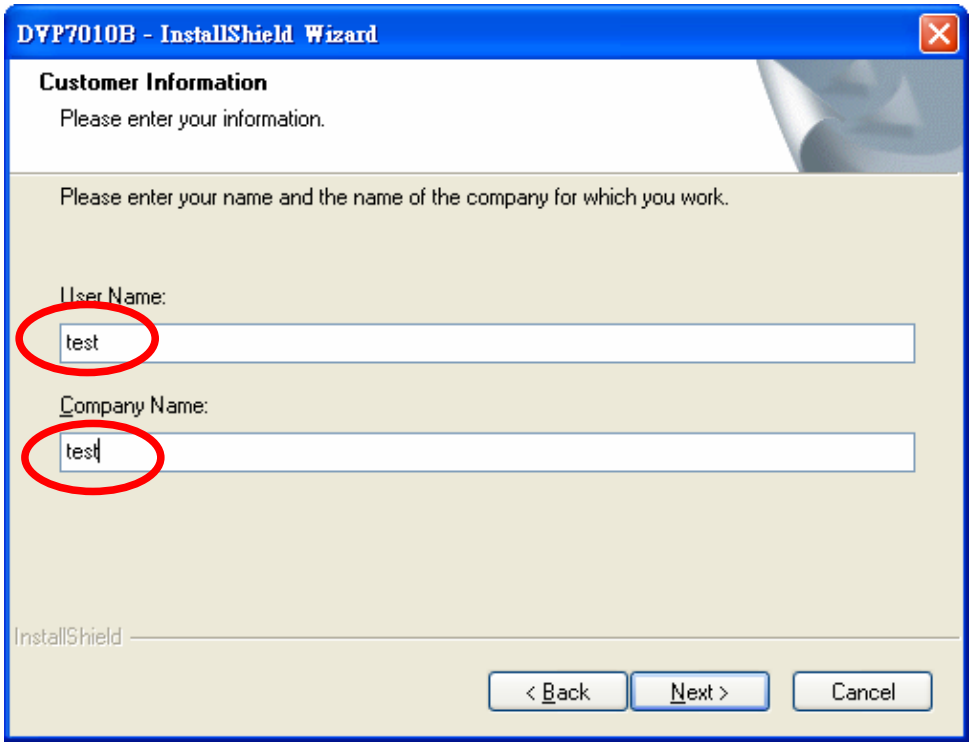
Step 2: Press the “Next” bottom to begin the installation.



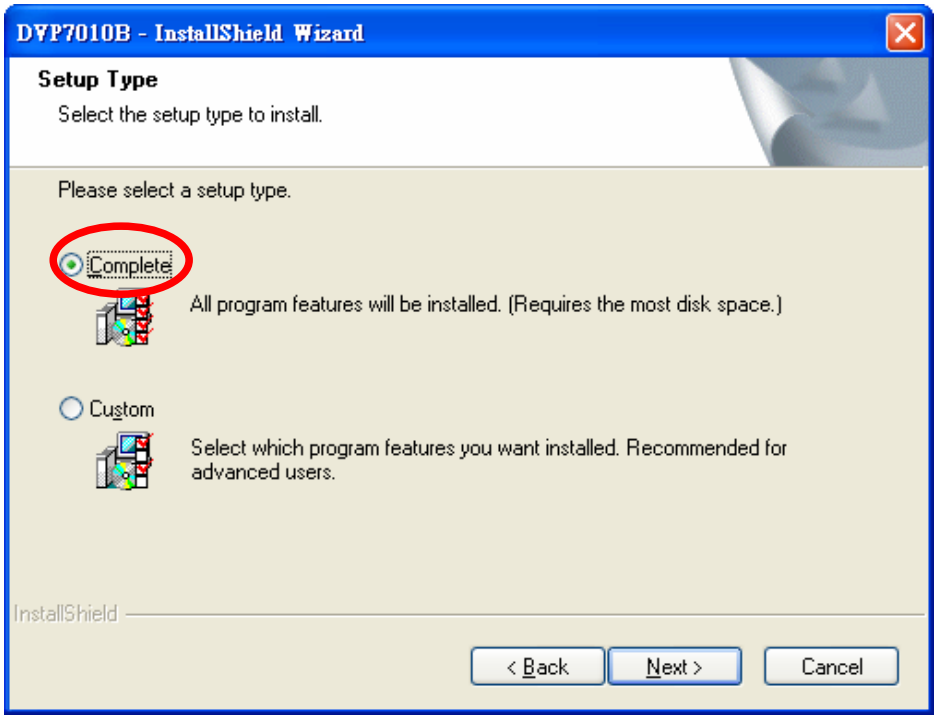
Step 3: Accept the license agreement and continue the installation.



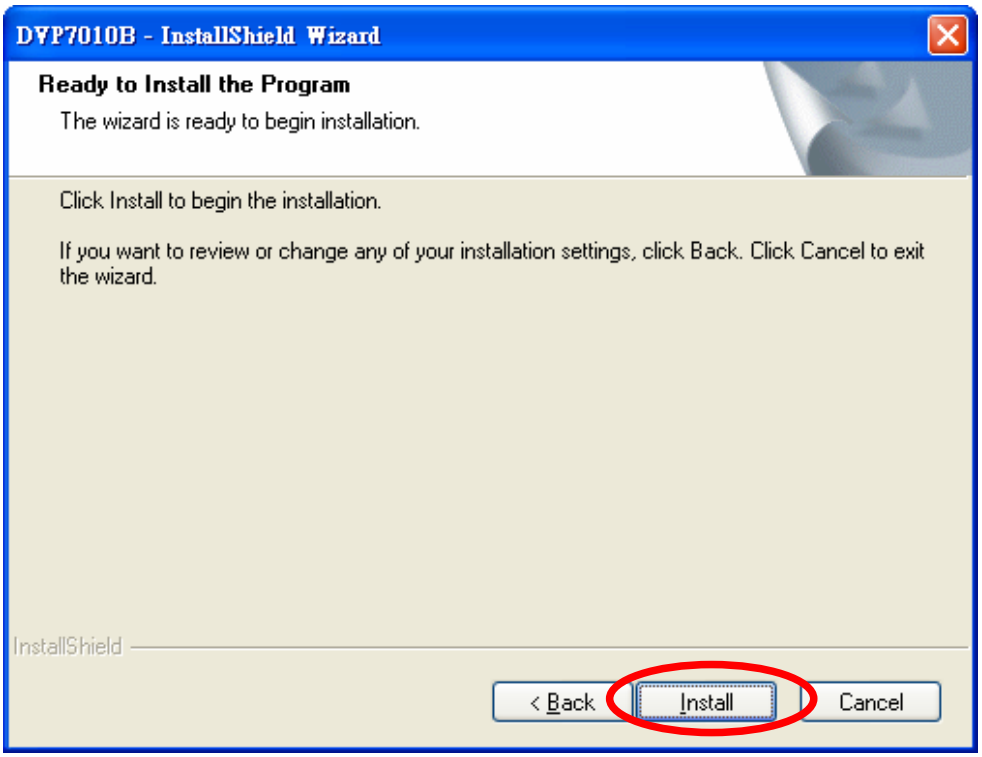
Step 4: Key in your name and company name. Then press the “Next” bottom to continue.



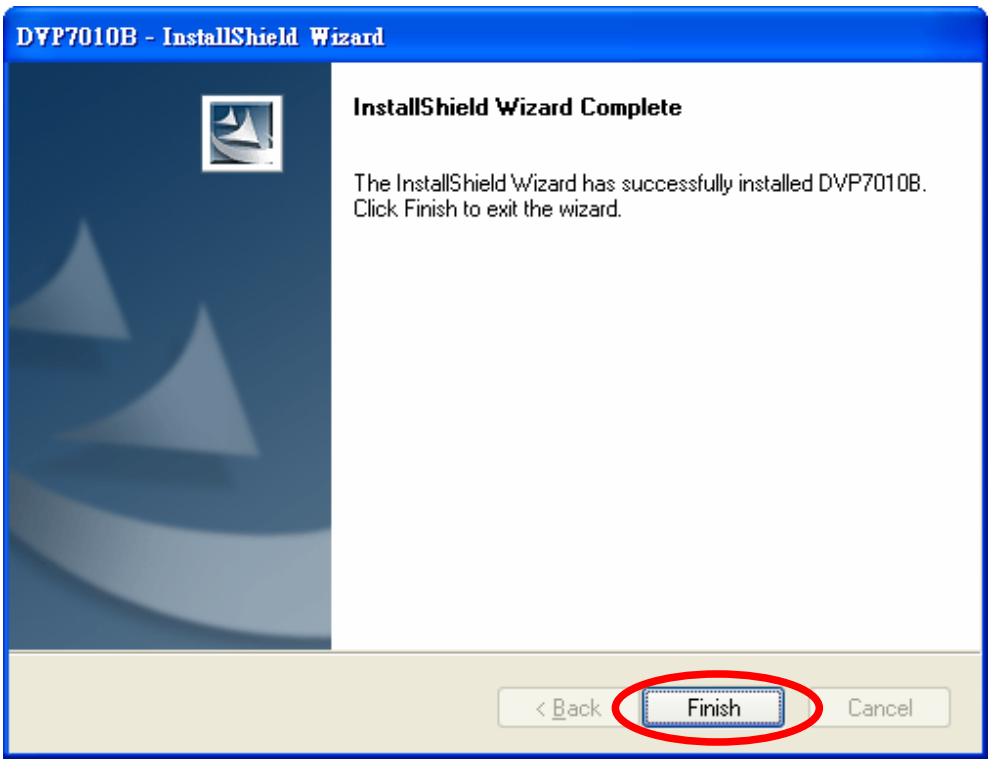
Step 5: Choose the setup type you want and next.



