PCI-MPG24

4-CH MPEG4 Hardware

Video Compression Card

Network Streaming Manual

Manual Rev. 2.00

Revision Date: January 24, 2008

Part No: 50-15062-1000

Recycled Paper

Advance Technologies; Automate the World.

The information in this document is subject to change without prior notice in order to improve reliability, design, and function and does not represent a commitment on the part of the manufacturer.

In no event will the manufacturer be liable for direct, indirect, special, incidental, or consequential damages arising out of the use or inability to use the product or documentation, even if advised of the possibility of such damages.

Trademarks

Microsoft®, Windows NT®, Windows 98®, Windows 2000®, and Windows XP® are registered trademarks of Microsoft Corporation. Borland C++ Builder® is a registered trademark of Borland International, Inc.

Other product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.

Getting Service from ADLINK

Customer Satisfaction is top priority for ADLINK Technology Inc. Please contact us should you require any service or assistance.

ADLINK TECHNOLOGY INC.

Web Site:	http://www.adlinktech.com
Sales & Service:	Service@adlinktech.com
TEL:	+886-2-82265877
FAX:	+886-2-82265717
Address:	9F, No. 166, Jian Yi Road, Chungho City, Taipei, 235 Taiwan

Please email or FAX this completed service form for prompt and satisfactory service.

Company Information	
Company/Organization	
Contact Person	
E-mail Address	
Address	
Country	
TEL	FAX:
Web Site	
Product Information	
Product Model	
Environment	OS: M/B: CPU: Chipset: BIOS:

Please give a detailed description of the problem(s):

Table of Contents

List of Tables

1 Introduction

This manual is used for PCI-MPG24 networking. With this capability, users can send camera images out of local computers and receive them on remote computers.

Two kinds of DirectShow filters are provided: unicast (TCP) and multicast (UDP). Unicast is used for point to point communication and multicast is used for group broadcast communication. We also provided network streaming API (DLL). It is an extension of PCI-MPG24 API.

2 Installation Guide

2.1 Software Requirements

Server Computer

1. DirectX 9.0 SDK for DirectShow developer or DirectX 9.0c runtime for end user
2. PCI-MPG24 setup package
3. Mpg24Net files listed in chapter 2.2

Client Computer

1. DirectX 9.0 SDK for DirectShow developer or DirectX 9.0c runtime for end user
2. DivX v5.x or above for DivX MPEG-4 video subtype and third-party MPEG-2 DirectShow software (InterVideo, Elecard… etc) for MPEG-2 video subtype
3. Mpg24Net files what listed in chapter 2.2

2.2 Software Driver Installation

Network streaming files are bound with the PCI-MPEG24 setup file. Users will not need to install them separately. If you want to install them manually, please follow the steps below:

1. Install DirectX 9.0C.
2. Register filters:

   Open a Windows Command Prompt window and enter the following commands:

   ```
   C> regsvr32 dsnet.ax
   C> regsvr32 dsnet_mcast.ax
   ```

3. Copy shared library:

   Copy Mpg24Net.dll to %windir%/system32.

Note: All files listed above are located on PCI-MPG24 installation directory.

2.3 Setup Build Environment

Include Files

All applications using the APIs must include the file shown in the following table.

Include File	Description
Dsnetifc_ucast.h	The header file required for all C/C++ DirectShow applications with point to point communication.
Dsnetifc_mcast.h	The header file required for all C/C++ DirectShow applications with group broadcast communication.
Mpg24Net.h	The header file required for all C/C++ applications which use PCI-MPG24 API.

Table 2-1: Include Files

Library File

All C/C++ applications using the APIs need the library file shown in the following table.

Library File	Description
Mpg24Net.lib	Exports API function definitions. Required for all C/C++ applications which use PCI-MPG24 API.

Table 2-2: Library File

Note: All of files listed above are located on PCI-MPG24 installation directory

3 DirectShow Programming

3.1 Descriptions of DirectShow Filters

ADLink Unicast Sender

Filter Name	ADLink Unicast Sender
Filter CLSID	{CB763BCE-9540-48fa-9974-69A625D478E3}
Supported Media Types	MEDIATYPE_Video Subtypes: DIVX_MPEG4, MICROSOFT_MPEG4, MPEG2, MJPG

ADLink Unicast Receiver

Filter Name	ADLink Unicast Receiver
Filter CLSID	{15089F31-ACEF-45fe-B497-A2E5D90A69D7}
Supported Media Types	MEDIATYPE_Video Subtypes: DIVX_MPEG4, MICROSOFT_MPEG4, MPEG2, MJPG

ADLink Multicast Sender

Filter Name	ADLink Multicast Sender
Filter CLSID	{CE3B76CB-9540-48fa-9974-69A625D478E3}
Supported Media Types	MEDIATYPE_Video Subtypes: DIVX_MPEG4, MICROSOFT_MPEG4, MPEG2, MJPG

ADLink Multicast Receiver

Filter Name	ADLink Multicast Receiver
Filter CLSID	{319F0815-ACEF-45fe-B497-A2E5D90A69D7}
Supported Media Types	MEDIATYPE_Video Subtypes: DIVX_MPEG4, MICROSOFT_MPEG4, MPEG2, MJPG

3.1.1 Media Types

DIVX_MPEG4

Major Type	MEDIATYPE_Video
SubType	‘D’, ’X’, ’5’, ’0’, 0x0000, 0x0010, 0x80, 0x00, 0x00, 0xaa, 0x00, 0x38, 0x9b, 0x71
Format Type	FORMAT_Videoinfo

