ADLINK
TECHNOLOGY INC.

PCI-MP4S

4-CH MPEG4 Software

Video Compression Card

User's Manual

Manual Rev.	2.00
Revision Date:	January 7, 2005
Part No:	50-15029-101

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

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

List of Figures

1 Introduction

The PCI-MP4S is a MPEG4 software video compression development kit that combines excellent real-time image acquisition and advanced MPEG4 software video compression algorithm for security and remote video surveillance applications. This 32-bit, 33MHz/66MHz PCI bus frame grabber simultaneously captures four video analog streams in real time.

It accepts standard composite color (PAL, NTSC) or monochrome video format (CCIR, EIA) camera input. The resolution can be programmed (640 x 480 or 768 x 576) and scaled down before images are transferred to PC memory.

The MPEG4 software video compression function library provides high quality video encoding and decoding. Image quality and bitrate are adjustable for more efficient data transmission via TCP/IP. The PCI-MP4S also has a sensitive partial or whole image motion detection area for smart video encoding or alarm signaling. Up to 4X image expansion technology for enhanced decoded image quality is available.

Each PCI-MP4S board has a unique hardware ID number. System integrators can use this to implement protection to lock their system product. Other useful features are a watchdog timer for fault-tolerant applications and easy-to-use standard connectors.

1.1 Features

Image Acquisition

Acquisition speed

NTSC	1 Camera	2 Cameras	3 Cameras	4 Cameras	8 Cameras
Fields	60	120	180	240	240
Frames	30	60	90	120	120
PAL	**1 Camera**	**2 Cameras**	**3 Cameras**	**4 Cameras**	**8 Cameras**
Fields	50	100	150	200	200
Frames	25	50	75	100	100

Table 1-1: Acquisition Speed

Color Image: The color video format is compatible with the following composite video input formats: NTSC-M, NTSC-Japan, PCL-B, PAL-D, PAL-G, PAL-H, PAL-I, PAM-M, PAL-N, and SECAM

Monochrome Image: The monochrome video acquisition is compatible with CCIR and EIA (RS-170).

Optional scaling: Optional scaling of acquired image or portions of an image is available as follows:

- Acquisition of a programmable area of interest.
- Scaling of the image (down to 1:16).
- Adjustment of hue (for NTSC signals), contrast (0 to 200%), brightness, and saturation (0 to 200% for U and V signals).
- Automatic chrominance gain control.

MPEG4 Encoding

Quality Index:

Quality Level	NTSC		
	4CIF (640 x 480)	**CIF (320 x 240)**	**QCIF (160 x 120)**
Lowest	Bit-rate = 400000*4 Frame rate = 5	Bit-rate = 400000 Frame rate = 5	Bit-rate = 400000/4 Frame rate = 5
Low	Bit-rate = 480000*4 Frame rate = 10	Bit-rate = 480000 Frame rate = 10	Bit-rate = 480000/4 Frame rate = 10
Normal	Bit-rate = 560000*4 Frame rate =15	Bit-rate = 560000 Frame rate =15	Bit-rate = 560000/4 Frame rate =15
High	Bit-rate = 560000*4 Frame rate = 30	Bit-rate = 560000 Frame rate = 30	Bit-rate = 560000/4 Frame rate = 30
Highest	Bit-rate = 1024000*4 Frame rate = 30	Bit-rate = 1024000 Frame rate = 30	Bit-rate = 1024000/4 Frame rate = 30
Quality Level	**PAL**		
	4CIF (768 x 576)	**CIF (384 x 288)**	**QCIF (192 x 144)**
Lowest	Bit-rate = 400000*4 Frame rate = 4	Bit-rate = 400000 Frame rate = 4	Bit-rate = 400000/4 Frame rate = 4
Low	Bit-rate = 480000*4 Frame rate = 8	Bit-rate = 480000 Frame rate = 8	Bit-rate = 480000/4 Frame rate = 8
Normal	Bit-rate = 560000*4 Frame rate =12	Bit-rate = 560000 Frame rate =12	Bit-rate = 560000/4 Frame rate =12
High	Bit-rate = 560000*4 Frame rate = 25	Bit-rate = 560000 Frame rate = 25	Bit-rate = 560000/4 Frame rate = 25
Highest	Bit-rate = 1024000*4 Frame rate = 25	Bit-rate = 1024000 Frame rate = 25	Bit-rate = 1024000/4 Frame rate = 25

Table 1-2: Quality Index

Supports setting of initial motion detection area and assigning or creating directory for saved files.

Save video file: Supports save continued video to M4V or AVI video file format. Users can play AVI files using MS Media Player after installing XVID CODEC (see ADLINK All-in-One CD).

Save single image file: Supports save single image to BMP or JPEG image file format.

MPEG4 Decoding

MPEG4 video decoding: Video decoding can be assigned a source from the local memory buffer, file access, or TCP/IP port. The quality of decoded video can be used to adjust the video encoding level. Decoding quality can also be used to control the flow rate between encoder and decoder and to adjust motion detection settings.

Save video file: Supports save continued video to M4V or AVI video file format, User may play AVI file by MS Media Player after installing XVID CODEC (see ADLINK All-in-One CD).

Save single image file: Supports save single image to BMP or JPEG image file format.

Motion Detection

Up to four detection areas can be set in one frame or the entire frame can be used for motion detection actions. The motion detection occurrence can be adjusted for sensitivity.

TCP/IP Data Transmission

Video data can be transferred by TCP/IP after connecting to the IP of the encoding site and data transmission speed can be monitored using the flow rate.

Watch Dog Timer

A hardware watchdog is available on the PCI-MP4S and is able to monitor PC application operation and will automatically reset the PC after a programmable inactivity time-out. This ensures reliable operation of remote systems.

I/O Lines

The PCI-MP4S is fitted with TTL compatible I/O lines protected against overloads and electrostatic discharges. Each line may be configured for input or output. They can be used to trigger acquisitions or to report alarm signals.

Supported Software

WDM driver - Supports VC++/VB/BCB/Delphi programming under Windows NT/98/2000/XP platforms with DLL.

ViewCreator - This package will assist in initial testing and functional evaluation.

1.2 Applications

- PC Based Surveillance System
- Digital Video Recorder (DVR)
- Factory Monitoring System
- Machine Vision Inspection System
- Scientific Research Instrumentation
- Medical Research Instrumentation

1.3 System requirement

For real-time* to compression 4-CH color MPEG4 video requirement, the PCI-MP4S minimum system requirement are as follows:

- Platform: Pentium 4, 2.4GHz CPU, 256MB DDRAM above.
- VGA display: AGP 4X above (No recommend VIA or SiS VGA chipset solution).
- Display setting: 800 x 600 above resolution, 16-bit above color format.
- OS: if OS uses Windows 2000 then please upgrade to Service Pack 4.0 or above.
- The CPU consumption average around 70% when based on the above system configuration.
- Less system configuration will lower acquisition performance.

* Real-time MPEG4 color video – Provides 320 x 240 pixels image resolution at RGB 16-bit color format, each channel acquired and compress speed at 30 frames per second and 4-CH with a total of up to 120 frames per second.

1.4 Compression Benchmark

PCI-33 Platform

- SBC: ADLINK NuPRO-842
- CPU: Intel® Pentium® 4 processor, 2.4HHz
- Memory: DDR266 256MB
- PCI Bus: 32-bit, 33MHz
- VGA: AGP 4X
- OS: Windows 2000/SP4
- Encoding color format: RGB 16-bit

Video Format: QCIF (160*120)

# Port Encoding	**Quality**	**CPU loading (max. %)**	**Decoding**	
			Microsoft Media Player	ADLINK M4V Player
1	Lowest	11	OK	OK
	Low	13	OK	OK
	Normal	16	OK	OK
	High	17	OK	OK
	Highest	20	OK	OK
2	Lowest	13	OK	OK
	Low	17	OK	OK
	Normal	17	OK	OK
	High	25	OK	OK
	Highest	23	OK	OK
3	Lowest	13	OK	OK
	Low	17	OK	OK
	Normal	20	OK	OK
	High	33	OK	OK
	Highest	30	OK	OK
4	Lowest	16	OK	OK
	Low	17	OK	OK
	Normal	22	OK	OK
	High	39	OK	OK
	Highest	34	OK	OK
8	Lowest	17	OK	OK
	Low	30	OK	OK
	Normal	39	OK	OK
	High	61	OK	OK
	Highest	61	OK	OK

Table 1-3: Video Format: QCIF (160*120)

12	Lowest	34	OK	OK
	Low	42	OK	OK
	Normal	61	OK	OK
	High	91	OK	OK
	Highest	91	OK	OK
13	Lowest	45	OK	OK
	Low	58	OK	OK
	Normal	72	OK	OK
	High	100*	-	-
	Highest	100*	-	-

Table 1-3: Video Format: QCIF (160*120)

* When CPU loading up to 100% then will start have time-lapse effect.

