IVC-4200

User Manual

Version 1.2

<Product Overview>

4-Channel MPEG-4Encoding Card

January 9, 2004

Copyright Notice

Trademarks

Support

For any questions regarding the content of this manual or the product, please contact us at: support@iei.com.tw

Table of Contents

Chapter 1 Introduction

Thank you for choosing IVC-4200 as your solution for high quality video applications. IVC-4200 can achieve real-time 4-channel MPEG-4 encoding and is the best tool to help you making high quality videos. IVC-4200 has 4 MPEG-4 encoding chips and is capable to record the video with quality that is similar to DVD. IVC-4200 can be used to develop multimedia applications such as video editing, streaming networks, medical video networks, corporate distance learning, and video on demand (VOD), etc…

1.1 Features

- Video Input:
  4-channel BNC Composite Video, NTSC/PAL/SECAM auto sensing
- PCI Interface:
  PCI V2.1 Compliance, Plug & Play support
- Video Compression:
  MPEG-4 Advanced Simple Profile @L3
  MPEG-2 MP@ML
  MPEG-1
- Video Resolution:
  720 x 480, 352 x 240, 176 x 120 (NTSC)
  720 x 576, 352 x 288, 176 x 144 (PAL)
- Frame Rate:
  30 FPS (NTSC) for each channel
  25 FPS (PAL) for each channel
- Video Quality:
  DVD quality full D1 video at 3 Mbps ～ 16 Mbps
  High quality full D1 video at 1 Mbps
  High quality CIF video at 384 Kbps
  High quality QCIF video at 96 Kbps
- Device Driver:
  Provides drivers for Windows 2000 and Windows XP Systems
- SDK:
  Provides SDK with demo program for software application development

1.2 Package Contents

- IVC-4200 video capture card x 1
- CD-ROM x 1
- User manual x 1

1.3 System Requirements

- IBM or IBM compatible computer
- Pentium-4 1.8 GHz CPU or better processor for reasonable decoding and display quality
- Minimum 128MB memory
- SVGA display adapter supporting DirectDraw
- At least one unoccupied PCI slot and IRQ
- Window Screen setting at 16 bits color or higher
- UDMA Bus Mastering IDE or SCSI HDD for video capture
- OS: Windows 2000 and Windows XP
- Microsoft DirectX 8.1 (or above version) installed

 **Note:**

- If the system has installed other video capture card before, please make sure the previous driver is removed from the system.
- Users need Microsoft Direct X 8.1 (or above) in their computer. Window XP users do not need to install DirectX 8.1 since Windows XP includes DirectX 8.1.
- About DirectX 8.1:
  Direct X 8.1 is a Microsoft shareware. Direct X 8.1 will help improve multimedia experiences on most PCs. Since DirectX 8.1 is a system component, it is impossible to remove DirectX 8.1 without uninstalled your OS.

Chapter 2 Hardware Installation

2.1 Hardware Installation

Please make sure the power of the computer is off when you install the IVC-4200 card:

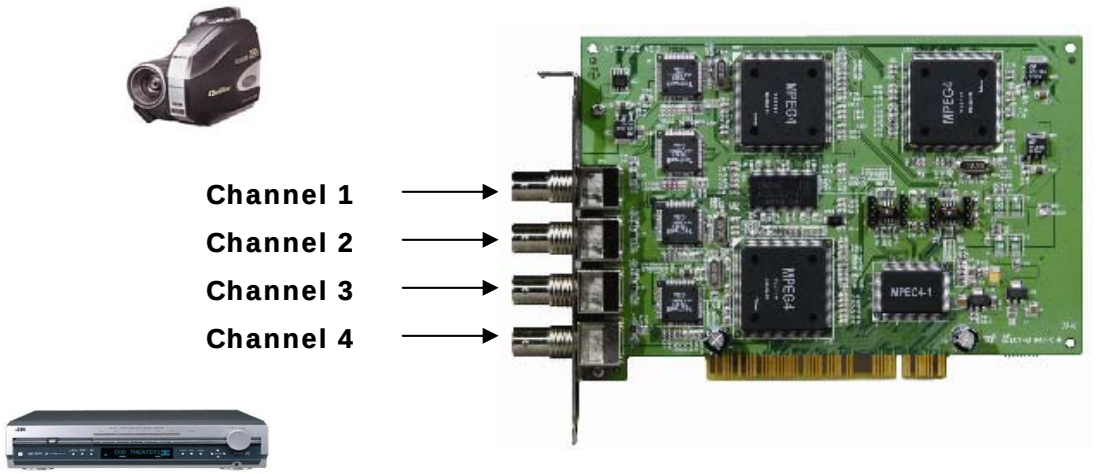
1. Power off the computer.
2. Plug the IVC-4200 into a free PCI slot in your computer.
3. Use the screw to fasten the IVC-4200 on the computer casing.

Note:

- IVC-4200 will take up or share several sets of IRQ and I/O Address. Please make sure that there is enough free sets of IRQ and I/O address for IVC-4200 to use. The IRQ of the PCI slot can be modified from the BIOS setting of the motherboard. Please make the necessary adjustment according to the motherboard's user manual.

2.2 Connections

- IVC-4200



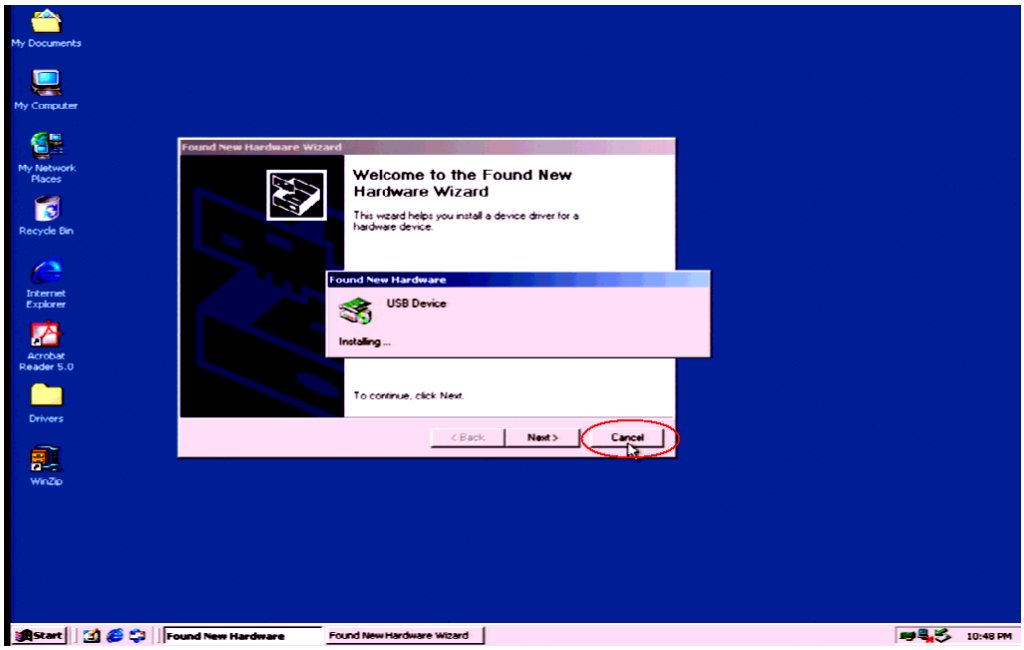
Chapter 3 Software Installation

After you have installed the IVC-4200 card on your computer, turn on the power. Then please follow the instructions for driver and software installation for your IVC-4200 card.

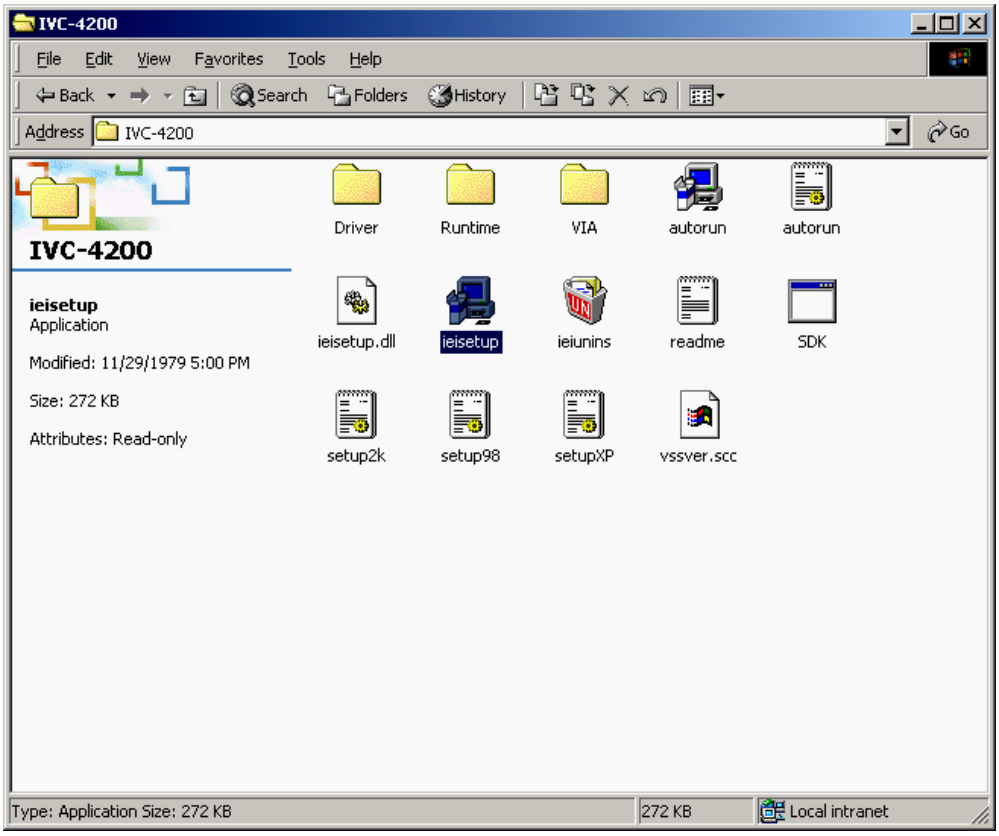
3.1 Driver Installation

Note: The following driver installation procedure applies to Windows 2000. And the procedure is quite similar for other Windows platform.

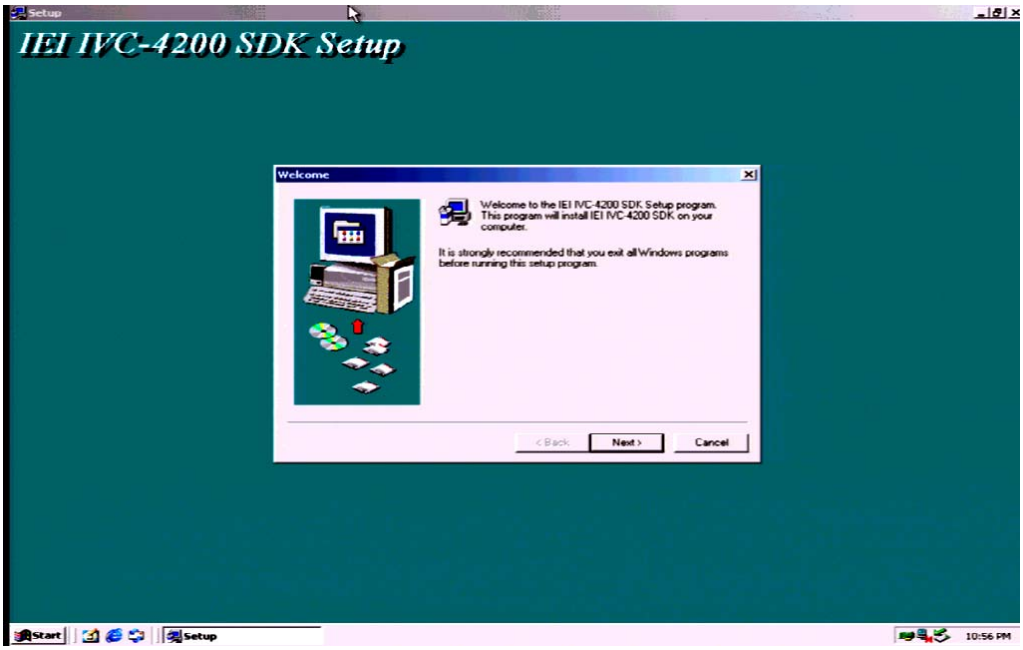
1. Log on to the system, you may see “Found New Hardware” dialogs several times. Click Cancel to skip them.