MICROSOFT_MPEG4

Major Type	MEDIATYPE_Video
SubType	‘M’, ’P’, ’4’, ’S’, 0x0000, 0x0010, 0x80, 0x00, 0x00, 0xaa, 0x00, 0x38, 0x9b, 0x71
Format Type	FORMAT_Videoinfo

MPEG2

Major Type	MEDIATYPE_Video
SubType	MEDIASUBTYPE_MPEG2_VIDEO
Format Type	FORMAT_MPEG2Video

MJPG

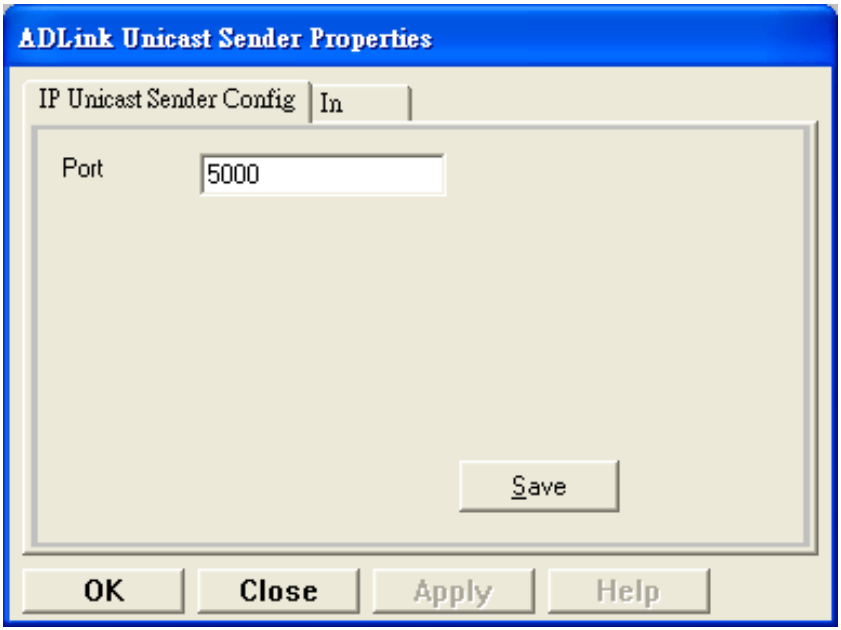
Major Type	MEDIATYPE_Video
SubType	‘M’, ’J’, ’P’, ’G’, 0x0000, 0x0010, 0x80, 0x00, 0x00, 0xaa, 0x00, 0x38, 0x9b, 0x71
Format Type	FORMAT_Videoinfo

3.2 Example Graphs

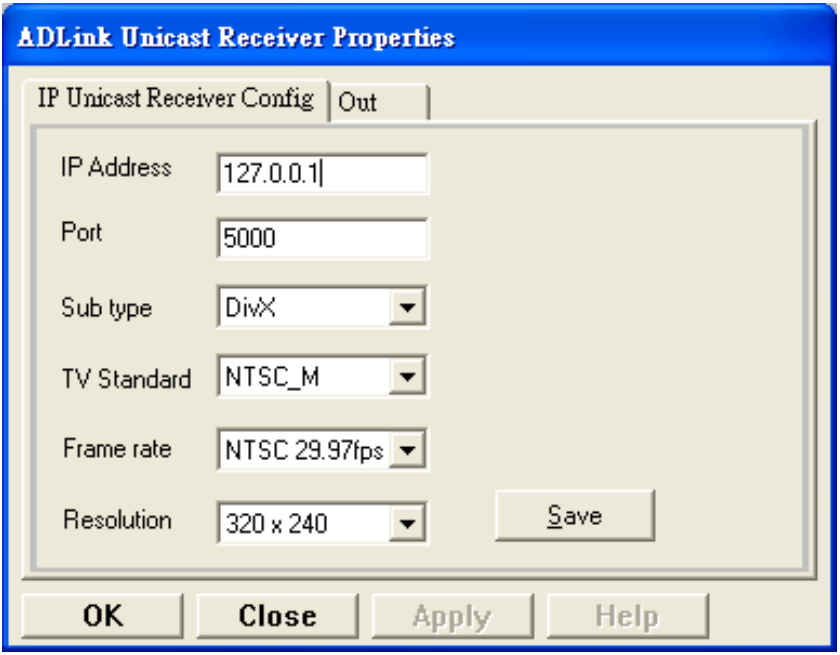
The Microsoft DirectX SDK provides a very useful debugging utility called GraphEdit, which can be used to simulate graph building. From the **Graph** menu of the GraphEdit application, click **Insert Filters…** and choose the desired filters. Filters are organized by categories. Click the **Insert Filter** button to add filters to a graph. Then connect the pins from two filters by dragging the mouse from one of the filters' output pin to another filters' input pin. An arrow will be drawn if these two pins agree on the connection.

After inserting ADLink Unicast Sender filter, ADLink Unicast Receiver, ADLink Multicast Sender filter, or ADLink Multicast Receiver, right click on the rectangle and click **Filter Properties….** The filter properties dialogue will appear. Use the property pages to set video settings before connecting video pins to other filters. The property pages are shown below:

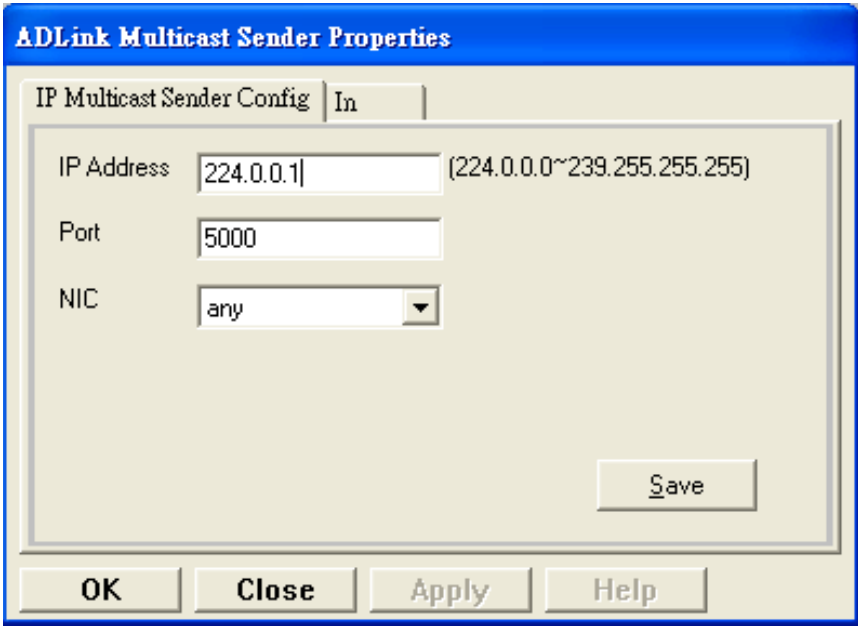
ADLink Unicast Sender



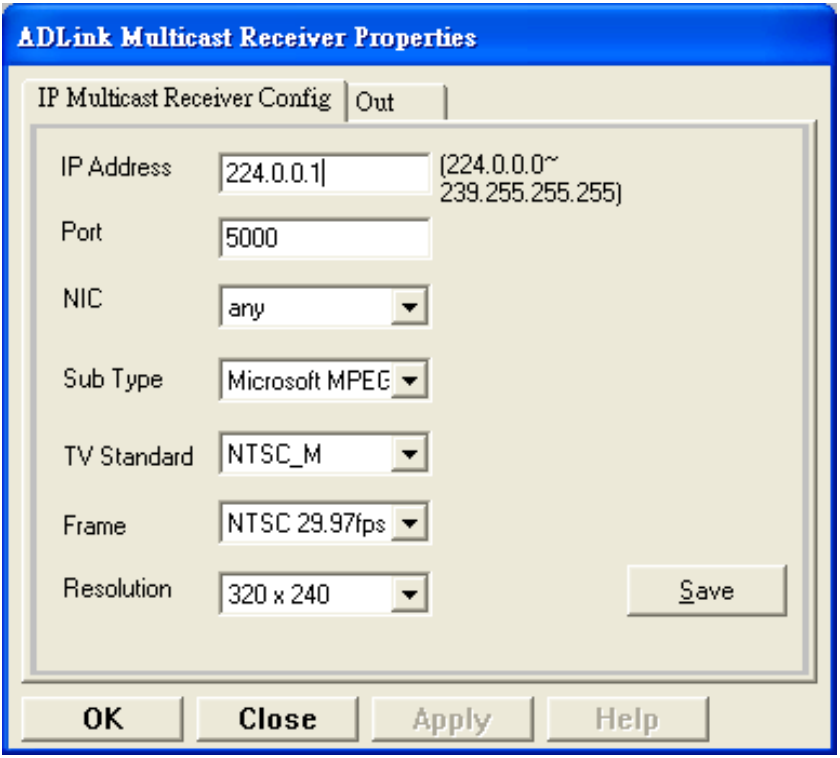
ADLink Unicast Receiver



ADLink Multicast Sender

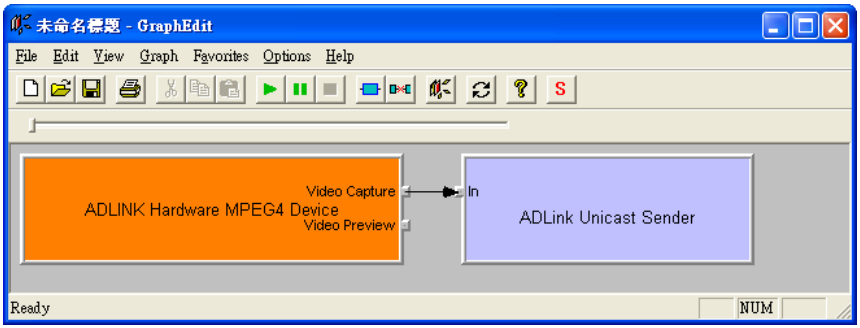


ADLink Multicast Receiver



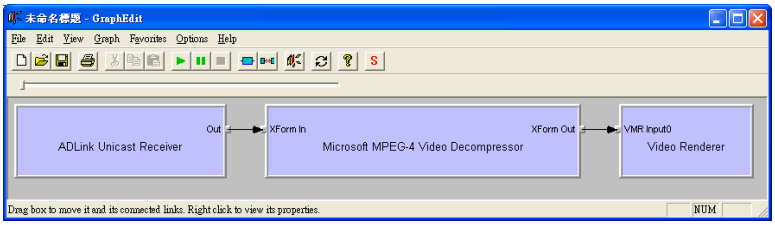
The graphs for sender and receiver are shown below:

Unicast Sender

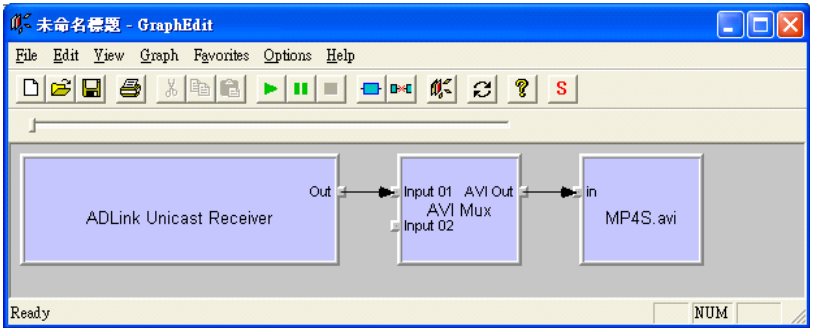


Unicast Receiver

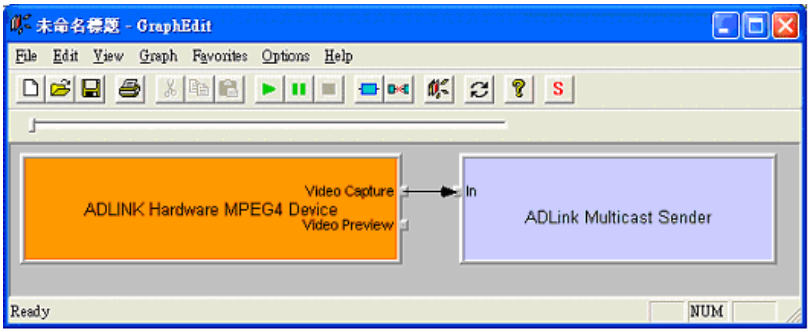
Preview:



File Save:

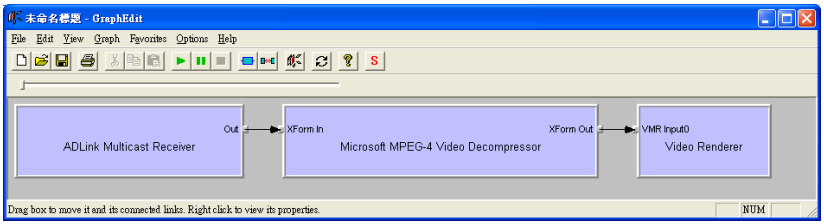


Multicast Sender

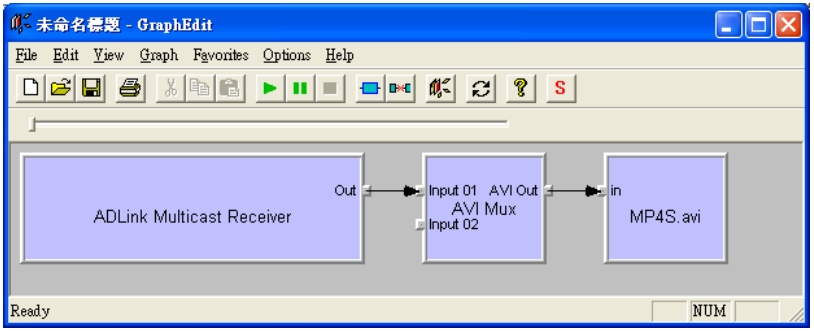


Multicast Receiver

Preview:



File Save:



3.3 Controlling Filters

The ADLink net filters expose COM interfaces as a means for applications to control video configurations.

ADLink Unicast Sender & ADLink Unicast Receiver

Interface	IUnicastConfig
CLSID	{C82CB41C-D32C-4f73-9267-C114DA470378}

ADLink Multicast Sender & ADLink Multicast Receiver

Interface	IMulticastConfig
CLSID	{1CB42CC8-D32C-4f73-9267-C114DA470378}

The methods of IUnicastConfig & IMulticastConfig

3.3.1 SetNetworkInterface

The SetNetworkInterface method sets the Network Interface Card (NIC) address that the filter will use.

Syntax

```
HRESULT SetNetworkInterface(
     ULONG ulNIC
);
```

Parameters

`ulNIC`

Specifies the NIC address, in network order (bytes ordered form left to right). You can use the inet_addr function to convert a standard dotted-format string (such as “255.255.255.255) to the correct binary numbers.

Return values

Value	Description
E_INVALIDARG	Invalid IP address.
E_UNEXPECTED	The operation could not be performed because the filter is not stopped.
S_OK	The method succeeded.

3.3.2 GetNetworkInterface

The GetNetworkInterface method returns the filters' current Network Interface Card (NIC) address.

Syntax

```
HRESULT GetNetworkInterface(
     ULONG* pNIC
);
```

Parameters

pNIC

Pointer to a variable that retrieves the NIC address, in network order (bytes ordered form left to right). You can use the inet_addr function to convert a standard dotted-format string (such as "255.255.255.255) to the correct binary number.

Return values

Value	Description
E_INVALIDARG	Invalid IP address.
S_OK	The method succeeded.

3.3.3 SetUnicastGroup / SetMulticastGroup

The SetUnicastGroup / SetMulticastGroup methods set the group that the filter will send or listen to.

Syntax

```
HRESULT SetUnicastGroup(
      ULONG ulIP,
      USHORT usPort
);
HRESULT SetMulticastGroup(
      ULONG ulIP,
      USHORT usPort
);
```

Parameters

`ulIP`

Specifies the IP address in network order (bytes ordered from left to right). You can use the inet_addr function to convert a standard dotted-format string (such as “255.255.255.255”) to the correct binary number.

`usPort`

Specifies the port number, in TCP/IP network byte order. You can use the htons function to convert the port number to the correct byte order.

Return values

Value	Description
E_INVALIDARG	Invalid IP address.
E_UNEXPECTED	The operation could not be performed because the filter is not stopped.
S_OK	The method succeeded.

3.3.4 GetUnicastGroup / GetMulticastGroup

The GetUnicastGroup / GetMulticastGroup methods retrieve the filters' curent group address.

Syntax

```
HRESULT GetUnicastGroup(
      ULONG* pIP,
      USHORT* pPort
);
HRESULT GetMulticastGroup(
      ULONG* pIP,
      USHORT* pPort
);
```

Parameters

`pIP`

Pointer to a variable that retrieves the IP address in network order (bytes ordered from left to right). You can use the inet_addr function to convert a standard dotted-format string (such as "255.255.255.255") to the correct binary number.