Step 6: Beginning the installation.



Step 7: Finished the installation of DVP-7020BE demo program.



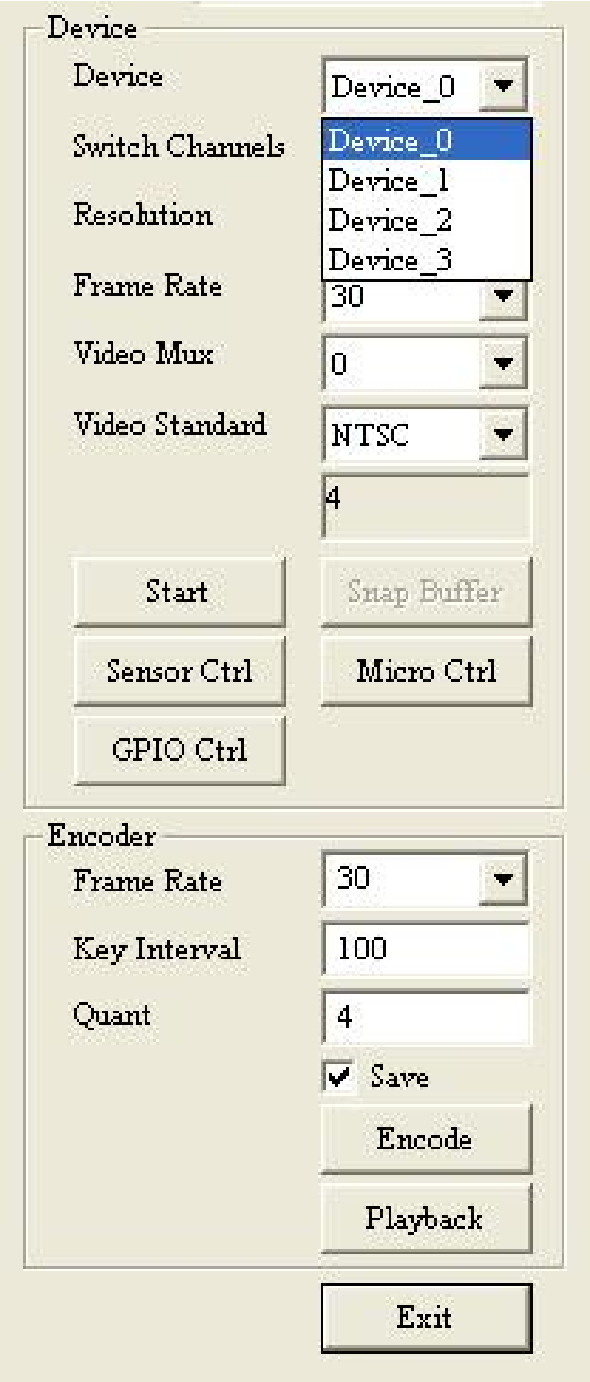
Step 8: There will be a “DVP7010B_4DEV.exe“icon on the desktop. Execute the demo program.

1.10 Demo Program Functionality

Below is the demo program window. The left side panels are the preview windows of video inputs. The right side panels are the function parameter settings.

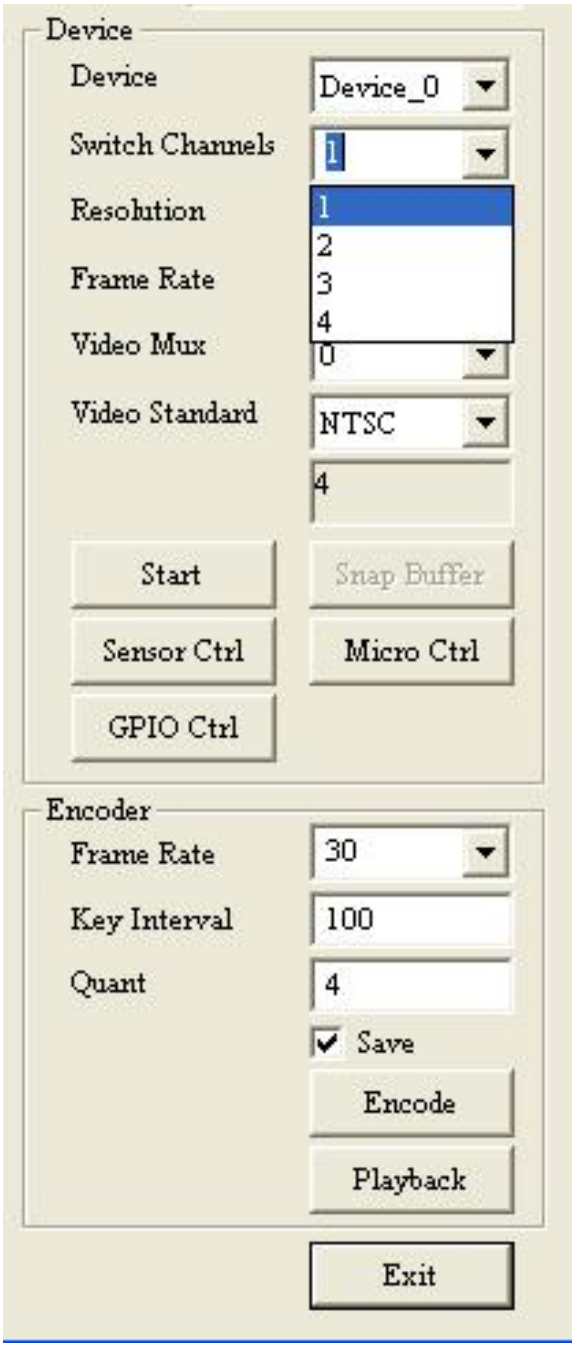
1.10.1 Device

Each device is representative of one Conexant Fusion 878A video capture chip. User can set different parameters to different 878A chip.



1.10.2 Switch Channels

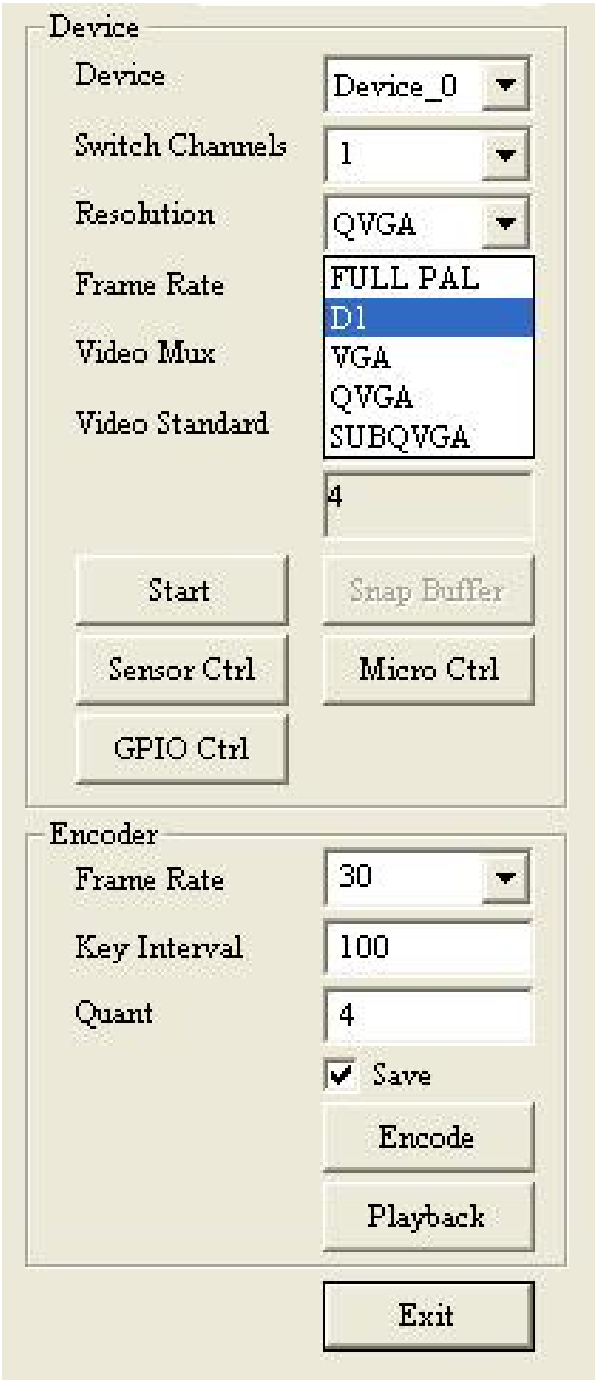
Set the “Switch Channels” to decide how many input for each 878A video chip. Each 878A chip can switch to 4 channel video inputs to share 30/25 frame per second. For more information, please refer to “Chapter 2.5.17 AdvDVP_SetVideoInput”.