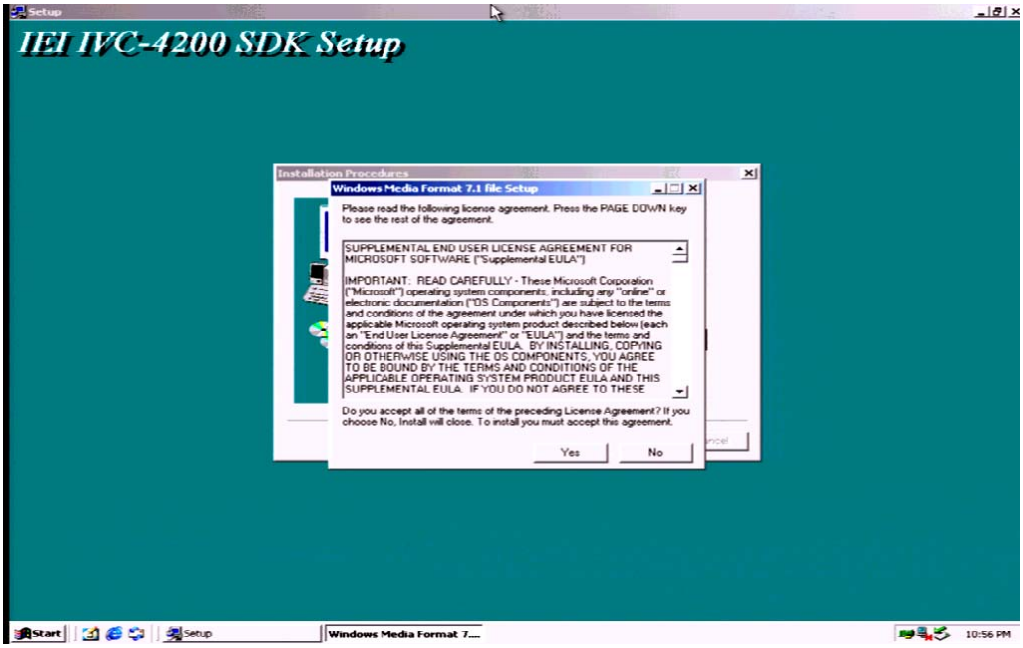
2. Insert the companion driver CD-ROM and browse it using Windows Explorer. Enter the IVC-4200 directory within the CD, then double click “ieisetup”.



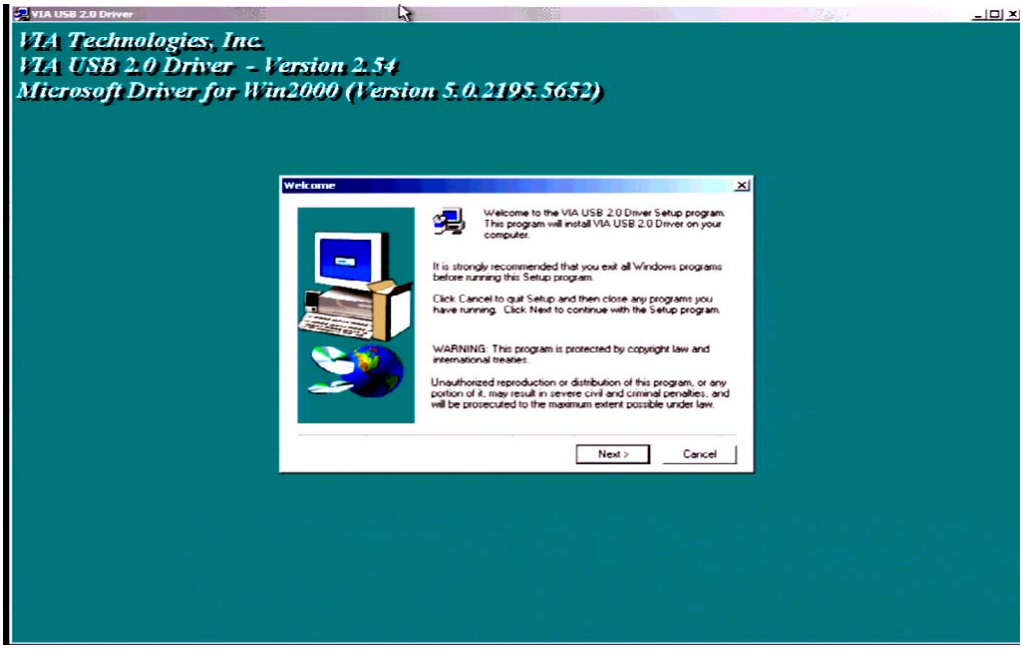
3. The IVC-4200 SDK Setup program will be launched to install the driver and SDK for your IVC-4200 card on your system.



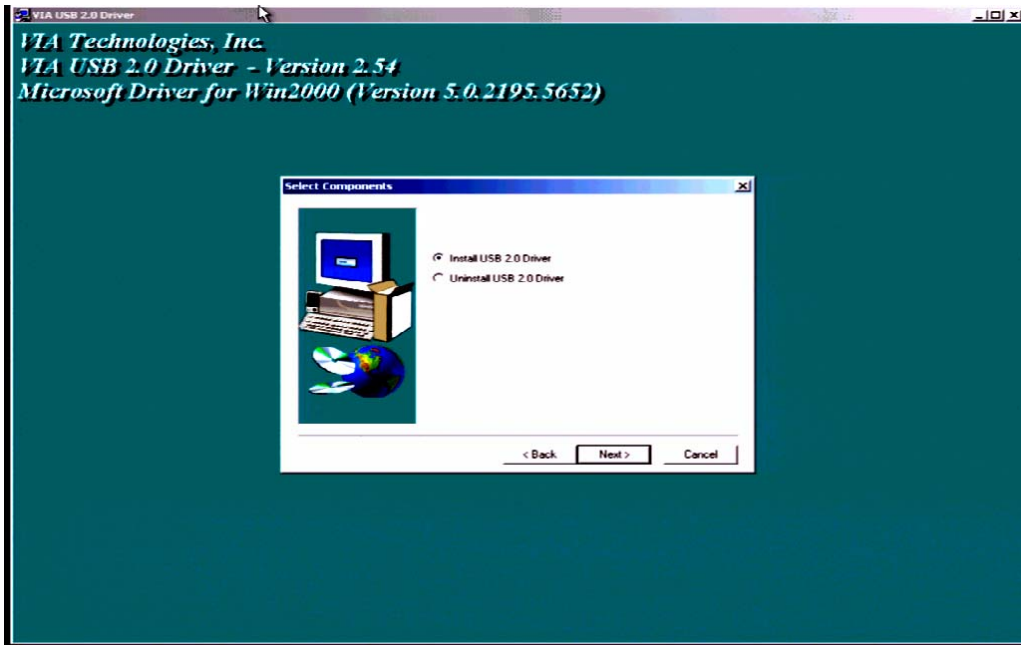
4. The setup program will also install the Microsoft Media Format 7.1 on your system.



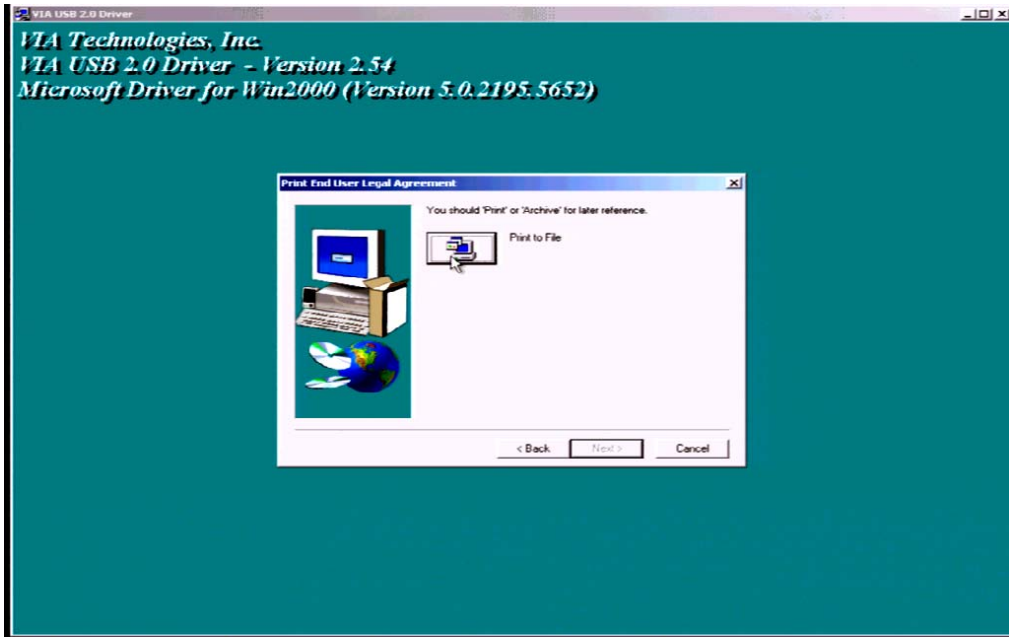
5. The setup program will also install the VIA USB 2.0 driver as well.



6. Select “Install USB 2.0 Driver” and click Next.

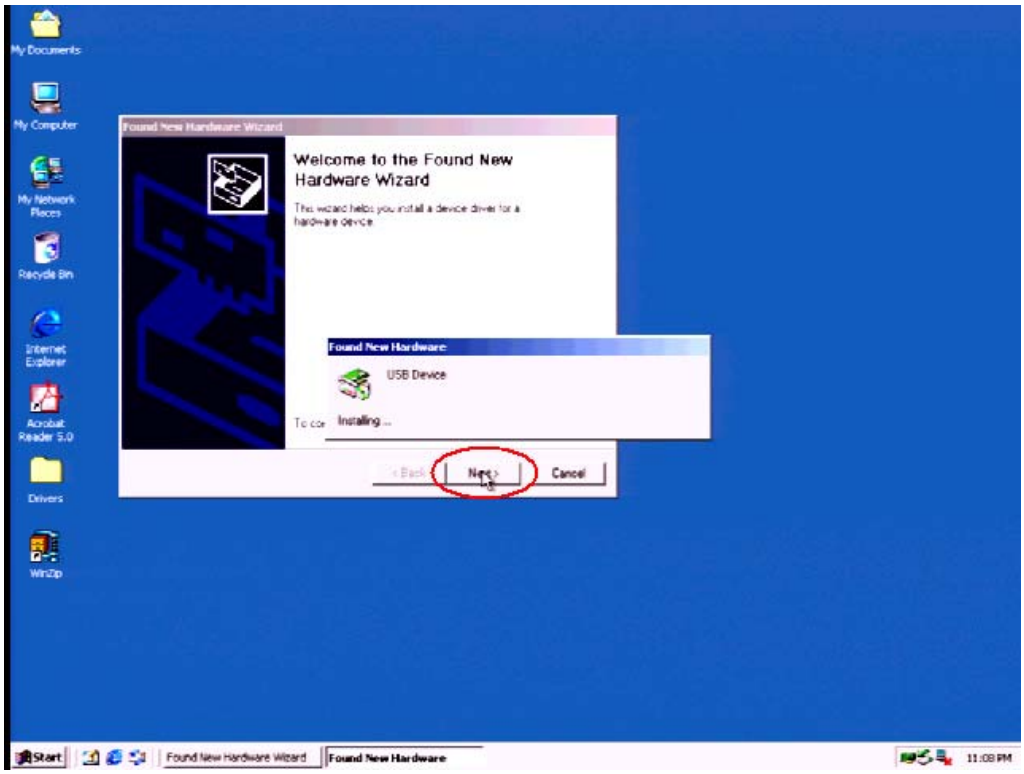


7. On Windows 2000, the setup program will install the Microsoft USB 2.0 Driver for Windows 2000. Follow the on-screen instructions to proceed.