`pPort`

Pointer to a variable that retrieves the port number, in TCP/IP network byte order. You can use the htons function to convert the port number to the correct byte order.

Return values

Value	Description
E_INVALIDARG	Invalid IP address.
S_OK	The method succeeded.

3.3.5 SetSubType

The SetSubType method set the filter's media subtype.

Syntax

```
HRESULT SetSubType(
     ULONG ulSubType
);
```

Paramters

`ulSubType`

Specifies the media subtype as one of the following:

0: DivX MPEG-4 (Need DivX V5.x)

1: Motion JPEG

2: Microsoft MPEG-4 (default)

3: MPEG-2 (Need third-party MPEG-2 DirectShow package, such as InterVideo, elecard, Nero …, etc.)

Return values

Value	Description
E_INVALIDARG	Invalid IP address.
E_UNEXPECTED	The operation could not be performed because the filter is not stopped.
S_OK	The method succeeded.

3.3.6 GetSubType

The GetSubType method retrieves the filter's current media sub-type.

Syntax

```
HRESULT GetSubType(
     ULONG* pSubType
);
```

Paramters

pSubType

Pointer to a variable that retrieves the media subtype as one of the following:

0: DivX MPEG-4 (Need DivX V5.x)

1: Motion JPEG

2: Microsoft MPEG-4 (default)

3: MPEG-2 (Need third-party MPEG-2 DirectShow package, such as InterVideo, elecard, Nero …, etc.)

Return values

Value	Description
E_INVALIDARG	Invalid IP address.
S_OK	The method succeeded.

3.3.7 SetTVStandard

The SetTVStandard method set the filter's TV Standard.

Syntax

```
HRESULT SetTVStandard(
     ULONG ulTVStandard
);
```

Paramters

`ulTVStandard`

Specifies the TV Standard as one of the following:

AnalogVideo_NTSC_M,
AnalogVideo_NTSC_M_J,
AnalogVideo_NTSC_433,
AnalogVideo_PAL_B,
AnalogVideo_PAL_D,
AnalogVideo_PAL_H,
AnalogVideo_PAL_I,
AnalogVideo_PAL_M,
AnalogVideo_PAL_N,
AnalogVideo_PAL_60

Return values

Value	Description
E_INVALIDARG	Invalid IP address.
E_UNEXPECTED	The operation could not be performed because the filter is not stopped.
S_OK	The method succeeded.

3.3.8 GetTVStandard

The GetTVStandard method retrieves the filter's current TV Standard.

Syntax

```
HRESULT GetTVStandard(
     ULONG* pTVStandard
);
```

Paramters

`pTVStandard`

Pointer to a variable that retrieves the TV standard.

Return values

Value	Description
E_INVALIDARG	Invalid IP address.
S_OK	The method succeeded.

3.3.9 SetFrameRate

The SetFrameRate method set the filters' frame rate ratio.

Syntax

```
HRESULT SetFrameRate(
     ULONG ulFrameRate
);
```

Paramters

`ulFrameRate`

Specifies the frame rate ratio. Frame rate ratio = 1001 * frame rate (fps). Fram rate ratio can be expressed as one of following values:

If TV Standard = NTSC,

30000: NTSC 29.97fps(30 fps)
15000: NTSC 15fps(30/2 fps)
10000: NTSC 10fps(30/3 fps)
7500: NTSC 7.5fps(30/4 fps)
6000: NTSC 6fps(30/5 fps)
5000: NTSC 5fps(30/6 fps)
4285: NTSC 4.3fps(30/7 fps)
3000: NTSC 3fps(30/10 fps)
2000: NTSC 2fps(30/15 fps)
1000: NTSC 1fps(30/30 fps)

If TV Standard = PAL,

25025: PAL 25fps(25 fps)
12512: PAL 12.5fps(25/2 fps)
8341: PAL 8.3fps(25/3 fps)
6256: PAL 6fps(25/4 fps)
5005: PAL 5fps(25/5 fps)
4170: PAL 4.2fps(25/6 fps)
3128: PAL 3fps(25/8 fps)
2085: PAL 2fps(25/12 fps)

1001: PAL 1fps(25/25 fps)

Return values

Value	Description
E_INVALIDARG	Invalid IP address.
E_UNEXPECTED	The operation could not be performed because the filter is not stopped.
S_OK	The method succeeded.

3.3.10 GetFrameRate

The GetFrameRate method retrieves the filter's current frame rate ratio.

Syntax

```
HRESULT GetFrameRate(
     ULONG* pFrameRate
);
```

Paramters

`pFrameRate`

Pointer to a variable that retrieves the frame rate ratio. Frame rate ratio = 1001* frame rate (fps).

Return values

Value	Description
E_INVALIDARG	Invalid IP address.
S_OK	The method succeeded.

3.3.11 SetResolution

The SetResolution method set the filter's resolution.

Syntax

```
HRESULT SetResolution(
      ULONG ulWidth,
      ULONG ulHeight
);
```

Paramters

(**`ulWidth`**, **`ulHeight`**)

Specifies the width and height. It could be one of the following pairs:

If TV Standard = NTSC,

(720, 480),
(352, 240),
(640, 480),
(320, 240),
(480, 480),
(176, 144),
(352, 480),
(720, 240)

If TV Standard = PAL,

(720, 576),
(352, 288),
(480, 576),
(176, 144),
(352, 576),
(640, 480),
(720, 288)

Return values

Value	Description
E_INVALIDARG	Invalid IP address.
E_UNEXPECTED	The operation could not be performed because the filter is not stopped.
S_OK	The method succeeded.