Video Format: CIF(320*240)

# Port Encoding	Quality	CPU loading (max. %)	Decoding	
			Microsoft Media Player	ADLINK M4V Player
1	Lowest	9	OK	OK
	Low	14	OK	OK
	Normal	17	OK	OK
	High	30	OK	OK
	Highest	28	OK	OK
2	Lowest	16	OK	OK
	Low	20	OK	OK
	Normal	31	OK	OK
	High	53	OK	OK
	Highest	55	OK	OK
3	Lowest	22	OK	OK
	Low	31	OK	OK
	Normal	44	OK	OK
	High	80	OK	OK
	Highest	80	OK	OK
4	Lowest	32	OK	OK
	Low	42	OK	OK
	Normal	56	OK	OK
	High	98	OK	OK
	Highest	95	OK	OK
5	Lowest	35	OK	OK
	Low	53	OK	OK
	Normal	73	OK	OK
	High	100*	-	-
	Highest	100*	-	-

Table 1-4: Video Format: CIF(320*240)

* When CPU loading up to 100% then will start have time-lapse effect.

Video Format: 4CIF(640*480)

# Port Encoding	Quality	CPU loading (max. %)	Decoding	
			Microsoft Media Player	ADLINK M4V Player
1	Lowest	22	OK	OK
	Low	39	OK	OK
	Normal	50	OK	OK
	High	86	OK	OK
	Highest	88	OK	OK
2	Lowest	42	OK	OK
	Low	63	OK	OK
	Normal	67	OK	OK
	High	100*	-	-
	Highest	100*	-	-

Table 1-5: Video Format: 4CIF(640*480)

* When CPU loading up to 100% then will start have time-lapse effect.

PCI-X Platform

- SBC: ADLINK NuPRO850
- CPU: Intel Pentium 4, Hyper Threading Disable
- Memory: DDR266 1GB
- PCI-X Bus: 32-bit, 66MHz
- VGA: AGP 8X
- OS: Windows 2000/SP4
- Encoding color format: RGB 16-bit

Video Format: QCIF (160*120)

# Port Encoding	Quality	CPU loading (max. %)	Decoding	
			Microsoft Media Player	ADLINK M4V Player
1	Lowest	5	OK	OK
	Low	6	OK	OK
	Normal	6	OK	OK
	High	8	OK	OK
	Highest	8	OK	OK
2	Lowest	6	OK	OK
	Low	8	OK	OK
	Normal	8	OK	OK
	High	13	OK	OK
	Highest	11	OK	OK
3	Lowest	6	OK	OK
	Low	8	OK	OK
	Normal	11	OK	OK
	High	16	OK	OK
	Highest	17	OK	OK
4	Lowest	8	OK	OK
	Low	9	OK	OK
	Normal	13	OK	OK
	High	22	OK	OK
	Highest	22	OK	OK
8	Lowest	11	OK	OK
	Low	17	OK	OK
	Normal	22	OK	OK
	High	39	OK	OK
	Highest	38	OK	OK

Table 1-6: Video Format: QCIF (160*120)

12	Lowest	18	OK	OK
	Low	25	OK	OK
	Normal	33	OK	OK
	High	58	OK	OK
	Highest	55	OK	OK
16	Lowest	28	OK	OK
	Low	45	OK	OK
	Normal	61	OK	OK
	High	87	OK	OK
	Highest	86	OK	OK

Table 1-6: Video Format: QCIF (160*120)

Video Format: CIF(320*240)

# Port Encoding	Quality	CPU loading (max. %)	Decoding	
			Microsoft Media Player	ADLINK M4V Player
1	Lowest	9	OK	OK
	Low	13	OK	OK
	Normal	16	OK	OK
	High	25	OK	OK
	Highest	26	OK	OK
2	Lowest	13	OK	OK
	Low	17	OK	OK
	Normal	23	OK	OK
	High	34	OK	OK
	Highest	34	OK	OK
3	Lowest	16	OK	OK
	Low	22	OK	OK
	Normal	27	OK	OK
	High	45	OK	OK
	Highest	47	OK	OK
4	Lowest	19	OK	OK
	Low	28	OK	OK
	Normal	36	OK	OK
	High	67	OK	OK
	Highest	67	OK	OK
5	Lowest	30	OK	OK
	Low	33	OK	OK
	Normal	48	OK	OK
	High	88	OK	OK
	Highest	88	OK	OK

Table 1-7: Video Format: CIF(320*240)

6	Lowest	29	OK	OK
	Low	38	OK	OK
	Normal	52	OK	OK
	High	92	OK	OK
	Highest	98	OK	OK
7	Lowest	33	OK	OK
	Low	44	OK	OK
	Normal	59	OK	OK
	High	100*	-	-
	Highest	100*	-	-

Table 1-7: Video Format: CIF(320*240)

* When CPU loads up to 100%, the time-lapse effect would commence.

Video Format: 4CIF(640*480)

# Port Encoding	Quality	CPU loading (max. %)	Decoding	
			Microsoft Media Player	ADLINK M4V Player
1	Lowest	19	OK	OK
	Low	31	OK	OK
	Normal	41	OK	OK
	High	67	OK	OK
	Highest	67	OK	OK
2	Lowest	37	OK	OK
	Low	52	OK	OK
	Normal	70	OK	OK
	High	98	OK	OK
	Highest	98	OK	OK
3	Lowest	52	OK	OK
	Low	70	OK	OK
	Normal	91	OK	OK
	High	100*	-	-
	Highest	100*	-	-

Table 1-8: Video Format: 4CIF(640*480)

* When CPU loads up to 100%, the time-lapse effect would commence.

2 Hardware Reference

2.1 PCI-MP4S Specification

Video Input

Four composite video color digitizers

Video input interface: four composite BNC connectors

Coaxial cable recommended

Channel Extension

Expandable up to 16 channels

Channel extension interface:

- 10-pin ribbon cable to onboard 10-pin header connector for channel extension, each header adds four video inputs channels
- Three 10-pin header connectors onboard

General Purpose I/O Lines

All I/Os are TTL compatible and support four inputs, four outputs, and four soft trigger lines

GPIO interface:

- Two 10-pin header connectors onboard
- The I/O lines are internally pulled up:

Voltage	MIN	MAX
Input high voltage (5µA)	2.0V	5.25V
Input low voltage (-5µA)	0.0V	0.80V
Output high voltage (-1.0mA)	5.0V	-
Output low voltage (100.0mA)	-	0.5V

Table 2-1: General Purpose I/O Lines

Watch Dog Timer

- For monitoring applications – will reset the PC after a programmable inactivity time-out.
- Interface: 2-pin header

4-channel software trigger output

4-channel programmable trigger scale (60μs – 16ms)

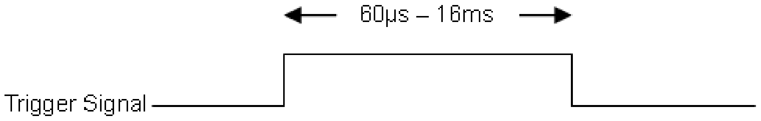


Figure 2-1: 4-channel software trigger output

User EEPROM

Includes 1kbit available EEPROM

Form Factor

32-bit, 33/66MHz PCI half-size board

PCI-MP4S Appearance

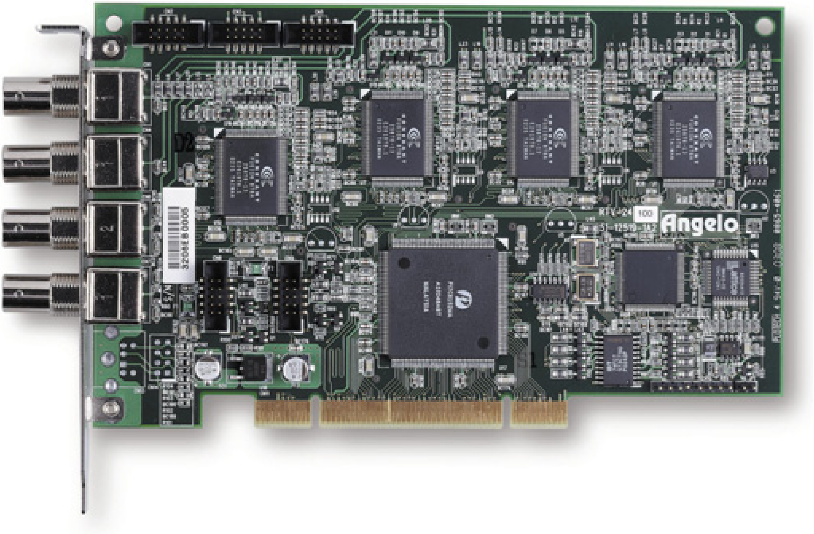


Figure 2-2: PCI-MP4S Appearance

PCI-MP4S Standard Accessories

- Watch dog reset cable
- GPIO bracket
- User's Manual
- All in One CD

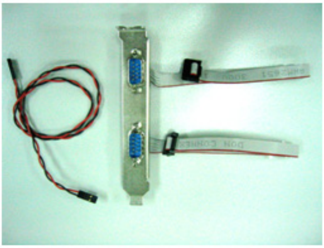


Figure 2-3: Watch dog reset cable, GPIO bracket, All in One CD

PCI-MP4S Connectors and Pin Definitions

Video Inputs

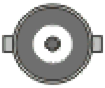
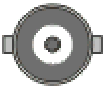
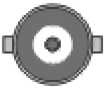
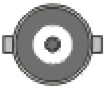
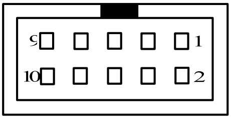
Connector	Definition
	Video IN – CH 0
	Video IN – CH 1
	Video IN – CH 2
	Video IN – CH 3