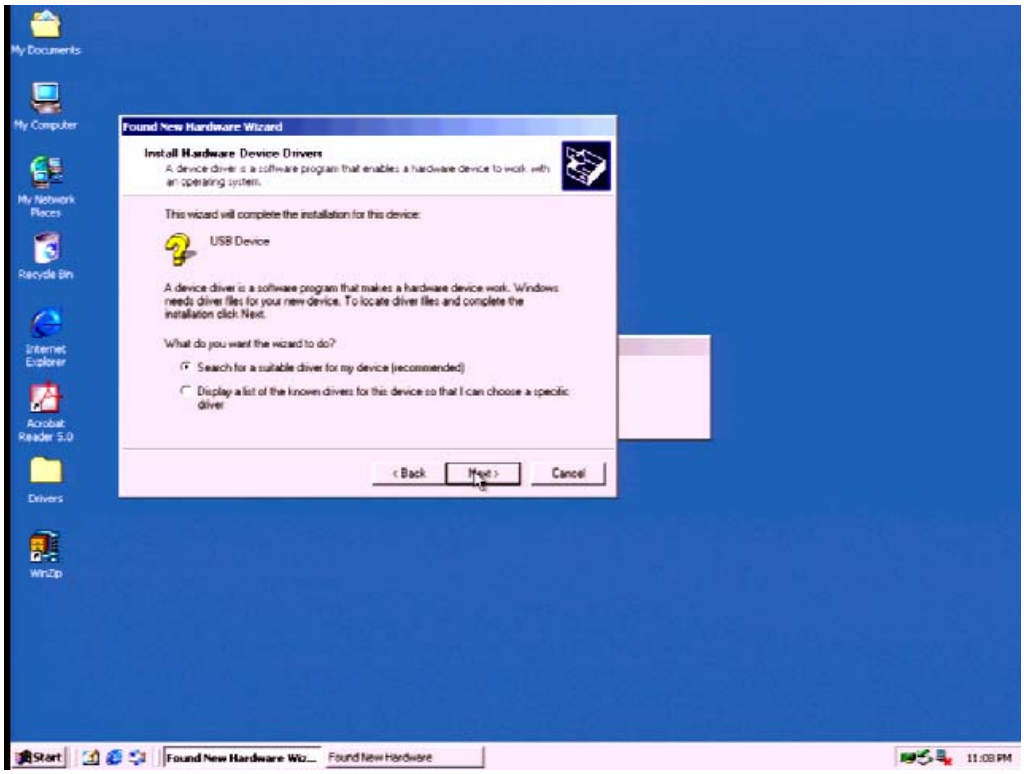


8. After VIA USB 2.0 Driver setup program completes, the system will be restarted automatically.

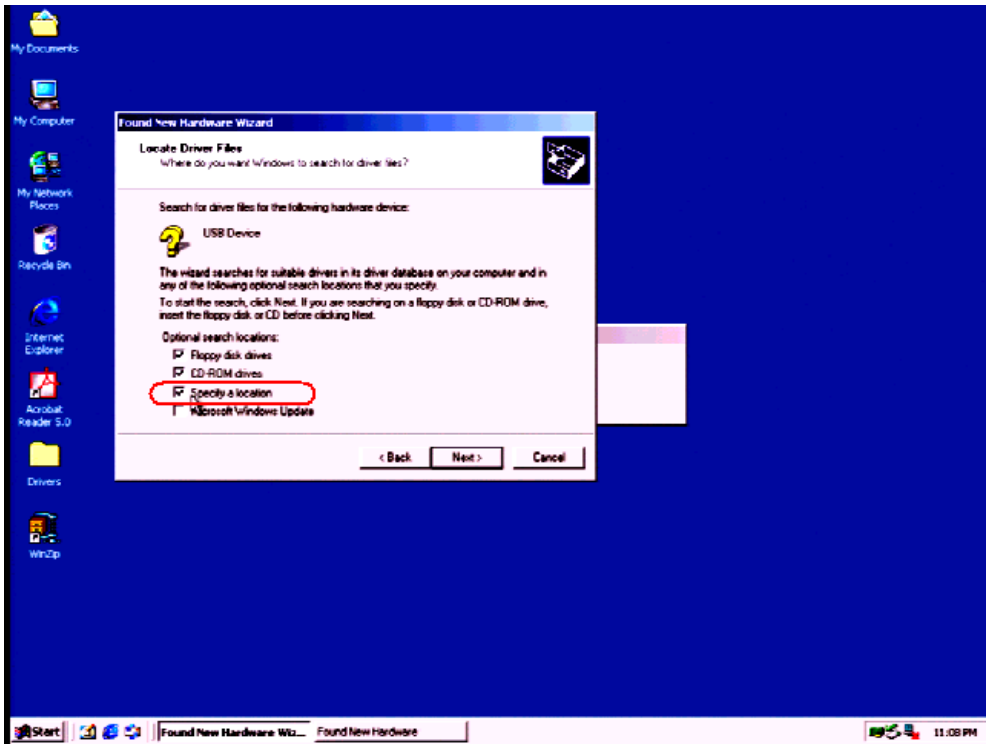
9. Log-on to the system again. The “Found New Hardware” dialog will appear again. Click “Next” to install the driver for the IVC-4200 card.



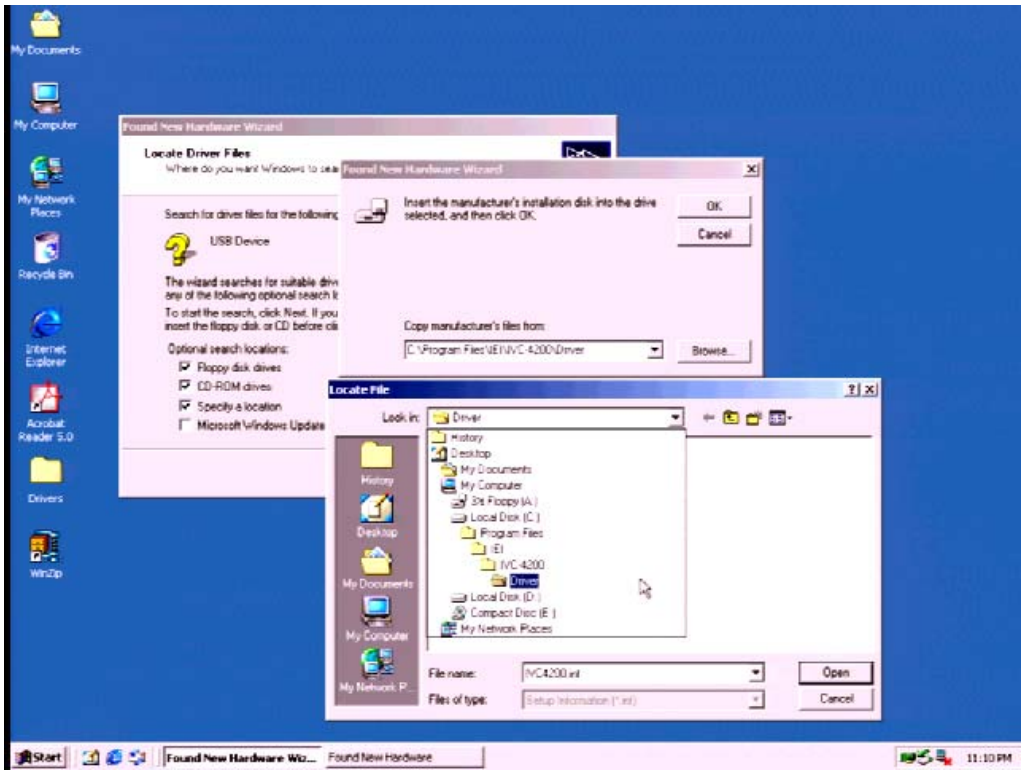
10. Install Hardware Device Drivers dialog will appear, click “Next” to proceed.



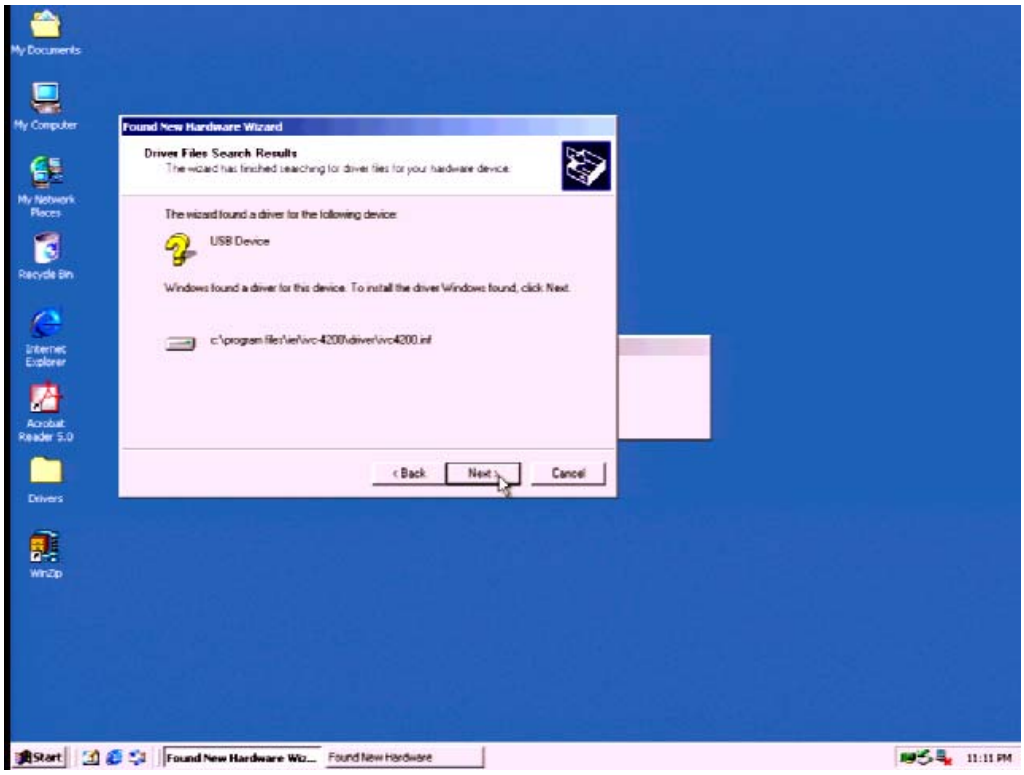
11. You will be asked to locate the device driver, be sure to select “Specify a location” before click the “Next” button.