3.3.12 GetResolution

The GetResolution method retrieves the filter's current width and height.

Syntax

```
HRESULT GetResolution(
     ULONG* pWidth,
     ULONG* pHeight
);
```

Paramters

`pWidth`

Pointer to a variable that retrieves the media width.

`pHeight`

Pointer to a variable that retrieves the media height.

Return values

Value	Description
E_INVALIDARG	Invalid IP address.
S_OK	The method succeeded.

4 Windows API Functions

4.1 Introduction

The Application Program Interface (API) functions are based on DirectX 9.0. DirectX 9.0 must be installed in order for the API functions to work.

The API library is additional software to PCI-MPG24.dll for network streaming extension.

4.2 Function List

ADLINK Unicast Sender

Function Name	Section
Mpg24Net_ServerOpen	4.3.1
Mpg24Net_ServerClose	4.3.2
Mpg24Net_ServerStart	4.3.3
Mpg24Net_ServerStop	4.3.4
Mpg24Net_ServerSend	4.3.5
Mpg24Net_ClientOpen	4.4.1
Mpg24Net_ClientClose	4.4.2
Mpg24Net_ClientSetSubType	4.4.3
Mpg24Net_ClientSetTVStandard	4.4.4
Mpg24Net_ClientSetResolution	4.4.5
Mpg24Net_ClientSetFrameRate	4.4.6
Mpg24Net_ClientSetPreviewDisplay	4.4.7
Mpg24Net_ClientGetConnectionStatus	4.4.8
Mpg24Net_ClientStartPreview	4.4.9
Mpg24Net_ClientStartSave	4.4.10
Mpg24Net_ClientStop	4.4.11
Mpg24Net_ClientPreviewShow	4.4.12
Mpg24Net_GetLastErrorInfo	4.5

4.3 Server functions

4.3.1 Mpg24Net_ServerOpen

This function will open a new server instance. A server instance is a network server that waits for client connections (unicast), or sends stream data to the same network group of receivers (multi-cast).

Syntax

```
int Mpg24Net_ServerOpen(
     int StreamType = 0
);
```

Parameters

`StreamType`

Specifies the type of the network streaming. It could be:

0: Unicast (TCP). Point to point communication.

1: Multicast (UDP). Group broadcast communication.

Return Values

Handle of server instance if return value is greater than or equal to 0. Or an error code if return value is negative. See chapter 4.5 for the error code.

4.3.2 Mpg24Net_ServerClose

This function will close a server instance. A server instance is a network server that waits for client connections (unicast), or sends stream data to the same network group of receivers (multicast).

Syntax

```
int Mpg24Net_ServerClose(
     int hHandle
);
```

Parameters

hHandle

Specifies the handle of the server instance.

Return Values

See chapter 4.5.

4.3.3 Mpg24Net_ServerStart

This function will start a server. The server will open a network socket and wait for client connections (unicast), or send stream data over the network (multicast). If the stream type is unicast, a server instance has a maximum of 16 concurrent connections.

Syntax

```
int Mpg24Net_ServerStart(
      int hHandle,
      short port,
      char *ServerIp=””
);
```

Parameters

`hHandle`

Specifies the handle of the server instance.

`Port`

Specifies the port number.

`ServerIp`

Specifies the IP address, in network order (bytes ordered from left to right). You can leave it blank for unicast type.

Return Values

See chapter 4.5.

4.3.4 Mpg24Net_ServerStop

This function can be used to stop a server. The server will close a network socket and close all client connections.

Syntax

```
int Mpg24Net_ServerStop(
     int hHandle
);
```

Parameters

`hHandle`

Specifies the handle of the server instance.

Return Values

See chapter 4.5.

4.3.5 Mpg24Net_ServerSend

This function can be used to send stream data to all clients. The stream data comes from external library (callback function of PCI_MPG24.dll).

Syntax

```
int Mpg24Net_ServerSend(
     int hHandle,
     BYTE * pBuffer,
     int iBufferSize,
     int iSyncPoint
);
```

Parameters

`hHandle`

Specifies the handle of the server instance.

`pBuffer`

Specifies the buffer pointer of stream data.

`iBufferSize`

Specifies the size of stream data.

`iSyncPoint`

Specifies whether the stream data is a key frame (I frame). Its value could be:

0: Key frame (I frame)

1: Other frame (P frame or B frame)

Return Values

See chapter 4.5.

Example

Below is a server sample which functions open a unicast server instance and which send data to all clients.

```
#include "PCI_MPG24.h"
#include "Mpg24Net.h"
int hHandle;
```

```
void __stdcall MyCallbackProc(
      BYTE * pBuffer,
      int iBufferSize,
      int iSyncPoint
)
{
      if(iBufferSize>0 && pBuffer)
            Mpg24Net_ServerSend(
                  hHandle,
                  pBuffer,
                  iBufferSize,
                  iSyncPoint
            );
}

start()
{
      // Open ADLink Hardware MPEG4 Device
      Mpg24_EncoderOpen(0, 0);
      // Set configuration: NTSC Full D1 30fps
      4Mbps Microsoft MPEG-4
      Mpg24_EncoderSetVideoFormat(0, 0,
      VideoFormat_Standard, 0);
      Mpg24_EncoderSetVideoFormat(0, 0,
      VideoFormat_Size, 0);
      Mpg24_EncoderSetVideoFormat(0, 0,
      VideoFormat_FrameRate, 0);
      Mpg24_EncoderSetVideoFormat(0, 0,
      VideoFormat_TargetBitrate, 0);
      Mpg24_EncoderSetVideoFormat(0, 0,
      VideoFormat_SubType, 0);
      Mpg24_EncoderSetVideoFormat(0, 0,
      VideoFormat_MPEG4Mode, 1);
      Mpg24_EncoderCallback(0, 0,
      MyCallbackProc);
      // Open a server instance
      hHandle = Mpg24Net_ServerOpen(0);

      // Start
      Mpg24Net_ServerStart(hHandle, 5000);
      Mpg24_EncoderRun(0, 0);
}
```

4.4 Client functions

4.4.1 Mpg24Net_ClientOpen

This function will open a new client instance. A client instance is a network client that connects to a server (unicast), or joins a network group (multicast).