Table 2-2: Video Inputs

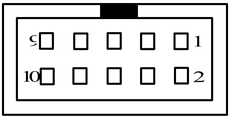
Channel Extension Video Input (CN2)



Pin	Function	Pin	Function
1	GND	2	CH4 video in
3	CH5 video in	4	GND
5	GND	6	CH6 video in
7	CH7 video in	8	GND
9	GND	10	GND

Table 2-3: Channel Extension Video Input (CN2)

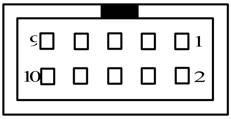
Channel Extension Video Input (CN3)



Pin	Function	Pin	Function
1	GND	2	CH8 video in
3	CH9 video in	4	GND
5	GND	6	CH10 video in
7	CH11 video in	8	GND
9	GND	10	GND

Table 2-4: Channel Extension Video Input (CN3)

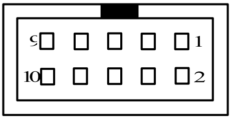
Channel Extension Video Input (CN5)



Pin	Function	Pin	Function
1	GND	2	CH12 video in
3	CH13 video in	4	GND
5	GND	6	CH14 video in
7	CH15 video in	8	GND
9	GND	10	GND

Table 2-5: Channel Extension Video Input (CN5)

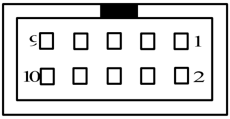
GPIO (CN8)



Pin	Function	Pin	Function
1	IN0 (External interrupt)	2	GND
3	OUT0	4	Software Trigger 0
5	IN1 (External interrupt)	6	Software Trigger 1
7	OUT1	8	+5V
9	GND	10	--

Table 2-6: GPIO (CN8)

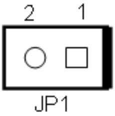
GPIO (CN9)



Pin	Function	Pin	Function
1	IN2 (External interrupt)	2	GND
3	OUT2	4	Software Trigger 2
5	IN3 (External interrupt)	6	Software Trigger 3
7	OUT3	8	+5V
9	GND	10	--

Table 2-7: GPIO (CN9)

Watchdog Timer Reset



Pin	Function
1	System reset
2	GND

Table 2-8: Watchdog Timer Reset

2.2 RTV-E4 Extension board for RTV-24 and PCI-MP4S

(Optional item, not a standard accessory)

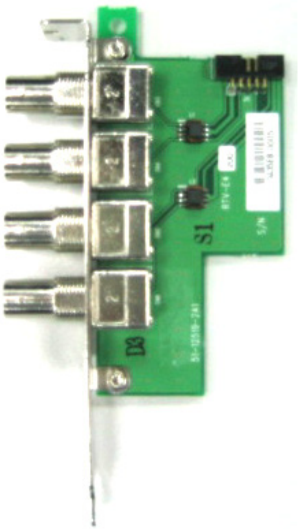
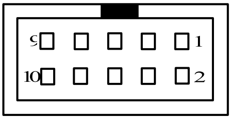


Figure 2-4: RTV-E4

RTV-E4 Connectors and Pin Definitions

Channel Extension Video Input (J11)



Pin	Function	Pin	Function
1	GND	2	CH4 video in
3	CH5 video in	4	GND
5	GND	6	CH6 video in
7	CH7 video in	8	GND
9	GND	10	GND

Table 2-9: RTV-E4 Connectors and Pin Definitions

2.3 RTV-I4 Isolation GPIO board for RTV-24 and PCI-MP4S

(Optional item, not a standard accessory)

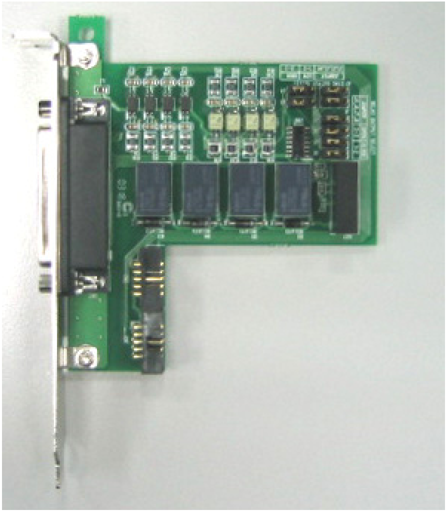
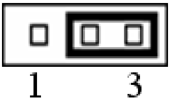
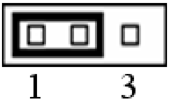


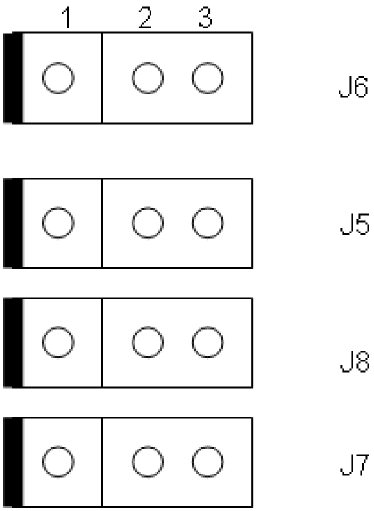
Figure 2-5: RTV-I4

RTV-I4 Connectors and Pin Definitions

Relay output signal select:

- Relay output types: Normal open or Normal closed
- Signal names: RY1, RY2, RY3, RY4
- Jumper addresses: J5, J6, J7, J8
- Type select: Normal open: 2-3, Normal close: 1-2

Normal Open	Normal Closed
	

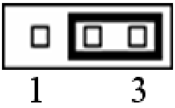
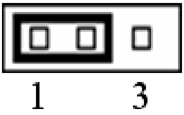


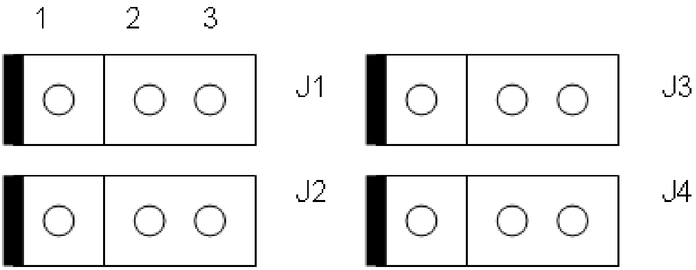
Relay I/O voltage requirement:

- Input: +5V to +24V
- Output: AC: 0.5A/125V, DC: 1A/30V or 0.3A/100V

STRG output signal select:

- STRG output signal types: Active high or Active low
- Signal names: STRG_OUT1, STRG_OUT2, STRG_OUT3, STRG_OUT4
- Jumper addresses: J1, J2, J3, J4
- Type select: Active high => 2-3
- Active low => 1-2

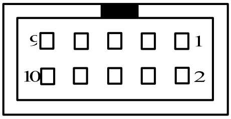
Active High	Active Low
	



Trigger output voltage: 0V to +5V

Input 2R10P pin header pin definition:

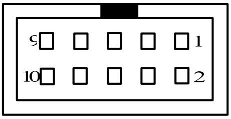
GPIO (CN1)



Pin	Function	Pin	Function
1	GPIO Input 1	2	GND
3	GPIO Output 1	4	PORT1 STRG Output
5	GPIO Input 2	6	PORT2 STRG Output
7	GPIO Output 2	8	VCC
9	GND	10	NC

Table 2-10: GPIO (CN1)

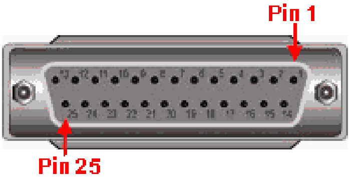
GPIO (CN2)



Pin	Function	Pin	Function
1	GPIO Input 3	2	GND
3	GPIO Output 3	4	PORT3 STRG Output
5	GPIO Input 4	6	PORT4 STRG Output
7	GPIO Output 4	8	VCC
9	GND	10	NC

Table 2-11: GPIO (CN2)

D-sub 25-pin output connector pin



Pin	Signal name	Pin	Signal name
1	DI1	14	RY3_COM
2	DI1_COM	15	RY4
3	DI2	16	RY4_COM
4	DI2_COM	17	STRG_OUT1
5	DI3	18	STRG_OUT2
6	DI3_COM	19	STRG_OUT3
7	DI4	20	STRG_OUT4
8	DI4_COM	21	STRG_GND
9	RY1	22	STRG_GNG
10	RY1_COM	23	NC
11	RY2	24	NC
12	RY2_COM	25	NC
13	RY3	26	