1.10.3 Resolution

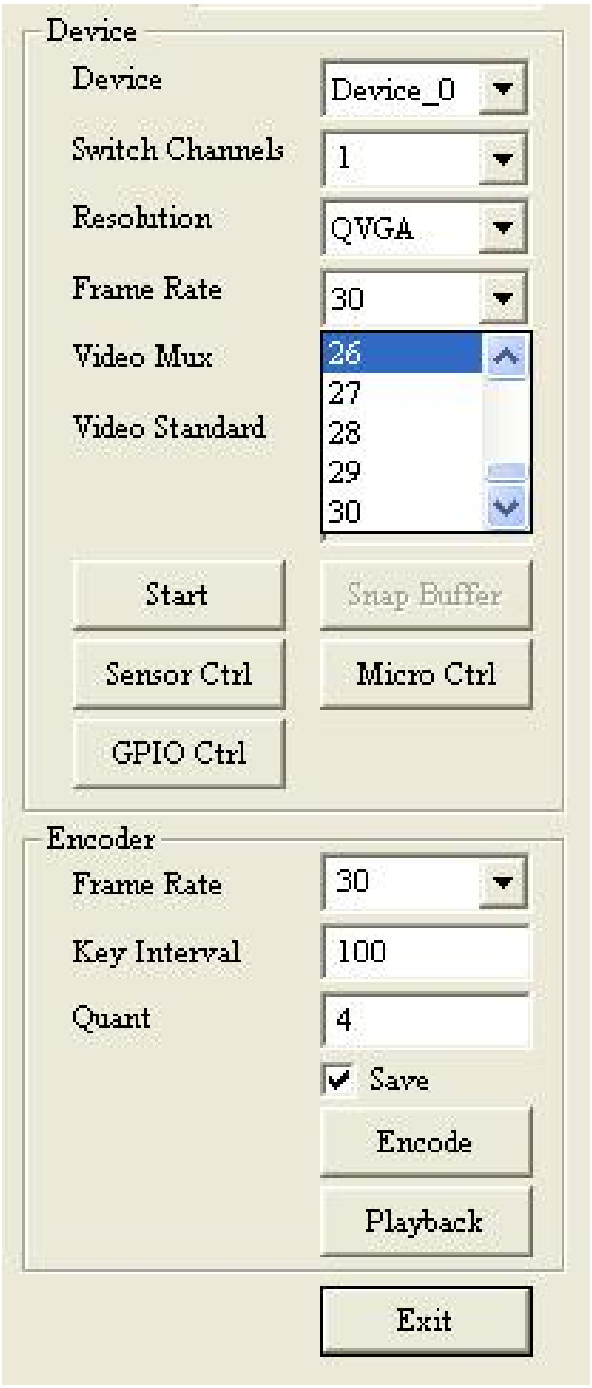
Set the video capturing resolution. Please refer to “Chapter 2.5.15 AdvDVP_SetResolution”.

Notice: *For the resolution of VGA or D1, the capture video will have the interlace effect on the video image. In other words, there will be lines in the capture image especially when the targeted image is moving. To eliminate this effect, user might need to set the resolution down to 640x240 and use specific algorisms to compensate the image interlace between the scanning even field image and odd field image. For CIF/320x240 resolution, there will be no interlace effect.*


Device
Device
Device_0
Switch Channels
1
Resolution
QVGA
Frame Rate
Video Mux
Video Standard
FULL PAL
D1
VGA
QVGA
SUBQVGA
4
Start
Snap Buffer
Sensor Ctrl
Micro Ctrl
GPIO Ctrl
Encoder
Frame Rate
30
Key Interval
100
Quant
4
Save
Encode
Playback
Exit

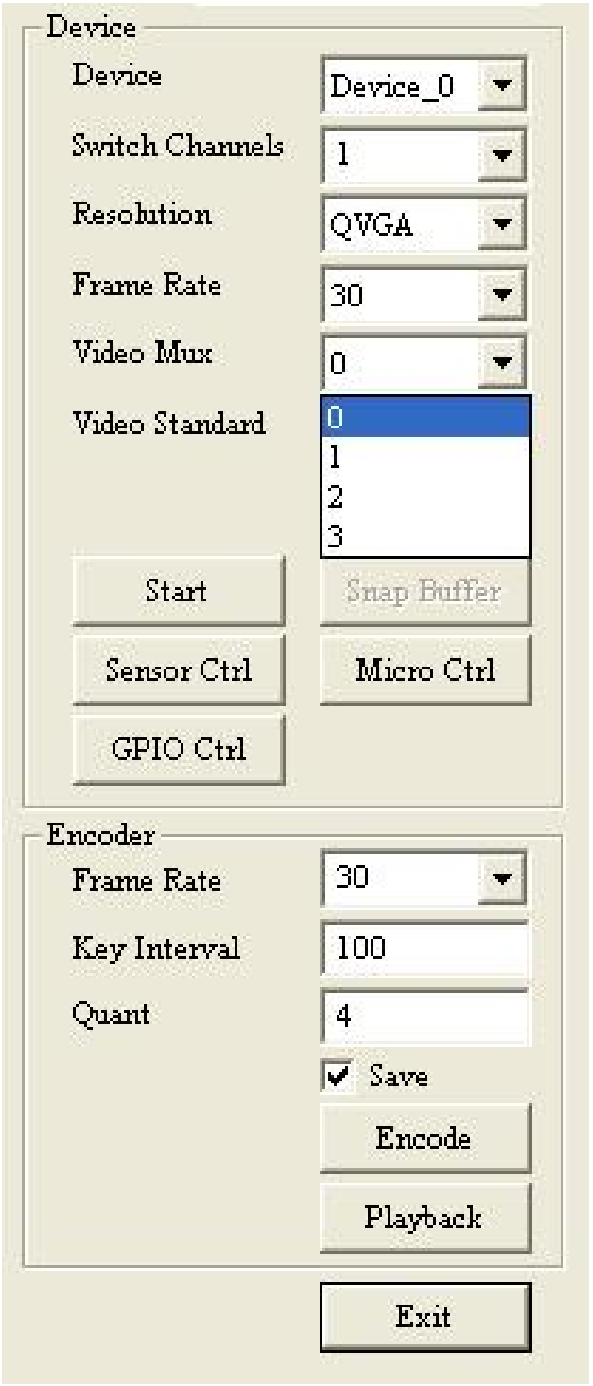
1.10.4 Frame Rate

Set the frame rate for video capturing for specific channel. Please refer to “Chapter 2.5.13 AdvDVP_SetFrameRate”



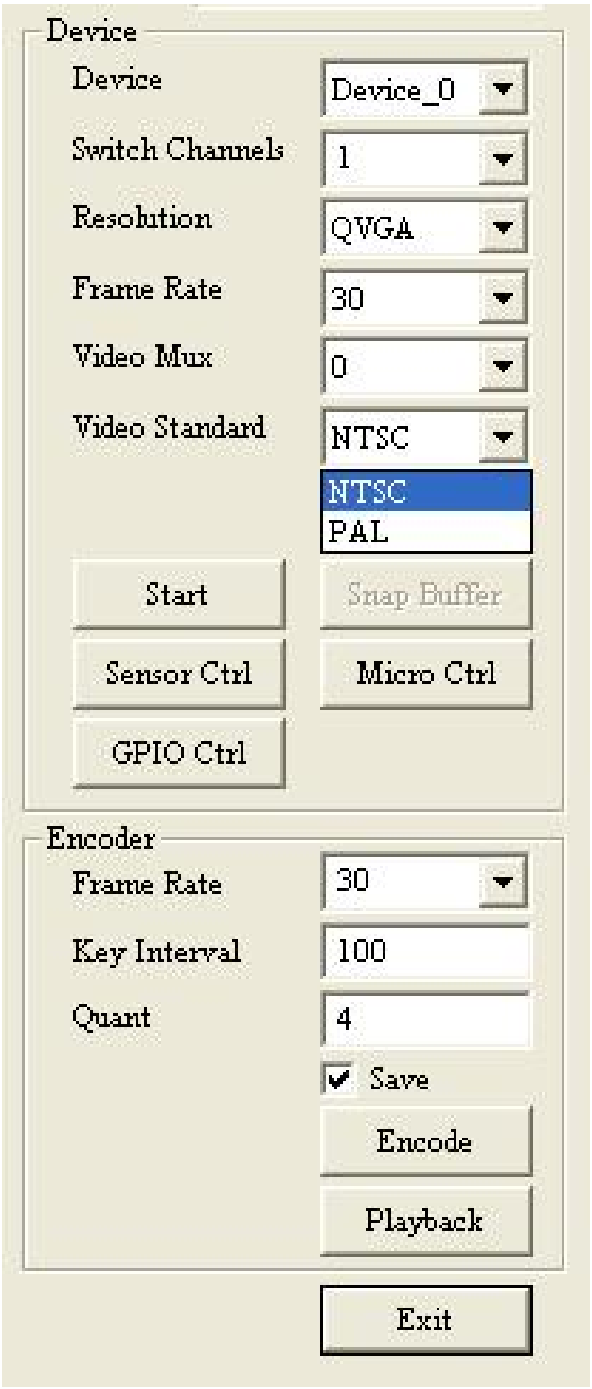
1.10.5 Video Mux

Set the “Video Mux” to specify the video input channel for setting parameter. Please refer to “Chapter 2.5.16 AdvDVP_GetVideoInput”.