Syntax

```
int Mpg24Net_ClientOpen(
     int StreamType = 0
);
```

Parameters

`StreamType`

Specifies the type of the network streaming. It could be:

0: Unicast (TCP), point to point communication.

1: Multicast (UDP), group broadcast communication.

Return Values

Handle of client instance if return value is greater than or equal to 0. Or an error code if return value is negative. See chapter 4.5 for the error code.

4.4.2 Mpg24Net_ClientClose

This function will close the client instance. A client instance is a network client that connects to a server (unicast), or joins a network group (multicast).

Syntax

```
int Mpg24Net_ClientClose (
     int hHandle
);
```

Parameters

`hHandle`

Specifies the handle of the client instance.

Return Values

Handle of client instance if return value is greater than or equal to 0. Or an error code if return value is negative. See chapter 4.5 for the error code.

4.4.3 Mpg24Net_ClientSetSubType

Specifies the video subtype.

Syntax

```
int Mpg24Net_ClientSetSubType (
     int hHandle,
     int iSubType
);
```

Parameters

`hHandle`

Specifies the handle of the client instance.

`iSubType`

Specifies the video subtype. It could be:

0: DivX MPEG-4(need DivX V5.x)

1: Motion JPEG

2: Microsoft MPEG-4 (default)

3: MPEG-2 (Need third-party MPEG-2 DirectShow software, such as InterVideo, Elecard, Nero…,etc.)

Return Values

See chapter 4.5.

4.4.4 Mpg24Net_ClientSetTVStandard

Specifies the TV standard.

Syntax

```
int Mpg24Net_ClientSetTVStandard (
     int hHandle,
     int iTVStandard
);
```

Parameters

`hHandle`

Specifies the handle of the client instance.

`iTVStandard`

Specifies the TV standard. It could be:

0: NTSC

1: PAL

Return Values

See chapter 4.5.

4.4.5 Mpg24Net_ClientSetResolution

This function will specifies the specific resolution (width and height of video).

Syntax

```
int Mpg24Net_ClientSetResolution (
     int hHandle,
     int iResolution
);
```

Parameters

`hHandle`

Specifies the handle of the client instance.

`iResolution`

Specifies the specific resolution. It depends on the value of the TV standard. It could be:

TV Standard = NTSC

0: Full D1 (720x480)
1: CIF (352x240)
2: QCIF (176x144)
3: VGA (640x480)
4: QVGA (320x240)

TV Standard = PAL

0: Full D1 (720x576)
1: CIF (352x288)
2: QCIF (176x144)
3: VGA (640x480)

Return Values

See chapter 4.5.

4.4.6 Mpg24Net_ClientSetFrameRate

Specifies the specific frame rate.

Syntax

```
int Mpg24Net_ClientSetFrameRate (
     int hHandle,
     int iFrameRate
);
```

Parameters

`hHandle`

Specifies the handle of the client instance.

`iResolution`

Specifies the specific frame rate which is dependant upon the value of the TV standard. It could be:

TV Standard = NTSC

0: 29.97 fps

1: 15 fps

2: 10 fps

3: 5 fps

4: 1 fps

TV Standard = PAL

0: 25 fps

1: 12.5 fps

2: 8.3 fps

3: 5 fps

4: 1 fps

Return Values

See chapter 4.5.

4.4.7 Mpg24Net_ClientSetPreviewDisplay

Specifies the configuration of preview window.

Syntax

```
int Mpg24Net_ClientSetPreviewDisplay (
     int hHandle,
     int hWnd,
     int Left,
     int Top,
     int Width,
     int Height,
     int AutoShow
);
```

Parameters

`hHandle`

Specifies the handle of the client instance.

`hWnd`

Specifies the handle of a parent window for the video window. The video images will overlay on this window. Setting it to null will create a new window and display the video images on it.

`Left`

Specifies the x-coordinate of the video window.

`Top`

Specifies the Y-coordinate of the video window.

`Width`

Specifies the width of the video window.

`Height`

Specifies the height of the video window.

`AutoShow`

Specifies whether the video renderer automatically shows the video window when it receives video data.

Return Values

See chapter 4.5.

4.4.8 Mpg24Net_ClientGetConnectionStatus

This function can be used to retrieve the connection status either to the server (unicast), or a network group multicast).

Syntax

```
int Mpg24Net_ClientGetConnectionStatus(
          int hHandle
);
```

Parameters

hHandle

Specifies the handle of the client instance.

Return Values

0: Disconnected

1: Connected

Others: Error occurs. See chapter 4.5.

4.4.9 Mpg24Net_ClientStartPreview

This function will start a client preview of a video graph. The client will open a network socket and establish a connection with the server (unicast), or it will join a network (multicast).

Syntax

```
int Mpg24Net_ClientStartPreview(
     int hHandle,
     char *ServerIp,
     short port
);
```

Parameters

`hHandle`

Specifies the handle of the client instance.

`ServerIp`

Specifies the IP addres, in network order (bytes ordered from left to right). For unicast, the IP address will be the IP address of the server. For multicast, the IP address is same as the parameter 'ServerIp' of Mpg24Net_ServerOpen().