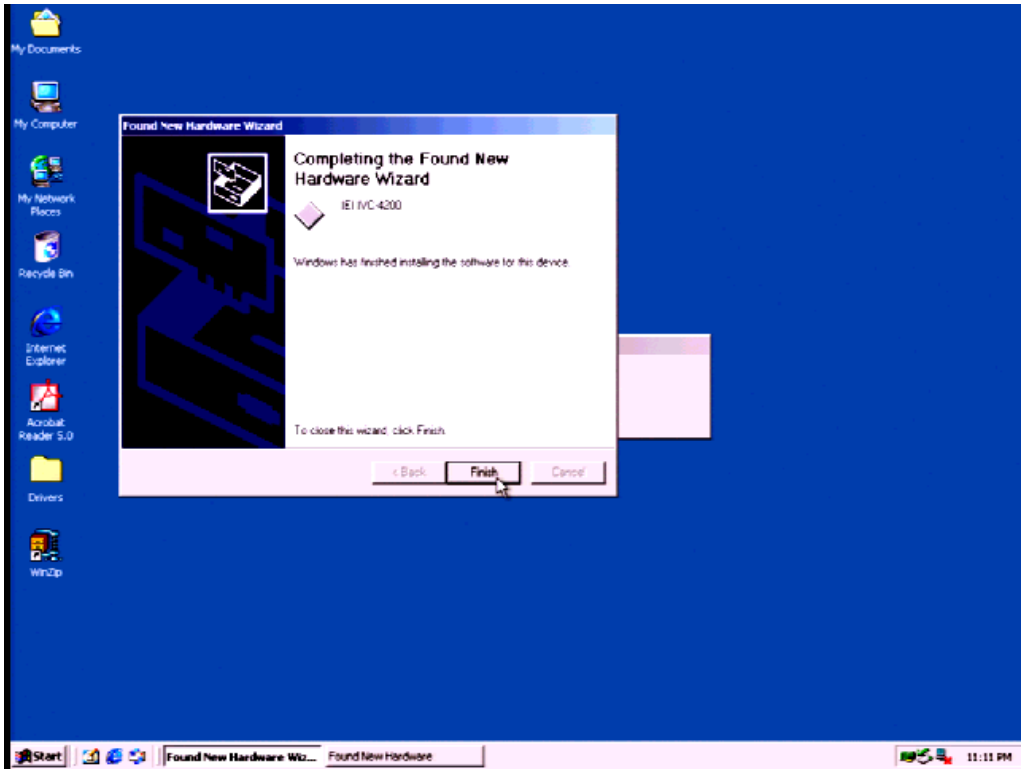
12. The IVC-4200 driver files has been installed in the “IEI\IVC-4200\Driver” directory under the Windows Program Files folder. Normally, just specify the driver location at “C:\Program Files\IEI\IVC-4200\Driver”.



13. The system will detect and find the driver for your IVC-4200, click “Next” to complete the driver installation.



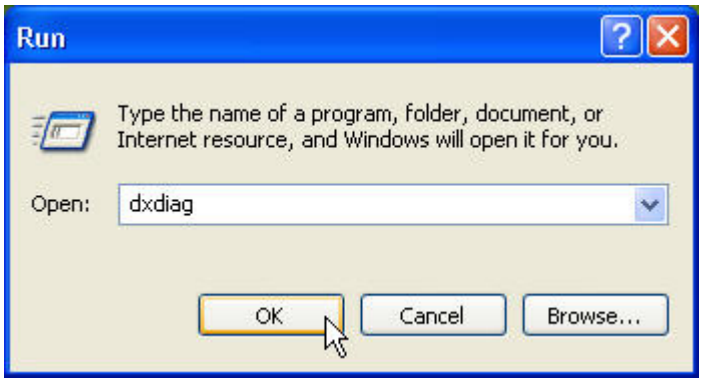
14. After the driver is installed successfully, the following screen will appear.



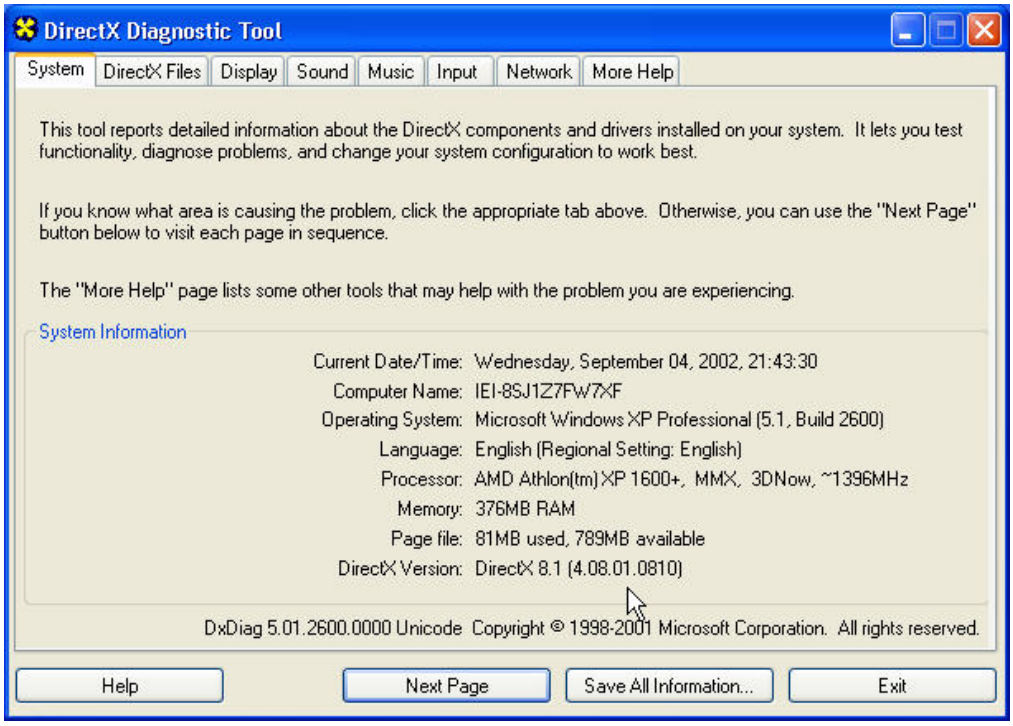
3.2 Checking the DirectX version

After you finish the driver installation, please check the version of DirectX:

1. Select Run in Start Menu, enter “dxdiag” and click OK to run the DirectX Diagnostic Tool.



2. If the version of DirectX is not “DirectX 8.1”, then please install DirectX form the CD packaged with IVC-4200.



Chapter 4 IVC-4200 SDK Programmer Guide

4.1 Introduction

IVC-4200 is a video capture card designed for multi-channel applications. It includes 4 channels MPEG-4 hardware encoder to provide real-time VGA quality video capture for each channel.

1. 4 Channels BNC Composite Video input
2. NSTC/PAL/SECAM Auto-Sensing
3. Full D1, CIF, QCIF with MPEG-4 Advanced Simple Profile @L3 quality stream
4. Output Format: WIS mp4 compatible, DivX, Microsoft WMV, Sigma Design MPEG4, MPEG II, MPEG I
5. Video input signal control of brightness, contrast and saturation.

In addition to drivers, IVC-4200 provides SDK to ease the integration of the IVC-4200 in various applications. The SDK only supports Windows platform with USB support and DirectX (version 8.1 or above) installed, including Windows 2000 and Windows XP. Linux platform is currently not supported. Please check with the technical support (see appendix) for further information if you need to use IVC-4200 under Linux.

4.2 System Requirement

- Windows 2000 or Windows XP
- 128 MB RAM or above
- At least 10 MB free disk space
- DirectX 8.1 (or above version) installed

Note: The SDK needs to decode the encoded MPEG-4 stream using installed software decoder. If you are seeing slow moving pictures, please use CPU or display card with higher performance instead.

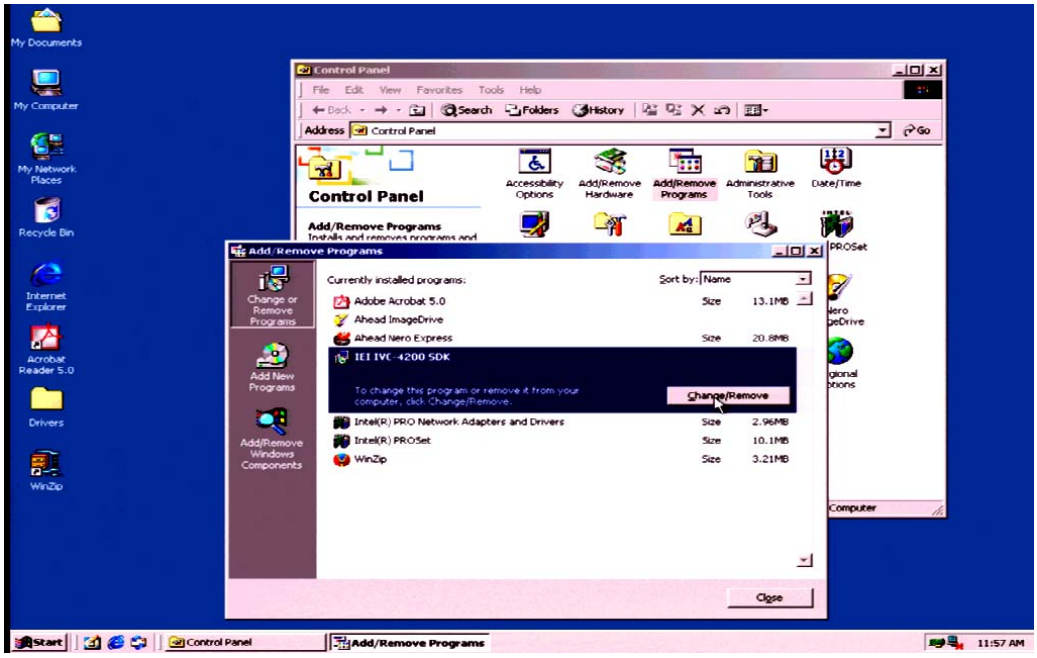
4.3 SDK Installation

If you have finished the software installation, the SDK is already installed on your system. Here are some brief steps to install the driver and SDK on your system:

1. Turn your system off.
2. Install your IVC-4200 MPEG-4 Video Encoder card on an empty PCI slot.
3. Restart your system.
4. Click "Cancel" for all pop-up "Found New Hardware" dialogs.
5. Run ieisetup.exe from the companion CD-ROM to install the SDK software for your IVC-4200 card.
6. Follow the on-screen instructions to install the driver and software. The system will automatically restart after the installation completed.
7. After system restarted, Windows will detect the all the new devices. You should assign the correct path (normally it's under C:\Program Files\IEI\IVC-4200\Driver) to install IVC-4200 driver properly.