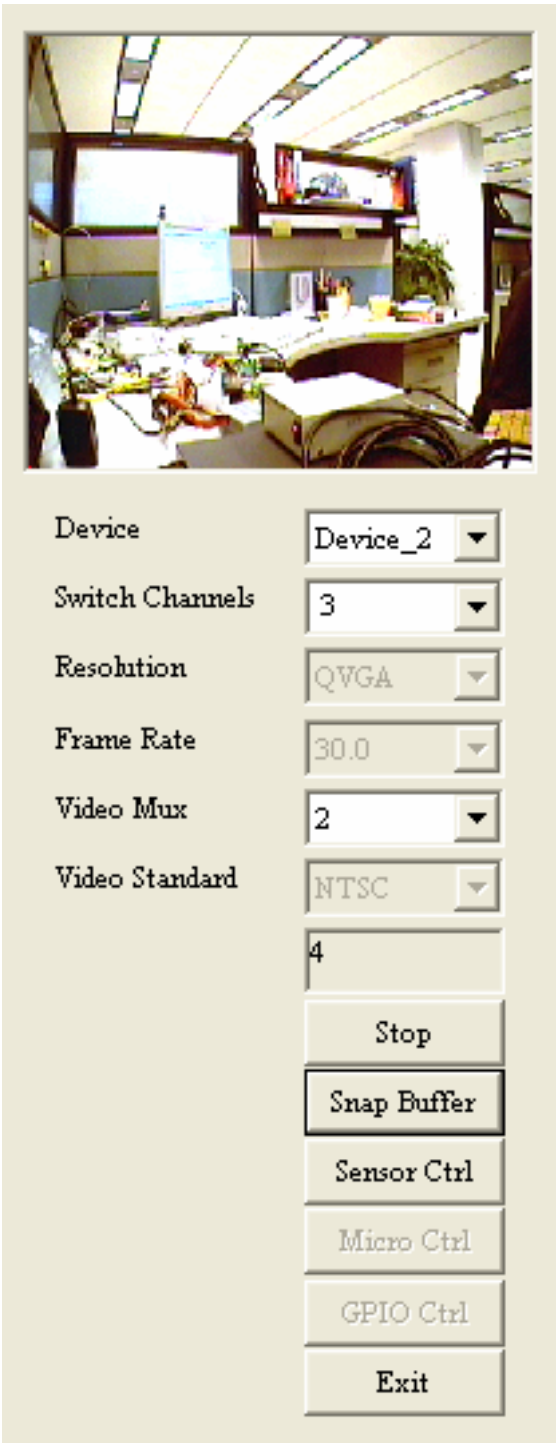
1.10.6 Video Standard

Set the video standard of your cameras. Please refer to “Chapter 2.5.10 AdvDVP_GetVideoFormat”.



1.10.7 Snap Buffer

Press the “Snap Buffer” to get the image data of specific channel video input. The snap image will be show on the up panel.



1.10.8 Sensor Control

To set the brightness, contrast, hue and saturation of specific channel. Please refer to chapter

2.5.18 AdvDVP_SetBrightness

2.5.18 AdvDVP_GetContrast

2.5.18 AdvDVP_SetContrast

2.5.18 AdvDVP_GetHue

2.5.18 AdvDVP_SetHue

2.5.18 AdvDVP_GetSaturation

2.5.18 AdvDVP_SetSaturation

1.10.9 GPIO control

To get a specified DI value or to set a specified DO value.

CHAPTER 2

Software Function Library

Chapter 2 Software Function Library

2.1 DVP7010B/7020B Functions Library
Library: DVP7010B.dll

2.1.1 Data Type Summary

Res	The method returned code

2.1.2 Method Summary

SDK Initialize and close	
AdvDVP_CreateSDKInstence	Creates SDK instance
AdvDVP_InitSDK	Initializes the SDK
AdvDVP_CloseSDK	Closes up the SDK.

Capture control	
AdvDVP_GetNoOfDevices	Gets number of video capture devices
AdvDVP_Start	Starts video capturing
AdvDVP_Stop	Stops video capturing
AdvDVP_GetCapState	Gets capture state
AdvDVP_SetNewFrameCallback	Sets a callback function to SDK
AdvDVP_GetCurFrameBuffer	Gets current frame

	buffer

Capture setting	
AdvDVP_GetVideoFormat	Gets video input format
AdvDVP_SetVideoFormat	Sets video input format
AdvDVP_GetFrameRate	Gets frame rate
AdvDVP_SetFrameRate	Sets frame rate
AdvDVP_GetResolution	Gets video resolution
AdvDVP_SetResolution	Sets video resolution
AdvDVP_GetVideoInput	Gets video input mux
AdvDVP_SetVideoInput	Sets video input mux

Sensor Control	
AdvDVP_GetBrightness	Gets brightness value
AdvDVP_SetBrightness	Sets brightness value
AdvDVP_GetContrast	Gets contrast value
AdvDVP_SetContrast	Sets contrast value
AdvDVP_GetHue	Gets hue value
AdvDVP_SetHue	Sets hue value
AdvDVP_GetSaturation	Gets saturation value
AdvDVP_SetSaturation	Sets saturation value

GPIO	
AdvDVP_GPIOGetData	Gets value of specified GPIO pin
AdvDVP_GPIOSetData	Sets value of specified

	GPIO pin

2.2 DVP7010B/7020B Encoding Functions Library
Library: DVP7010BEnc.dll
Encoder: rmp4.dll

Before using the DVP7010B/7020B encoding functions library, the "RMP4" codec must be installed to the system. After installing the sample program, the codec will be installed automatically. You can install the codec manually by using the "rmp4.inf" file. Right click on the file, and then click "Install".

2.2.1 Data Type Summary

EncRes	The method returned code
PSTREAMREADBEGIN	The stream Read Begin function pointer
PSTREAMREADPROC	The Stream Read Process function pointer
PSTREAMREADEND	The Stream Read End function pointer
STREAMREAD_STRUCT	The structure stores the Stream Read callback function pointers

2.2.2 Method Summary

SDK Initialize and close	
AdvDVP_CreateEncSDKInstence	Creates encoding SDK instance
AdvDVP_InitSDK	Initializes the SDK
AdvDVP_CloseSDK	Closes up the SDK
AdvDVP_InitEncoder	Opens and initializes video encoder
AdvDVP_CloseEncoder	Closes and release video encoder

Encode control	
AdvDVP_StartVideoEncode	Starts video encoding
AdvDVP_VideoEncode	Encodes one video frame
AdvDVP_StopVideoEncode	Stops video encoding
AdvDVP_GetState	Gets encoder state
AdvDVP_CreateAVIFile	Creates an AVI file
AdvDVP_WriteAVIFile	Writes video data to the AVI file
AdvDVP_CloseAVIFile	Closes AVI file
AdvDVP_SetStreamReadCB	Sets the stream read callback functions to SDK

Encode setting	
AdvDVP_GetVideoQuant	Gets video encoding quant
AdvDVP_SetVideoQuant	Sets video encoding quant
AdvDVP_GetVideoFrameRate	Gets video encoding frame rate
AdvDVP_SetVideoFrameRate	Sets video encoding frame rate
AdvDVP_GetVideoResolution	Gets video encoding resolution
AdvDVP_SetVideoResolution	Sets video encoding resolution
AdvDVP_GetVideoKeyInterval	Gets video encoding key interval
AdvDVP_SetVideoKeyInterval	Sets video encoding key interval

2.3 DVP7010B/7020B Player Functions Library
Library: DVP7010BPlayer.dll
Decoder: rmp4.dll

Before using the DVP7010B/7020B player functions library, the “RMP4” codec must be installed to the system. After installing the sample program, the codec will be installed automatically. You can install the codec manually by using the “rmp4.inf” file. Right click on the file, and then click “Install”.

2.3.1 Data Type Summary

PlayerRes	The method returned code

2.3.2 Method Summary

Playback SDK initialize	
AdvDVP_CreatePlayerSDKInstence	Creates Player SDK instance

Playback control	
AdvDVP_OpenFile	Opens file and initialize player
AdvDVP_CloseFile	Closes file that has been opened
AdvDVP_Play	Plays file that has been opened
AdvDVP_Pause	Pauses or continues

AdvDVP_Stop	Stops to play file
AdvDVP_Fast	Plays file with faster speed
AdvDVP_Slow	Plays file with slower speed
AdvDVP_PlayStep	Plays by single frame
AdvDVP_GetStatus	Gets playback state
AdvDVP_GetCurImage	Gets frame that is rendered
AdvDVP_RegNotifyMsg	Registers message sent to player when event occurs
AdvDVP_CheckFileEnd	Checks if file is finished playing

Playback setting	
AdvDVP_GetVideoResolution	Gets video resolution of file
AdvDVP_GetFileTime	Gets total file time
AdvDVP_GetPlayedTime	Gets current file time
AdvDVP_SetPlayPosition	Locates position of file
AdvDVP_GetFileTotalFrames	Gets total frame number of file
AdvDVP_GetPlayedFrames	Gets current frame

	number of file
AdvDVP_GetPlayRate	Gets current played rate

2.4 DVP7010B/7020B Functions Reference

Data Type
2.4.1 Res

Syntax

```
typedef enum tagRes
{
    SUCCEEDED           = 1,
    FAILED              = 0,
    SDKINITFAILED       =-1,
    PARAMERROR          =-2,
    NODEVICES       =-3,
    NOSAMPLE            =-4,
    DEVICENUMERROR      =-5,
    INPUTERROR          =-6,
} Res;
```

Description

The method returned code.

2.5 Method

2.5.1 AdvDVP_CreateSDKInstence

Syntax

int AdvDVP_CreateSDKInstence(void **pp)

Parameters

pp: A pointer to the SDK instance.

Return Value

SUCCEEDED: Function succeeded.
FAILED: Function failed.
PARAMERROR: Parameter error.

Description

This function creates SDK instance.

2.5.2 AdvDVP_InitSDK

Syntax
int AdvDVP_InitSDK()

Parameters
None

Return Value
SUCCEEDED: Function succeeded.
FAILED: Function failed.
NODEVICES: No devices found.

Description
This function initializes all video capture devices in the system. After initializing each device, the capture status would be set as “STOPPED”.

See Also
AdvDVP_GetNoOfDevices
AdvDVP_GetCapState
AdvDVP_CloseSDK

2.5.3 AdvDVP_CloseSDK

Syntax
int AdvDVP_CloseSDK(void)

Parameters
None

Return Value
SUCCEEDED: Function succeeded.
SDKINITFAILED: SDK not initialized.

Description
This function cleans all instances of capture devices and closes up the SDK.