Table 2-12: D-sub 25-pin output connector pin

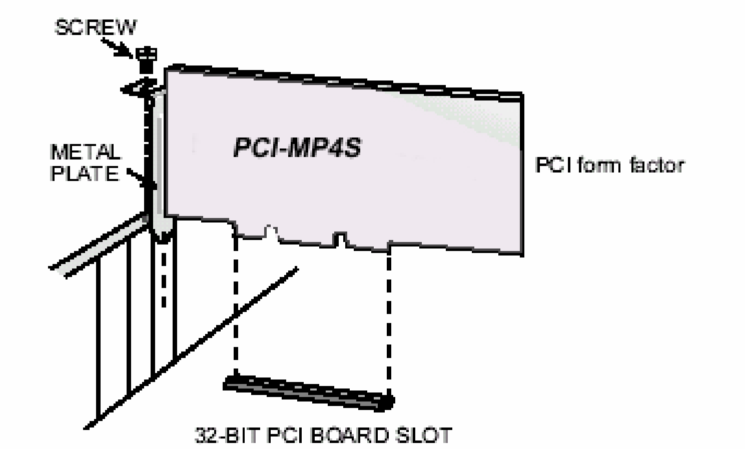
3 Installation Guide

3.1 Hardware Installation

PCI-MP4S

Use the following steps to install the PCI-MP4S board on the PCI bus:

1. Remove the computer cover using the instructions from the computer manual.
2. Check that there is an empty PCI (32-bit) slot to accommodate the board. If there is no empty slot, remove a PCI board from the computer to make room for the PCI-MP4S board and note the chosen slot number (i.e. card index).
3. Remove the blank metal plate located at the back of the selected slot (if any). Keep the removed screw to fasten the PCI-MP4S board after installation.
4. Carefully position the PCI-MP4S in the selected PCI slot as illustrated below. If using a tower computer, orient the board to suit the board slots.



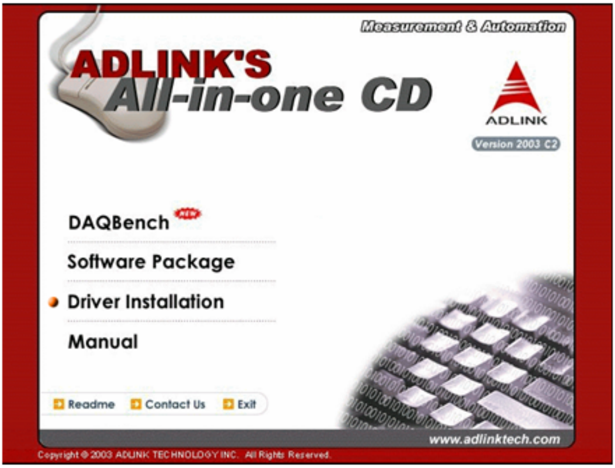
5. Once perfectly aligned with an empty slot, press the board firmly but carefully into the connector.
6. Anchor the board by replacing the screw.
7. Connect your video sources for image acquisition tests. For details, refer to the “ViewCreator” utility.
8. Turn on the computer. In some cases, when the computer boots up, the “Plug and Play” feature of Windows will detect the new PCI card eight times (four videos and four audios) and you will require drivers. For details, see the “Installation Guide.”

3.2 Driver Installation

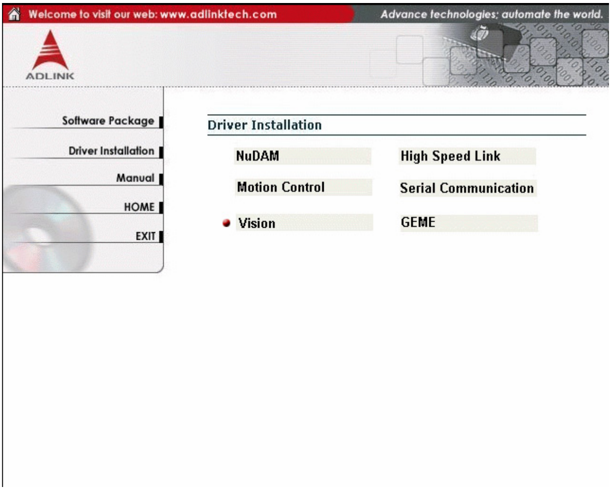
WDM Driver Installation

1. Insert the Automation All-in-one CD into the CD-ROM drive and click **Driver Installation**.

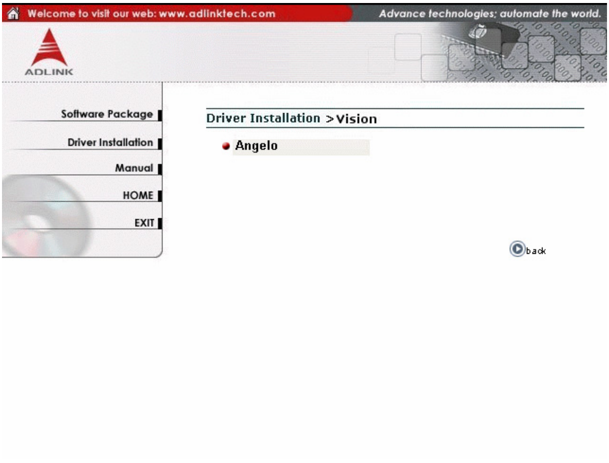
Note: Do not plug the hardware before installing the software driver.



2. Select **Vision**.



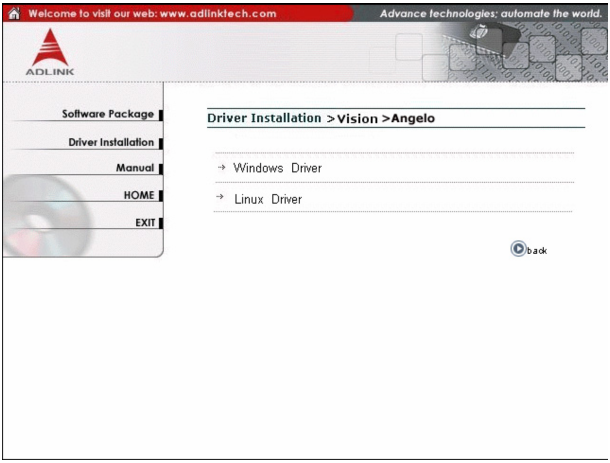
3. Click **Angelo**


Welcome to visit our web: www.adlinktech.com
Advance technologies; automate the world.
ADLINK
Software Package
Driver Installation
Manual
HOME
EXIT
back

Driver Installation >Vision

● Angelo

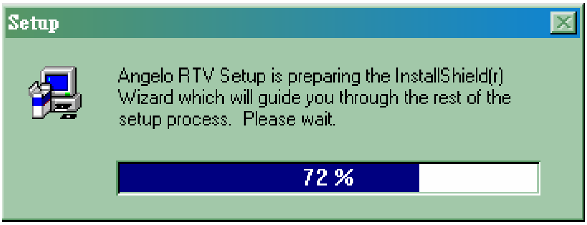
4. Select **Windows Driver** for Windows 98/NT/2000/XP.


Welcome to visit our web: www.adlinktech.com
Advance technologies; automate the world.
ADLINK
Software Package
Driver Installation
Manual
HOME
EXIT
back

Driver Installation >Vision >Angelo

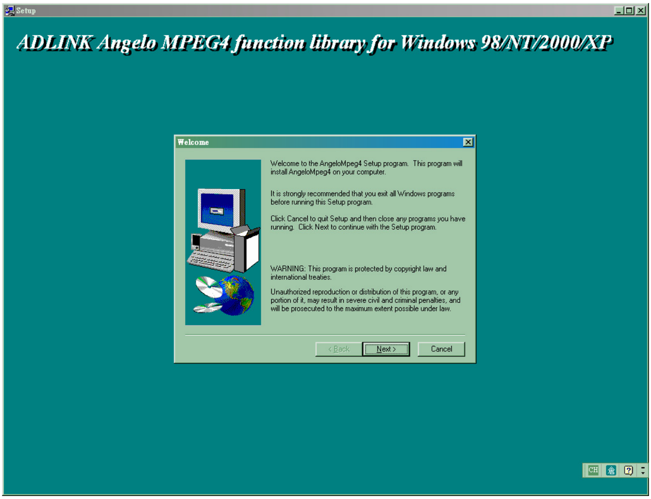
- Windows Driver
- Linux Driver

5. The driver will begin installing.



Angelo RTV Setup is preparing the InstallShield(r) Wizard which will guide you through the rest of the setup process. Please wait.