`Port`

Specifies the port number that is same as the parameter 'port' of Mpg24Net_ServerOpen().

Return Values

See chapter 4.5.

Note: If MPEG-2 sub type is selected, you must install a third-party MPEG-2 decoder.

4.4.10 Mpg24Net_ClientStartSave

This function can be used to start a client. The client will open a network socket and establish a connection with the server (unicast), or join a network (multicast). This function also receives the stream data and saves them to a media file.

Syntax

```
int Mpg24Net_ClientStartSave(
     int hHandle,
     char *ServerIp,
     short port,
     char *Filename
);
```

Parameters

`hHandle`

Specifies the handle of the client instance.

`ServerIp`

Specifies the IP address in network order (bytes ordered from left to right). For unicast, the IP address will be the IP address of the server. For multicast, the IP address is same as the parameter 'ServerIp' of Mpg24Net_ServerOpen().

`Port`

Specifies the port number that is same as the parameter 'port' of Mpg24Net_ServerOpen().

`Filename`

Specifies the media file name. Usually the file extension of MPEG-4 and Motion JPEG is 'avi'.

Note: We don't support MPEG-2 subtype for file saving.

4.4.11 Mpg24Net_ClientStop

Stop a client. The client will close a network socket and terminate the server connection (unicast) or leave the network (multicast).

Syntax

```
int Mpg24Net_ClientStop(
     int hHandle
);
```

Parameters

`hHandle`

Specifies the handle of the client instance.

`Return Values`

See chapter 4.5.

4.4.12 Mpg24Net_ClientPreviewShow

This function can be used to show or hide the preview graph.

Syntax

```
int Mpg24Net_ClientPreviewShow(
     int hHandle,
     short Visible
);
```

Parameters

`hHandle`

Specifies the handle of the client instance.

`Visible`

Specifies whether the preview graph is shown. It could be:

0: Invisible

1: Visible

Return Values

See chapter 4.5.

4.5 Error Codes

`Mpg24Net_GetLastErrorInfo`

Use this function to get an error message of last error.

Syntax

```
Int Mpg24Net_GetLastErrorInfo(
     TCHAR *ErrorInfo

);
```

Parameters

`ErrorInfo`

Specifies the pointer of a text buffer for storing the error message. Users need to allocate a buffer space enough to get this error message (max. 256 bytes).

All error codes are listed below:

Error Name	Value	Description
S_OK	0	Success
Mpg24Net_INSUFFICIENT_MEMORY	-1	Insufficient memory
Mpg24Net_INVALID_HANDLE	-2	Invalid handle number
Mpg24Net_INSTANCE_NOT_OPEN	-3	Instance has not been opened
Mpg24Net_INSTANCE_EXCEED	-4	Exceeds the maximum instance
Mpg24Net_INVALID_PORT	-5	Invalid port number
Mpg24Net_FILTER_NOT_REGISTER	-6	dsnet.ax or dsnet_mcast.ax has not been registered
Mpg24Net_INTERFACE_ERROR	-7	The interface of ADLINK Unicast/Multicast Sender/ Receiver error
Mpg24Net_DECODER_NOT_INSTALL	-8	DivX decoder, MJPEG decompressor, Microsoft MPEG-4 video decompressor, or MPEG-2 decoder has not been installed

Error Name	Value	Description
Mpg24Net_DIRECTX_ERROR	-9	An error message comes form DirectX
Mpg24Net_INVALID_PARAMETER	-10	Input parameters exceed allowable range
Mpg24Net_NOT_STOPPED	-11	Invalid operation while running

Warranty Policy

Thank you for choosing ADLINK. To understand your rights and enjoy all the after-sales services we offer, please read the following carefully.

1. Before using ADLINK's products please read the user manual and follow the instructions exactly. When sending in damaged products for repair, please attach an RMA application form which can be downloaded from: http://rma.adlinktech.com/policy/.
2. All ADLINK products come with a limited two-year warranty, one year for products bought in China:
   - The warranty period starts on the day the product is shipped from ADLINK's factory.
   - Peripherals and third-party products not manufactured by ADLINK will be covered by the original manufacturers' warranty.
   - For products containing storage devices (hard drives, flash cards, etc.), please back up your data before sending them for repair. ADLINK is not responsible for any loss of data.
   - Please ensure the use of properly licensed software with our systems. ADLINK does not condone the use of pirated software and will not service systems using such software. ADLINK will not be held legally responsible for products shipped with unlicensed software installed by the user.
   - For general repairs, please do not include peripheral accessories. If peripherals need to be included, be certain to specify which items you sent on the RMA Request & Confirmation Form. ADLINK is not responsible for items not listed on the RMA Request & Confirmation Form.

3. Our repair service is not covered by ADLINK's guarantee in the following situations:
   - Damage caused by not following instructions in the User's Manual.
   - Damage caused by carelessness on the user's part during product transportation.
   - Damage caused by fire, earthquakes, floods, lightening, pollution, other acts of God, and/or incorrect usage of voltage transformers.
   - Damage caused by unsuitable storage environments (i.e. high temperatures, high humidity, or volatile chemicals).
   - Damage caused by leakage of battery fluid during or after change of batteries by customer/user.
   - Damage from improper repair by unauthorized ADLINK technicians.
   - Products with altered and/or damaged serial numbers are not entitled to our service.
   - This warranty is not transferable or extendible.
   - Other categories not protected under our warranty.
4. Customers are responsible for shipping costs to transport damaged products to our company or sales office.
5. To ensure the speed and quality of product repair, please download an RMA application form from our company website: http://rma.adlinktech.com/policy. Damaged products with attached RMA forms receive priority.

If you have any further questions, please email our FAE staff: service@adlinktech.com.