See Also
AdvDVP_InitSDK

2.5.4 AdvDVP_GetNumberOfDevices

Syntax

int AdvDVP_GetNoOfDevices(int *pNoOfDevs)

Parameters

pNoOfDevs:	A pointer to get number of video capture devices.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.

Description

This function gets number of video capture devices in the system.

2.5.5 AdvDVP_Start

Syntax

int AdvDVP_Start(int nDevNum, int SwitchingChans, HWND Main, HWND hwndPreview)

Parameters

nDevNum: Specifies the device number(0~3).

SwitchingChans: Single video input or switching between video muxes.
0: single channel.
1: two channels (mux0, mux1).
2: three channels (mux0, mux1, mux2).
3: four channels (mux0, mux1, mux2, mux3).

Main: A main window handle.

hwndPreview: A windows handle for display area. This parameter is only valid, when the “SwitchChans” is zero. When the value of this parameter is NULL, the video will not be rendered.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.

Description

This function starts video capturing on a specified capture port.

The capture state would be set as “RUNNING” after a successful start. If the channels share frames (i.e.

SwitchingChans>0), the video input mux will be set 0.

See Also

AdvDVP_Stop

AdvDVP_GetCapState

2.5.6 AdvDVP_Stop

Syntax

int AdvDVP_Stop(int nDevNum)

Parameters

nDevNum: Specifies the device number(0~3).

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.

Description

This function stops video capturing on a specified capture port. The capture state would be set as “STOPPED” after a successful stop.

See Also

AdvDVP_Start

AdvDVP_GetCapState

2.5.7 AdvDVP_GetCapState

Syntax

int AdvDVP_GetCapState(int nDevNum)

Parameters

nDevNum: Specifies the device number(0~3).

Return Value

DEVICENUMERROR: Invalid device number.
SDKINITFAILED: SDK not initialized.

Description

This function gets capture state of a specified capture port.

```
typedef enum {
STOPPED          = 1,
RUNNING          = 2,
UNINITIALIZED    =-1,
UNKNOWNSTATE     =-2
} CapState;
```

See Also

AdvDVP_InitSDK
AdvDVP_Start
AdvDVP_Stop

2.5.8 AdvDVP_GetCurFrameBuffer

Syntax

int AdvDVP_GetCurFrameBuffer(int nDevNum, long* bufSize, BYTE* buf, int VMux)

Parameters

nDevNum:	Specifies the device number(0~3).
bufSize:	Frame buffer size.
buf:	Frame buffer.
VMux:	Video mux.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.
NOSAMPLE:	No buffer sample.

Description

This function gets current frame buffer of a specified capture port. Start capturing before the function is called.

See Also

AdvDVP_Start

2.5.9 AdvDVP_SetNewFrameCallback

Syntax

int AdvDVP_SetNewFrameCallback(int nDevNum, int callback)

Parameters

nDevNum: Specifies the device number(0~3).
callback: Callback function.
Callback fumction type:
typedef int (*CAPCALLBACK)(int nID, int nDevNum, int VMux, int bufsize, BYTE* buf);
nID: Single video input ID or the video mux ID. The value of IDs is showed as following:
#define ID_NEW_FRAME 37810
#define ID_MUX0_NEW_FRAME 37800
#define ID_MUX1_NEW_FRAME 37801
#define ID_MUX2_NEW_FRAME 37802
#define ID_MUX3_NEW_FRAME 37803
nDevNum: Specifies the device number(0~3).
VMux: Specifies the video mux number(0~3).
bufsize: An integer pointer of the frame buffer size.
buf: A BYTE pointer of the frame buffer.

Return Value

SUCCEEDED:	Function succeeded.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.

Description

This function sets a callback function to SDK. When new frame arrived, messages and frame information will be sent to callback function.

See Also

2.5.10 AdvDVP_GetVideoFormat

Syntax

int AdvDVP_GetVideoFormat(int nDevNum, AnalogVideoFormat* vFormat)

Parameters

nDevNum: Specifies the device number(0~3).
Vformat: A pointer to get video format.

```
typedef enum tagAnalogVideoFormat
{
    Video_None      = 0x00000000,
    Video_NTSC_M        = 0x00000001,
    Video_NTSC_M_J      = 0x00000002,
    Video_PAL_B         = 0x00000010,
    Video_PAL_M         = 0x00000200,
    Video_PAL_N         = 0x00000400,
    Video_SECAM_B = 0x00001000
} AnalogVideoFormat;
```

Return Value

SUCCEEDED: Function succeeded.
FAILED: Function failed.
SDKINITFAILED: SDK not initialized.
DEVICENUMERROR: Invalid device number.
PARAMERROR: Invalid parameter.

Description

This function gets video input format of a specified capture port.

See Also

AdvDVP_SetVideoFormat

2.5.11 AdvDVP_SetVideoFormat

Syntax

int AdvDVP_SetVideoFormat(int nDevNum, AnalogVideoFormat* vFormat)

Parameters

nDevNum: Specifies the port device number(0~3).

Vformat: video format:

```
typedef enum tagAnalogVideoFormat
{
    Video_None      = 0x00000000,
    Video_NTSC_M        = 0x00000001,
    Video_NTSC_M_J      = 0x00000002,
    Video_PAL_B         = 0x00000010,
    Video_PAL_M         = 0x00000200,
    Video_PAL_N         = 0x00000400,
    Video_SECAM_B = 0x00001000
} AnalogVideoFormat;
```

Return Value

SUCCEEDED: Function succeeded.
FAILED: Function failed.
SDKINITFAILED: SDK not initialized.
DEVICENUMERROR: Invalid device number.

Description

This function sets video input format a specified capture port. This function should be called before “AdvDVP_Start”.

See Also

AdvDVP_GetVideoFormat

2.5.12 AdvDVP_GetFrameRate

Syntax

int AdvDVP_GetFrameRate(int nDevNum, int *FrameRate)

Parameters

nDevNum: Specifies the device number(0~3).
FrameRate: A pointer to get video frame rate.

Return Value

SUCCEEDED: Function succeeded.
FAILED: Function failed.
SDKINITFAILED: SDK not initialized.
DEVICENUMERROR: Invalid device number.
PARAMERROR: Invalid parameter.

Description

This function gets frame rate of a specified capture port.

See Also

AdvDVP_SetFrameRate

2.5.13 AdvDVP_SetFrameRate

Syntax

int AdvDVP_SetFrameRate(int nDevNum , int SwitchingChans, int FrameRate)

Parameters

nDevNum: Specifies the device number(0~3).

SwitchingChans: Single video input or switching between video muxes(0~3).
0: single channel.
1: two channels (mux0, mux1).
2: three channels (mux0, mux1, mux2).
3: four channels (mux0, mux1, mux2, mux3).

FrameRate: A value to set frame rate. (0<FrameRate<=30, Default value is 30)

Return Value

SUCCEEDED: Function succeeded.
FAILED: Function failed.
SDKINITFAILED: SDK not initialized.
DEVICENUMERROR: Invalid device number.
PARAMERROR: Invalid parameter.

Description

This function sets frame rate of a specified capture port. This function should be called before "AdvDVP_Start". If the channels share frames (i.e.

SwitchingChans>0), the frame rate must be set 30. Otherwise, the function will return PARAMERROR.

See Also

AdvDVP_GetFrameRate

2.5.14 AdvDVP_GetResolution

Syntax

int AdvDVP_GetResolution(int nDevNum, VideoSize *Size)

Parameters

nDevNum: Specifies the device number(0~3).
Size: A pointer to get video resolution.

```
typedef enum
{
FULLPAL=0,      // (PAL: 768x576)
    SIZED1,         // (NTSC: 720x480, PAL: 720x576)
    SIZEVGA,        // (640x480)
    SIZEQVGA,       // (320x240)
    SIZESUBQVGA     // (160x120)
} VideoSize;
```

Return Value

SUCCEEDED: Function succeeded.
FAILED: Function failed.
SDKINITFAILED: SDK not initialized.
DEVICENUMERROR: Invalid device number.
PARAMERROR: Invalid parameter.

Description

This function gets video resolution of a specified capture port.

See Also

AdvDVP_SetResolution

2.5.15 AdvDVP_SetResolution

Syntax

int AdvDVP_SetResolution(int nDevNum, VideoSize Size)

Parameters

nDevNum: Specifies the device number(0~3).
Size: A value to set video resolution.

```
typedef enum
{
FULLPAL=0,      // (PAL: 768x576)
    SIZED1,         // (NTSC: 720x480, PAL: 720x576)
    SIZEVGA,        // (640x480)
    SIZEQVGA,       // (320x240)
    SIZESUBQVGA     // (160x120)
} VideoSize;
```

Return Value

SUCCEEDED: Function succeeded.
FAILED: Function failed.
SDKINITFAILED: SDK not initialized.
DEVICENUMERROR: Invalid device number.

Description

This function sets video resolution of a specified capture port. This function should be called before “AdvDVP_Start”.

See Also

AdvDVP_GetResolution

2.5.16 AdvDVP_GetVideoInput

Syntax

int AdvDVP_GetVideoInput(int nDevNum, int* pInput)

Parameters

nDevNum: Specifies the device number(0~3).
pInput: A pointer to get video input mux.

Return Value

SUCCEEDED: Function succeeded.
FAILED: Function failed.
SDKINITFAILED: SDK not initialized.
DEVICENUMERROR: Invalid device number.
PARAMERROR: Invalid parameter.

Description

This function gets video input mux of a specified capture port.
It returns “FAILED” when argument “SwitchingChans” of AdvDVP_Start was set nonzero. And, the video input mux will be set 0 automatically when argument “SwitchingChans” of AdvDVP_Start was set nonzero.