6. Click **Next** until the driver installs completely.


Setup
ADLINK Angelo MPEG4 function library for Windows 98/NT/2000/XP
Welcome

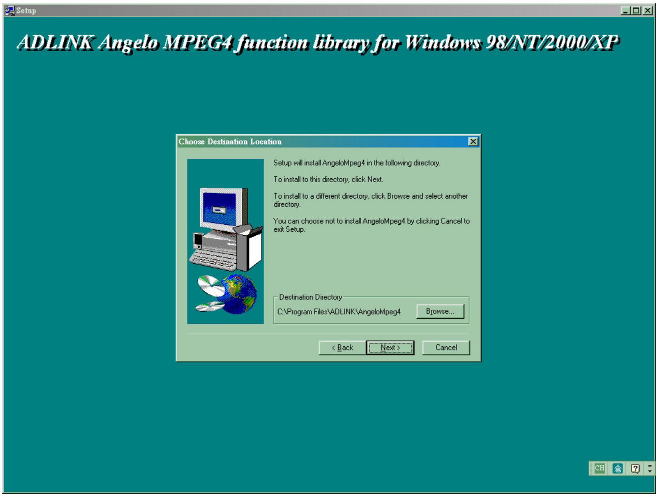
Welcome to the AngeloMpeg4 Setup program. This program will install AngeloMpeg4 on your computer.

It is strongly recommended that you exit all Windows programs before running this Setup program.

Click Cancel to quit Setup and then close any programs you have running. Click Next to continue with the Setup program.

WARNING: This program is protected by copyright law and international treaties.

Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.



ADLINK Angelo MPEG4 function library for Windows 98/NT/2000/XP

Choose Destination Location

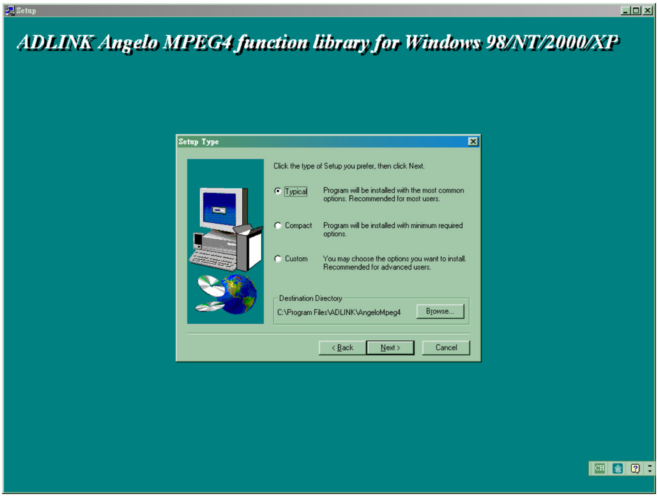
Setup will install AngeloMpeg4 in the following directory.

To install to this directory, click Next.

To install to a different directory, click Browse and select another directory.

You can choose not to install AngeloMpeg4 by clicking Cancel to exit Setup.

Destination Directory: C:\Program Files\ADLINK\AngeloMpeg4



ADLINK Angelo MPEG4 function library for Windows 98/NT/2000/XP

Setup Type

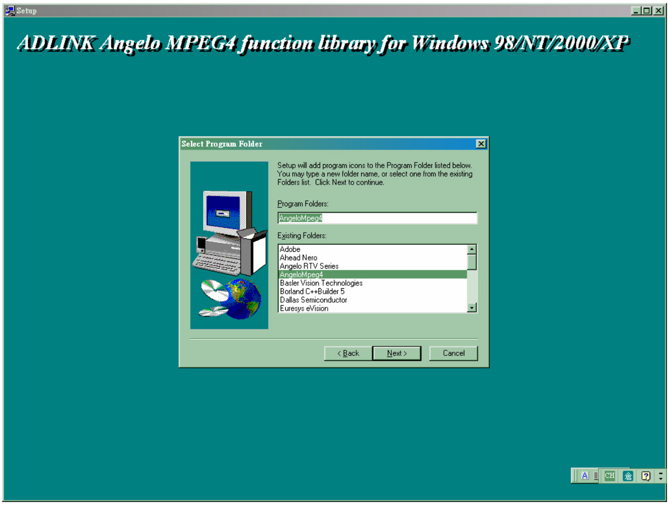
Click the type of Setup you prefer, then click Next.

● Typical — Program will be installed with the most common options. Recommended for most users.

○ Compact — Program will be installed with minimum required options.

○ Custom — You may choose the options you want to install. Recommended for advanced users.

Destination Directory: C:\Program Files\ADLINK\AngeloMpeg4



ADLINK Angelo MPEG4 function library for Windows 98/NT/2000/XP

Select Program Folder

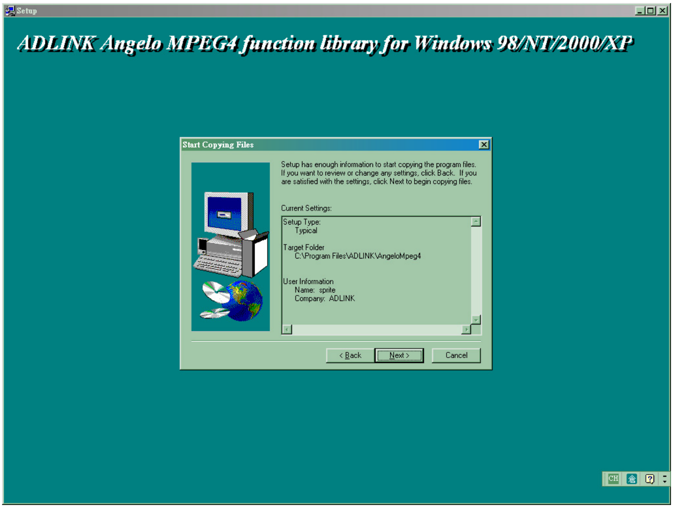
Setup will add program icons to the Program Folder listed below. You may type a new folder name, or select one from the existing Folders list. Click Next to continue.

Program Folders:

AngeloMpeg4

Existing Folders:

Adobe
Ahead Nero
Angelo RTV Series
AngeloMpeg4
Basler Vision Technologies
Borland C++Builder 5
Dallas Semiconductor
Euresys eVision



ADLINK Angelo MPEG4 function library for Windows 98/NT/2000/XP

Start Copying Files

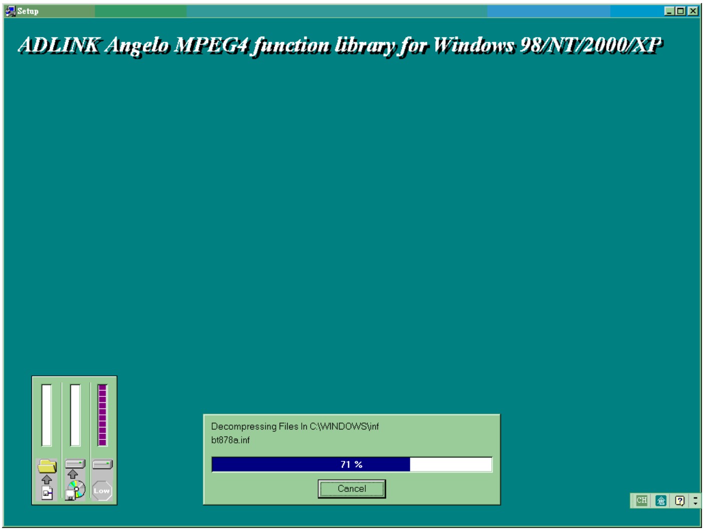
Setup has enough information to start copying the program files. If you want to review or change any settings, click Back. If you are satisfied with the settings, click Next to begin copying files.

Current Settings:

Setup Type:
    Typical

Target Folder
    C:\Program Files\ADLINK\AngeloMpeg4

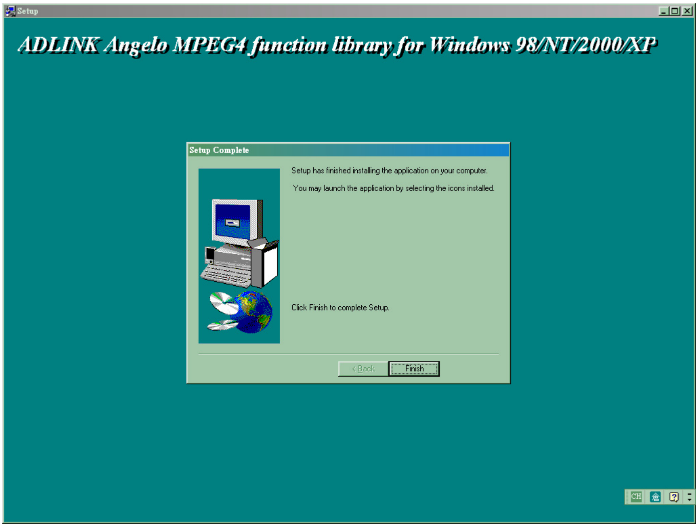
User Information
    Name: sprite
    Company: ADLINK



Setup
ADLINK Angelo MPEG4 function library for Windows 98/NT/2000/XP
Low
71 %
Cancel

Decompressing Files In C:\WINDOWS\inf
bt878a.inf

7. Click Finish and restart the system.



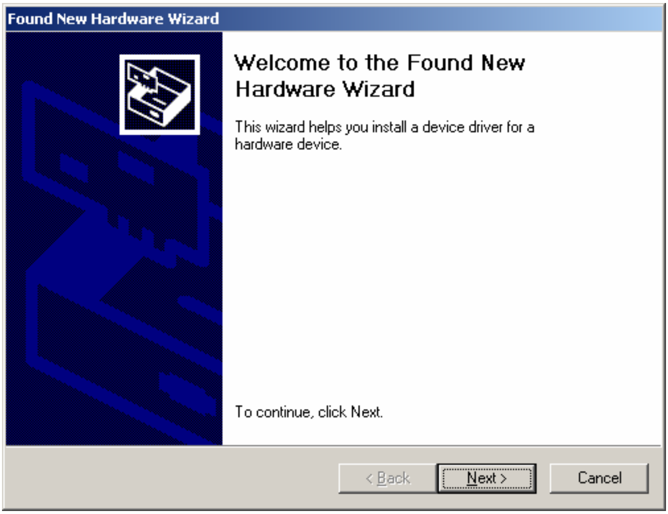
Setup
ADLINK Angelo MPEG4 function library for Windows 98/NT/2000/XP
Setup Complete
< Back
Finish

Setup has finished installing the application on your computer.