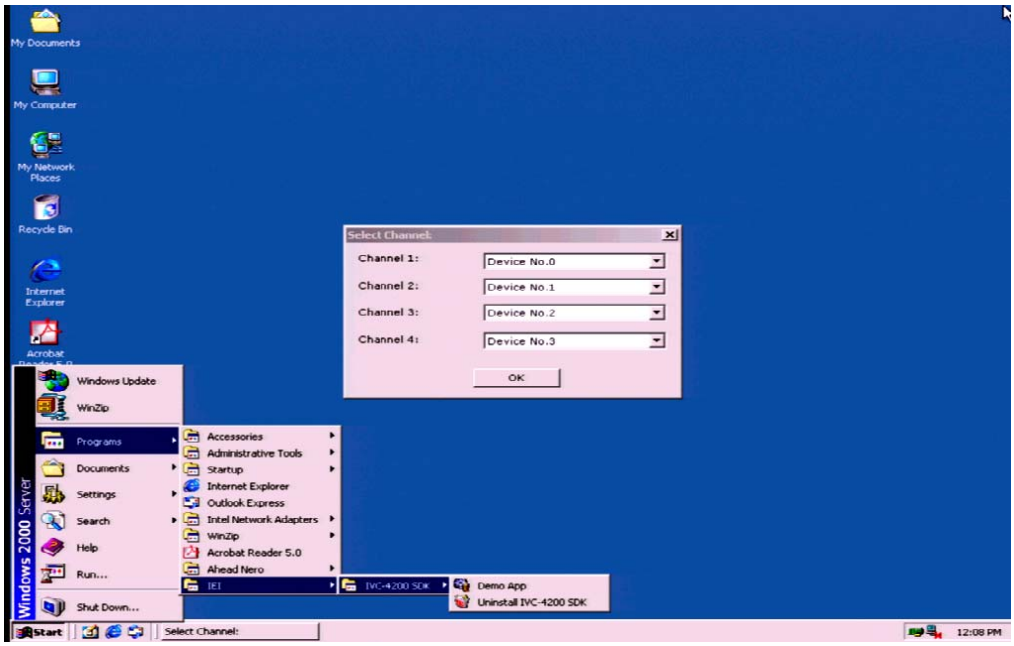
To remove this software, please follow below steps:

1. Open the Control Panel by clicking Settings from Start Menu.
2. Open the "Add/Remove Programs" and select "IEI IVC-4200 SDK" from the list.
3. Click on the "Add/Remove" button and follow the on-screen instructions to remove the software.



4.4 Demo Program (Demo App)

From the Programs menu, run “Demo App” under “IEI / IVC-4200 SDK” to test IVC-4200 card.



4.4.1 Using Demo Program

Here are some basic operations about this demo program:

1. Select Channel:
   You can arrange the channel order of all installed encoder device.
2. Open Channel:
   To start a single channel, just select the channel by clicking on the title bar of that channel and click “Open” button. Or you can click “Open All Channels” to start 4 channel in the same time.
3. Configuration:
   There are two kind of configuration. One is for settings of MPEG-4 stream; the other is for video input control. All MPEG-4 steam settings must be done before device was opened.
4. Full Screen Mode:
   You can select full screen mode while any one device has opened. To exit full screen mode, please press “ESC”.

4.4.2 Demo Program and SDK

The source of the demo application is included in SDK and can be used as a good reference about how to use SDK library functions.

Note: The source code for this demo program is installed in the IEI\IVC-4200\SDK\src directory under the Program Files folder.

Project files:

- IVC4200SDK.cpp: IVC4200 SDK App class definition
- IVC4200SDK.h: IVC4200 SDK App class declaration
- MainFrm.cpp: main frame window class definition
- MainFrm.h: main frame window class declaration
- ChildFrm.cpp: Child frame window class definition
- ChildFrm.h: child frame window class declaration
- DevSelectDlg.cpp: device selection dialog class definition
- DevSelectDlg.h: device selection dialog class declaration
- FSPanel.cpp: full-screen-panel class definition
- FSPanel.h: full-screen-panel class declaration
- IVC4200SDK.rc: resource script file
- Resource.h: resource header file
- StdAtx.cpp: pre-compiler class

- StdAtx.h: pre-compiler header
- Res\IVC4200.ico: main icon file
- Res\IVC4200SDK.rc2: resource script file
- Res\openfile_down.bmp: open file dialog icon
- Res\openfile_up.bmp: open file dialog icon
- Res\Toolbar.bmp: toolbar icon collection

Class description:

- CIVC4200SDKApp:
  Main application class, derived from CWinApp class. Most of SDK API functions are called from here. When InitInstance was called, a device selection dialog will ask user to select the devices to be handled in this session. After main frame window was created, all selected devices will be created by calling CreateAllDevice().
- CDevSelectDlg:
  Derived from CDialog class. This class is a simple common dialog with for combo box. User can specify 4 devices to be used from all available devices installed on current system. After user closes this dialog, the selected device number will be saved in Windows registry in sequence. It will be loaded when this program was launched next time.
- CMainFrame:
  Derived from CMDIFrameWnd. It is a MDI style frame window. It contains four child frame windows, one tool bar and one status bar. This class is designed to pass through the command between child frame and CIVC4200SDKApp class. CMainFrame class also maintains the tool bar and the status bar.
- CchildFrame:
  Derived from CMDIChildWnd. This class will be the owner window of created devices. So it will receive window message post from device driver. And it is also a container of video render window. This class process most user interface operation and pass the command to CIVC4200SDKApp class to do the real things.
- CFSPanel:
  This class is a CWnd class. It creates a none-parent, full-screen window to contain all four video render windows.

4.5 IVC-4200 SDK

4.5.1 SDK File List

Driver and runtime library setup program:

- IVC4200.sys/IVC4200_XP.sys/IVC4200_NT.sys: drivers for IVC-4200
- IVC4200.inf: driver installation description file
- WMFDist.exe: Windows media format 7.1 SDK setup program
- VIA directory: VIA 6202 VT chip driver

Firmware file:

Installed to [windows] directory.

- Go7007fw.bin: firmware for driver.
- Ivc4200in.bin: firmware data for SDK library
- Ivc4200.prf: setting data file
- Tw9903.snr: video sensor data

Runtime filter and DLL:

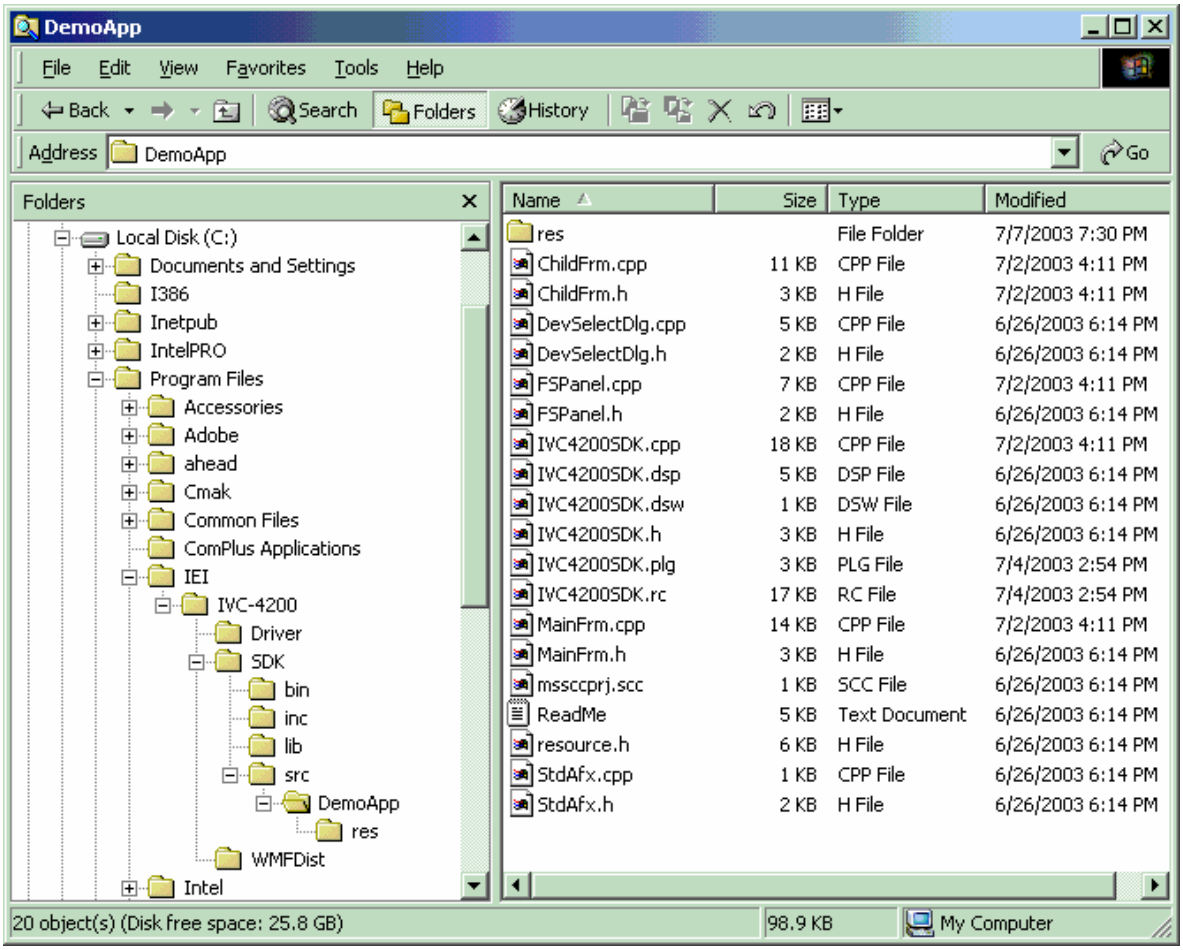
Installed in [Windows System] directory and registered to system registry.

- Mp2filter.ax: MP2 filter
- Wismp3.ax: MP3 filter
- Unidec.ax: DirectShow filter for media player
- StreamRender.dll: preview render module
- SampleTerminator.dll: sample data grabber filter
- msvcr70.dll: runtime library
- atl.dll: ATL runtime library

SDK dynamic linking library, programs and source files:

- IVC-4200\SDK\bin:
  SDKLib.dll: SDK main function DLL
  IVC4200SDK.exe: demo application
  IVC4200SDKAtx.ocx: IVC-4200 SDK ActiveX
- IVC-4200\SDK\inc:
  IVC4200.h: SDKLib DLL header file
- IVC-4200\SDK\lib:
  SDKLib.lib: SDKLib DLL import library

- IVC-4200\SDK\src:
  Source code of demo application (IVC4200SDK.exe)



4.5.2 SDK Library Reference

The IVC-4200 SDK Library contains a set of API that can perform the core function of IVC-4200. It provides an easy way to manipulate IVC-4200 driver and to preview, record and configure with your IVC-4200 video capture card.