See Also

AdvDVP_Start
AdvDVP_SetVideoInput

2.5.17 AdvDVP_SetVideoInput

Syntax

int AdvDVP_SetVideoInput(int nDevNum, int nInput)

Parameters

nDevNum:	Specifies the device number(0~3).
nInput:	A value to set video input mux(0~3).

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.

Description

This function sets video input mux of a specified capture port.

It returns “FAILED” when argument “SwitchingChans” of AdvDVP_Start was set nonzero. And, the video input mux will be set 0 automatically when argument “SwitchingChans” of AdvDVP_Start was set nonzero.

See Also

AdvDVP_Start

AdvDVP_GetVideoInput

2.5.18 AdvDVP_GetBrightness

Syntax

AdvDVP_GetBrightness(int nDevNum, int nInput, long *lpValue)

Parameters

nDevNum:	Specifies the device number(0~3).
nInput:	Specifies the video input mux(-1~3). This value must be set -1 when no switching channels.
lpValue:	A long pointer to get brightness value.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.
INPUTERROR:	Invalid video input mux.

Description

This function gets brightness value of a specified capture port.

See Also

AdvDVP_SetBrightness

2.5.19 AdvDVP_SetBrightness

Syntax

int AdvDVP_SetBrightness(int nDevNum , int nInput, long lValue)

Parameters

nDevNum:	Specifies the device number(0~3).
nInput:	Specifies the video input mux(-1~3). This value must be set -1 when no switching channels.
lValue:	A value to set brightness(0~100).

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.
INPUTERROR:	Invalid video input mux.

Description

This function sets brightness value of a specified capture port.

See Also

AdvDVP_GetBrightness

2.5.20 AdvDVP_GetContrast

Syntax

int AdvDVP_GetContrast(int nDevNum, int nInput, long *lpValue)

Parameters

nDevNum:	Specifies the device number(0~3).
nInput:	Specifies the video input mux(-1~3). This value must be set -1 when no switching channels.
lpValue:	A long pointer to get contrast value.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.
INPUTERROR:	Invalid video input mux.

Description

This function gets contrast value of a specified capture port.

See Also

AdvDVP_SetContrast

2.5.21 AdvDVP_SetContrast

Syntax

int AdvDVP_SetContrast(int nDevNum, int nInput, long lValue)

Parameters

nDevNum:	Specifies the device number(0~3).
nInput:	Specifies the video input mux(-1~3). This value must be set -1 when no switching channels.
lValue:	A value to set contrast(0~100).

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device.
PARAMERROR:	Invalid parameter.
INPUTERROR:	Invalid video input mux.

Description

This function sets contrast value of a specified capture port.

See Also

AdvDVP_GetContrast

2.5.22 AdvDVP_GetHue

Syntax

int AdvDVP_GetHue(int nDevNum, int nInput, long *lpValue)

Parameters

nDevNum:	Specifies the device number(0~3).
nInput:	Specifies the video input mux(-1~3). This value must be set -1 when no switching channels.
lpValue:	A long pointer to get hue value.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.
INPUTERROR:	Invalid video input mux.

Description

This function gets hue value of a specified capture port.

See Also

AdvDVP_SetHue

2.5.23 AdvDVP_SetHue

Syntax

int AdvDVP_SetHue(int nDevNum, int nInput, long lValue)

Parameters

nDevNum:	Specifies the device number(0~3).
nInput:	Specifies the video input mux(-1~3). This value must be set -1 when no switching channels.
lValue:	A value to set hue(0~100).

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.
INPUTERROR:	Invalid video input mux.

Description

This function sets hue value of a specified capture port.

See Also

AdvDVP_GetHue

2.5.24 AdvDVP_GetSaturation

Syntax

int AdvDVP_GetSaturation(int nDevNum, int nInput, long *lpValue)

Parameters

nDevNum:	Specifies the device number(0~3).
nInput:	Specifies the video input mux(-1~3). This value must be set -1 when no switching channels.
lpValue:	A long pointer to get saturation value.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.
INPUTERROR:	Invalid video input mux.

Description

This function gets saturation value of a specified capture port.

See Also

AdvDVP_SetSaturation

2.5.25 AdvDVP_SetSaturation

Syntax

int AdvDVP_SetSaturation(int nDevNum , int nInput, long lValue)

Parameters

nDevNum:	Specifies the device number(0~3).
nInput:	Specifies the video input mux(-1~3). This value must be set -1 when no switching channels.
lValue:	A value to set saturation(0~100).

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.
INPUTERROR:	Invalid video input mux.

Description

This function sets saturation value of a specified capture port.

See Also

AdvDVP_GetSaturation

2.5.26 AdvDVP_GPIOGetData

Syntax

int AdvDVP_GPIOGetData(int nDINum, BOOL* pValue)

Parameters

nDINum:	Specifies the digital input number(0~3).
pValue:	A pointer to get the value of the specified digital input.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
PARAMERROR:	Invalid parameter.

Description

This function gets the value of the specified digital input.

See Also

AdvDVP_GPIOSetData

2.5.27 AdvDVP_GPIOSetData

Syntax

int AdvDVP_GPIOSetData(int nDONum, BOOL bValue)

Parameters

nDONum:	Specifies the digital output number(0~3).
bValue:	A value to set the value of the specified digital output.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
PARAMERROR:	Invalid parameter.

Description

This function sets the value of the specified digital output.

See Also

AdvDVP_GPIOGetData

2.6 DVP7010B/7020B Encoding Functions Reference

Data Type
2.6.1 EncRes

Syntax

```
typedef enum tagRes
{
    ENC_SUCCEEDED       = 1,
    ENC_FAILED          = 0,
    ENC_SDKINITFAILED   = -1,
    ENC_ENCINITFAILED   = -2,
    ENC_PARAMERROR      = -3,
    ENC_ENCNUMERROR     = -4,
    ENC_BUFFERFULL      = -5
} EncRes;
```

Description

The method returned code.

2.6.2 PSTREAMREADBEGIN

Syntax

void (*PSTREAMREADBEGIN)(int nEncNum)

Parameters

nEncNum: Specifies the encoder number.

Return Value

None

Description

The pointer to the Stream Read Begin callback function called when begins the video stream read process.

See Also

STREAMREAD_STRUCT

2.6.3 PSTREAMREADPROC

Syntax

void (*PSTREAMREADPROC)(int nEncNum, LPVOID pStreamBuf, long lBufSize, DWORD dwCompFlags)

Parameters

nEncNum:	Specifies the encoder number.
pStreamBuf:	A point to the data buffer stores an encoded video frame.
lBufSize:	Specifies the size of the encoded video frame.
dwCompFlags	Specifies if this encoded video frame is I-frame. The AVIIF_KEYFRAME value means the frame is I-frame.

#define AVIIF_KEYFRAME 0x00000010L

Return Value

None

Description

The pointer to the Stream Read Process callback function called after every video frame is encoded. User can use this function to get every encoded video frame.

See Also

STREAMREAD_STRUCT

2.6.4 PSTREAMREADEND

Syntax
void (*PSTREAMREADEND)(int nEncNum)

Parameters
nEncNum: Specifies the encoder number.

Return Value
None

Description
The pointer to the Stream Read End callback function called when the video stream read process is finished.

See Also
STREAMREAD_STRUCT

2.6.5 STREAMREAD_STRUCT structure

Syntax

```
typedef struct
{
    void (*PSTREAMREADBEGIN)(int nEncNum);
    void (*PSTREAMREADPROC)(int nEncNum,
    LPVOID pStreamBuf,  long lBufSize, DWORD
    dwCompFlags);
    void (*PSTREAMREADEND)(int nEncNum);
}STREAMREAD_STRUCT;
```

Parameters:

PSTREAMREADBEGIN:	The pointer to the Stream Read Begin callback function called when begins the video stream read process.
PSTREAMREADPROC:	The pointer to the Stream Read Process callback function called after every video frame is encoded.
PSTREAMREADEND:	The pointer to the Stream Read End callback function called when the video stream read process is finished.

Description

This structure stores the Stream Read callback function pointers.

See Also

PSTREAMREADBEGIN
PSTREAMREADPROC
PSTREAMREADEND
AdvDVP_SetStreamReadCB

2.7 Method

2.7.1 AdvDVP_CreateEncSDKInstence

Syntax

int AdvDVP_CreateEncSDKInstence (void **pp)

Parameters

pp: A pointer to the encoding SDK instance.

Return Value

ENC_SUCCEEDED: Function succeeded.
ENC_FAILED: Function failed.
ENC_PARAMERROR: Parameter error.

Description

This function creates the encoding SDK instance.

2.7.2 AdvDVP_InitSDK

Syntax
int AdvDVP_InitSDK(void)

Parameters
None

Return Value
ENC_SUCCEEDED: Function succeeded.

Description
This function initializes all parameters of the SDK in the system.

See Also
AdvDVP_CloseSDK

2.7.3 AdvDVP_CloseSDK

Syntax
int AdvDVP_CloseSDK(void)

Parameters
None

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_SDKINITFAILED:	SDK does not be initialized successfully.

Description
This function cleans all parameters of the SDK and closes up the SDK.

See Also
AdvDVP_InitSDK

2.7.4 AdvDVP_InitEncoder

Syntax

int AdvDVP_InitEncoder(int nEncNum, int nEncBufSize)

Parameters

nEncNum: Specifies the encoder number (0~15).

nEncBufSize: Specifies the encoding buffer size.

Return Value

ENC_SUCCEEDED: Function succeeded.

ENC_FAILED: Function failed.

ENC_SDKINITFAILED: SDK does not be initialized successfully.