You may launch the application by selecting the icons installed.

Click Finish to complete Setup.

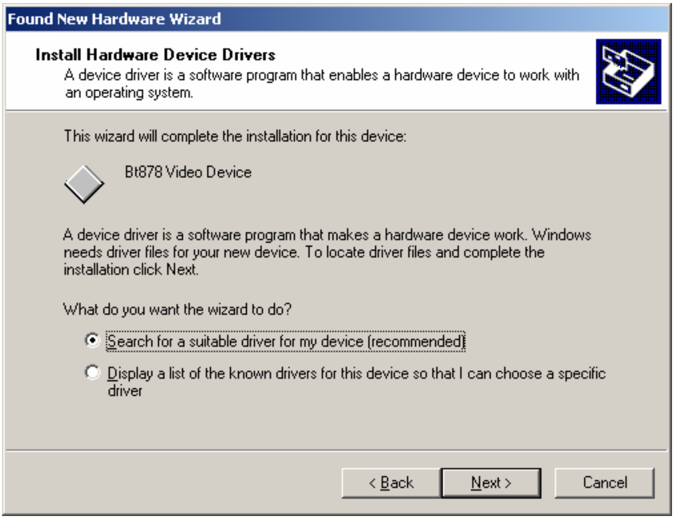
- The **Found New Hardware Wizard** window appears after system restarts. Click **NEXT** and follow the following steps to complete the new hardware wizard.



Welcome to the Found New Hardware Wizard

This wizard helps you install a device driver for a hardware device.

To continue, click Next.



Found New Hardware Wizard

Install Hardware Device Drivers

A device driver is a software program that enables a hardware device to work with an operating system.

This wizard will complete the installation for this device:

Bt878 Video Device

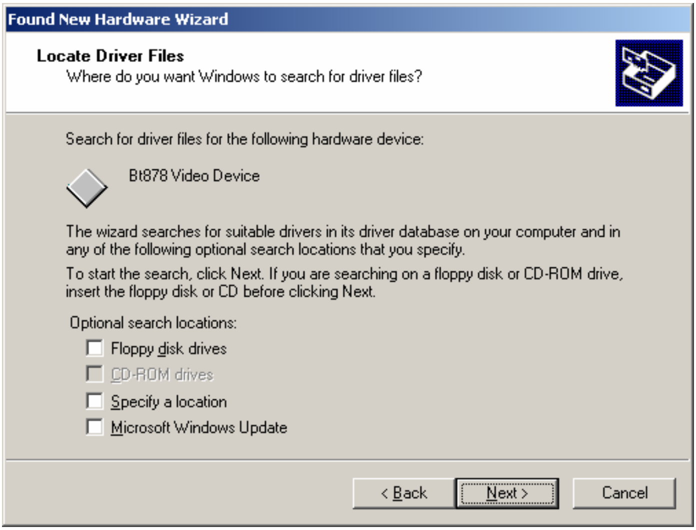
A device driver is a software program that makes a hardware device work. Windows needs driver files for your new device. To locate driver files and complete the installation click Next.

What do you want the wizard to do?

● Search for a suitable driver for my device (recommended)

○ Display a list of the known drivers for this device so that I can choose a specific driver

▷ Click **Next.**



Found New Hardware Wizard

Locate Driver Files

Where do you want Windows to search for driver files?

Search for driver files for the following hardware device:

Bt878 Video Device

The wizard searches for suitable drivers in its driver database on your computer and in any of the following optional search locations that you specify.

To start the search, click Next. If you are searching on a floppy disk or CD-ROM drive, insert the floppy disk or CD before clicking Next.

Optional search locations:

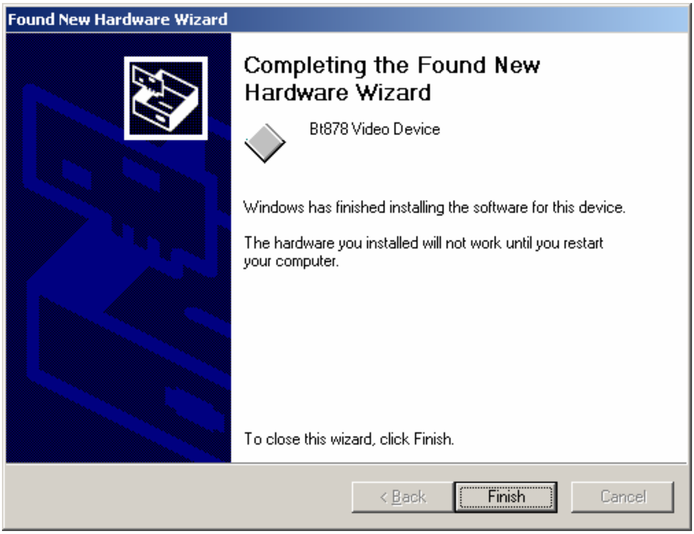
□ Floppy disk drives

□ CD-ROM drives

□ Specify a location

□ Microsoft Windows Update

▷ Click **Next.**



Found New Hardware Wizard

Completing the Found New Hardware Wizard

Bt878 Video Device

Windows has finished installing the software for this device.

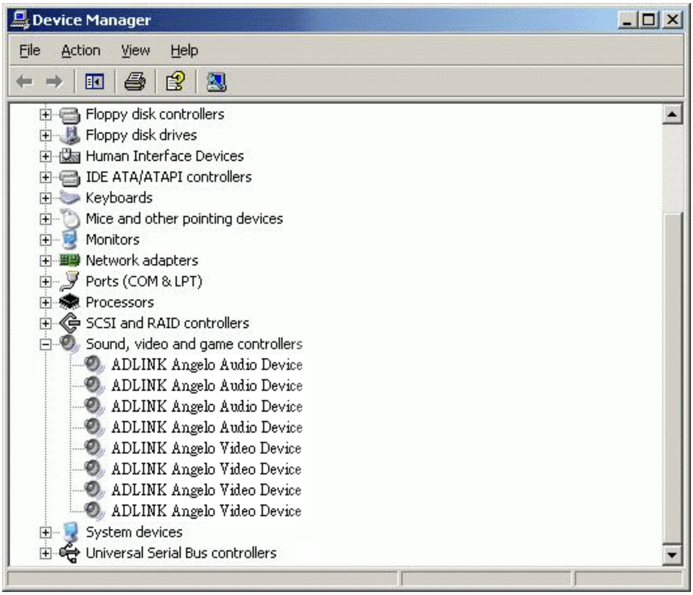
The hardware you installed will not work until you restart your computer.

To close this wizard, click Finish.

▷ Click **Finish**.

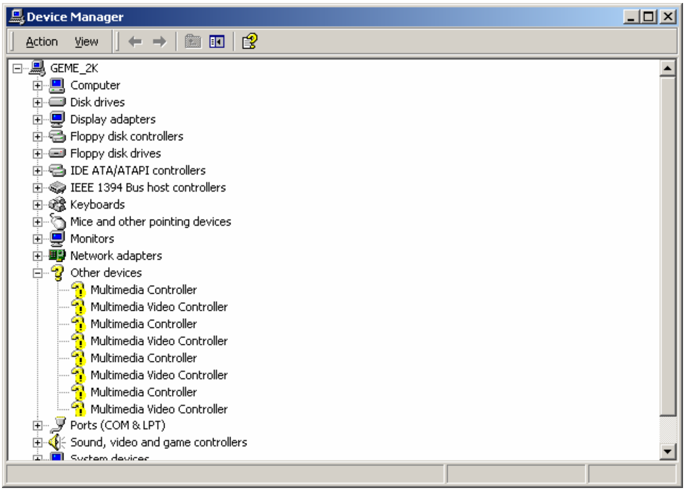
▶ Another Found New Hardware Wizard window appears when you finished the wizard. Repeat step 7.1 until you finished all wizards.