Macro

```
// Stream Format Choice:
#define IVC4200_SFC_PRV_AVI                0
#define IVC4200_SFC_MS_WMV                 1
#define IVC4200_SFC_DIVX_AVI               2
#define IVC4200_SFC_SIGMA_AVI              3
#define IVC4200_SFC_MPEG1                  4
#define IVC4200_SFC_MPEG2                  5
#define IVC4200_SFC_H263                   6

// Video stream resoultion
#define IVC4200_RES_D1                     0
#define IVC4200_RES_VGA                    1
#define IVC4200_RES_CIF                    2
#define IVC4200_RES_QCIF                   3

// Frame rate control type
#define IVC4200_FRC_KEEP_ORIGINAL          0
#define IVC4200_FRC_ADPTIVE                1
#define IVC4200_FRC_USER_SPEC              2

// User specific frame rate range
#define IVC4200_FR_MIN                     1
#define IVC4200_FR_MAX                     30

// CBR Level range
#define IVC4200_CBR_LVL_MIN                10
#define IVC4200_CBR_LVL_MAX                60

// CBR bit rate range (in Kbits)
#define IVC4200_CBR_BR_MIN                 1
#define IVC4200_CBR_BR_MAX                 20000

// VBR Level range
#define IVC4200_VBR_LVL_MIN                2
#define IVC4200_VBR_LVL_MAX                30
```

```
// Playback Choice
#define IVC4200_PC_I_DBDR //(Reserved)
#define IVC4200_PC_IP                   1
#define IVC4200_PC_IP_DB                2
#define IVC4200_PC_IP_DBDR              3
#define IVC4200_PC_IPB                  4
#define IVC4200_PC_IPB_DB               5
#define IVC4200_PC_IPB_DBDR             6

// Playback quality on recording
#define IVC4200_PQR_NONE                0
#define IVC4200_PQR_DEGRADE             1
#define IVC4200_PQR_KEEP_QUALITY        2

// Video system type
#define IVC4200_VST_NTSC                0
#define IVC4200_VST_PAL_B               1
#define IVC4200_VST_SECAM               2
#define IVC4200_VST_NTSC_443            3
#define IVC4200_VST_PAL_M               4
#define IVC4200_VST_PAL_CN              5
#define IVC4200_VST_PAL_60              6

// Video render mode: (Only for preview)
#define IVC4200_RM_RGB                  0
#define IVC4200_RM_YUV                  1

// Video signal state:
#define IVC4200_VSS_LOSS                0
#define IVC4200_VSS_SIGNAL              1
#define IVC4200_VSS_UNKNOWN             -1

// Status code...
#define IVC4200_SCODE_NOTCREATED        -1
#define IVC4200_SCODE_READY             0
#define IVC4200_SCODE_RUNNING           1
#define IVC4200_SCODE_DEVICENOTEXIST    0x101
#define IVC4200_SCODE_INITFILELOST      0x102
```

```
#define IVC4200_SCODE_SENSORFILELOST        0x103
#define IVC4200_SCODE_INITSENSORFAIL        0x104
#define IVC4200_SCODE_INITDEVFAIL           0x105
#define IVC4200_SCODE_INVDEVICE             0x106
#define IVC4200_SCODE_BOOTUPFAIL            0x107
#define IVC4200_SCODE_REGDEVNOTIFYERROR     0x1001

// Driver event notification type:
#define IVC4200_DRVEVENT_UNKNOWERROR        -1
#define IVC4200_DRVEVENT_REPORTTS           1
#define IVC4200_DRVEVENT_SNAPSHOT           2
#define IVC4200_DRVEVENT_FATALERROR         3
#define IVC4200_DRVEVENT_DEVICEREMOVED      4
#define IVC4200_DRVEVENT_TIMEOUT            5
#define IVC4200_DRVEVENT_TOOMUCHDELAY       6

// Decode format:

#define IVC4200_DECODE_DDRAW_RGB24          0x03
#define IVC4200_DECODE_DDRAW_RGB32          0x04
#define IVC4200_DECODE_DDRAW_DIB24          0x13
#define IVC4200_DECODE_DDRAW_DIB32          0x14

// Decode mode:
#define IVC4200_DECODE_IONLY                1
#define IVC4200_DECODE_IPONLY               2
#define IVC4200_DECODE_IPB                  3
#define IVC4200_DECODE_IPBDROP              4

// Decode error:
#define IVC4200_DECODE_ERR_NODECODER        1
#define IVC4200_DECODE_ERR_NOFRAME          2
#define IVC4200_DECODE_ERR_NOBUFFER         3
#define IVC4200_DECODE_ERR_NOHEADER         4
#define IVC4200_DECODE_ERR_INVALIDFRAME     5
#define IVC4200_DECODE_ERR_SYNCFAIL         6
#define IVC4200_DECODE_ERR_NOTSYNC          7
#define IVC4200_DECODE_ERR_BUFSMALL         8
```

```
// The device owner must process this window message...
#define WM_IVC4200DRVEVENT                    (0x0400 + 1000)
```

Structure and enumerator

```
enum RECORD_STATE
{
        RS_NONE_ALLOWED = -1,
        RS_ALLOW_RECORD = 0,
        RS_ALLOW_STOP = 1
}

typedef struct {
        double          timeStamp;
        int             nGOP;
        int             nSubGOP;
        int             nPicture;
        double          fno;
        char            ftype;
        double          fq;
} TMP_FrmInfo;

typedef struct {
        TMP_FrmInfo     FrmInfo;
        int             nWidth;
        int             nHeight;
} GOSTRMFrmInfo;
```

Callback function type definition

```
typedef void (CALLBACK* GETENCFRAMEPROC)(GOSTRMFrmInfo*, LPBYTE,
DWORD, LPVOID)
```

API Functions

Device operation function:

Function:

void **SetVendorName**(LPCTSTR lpszName)

Description:

Set vendor name for Windows registry. IVC-4200 SDK library will keep some setting data in Windows registry. You can specify a special key name as the root of those kept data.

Parameter:

lpszName: Name of main key of setting data.

Return:

None.

Function:

int **GetDeviceCount**()

Description:

Get total video channel device count. Each IVC-4200 card has 4 MPEG-4 hardware devices.

Parameter:

None.

Return:

The number of device installed.

Function:

BOOL **IsDeviceExisting**(int nID)

Description:

Determine whether device of given ID exists.

Parameter:

nID: a number between 0 and the device count minus 1.

Return:

TRUE: Device with specified ID exists.

FALSE: No such device id is valid.

Function:

HRESULT **CreateDevice**(int nID, HWND hWnd, HANDLE *phDevice)

Description:

Create a video channel device.

Parameter:

nID: a number between 0 and the device count minus 1.

hWnd: Device's owner window. It will contain the window of live video and it also receive the notification message from device driver.

phDevice: Pointer to a handle variable to receive the created device handle.

Return:

S_OK: Create device successfully.

If CreateDevice failed, get the error code by mask return value with "0x0000FFFF". The error code could be one of the following (same as Win32 error code):

ERROR_OUTOFMEMORY(14): Out of memory.

ERROR_ALREADY_EXISTS(183): Device has been created.

ERROR_IO_DEVICE(1117): Can't register device interface, driver may be corrupted.

ERROR_FILE_NOT_FOUND(2): Sensor configuration file or firmware file not found.

ERROR_NOT_READY(21): Device create failed or firmware file corrupts.

Function:

void **DestroyDevice**(HANDLE hDev)

Description:

Destroy opened video channel device.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

None.

Function:

int **GetDevStatus**(HANDLE hDev)

Description:

Get last status of specific device.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

IVC4200_SCODE_NOTCREATED: undefined. Invalid device handle.
IVC4200_SCODE_READY: Device is ready and can accept StartDevice command.
IVC4200_SCODE_RUNNING: Device is running. While device is running, any configuration change is not acceptable.
IVC4200_SCODE_DEVICENOTEXIST: Device is gone. You may need to reboot the system.
IVC4200_SCODE_INITFILELOST: Firmware file (IVC4200in.bin) can't be found.
IVC4200_SCODE_SENSORFILELOST: Sensor initial file (TW9903.snr) can't be found.
IVC4200_SCODE_INITSENSORFAIL: Device can't initiate sensor. The sensor initial file (TW9903.snr) could be corrupt.
IVC4200_SCODE_INITDEVFAIL: Device initialization error.

IVC4200_SCODE_INVDEVICE: Device open failed.

IVC4200_SCODE_BOOTUPFAIL: Device boot-up failed. System reboot is required.

IVC4200_SCODE_REGDEVNOTIFYERROR: Register device notification failed. Device driver could be corrupt and it should re-install driver again.

Function:

void **GetDevStatusText**(int nStatus, LPTSTR lpBuffer, DWORD cbSize)

Description:

Translate status code to a string text.

Parameter:

nStatus: Status code returned by **GetDevStatus**.

lpBuffer: pointer to a string buffer to receive translated text

cbSize: size of buffer which is pointed by lpBuffer.

Return:

None.

Function:

int **GetDeviceID**(HANDLE hDev)

Description:

Get created video device id number.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

Device id number.

Function:

HWND **GetOwnerWnd**(HANDLE hDev)

Description:

Get owner window of opened video channel device.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

The window handle of owner window, which is the same as the second parameter of function **CreateDevice**.

Function:

BOOL **Start**(HANDLE hDev)

Description:

Start stream output.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

TRUE: successful.

FALSE: failed.

Function:

void **Stop**(HANDLE hDev)

Description:

Stop stream output.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

None.

Function:

BOOL **IsStarted**(HANDLE hDev)

Description:

Check if stream output is stated.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

TRUE: Device is running.

FALSE: device is not running.

Function:

HWND **GetRenderWnd**(HANDLE hDev)

Description:

Get preview render window handle.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

Returns the window handle of video window. You can use this handle to adjust the position and size of this video window.

Function:

void **ShowConfigDlg**(HANDLE hDev)

Description:

Show configuration dialog.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

None.

Function:

void **ShowSensorDlg**(HANDLE hDev)

Description:

Show sensor setting dialog.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

None.

Function:

void **Restart**(HANDLE hDev)

Description:

Restart stream output.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

None.

Function:

BOOL **Snapshot**(HANDLE hDev, LPCTSTR lpszFilepath = NULL)

Description:

Take a snapshot of specific channel device.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

lpszFilepath: File path to save snapshot bitmap file.

Return:

TRUE: successful.

FALSE: failed.

Function:

BOOL **SetSnapshotFile**(HANDLE hDev, LPCTSTR lpszFilepath)

Description:

Pre-set snapshot file path.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

lpszFilepath: File path to save snapshot bitmap file.

Return:

TRUE: successful.

FALSE: failed.

Function:

BOOL **GetSnapshotFile**(HANDLE hDev, LPTSTR lpBuffer, DWORD *pcbSize)

Description:

Get pre-set snapshot file path.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

lpBuffer: Buffer to store pre-set snapshot file path.

pcbSize: size of buffer pointed by lpBuffer.

Return:

TRUE: successful.

FALSE: failed.

Function:

BOOL **SetPreview**(HANDLE hDev, BOOL bValue)

Description:

Set whether show preview window or not while device generating streaming data.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

bValue: TRUE means enabled preview, FALSE means disable.

Return:

TRUE: successful.

FALSE: failed.

Function:

BOOL **IsPreviewing**(HANDLE hDev)

Description:

Check if preview was set.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

TRUE: Preview is enabled.

FALSE: preview is disabled.

Function:

BOOL **SetOnScreenDisplay**(HANDLE hDev, BOOL bValue)

Description:

Show or hide on screen display (OSD) information.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

bValue: TRUE means OSD is visible, FALSE means hide OSD.

Return:

TRUE: successful.

FALSE: failed.

Note:

Only IVC4200_SFC_PRV_AVI format file contains the OSD information.

Function:

BOOL **IsOnScreenDisplay**(HANDLE hDev)

Description:

Check if OSD was set.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

TRUE: OSD is on.

FALSE: OSD is off.

Function:

BOOL **SetOSDText**(HANDLE hDev, LPCTSTR lpszText)

Description:

Set OSD text.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

lpszText: user defined text shown on OSD.

Return:

TRUE: successful.

FALSE: failed.

Function:

BOOL **SetUseOSDTime**(HANDLE hDev, BOOL bValue)

Description:

Determine whether show time stamp on OSD or not.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

bValue: If true, current time is shown on OSD, otherwise OSD is without time info.

Return:

TRUE: successful.

FALSE: failed.

Function:

BOOL **GetUseOSDTime**(HANDLE hDev)

Description:

Check if time stamp was shown or not.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

TRUE: OSD is with time info.

FALSE: no time info on OSD.

Function:

BOOL **StartRecord**(HANDLE hDev)

Description:

Start stream recording.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

TRUE: successful.

FALSE: failed.

Note:

It always returns FALSE if **StartDevice** is not called properly.

Function:

void **StopRecord**(HANDLE hDev)

Description:

Stop stream recording.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

None.

Function:

RECORD_STATE **GetRecordState**(HANDLE hDev)

Description:

Get recording state. Before you call **StartRecord** or **StopRecord**, you should call this function to check whether a recording action is allowed or not.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

RS_NONE_ALLOWED: Neither of recording action (starting and stopping) is allowed.

RS_ALLOW_RECORD: Indicates that you can call **StartRecord** now.