ENC_ENCNUMERROR: Invalid encoder number.

Description

This function opens and initializes the specified video encoder. After initializing the encoder, the encoding state would be set as “ENC_STOPPED”.

See Also

AdvDVP_CloseEncoder

AdvDVP_GetState

2.7.5 AdvDVP_CloseEncoder

Syntax

int AdvDVP_CloseEncoder(int nEncNum)

Parameters

nEncNum: Specifies the encoder number (0~15).

Return Value

ENC_SUCCEEDED: Function succeeded.
ENC_FAILED: Function failed.
ENC_SDKINITFAILED: SDK does not be initialized successfully.
ENC_ENCNUMERROR: Invalid encoder number.
ENC_ENCINITFAILED: Encoder does not be initialized successfully.

Description

This function closes and releases the specified video encoder. After successfully calling this function, the encoding state would be set as “ENC_UNINITIALIZED”.

See Also

AdvDVP_InitEncoder
AdvDVP_GetState

2.7.6 AdvDVP_StartVideoEncode

Syntax

int AdvDVP_StartVideoEncode(int nEncNum)

Parameters

nEncNum: Specifies the encoder number (0~15).

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK does not be initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder does not be initialized successfully.

Description

This function notifies the specified video encoder to prepare to encode the video. The encode state would be set as “ENC_RUNNING” after a successful beginning.

See Also

AdvDVP_VideoEncode
AdvDVP_StopVideoEncode
AdvDVP_GetState

2.7.7 AdvDVP_VideoEncode

Syntax

int AdvDVP_VideoEncode(int nEncNum, LPVOID lpInBuf,
int InBufSize, BOOL bKeyFrame)

Parameters

nEncNum:	Specifies the encoder number (0~15).
lpbiIn:	Specifies the input buffer stores the source video frame.
InBufSize:	Specifies the size of the input buffer.
bKeyFrame:	Specifies if the video frame is encoded as a I-frame.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK does not be initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder does not be initialized successfully.
ENC_PARAMERROR:	Parameter error.
ENC_BUFFERFULL:	Encoding buffer is full, the video frame can not be written to the

buffer.

Description

This function writes the video frame to the encoding buffer to encode it by the specified encoder.

See Also

AdvDVP_StartVideoEncode
AdvDVP_StopVideoEncode

2.7.8 AdvDVP_StopVideoEncode

Syntax

int AdvDVP_StopVideoEncode(int nEncNum)

Parameters

nEncNum: Specifies the encoder number (0~15).

Return Value

ENC_SUCCEEDED: Function succeeded.
ENC_FAILED: Function failed.
ENC_SDKINITFAILED: SDK does not be initialized successfully.
ENC_ENCNUMERROR: Invalid encoder number.
ENC_ENCINITFAILED: Encoder does not be initialized successfully.

Description

This function notifies the specified video encoder to stop encoding and releases all relational resources. The encoding state would be set as "ENC_STOPPED" after a successful stop.

See Also

AdvDVP_StartVideoEncode
AdvDVP_VideoEncode
AdvDVP_GetState

2.7.9 AdvDVP_GetState

Syntax

int AdvDVP_GetState(int nEncNum)

Parameters

nEncNum: Specifies the encoder number (0~15).

Return Value

ENC_ENCNUMERROR: Invalid encoder number.

Description

This function gets encoding state of a specified video encoder.

```
typedef enum
{
ENC_STOPPED          = 1,
ENC_RUNNING          = 2,
ENC_UNINITIALIZED = -1,
} EncoderState;
```

See Also

AdvDVP_InitEncoder
AdvDVP_CloseEncoder
AdvDVP_StartVideoEncode
AdvDVP_StopVideoEncode

2.7.10 AdvDVP_SetStreamReadCB

Syntax

void
AdvDVP_SetStreamReadCB(STREAMREAD_STRUCT
*pStreamRead)

Parameters

pStreamRead:	A pointer to STREAMREAD_STRUCT structure recording the pointers to the StreamRead callback functions.

Return Value

None

Description

This function registers the Stream Read callback functions to the SDK.

See Also

STREAMREAD_STRUCT structure

2.7.11 AdvDVP_GetVideoQuant

Syntax

int AdvDVP_GetVideoQuant(int nEncNum, int *nQuant)

Parameters

nEncNum:	Specifies the encoder number (0~15).
nQuant:	A pointer to get the video quant. The range is 1~31. The default video quality is 4.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK does not be initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder does not be initialized successfully.

Description

This function gets video quant of the specified video encoder. The lower video quant can get the compressed video with higher quality and bit rate, vice versa.

See Also

AdvDVP_SetVideoQuant

2.7.12 AdvDVP_SetVideoQuant

Syntax

int AdvDVP_SetVideoQuant(int nEncNum, int nQuant)

Parameters

nEncNum:	Specifies the encoder number (0~15).
nQuant:	A value to set the video quant. The range is 1~31. The default video quality is 4.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK does not be initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder does not be initialized successfully.

Description

This function sets video quant of the specified video encoder. The lower video quant can get the compressed video with higher quality and bit rate, vice versa.

See Also

AdvDVP_GetVideoQuant

2.7.13 AdvDVP_GetVideoFrameRate

Syntax

int AdvDVP_GetVideoFrameRate(int nEncNum, int *nFrameRate)

Parameters

nEncNum:	Specifies the encoder number (0~15).
nFrameRate:	A pointer to get the video frame rate.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK does not be initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder does not be initialized successfully.

Description

This function gets video frame rate of the specified video encoder.

See Also

AdvDVP_SetVideoFrameRate

2.7.14 AdvDVP_SetVideoFrameRate

Syntax

int AdvDVP_SetVideoFrameRate(int nEncNum, int nFrameRate)

Parameters

nEncNum:	Specifies the encoder number (0~15).
nFrameRate:	A value to set the video frame rate. The range is 1~30. The default video frame rate is 30.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK does not be initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder does not be initialized successfully.

Description

This function sets video frame rate of the specified video encoder.

See Also

AdvDVP_GetVideoFrameRate

2.7.15 AdvDVP_GetVideoResolution

Syntax

int AdvDVP_GetVideoResolution(int nEncNum, int *nWidth, int *nHeight)

Parameters

nEncNum: Specifies the encoder number (0~15).

nWidth: A pointer to get the width of the video.

nHeight: A pointer to get the height of the video.

Return Value

ENC_SUCCEEDED: Function succeeded.

ENC_SDKINITFAILED: SDK does not be initialized successfully.

ENC_ENCNUMERROR: Invalid encoder number.

ENC_ENCINITFAILED: Encoder does not be initialized successfully.

Description

This function gets video resolution of the specified video encoder.

See Also

AdvDVP_SetVideoResolution

2.7.16 AdvDVP_SetVideoResolution

Syntax

int AdvDVP_SetVideoResolution(int nEncNum, int nWidth, int nHeight)

Parameters

nEncNum:	Specifies the encoder number (0~15).
nWidth:	A value to set the width of the video. The default width is 320.
nHeight	A value to set the height of the video. The default height is 240.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK does not be initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder does not be initialized successfully.

Description

This function sets video resolution of the specified video encoder.

See Also

AdvDVP_GetVideoResolution

2.7.17 AdvDVP_GetVideoKeyInterval

Syntax

int AdvDVP_GetVideoKeyInterval(int nEncNum, int *nKeyInterval)

Parameters

nEncNum:	Specifies the encoder number (0~15).
nKeyInterval:	A pointer to get the interval of the video key frame.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK does not be initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder does not be initialized successfully.

Description

This function gets the interval of the video key frame of the specified video encoder.

See Also

AdvDVP_SetVideoKeyInterval

2.7.18 AdvDVP_SetVideoKeyInterval

Syntax

int AdvDVP_SetVideoKeyInterval(int nEncNum, int nKeyInterval)

Parameters

nEncNum:	Specifies the encoder number (0~15).
nKeyInterval:	A value to set the interval of the video key frame. The range is 1~99. The default video frame rate is 60.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK does not be initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder does not be initialized successfully.

Description

This function sets the interval of the video key frame of the specified video encoder.

See Also

AdvDVP_GetVideoKeyInterval

2.7.19 AdvDVP_CreateAVIFile

Syntax

HANDLE AdvDVP_CreateAVIFile(LPCSTR lpcsFileName, int nWidth, int nHeight, int nFrameRate)

Parameters

lpcsFileName:	Specifies the file name of the AVI file.
nWidth:	
nHeight	
nFrameRate	Specifies the frame rate of the video.

Return Value

If the function succeeds, the file handle is returned. Otherwise, the function returns NULL.

Description

This function creates the AVI file to save the encoded video stream.

See Also

AdvDVP_WriteAVIFile
AdvDVP_CloseAVIFile

2.7.20 AdvDVP_WriteAVIFile

Syntax

int AdvDVP_WriteAVIFile(HANDLE hAVIFile, LPVOID lpStreamBuf, long lBufSize, DWORD dwCompFlags)

Parameters

hAVIFile:	Specifies the AVI file handle.
lpStreamBuf:	A pointer to the video stream data buffer written into the file.
lBufSize:	Specifies the size of the video stream data buffer.
dwCompFlags:	Flag associated with this data. The AVIIF_KEYFRAME flag is defined to indicate this data does not rely on preceding data in the file.

#define AVIIF_KEYFRAME 0x00000010L

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK does not be initialized successfully.

Description

This function writes the video stream data into the specified AVI file.

See Also

AdvDVP_CreateAVIFile

AdvDVP_CloseAVIFile

2.7.21 AdvDVP_CloseAVIFile

Syntax

int AdvDVP_CloseAVIFile(HANDLE hAVIFile)

Parameters

hAVIFile:	Specifies the AVI file handle.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK does not be initialized successfully.