8. Go to the System Control Panel and check to see that four “ADLINK Angelo Audio Device” and four “ADLINK Angelo Video Device” are installed as shown.



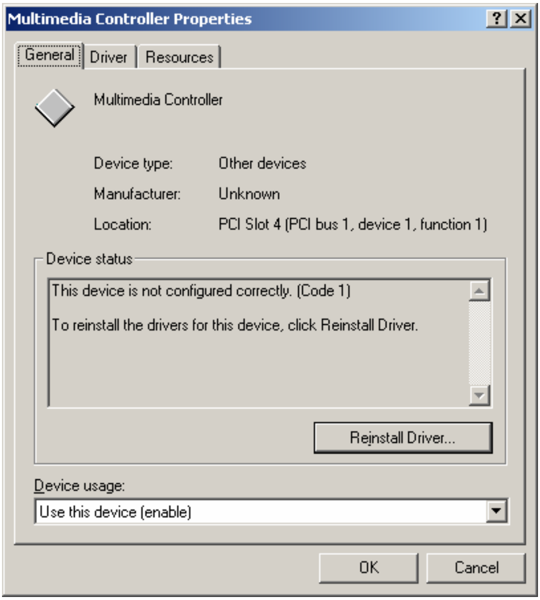
Floppy disk controllers
Floppy disk drives
Human Interface Devices
IDE ATA/ATAPI controllers
Keyboards
Mice and other pointing devices
Monitors
Network adapters
Ports (COM & LPT)
Processors
SCSI and RAID controllers
Sound, video and game controllers
ADLINK Angelo Audio Device
ADLINK Angelo Audio Device
ADLINK Angelo Audio Device
ADLINK Angelo Audio Device
ADLINK Angelo Video Device
ADLINK Angelo Video Device
ADLINK Angelo Video Device
ADLINK Angelo Video Device
System devices
Universal Serial Bus controllers

- If you see a yellow question mark appear in front of the new driver’s name, you need to setup the driver manually.



GEME_2K
Computer
Disk drives
Display adapters
Floppy disk controllers
Floppy disk drives
IDE ATA/ATAPI controllers
IEEE 1394 Bus host controllers
Keyboards
Mice and other pointing devices
Monitors
Network adapters
Other devices
Multimedia Controller
Multimedia Video Controller
Multimedia Controller
Multimedia Video Controller
Multimedia Controller
Multimedia Video Controller
Multimedia Controller
Multimedia Video Controller
Ports (COM & LPT)
Sound, video and game controllers
System devices

- Right click on the driver name, **Multimedia Controller** which is a audio device, then select **Properties** on the popup menu. Follow the steps below to complete the driver re-installment.



Multimedia Controller

Device type: Other devices

Manufacturer: Unknown

Location: PCI Slot 4 (PCI bus 1, device 1, function 1)

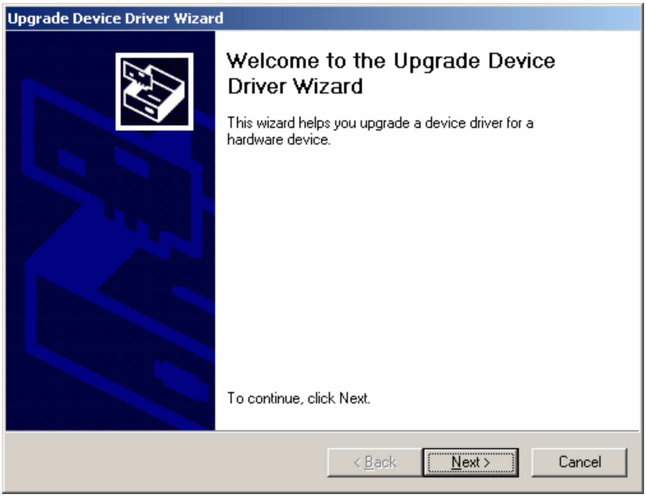
Device status

This device is not configured correctly. (Code 1)

To reinstall the drivers for this device, click Reinstall Driver.

Device usage: Use this device (enable)

▷ Click Reinstall Driver.



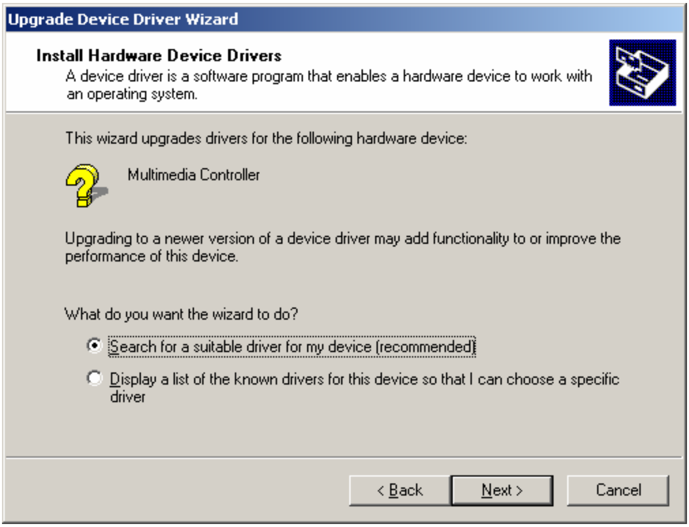
Upgrade Device Driver Wizard

Welcome to the Upgrade Device Driver Wizard

This wizard helps you upgrade a device driver for a hardware device.

To continue, click Next.

▷ Click Next.



Upgrade Device Driver Wizard

Install Hardware Device Drivers

A device driver is a software program that enables a hardware device to work with an operating system.

This wizard upgrades drivers for the following hardware device:

Multimedia Controller

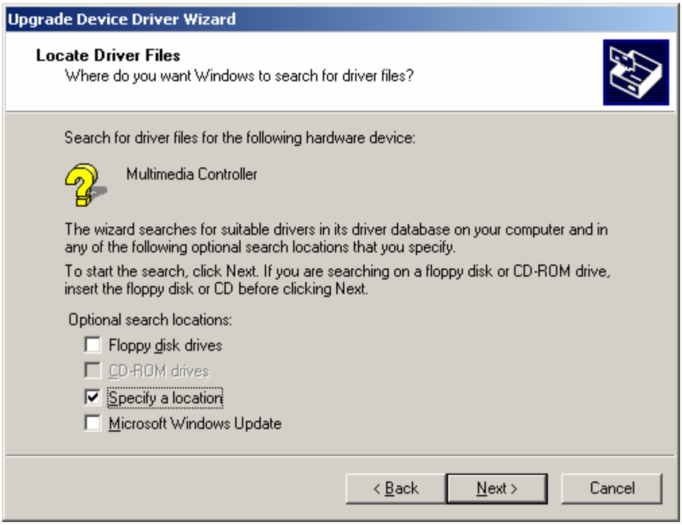
Upgrading to a newer version of a device driver may add functionality to or improve the performance of this device.

What do you want the wizard to do?

● Search for a suitable driver for my device (recommended)

○ Display a list of the known drivers for this device so that I can choose a specific driver

- ▷ Click Next.



Upgrade Device Driver Wizard

Locate Driver Files

Where do you want Windows to search for driver files?

Search for driver files for the following hardware device:

Multimedia Controller

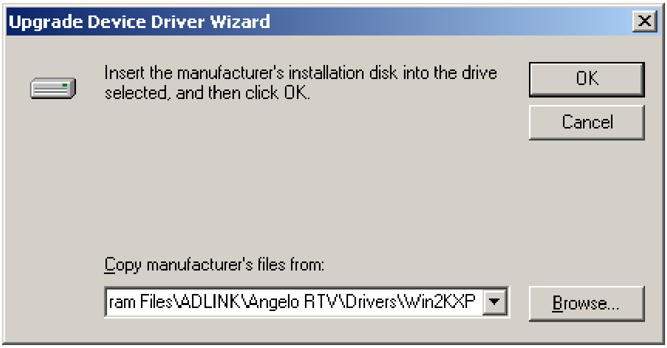
The wizard searches for suitable drivers in its driver database on your computer and in any of the following optional search locations that you specify.

To start the search, click Next. If you are searching on a floppy disk or CD-ROM drive, insert the floppy disk or CD before clicking Next.

Optional search locations:

- □ Floppy disk drives
- □ CD-ROM drives
- ☑ Specify a location
- □ Microsoft Windows Update

- ▷ Check Specify a location then click Next.