RS_ALLOW_STOP: Indicates that you can call **StopRecord** now.

Function:

int **GetVideoSignalState**(HANDLE hDev)

Description:

Get video signal status.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

-1: undetermined.

0: Video signal is lost.

1: Video signal exists.

Function:

int **ProcessDrvMessage**(HWND hOwnerWnd, WPARAM wParam, LPARAM lParam)

Description:

Process the window message post by the driver. While the owner window has received message: WM_IVC4200DRVEVENT, it must call this function with 2 parameters to get the driver notification event.

Parameter:

hOwnerWnd: Window handle of owner.

wParam: The WPARAM parameter come with message WM_IVC4200DRVEVENT.

lParam: The LPARAM parameter come with message WM_IVC4200DRVEVENT.

Return:

IVC4200_DRVEVENT_UNKNOWERROR: An unknown error occurred in device driver.

IVC4200_DRVEVENT_SNAPSHOT: A snapshot was taken by device.

IVC4200_DRVEVENT_FATALERROR: A fatal error occurred. A system reboot is required.

IVC4200_DRVEVENT_DEVICEREMOVED: Device has been moved.

IVC4200_DRVEVENT_TIMEOUT: Time out.

IVC4200_DRVEVENT_TOOMUCHDELAY: Output stream is too much delay. And can't be render live video for a while.

Function:

BOOL **SetGetFrameCallback**(HANDLE hDev, GETENCFRAMEPROC lpGetFrameProc, LPVOID lpParam)

Description:

Set callback function to get each encoded frame generated by device. The format of frame data received by callback function is specified by function call: **SetStreamFormat**

Parameter:

hDev: Device handle, returned by **CreateDevice**.

lpGetFrameProc: a function pointer pointed.

lpParam: User defined parameter that it will be passed to function pointed by lpGetFrameProc.

Return:

TRUE: successful.

FALSE: failed.

Function:

void **EnableLog**(BOOL bValue)

Description:

Enable to write status and driver events to log file.

Parameter:

bValue: TRUE : Enable, FALSE : Disable.

Return:

TRUE: successful.

FALSE: failed.

Function:

BOOL **IsLogEnabled**()

Description:

To check if log function is turn on.

Return:

TRUE: Log is enabled

FALSE: Log is disabled

Function:

void **SetLogFile**(LPCTSTR lpszFile)

Description:

Set log file pathname. If you did not call this function to specify the log file path and enable the log function, default log file is “SDKlib.log” in the same directory of program.

Parameter:

lpszFile: Log file pathname

Configuration function:

Note: All configuration function can't be called while device is running.

Function:

BOOL **SetResolution**(HANDLE hDev, int nIndex)

Description:

Set video resolution.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

nIndex: Can be one of the following value:

IVC4200_RES_D1: 720 X 480 (NTSC), 720 X 576 (PAL)

IVC4200_RES_VGA: 640 X 480 (VGA)

(default) IVC4200_RES_CIF: 352 X 240 (NTSC), 352 X 288 (PAL)

IVC4200_RES_QCIF: 176 X 112 (NTSC), 176 X 144 (PAL)

Return:

TRUE: successful.

FALSE: failed.

Function:

int **GetResolution**(HANDLE hDev)

Description:

Get current video resolution.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

Returns the index value of resolution. See **SetResolution** function.

Function:

BOOL **SetFrameRateType**(HANDLE hDev, int nIndex)

Description:

Set frame rate setting.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

nIndex: Can be one of the following value: (Default is IVC4200_FRC_ADPTIVE)

IVC4200_FRC_KEEP_ORIGINAL: It will keep the original frame rate from the video source. For example, in NTSC, frate rate is 29.97 fps

IVC4200_FRC_ADPTIVE: It will adjust the frame rate automatically;

IVC4200_FRC_USER_SPEC: It requires the user to specify a frame rate by calling **SetSpecFrameRate** function.

Return:

TRUE: successful.

FALSE: failed.

Function:

int **GetFrameRateType**(HANDLE hDev)

Description:

Get current frame rate setting.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

Returns the index value of frame rate type. See **SetFrameRateType** function.

Function:

BOOL **SetSpecFrameRate**(HANDLE hDev, int nfps)

Description:

Set user specified frame rate.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

nfps: must be any integer between 3 and 30.

Return:

TRUE: successful.

FALSE: failed.

Function:

int **GetSpecFrameRate**(HANDLE hDev)

Description:

Get user specified frame rate.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

User specified frame rate, ranged from 3 to 30.

Function:

BOOL **EnableCBR**(HANDLE hDev, BOOL bValue)

Description:

Enable / disable constant bit-rate stream. If disabled, variable bit-rate stream is used.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

bValue: if TRUE means enable CBR, otherwise use VBR.

Return:

TRUE: successful.

FALSE: failed.

Function:

BOOL **IsCBREnabled**(HANDLE hDev)

Description:

Check if CBR is enabled.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

TRUE: CBR is enabled.

FALSE: VBR is enabled.

Function:

BOOL **SetCBRLevel**(HANDLE hDev, int nValue)

Description:

Set constant bit-rate level.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

nValue: can be any integer between IVC4200_CBR_LVL_MIN (10) and IVC4200_CBR_LVL_MAX (60).

Return:

TRUE: successful.

FALSE: failed.

Note:

The CBR level is higher the bit-rate of output stream is more constant. CBR level is valid on CBR mode.

Function:

int **GetCBRLevel**(HANDLE hDev)

Description:

Get current CBR level.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

Returns CBR level, ranged from IVC4200_CBR_LVL_MIN (10) to IVC4200_CBR_LVL_MAX (60).

Function:

BOOL **SetCBRBitRate**(HANDLE hDev, int nValue)

Description:

Set stream bit-rate in CBR mode. The unit is in Kbit.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

nValue: can be from IVC4200_CBR_BR_MIN (1) to IVC4200_CBR_BR_MAX (20000).

Return:

TRUE: successful.

FALSE: failed.

Note:

The CBR bit-rate setting is the upper constraint of output stream bit-rate. CBR bit-rate setting is only valid on CBR mode.

Function:

int **GetCBRBitRate**(HANDLE hDev)

Description:

Get stream bit-rate in CBR mode.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

Output stream bit-rate, in Kbit, ranged from IVC4200_CBR_BR_MIN (1) to IVC4200_CBR_BR_MAX (20000).

Function:

BOOL **SetVBRLevel**(HANDLE hDev, int nValue)

Description:

Set variable bit-rate level.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

nValue: can be any integer between IVC4200_VBR_LVL_MIN(2) and IVC4200_VBR_LVL_MAX (30).

Return:

TRUE: successful.

FALSE: failed.

Note:

The higher VBR level value represents that the higher bit-rate of output stream. VBR level is valid on VBR mode.

Function:

int **GetVBRLevel**(HANDLE hDev)

Description:

Get variable bit-rate level.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

From IVC4200_VBR_LVL_MIN(2) and IVC4200_VBR_LVL_MAX (30).

Function:

BOOL **SetPlaybackLevel**(HANDLE hDev, int nIndex)

Description:

Set the decoding quality level of live video (preview). Sacrificing the playback quality helps decrease CPU usage substantially, and thus improve the quality of recorded file.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

nIndex: can be one of following value: (Default is IVC4200_PC_IP)

IVC4200_PC_IP, IVC4200_PC_IP_DB, IVC4200_PC_IP_DBDR, IVC4200_PC_IPB, IVC4200_PC_IPB_DB, IVC4200_PC_IPB_DBDR.

Return:

TRUE: successful.

FALSE: failed.

Note:

The quality of live video depends on playback level value. IVC4200_PC_IP is lowest and IVC4200_PC_IPB_DBDR is highest. The range of possible value depends on what format of output stream. Only “WIS MPEG4 AVI” can support up to IVC4200_PC_IPB_DBDR. The quality level is higher the CPU loading is more.

Function:

int **GetPlaybackLevel**(HANDLE hDev)

Description:

Get current decoding level in preview mode.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

Returns the level of live video quality. See **SetPlaybackLevel** function.

Function:

BOOL **SetStreamFormat**(HANDLE hDev, int nIndex)

Description:

Set the format of stream data which is transcoded from GoStream.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

nIndex: can be one of following value: (Default is IVC4200_SFC_PRV_AVI)

IVC4200_SFC_PRV_AVI: Private MPEG-4 AVI File.

IVC4200_SFC_MS_WMV: Microsoft Windows Media Format

IVC4200_SFC_DIVX_AVI: DivX MPEG-4 AVI file

IVC4200_SFC_SIGMA_AVI: Sigma Design MPEG-4 AVI file.

IVC4200_SFC_MPEG1: MPEG-I format

IVC4200_SFC_MPEG2: MPEG-II format

IVC4200_SFC_H263: H.263 format

Return:

TRUE: successful.

FALSE: failed.

Function:

int **GetStreamFormat**(HANDLE hDev)

Description:

Get current stream format.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

Returns the index value of output stream format. See **SetStreamFormat** function.

Function:

BOOL **EnableSplitFile**(HANDLE hDev, BOOL bEnabled)

Description:

Let IVC-4200 to record a long video then save them into several small files.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

bEnabled: Enable to use split recording file feature or not.

Return:

TRUE: successful.

FALSE: failed.

Function:

BOOL **IsSplitFileEnabled**(HANDLE hDev)

Description:

Check if split file is enabled.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

TRUE: Split file is enabled.

FALSE: disabled.

Function:

BOOL **SetSplitFileDuration**(HANDLE hDev, int nDuration)

Description:

Set the time for each split recording file.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

nDuration: Must be any integer between 10 and 10000.

Return:

TRUE: successful.

FALSE: failed.

Function:

int **GetSplitFileDuration**(HANDLE hDev)

Description:

Get file duration of each split recording file in second.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

The time period of each split recording file.

Function:

BOOL **SetRecFilename**(HANDLE hDev, LPCTSTR lpszFilename)

Description:

Set recording file path.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

lpszFilename: set recording file path.

Return:

TRUE: successful.

FALSE: failed.

Function:

BOOL **IsRecFileSpecified**(HANDLE hDev)

Description:

Check if record file path has been specified.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

TRUE: recording file path has been set.

FALSE: recording file path is not set.

Function:

BOOL **GetRecFilename**(HANDLE hDev, LPTSTR lpBuffer, DWORD *pcbSize)

Description:

Get current recording file path.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

lpBuffer: Pointer to a buffer to receive the return recording file path string.

pcbSize: Input size of buffer that pointed by lpBuffer. If lpBuffer is too small, a proper size is put in buffer pointed by pcbSize.

Return:

TRUE: successful.

FALSE: failed.

If failed, check the value of the address pointed by pcbSize. If it is larger than zero, please re-allocate buffer with enough space and pass it again to this function to get file path name.

Function:

BOOL **SetPlayLevelOnRec**(HANDLE hDev, int nIndex)

Description:

Set quality level of live video while recording started.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

nIndex: can be one of following value:

IVC4200_PQR_NONE: Stop previewing.

IVC4200_PQR_DEGRADE: grade down the quality of live video to improve the quality of recorded video when CPU usage is high.

IVC4200_PQR_KEEP_QUALITY: Do not change the quality of live video.

Return:

TRUE: successful.

FALSE: failed.

Function:

int **GetPlayLevelOnRec**(HANDLE hDev)

Description:

Get play level of preview while recording started.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

Returns the level of live video quality on recording. See **SetPlayLevelOnRec** function.

Function:

BOOL **SetBrightness**(HANDLE hDev, int brightness)

Description:

Set video brightness level.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

brightness: from 0 (darkest) to 100 (brightest)

Return:

TRUE: successful.

FALSE: failed.