Description

This function closes the specified AVI file.

See Also

AdvDVP_CreateAVIFile
AdvDVP_WriteAVIFile

2.8 DVP7010B/7020B Player Functions Reference

Data Type
2.8.1 PlayerRes

Syntax

```
typedef enum tagRes
{
    PLAYER_SUCCEEDED            = 1,
    PLAYER_FAILED               = 0,
    PLAYER_SDKINITFAILED        = -1,
    PLAYER_PARAMERROR           = -2,
} PlayerRes;
```

Description

The method returned code.

2.9 Method

2.9.1 AdvDVP_CreatePlayerSDKInstence

Syntax

int AdvDVP_CreatePlayerSDKInstence(void **pp)

Parameters

pp:	A pointer to the player SDK instance.

Return Value

PLAYER_SUCCEEDED:	Function succeeded.
PLAYER_FAILED:	Function failed.
PLAYER_PARAMERROR:	Parameter error.

Description

This function creates playback SDK instance.

2.9.2 AdvDVP_OpenFile

Syntax

int AdvDVP_OpenFile(LPCSTR lpcsFileName)

Parameters

lpcsFileName:	Specifies the file name of the source video file.

Return Value

PLAYER_SUCCEEDED:	Function succeeded.
PLAYER_FAILED:	Function failed.

Description

This function opens the source video file and initializes the video player. The playback status would be set as “PLAYER_STOPPED” after successfully calling this function.

See Also

AdvDVP_CloseFile
AdvDVP_GetStatus

2.9.3 AdvDVP_CloseFile

Syntax
int AdvDVP_CloseFile()

Parameters
None.

Return Value
PLAYER_SUCCEEDED: Function succeeded.
PLAYER_FAILED: Function failed.

Description
This function closes the source video file and free resources allocated for video player. The playback status would be set as “PLAYER_NOTOPENED” after successfully calling this function.

See Also
AdvDVP_OpenFile
AdvDVP_GetStatus

2.9.4 AdvDVP_Play

Syntax

int AdvDVP_Play(HWND hwndApp, BOOL bAutoResizeWnd)

Parameters

hwndApp:	A windows handle for display area.
bAutoResizeWnd:	Specifies if the display area is resized automatically according to the video resolution.

Return Value

PLAYER_SUCCEEDED:	Function succeeded.
PLAYER_FAILED:	Function failed.

Description

This function plays the file that has been opened. The playback status would be set as “PLAYER_PLAYING” after successfully calling this function.

See Also

AdvDVP_Pause
AdvDVP_Stop
AdvDVP_GetStatus

2.9.5 AdvDVP_Pause

Syntax
int AdvDVP_Pause()

Parameters
None.

Return Value
PLAYER_SUCCEEDED: Function succeeded.
PLAYER_FAILED: Function failed.

Description
This function pauses or continues the file that has been opened. The playback status would be set as “PLAYER_PAUSED” after successfully calling this function.

See Also
AdvDVP_Play
AdvDVP_Stop
AdvDVP_GetStatus

2.9.6 AdvDVP_Stop

Syntax
int AdvDVP_Stop()

Parameters
None.

Return Value
PLAYER_SUCCEEDED: Function succeeded.
PLAYER_FAILED: Function failed.

Description
This function stops the file that is playing. The playback status would be set as “PLAYER_STOPPED” after successfully calling this function.

See Also
AdvDVP_Play
AdvDVP_Pause
AdvDVP_GetStatus

2.9.7 AdvDVP_Fast

Syntax
int AdvDVP_Fast()

Parameters
None.

Return Value
PLAYER_SUCCEEDED: Function succeeded.
PLAYER_FAILED: Function failed.

Description
This function improves the current play speed by one time, 4 times at most. The playback status would be set as “PLAYER_PLAYING” after successfully calling this function.

See Also
AdvDVP_Pause
AdvDVP_Stop
AdvDVP_Slow
AdvDVP_GetStatus

2.9.8 AdvDVP_Slow

Syntax
int AdvDVP_Slow()

Parameters
None.

Return Value
PLAYER_SUCCEEDED: Function succeeded.
PLAYER_FAILED: Function failed.

Description
This function slows the current play speed by one time, 4 times at most. The playback status would be set as “PLAYER_PLAYING” after successfully calling this function.

See Also
AdvDVP_Pause
AdvDVP_Stop
AdvDVP_Fast
AdvDVP_GetStatus

2.9.9 AdvDVP_PlayStep

Syntax
int AdvDVP_PlayStep()

Parameters
None.

Return Value
PLAYER_SUCCEEDED: Function succeeded.
PLAYER_FAILED: Function failed.

Description
This function makes the video to step forward one frame. The playback status would be set as “PLAYER_PAUSED” after successfully calling this function.

See Also
AdvDVP_Pause
AdvDVP_Stop
AdvDVP_GetStatus

2.9.10 AdvDVP_GetStatus

Syntax
int AdvDVP_GetStatus ()

Parameters
None

Return Value
PLAYER_SUCCEEDED: Function succeeded.
PLAYER_FAILED: Function failed.

Description
This function gets playback status.

typedef enum tagPlayerStatus{
PLAYER_NOTOPENED = 0,
PLAYER_OPENED = 1,
PLAYER_PLAYING = 2,
PLAYER_STOPPED = 3,
PLAYER_PAUSED = 4
} PlayerStatus;

See Also
AdvDVP_OpenFile
AdvDVP_CloseFile
AdvDVP_Play
AdvDVP_Pause
AdvDVP_Stop
AdvDVP_Fast
AdvDVP_Slow
AdvDVP_PlayStep

2.9.11 AdvDVP_GetCurImage

Syntax

int AdvDVP_GetCurImage(LPBYTE *lpImage,
long *pBufSize)

Parameters

lpImage:	A pointer to a image buffer.
pBufSize:	A long pointer to receive the returned image buffer size.

Return Value

PLAYER_SUCCEEDED:	Function succeeded.
PLAYER_FAILED:	Function failed.

Description

This function gets current played image.

See Also

2.9.12 AdvDVP_RegNotifyMsg

Syntax

int AdvDVP_RegNotifyMsg(HWND hWnd, UINT nMsg)

Parameters

hWnd:	Specifies the handle of the window receiving this message.
nMsg:	Specifies the user-define message. When this message is received, it means some event of the playback occur such as the file playing is end.

Return Value

PLAYER_SUCCEEDED:	Function succeeded.
PLAYER_FAILED:	Function failed.

Description

This function registers a user-define message. When an event of the playback occurs, this message will be sent to the specified window.
This function must be called after “AdvDVP_OpenFile” function.

See Also

AdvDVP_CheckFileEnd

2.9.13 AdvDVP_CheckFileEnd

Syntax
BOOL AdvDVP_CheckFileEnd ()

Parameters
None

Return Value
If the event that the file playing end is detected, this function returns TRUE. Otherwise, the function returns FALSE.

Description
This function checks if the file playing is end.

See Also
AdvDVP_RegNotifyMsg

2.9.14 AdvDVP_GetVideoResolution

Syntax

int AdvDVP_GetVideoResolution(int *nWidth, int *nHeight)

Parameters

nWidth:	An integer pointer to get the width of the video.
nHeight:	An integer pointer to get the height of the video.

Return Value

PLAYER_SUCCEEDED:	Function succeeded.
PLAYER_FAILED:	Function failed.

Description

This function gets width and the height of the video.

See Also

2.9.15 AdvDVP_GetPlayRate

Syntax
double AdvDVP_GetPlayRate()

Parameters
None

Return Value
If the function succeeded, the playback ratio is returned. Otherwise, the function returns 0.

Description
This function retrieves the playback rate.

See Also
AdvDVP_Play
AdvDVP_Fast
AdvDVP_Slow

2.9.16 AdvDVP_GetFileTime

Syntax
double AdvDVP_GetFileTime()

Parameters
None

Return Value
If the function succeeded, the total file time is returned. Otherwise, the function returns 0.

Description
This function retrieves total file time in seconds.

See Also
AdvDVP_GetPlayedTime
AdvDVP_SetPlayPosition

2.9.17 AdvDVP_GetPlayedTime

Syntax
double AdvDVP_GetPlayedTime()

Parameters
None

Return Value
If the function succeeded, the current file time is returned. Otherwise, the function returns 0.

Description
This function retrieves current file time in seconds.

See Also
AdvDVP_GetFileTime
AdvDVP_SetPlayPosition

2.9.18 AdvDVP_SetPlayPosition

Syntax

int AdvDVP_SetPlayPosition (double dTime)

Parameters

dTime: Specifies the file time in seconds.

Return Value

PLAYER_SUCCEEDED: Function succeeded.

PLAYER_FAILED: Function failed.

Description

This function seeks the file to the specified file time.

See Also

AdvDVP_GetFileTime

AdvDVP_GetPlayedTime

2.9.19 AdvDVP_GetFileTotalFrames

Syntax
LONGLONG AdvDVP_GetFileTotalFrames()

Parameters
None

Return Value
If the function succeeded, the total number of the frame of the file is returned. Otherwise, the function returns 0.

Description
This function retrieves total number of the frame of the file.

See Also
AdvDVP_GetPlayedFrames

2.9.20 AdvDVP_GetPlayedFrames

Syntax
LONGLONG AdvDVP_GetPlayedFrames()

Parameters
None

Return Value
If the function succeeded, the current frame number of the file is returned. Otherwise, the function returns 0.

Description
This function retrieves current frame number of the file.

See Also
AdvDVP_GetFileTotalFrames