Video function:

Function:

int **GetBrightness**(HANDLE hDev)

Description:

Get video brightness level.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

Brightness level.

Function:

BOOL **SetHue**(HANDLE hDev, int hue)

Description:

Set video HUE.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

hue: from 0 to 100.

Return:

TRUE: successful.

FALSE: failed.

Function:

int **GetHue**(HANDLE hDev)

Description:

Get video HUE.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

Returns the level of video HUE.

Function:

BOOL **SetSaturation**(HANDLE hDev, int saturation)

Description:

Set video saturation.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

saturation: from 0 to 100.

Return:

TRUE: successful.

FALSE: failed.

Function:

int **GetSaturation**(HANDLE hDev)

Description:

Get video saturation.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

Returns the level of video saturation.

Function:

BOOL **SetContrast**(HANDLE hDev, int contrast)

Description:

Set video contrast value.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

contrast: from 0 to 100.

Return:

TRUE: successful.

FALSE: failed.

Function:

int **GetContrast**(HANDLE hDev)

Description:

Get video contrast value.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

Returns the level of video contrast.

Function:

BOOL **SetAutoVideoSystem**(HANDLE hDev, BOOL bAuto)

Description:

Enable/disable automatic video input system detection.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

bAuto: TRUE to enable auto detection, or FALSE to disable it.

Return:

TRUE: successful.

FALSE: failed.

Function:

BOOL **GetAutoVideoSystem**(HANDLE hDev)

Description:

Check if use automatic video input system detection.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

TRUE: auto detection is enabled.

FALSE: video system selection is by manual setting.

Function:

BOOL **SetVideoSystem**(HANDLE hDev, int colorSystem)

Description:

Specify video input system format.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

colorSystem: can be one of following value:

IVC4200_VST_NTSC, IVC4200_VST_PAL_B, IVC4200_VST_SECAM,

IVC4200_VST_NTSC_443, IVC4200_VST_PAL_M,

IVC4200_VST_PAL_CN, IVC4200_VST_PAL_60

Return:

TRUE: successful.

FALSE: failed.

Function:

int **GetVideoSystem**(HANDLE hDev)

Description:

Get current video input system format.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

Index value of video system type. See **SetVideoSystem** function.

Function:

BOOL **SetInputType**(HANDLE hDev, int inputType)

Description:

This function is reserved.

Function:

int **GetInputType**(HANDLE hDev)

Description:

Get video input type.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

None.

Note:

This function always returns 0.

Function:

BOOL **SetRenderMode**(HANDLE hDev, int nValue)

Description:

Set rendering mode of preview video window. It must be called before start streaming device.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

nValue: Render mode index, can be one of following value:

IVC4200_RM_RGB: Default value. RGB data of each frame will be sent to VGA memory to display.

IVC4200_RM_YUV: Use YUV data to display on VGA. This mode will use less data transfer rate for VGA card.

Return:

TRUE: successful.

FALSE: failed.

Function:

int **GetRenderMode**(HANDLE hDev)

Description:

Get current rendering mode setting of preview video window.

Parameter:

hDev: Device handle, returned by **CreateDevice**.

Return:

Returns the index of rendering mode (see **SetRenderMode**).

Decoder function:

Function:

HANDLE **DecoderInitialize**()

Description:

Create a new decoder.

Parameter:

none.

Return:

Returns a handle of new decoder. Return NULL if failed.

Function:

BOOL **SetDecodeFormat**(HANDLE decoder, int format)

Description:

Set the format of decoded frame data.

Parameter:

decoder: Decoder handle, returned by **DecoderInitialize**.

format: index of format, can be following value:

IVC4200_DECODE_DDRAW_DIB24: DIB24 format

IVC4200_DECODE_DDRAW_DIB32: DIB32 format(default)

IVC4200_DECODE_DDRAW_RGB24: RGB24 format

IVC4200_DECODE_DDRAW_RGB32: RGB32 format

Return:

TRUE: successful.

FALSE: failed.

Note:

This function should be called before calling DecodeOneFrame, otherwise, decoder might lost some frames.

Function:

BOOL **SetDecodeMode**(HANDLE decoder, int mode)

Description:

Set decoding mode.

Parameter:

decoder: Decoder handle, returned by **DecoderInitialize**.

mode: index of decoding mode, can be following value:

IVC4200_DECODE_IONLY: Decode I frame only.

IVC4200_DECODE_IPONLY: Decode I and P frame only.

IVC4200_DECODE_IPB: Decode I, P, B frame.(default)

IVC4200_DECODE_IPBDROP: Decode I, P, B frame, and drop frame is allowed.

Return:

TRUE: successful.

FALSE: failed.

Note:

This function should be called before calling DecodeOneFrame, otherwise, decoder might lost some frames.

Function:

BOOL **DecodeOneFrame**(HANDLE decoder, GOSTRMFrmInfo *frminfo, LPBYTE frame, UINT32 frmsize, BITMAPINFOHEADER* bmphd, LPBYTE buffer, UINT32& bufsize)

Description:

Decode one frame encoded in GO Stream format.

Parameter:

decoder: Device handle, returned by **DecoderInitialize**.

frminfo: Frame information got from callback function.

frame: Encoded frame data buffer.

frmsize: Size of encoded frame.

bmphd: Bitmap header of decoded frame data.

buffer: A buffer pointer for receiving decoded frame data.

bufsize: Size of the buffer. Buffer size is depended on the resolution of video and the format specified by SetDecodeFormat. To calculate the size in need, you can use this formula:

Min buffer size = [WidthBytes] * height

[WidthBytes] = ((width * [bitsdepth] + 0x1f) & 0xfffffe0) >> 3

[bitsdepth] is the bits width of each pixel (ex. DIB24 is 24 bits)

If the buffer size is smaller than the minimum value, it will return false and the minimum value will be set to the parameter bufsize.

Return:

TRUE: successful.

FALSE: failed.

Note:

You can get the encoded frame data by receiving data frome callback function which is set by **SetGetFrameCallback**. The decoder can only support “Private MPEG4” (IVC4200_SFC_PRV_AVI) format currently.

Function:

void **DecoderUninitialize**(HANDLE decoder)

Description:

Close and destroy decoder.

Parameter:

decoder: Device handle, returned by **DecoderInitialize**.

Return:

None. Decoder handle can’t be referenced after calling DecoderUninitialize.

4.5.3 IVC-4200 ActiveX Control Object

IVC-4200 SDK also provides an ActiveX Control (IVC4200SDKAtx.ocx), which has wrapped SDK library function in it.

Description:

- CLSID: 9C07D0E5-8D88-4D78-B038-DDD792FDFB18
- Program ID: IVC4200SDKATX.IVC4200SDKAtxCtrl.1
- File Description: IVC4200SDKAtx ActiveX Control Module
- Type Library name: IVC4200SDK ActiveX Control module
- Type library id: 2054F663-2DBD-416C-951C-3F59CDF8516E

Component feature list:

Properties		
[Name]	**[Range]**	**[Note]**
BSTR VendorName		This string is the key name for Windows registry.
BSTR SnapshotFile		Specify file path for snapshot. **Note**: If the file exists, the picture data will be saved to a file with filename that appends a number behind.
boolean Preview		Turn on/off live video.
boolean OnScreenDisplay		Indicates if you want some text is shown on video window. **Note**: The OSD information only embedded in video file with "WIS MPEG4 format".
boolean UseOSDTime		Indicates if a time stamp should be displayed on OSD. **Note**: this property only works while OnScreenDisplay is true.
long Resolution	0 – 3 0:720 X 480 (D1) 1:640 X 480 (VGA) 2:352 X 240 (CIF) 3:176 X 112(QCIF)	Specify the resolution of video stream.
long FrameRateType	0-2 0: Keep original 1: Adptive 2: User defined	Specify the frame rate control type.
long SpecFrameRate	3 – 30	Specify a user-defined frame rate.
boolean CBREnabled	True: CBR False: VBR	Indicate whether use CBR or VBR output stream.

long CBRLevel	10 – 60	Indicate constant bit-rate level.
long CBRBitRate	1 – 20,000	Indicate specific bit-rate on CBR mode.
long VBRLevel	2 – 30	Indicate variable bit-rate level.
long PlaybackLevel	1 – 6	The quality of live video. (preview)
long StreamFormat	0 – 6 0: PRV Mpeg4 1: MS WMV 2: DivX Mepg4 3: Sigma Design Mpeg 4 4: MPEG-I 5: MPEG-II 6: H.263	The format of output stream.
boolean SplitFileEnabled		Use split file feature.
long SplitFileDuration	10 - 10000	Set time period of each split file.
BSTR RecFilename		Indicate the recording file path.
long PlayLevelOnRec	0 – 2 0: Turn off live video 1: Degrade live video quality 2: Keep live video quality	Indicate quality level of live video while it is recording.
long Brightness	0-100	Indicate brightness value of video.
long Hue	0-100	Indicate hue value of video.
long Saturation	0-100	Indicate saturation value of video.
long Contrast	0-100	Indicate contrast value of video.
boolean AutoVideoSystem		Indicate whether use video system auto-detection feature.
long VideoSystem	0 = NTSC(M) 1 = PAL (B,D,G,H,I) 2 = SECAM 3 = NTSC4.43 4 = PAL (M) 5 = PAL (CN) 6 = PAL 60	Assign video system of video source manually.
long InputType	0	[Reserved]

Methods		
[Name]	**[Param]**	**[Note]**
long GetDeviceCount()		Get the number of installed device. There are four devices in one IVC-4200 card.
boolean IsDeviceExisting(long nID)	nID: id of device, must less than device count.	Check if the device with specified id exists.
boolean CreateDevice(long nID, OLE_HANDLE hOwner)	nID: id of device hOwner: window handle of owner window.	Create device with specified id. The owner window will receive the window message from device driver.
void DestroyDevice()		Destroy created device.
OLE_HANDLE GetOwnerWnd()		Get window handle of owner window.
boolean Start()		Start to output video stream.
void Stop()		Stop outputting video stream.
void ShowConfigDlg()		Display configuration dialog. This method can't be called while device is running.
boolean IsStarted()		Check if device is running.
OLE_HANDLE GetRenderWnd()		Get video render window.
void ResetDevice()		Reset device. Destroy current device then create it again. Use this function if device has trouble in output video stream.
boolean Snapshot(BSTR Filename)	Filename: Assign snapshot file path.	Take a snapshot bitmap to a file.
boolean StartRecord()		Start to record video stream to file.
void StopRecord()		Stop recording.
long GetRecordState()		To determine what recording action is allowed Return value: -1: None of recording action is allowed. 0: Ready for start recording. 1: Stop recording action is allowed.
long GetVideoSignalState()		Check video signal state. Return value: 0: signal lost 1: signal normal -1: undetermined
long ProcessDrvMessage(OLE_HANDLE hOwnerWnd, long wParam, long lParam)	hOwnerWnd: window handle of owner window wParam, lParam is the parameters come with message WM_IVC4200DRVEVENT	Use this method to translate driver notification message: (WM_IVC4200DRVEVENT) while owner window receive it.
void ShowSensorDlg()		Display video control dialog.

void ShowDlg()		This method will automatically determine which configuration dialog can be shown then invoke it.
long GetDevState()		Get last device status: The possible return value: Please see the description of API function: **GetDevStatus()**
BSTR GetDevStateText()		Get the status of device in text.
long GetCreateError()		Get error code on creating device.
boolean SetOSDText(BSTR Text)	Text: Text string on OSD	Set user defined text on preview video window.

Event		
[Name]	**[Param]**	**[Note]**
void OnDrvEvent(long Event)	Event: Indicates which event from driver happened.	Fired on driver event was issued. This event only fired on no owner window.