Upgrade Device Driver Wizard

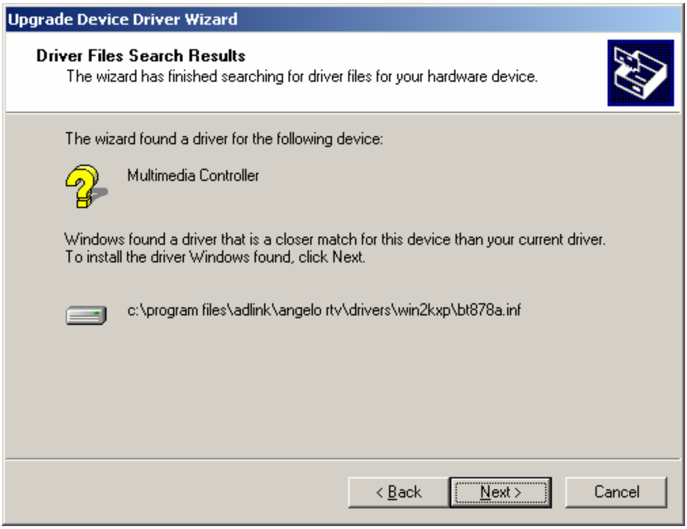
Insert the manufacturer's installation disk into the drive selected, and then click OK.

Copy manufacturer's files from:

ram Files\ADLINK\Angelo RTV\Drivers\Win2KXP

- ▷ In **Copy manufacture's files from**: text box insert the location of driver installed in step 6, for example, 'c:\Program Files\ADLINK\Angelo.RTV\Drivers\Win2KXP'.

Then click **OK**.



Upgrade Device Driver Wizard

Driver Files Search Results

The wizard has finished searching for driver files for your hardware device.

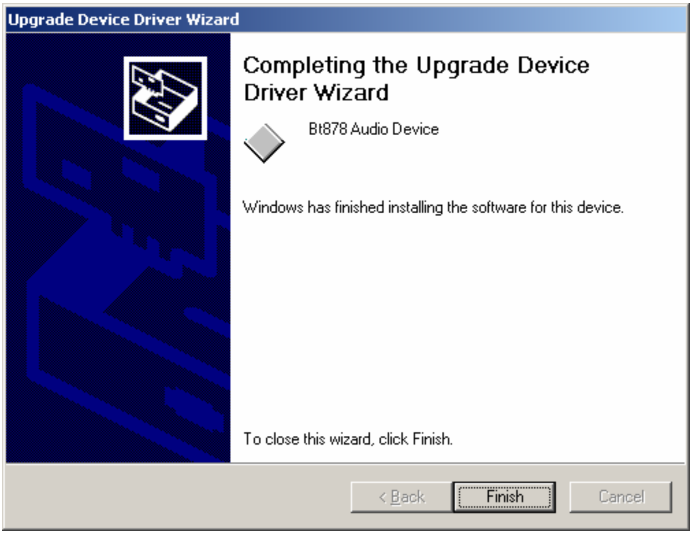
The wizard found a driver for the following device:

Multimedia Controller

Windows found a driver that is a closer match for this device than your current driver. To install the driver Windows found, click Next.

c:\program files\adlink\angelo rtv\drivers\win2kxp\bt878a.inf

▷ Click **Next**.



Upgrade Device Driver Wizard

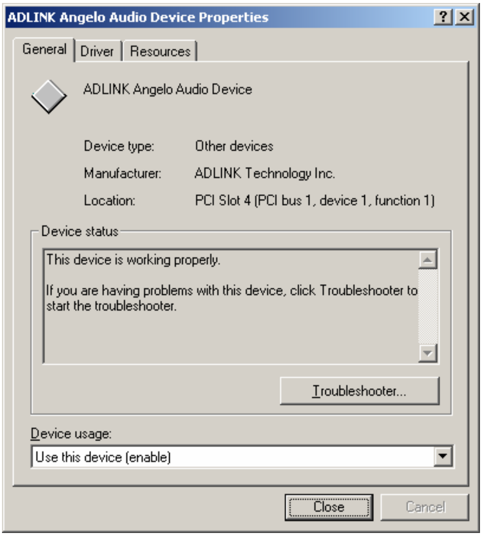
Completing the Upgrade Device Driver Wizard

Bt878 Audio Device

Windows has finished installing the software for this device.

To close this wizard, click Finish.

▷ Click **Finish** to complete this wizard.



ADLINK Angelo Audio Device

Device type: Other devices

Manufacturer: ADLINK Technology Inc.

Location: PCI Slot 4 (PCI bus 1, device 1, function 1)

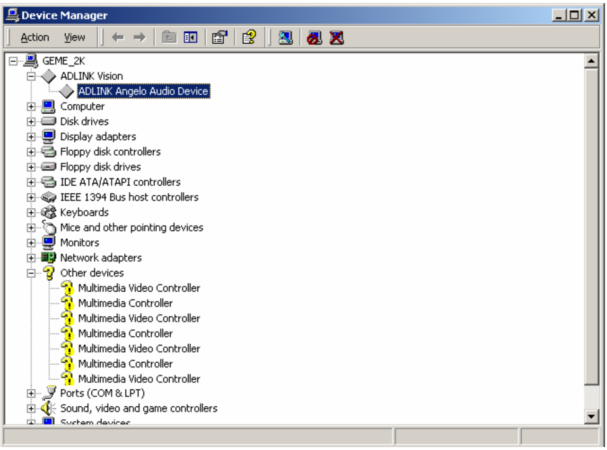
Device status

This device is working properly.

If you are having problems with this device, click Troubleshooter to start the troubleshooter.

Device usage: Use this device (enable)

▷ This device is working properly.



```
GEME_2K
  ADLINK Vision
    ADLINK Angelo Audio Device
  Computer
  Disk drives
  Display adapters
  Floppy disk controllers
  Floppy disk drives
  IDE ATA/ATAPI controllers
  IEEE 1394 Bus host controllers
  Keyboards
  Mice and other pointing devices
  Monitors
  Network adapters
  Other devices
    Multimedia Video Controller
    Multimedia Controller
    Multimedia Video Controller
    Multimedia Controller
    Multimedia Video Controller
    Multimedia Controller
    Multimedia Video Controller
  Ports (COM & LPT)
  Sound, video and game controllers
  System devices
```

▷ The yellow question mark disappears.

- Right click on the driver name, Multimedia Video Controller which is a video device. Repeat step 8.2 onwards.
- Repeat steps 8.2 and 8.3 until all audio devices and video devices are working properly.

Note: If the system prompts you to restart the computer, select No until all drivers are reinstalled, then restart the computer.

9. XVID CODEC

User must install the XVID CODEC in our setup disk in order to play ".avi" file in Microsoft Media Player.

4 ViewCreator Utility

Once hardware installation is complete, ensure that the system is correctly configured before running the ViewCreator utility. This chapter outlines how to set up a vision system and manually control Angelo series cards to verify correct operation. ViewCreator provides a simple yet powerful means to setup, configure, test, and debug the vision system.

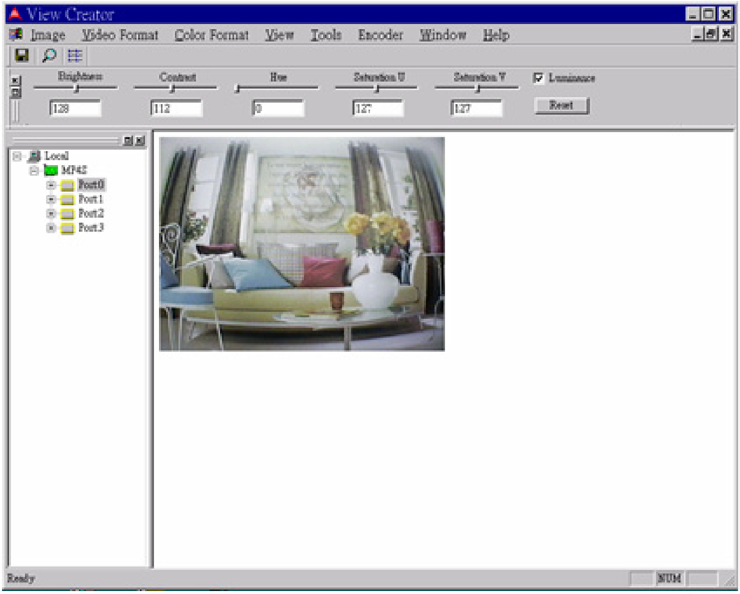
Note: ViewCreator is only available for Windows 98/NT/2k/XP with a recommended screen resolution of 800x600 or higher.

4.1 Overview

ViewCreator offers the following features:

1. 32-bit operation under Windows 98/2000/XP
2. Angelo series cards access and configuration
3. Video picture adjustments
4. MPEG4 video encoding
5. Recording (AVI video format)
6. Direct access to general purpose I/Os
7. FULL, CIF, or QCIF image size, 2x2 or 4x4 display
8. Software triggering

4.2 Component Description



Brightness 128

Contrast 112

Hue 0

Saturation U 127

Saturation V 127

☑ Luminance

Tree Browser

The Tree Browser window lists the PCI-MP4S cards and video ports available at the local computer.

Image View

The Image View window displays Full, CIF, and QCIF size images and image effects.

Control Panel

The control panel allows for making video adjustments, including brightness, hue, contrast, etc.

4.3 Operation Theory

ViewCreator provides many functions for the Angelo series card as described below.

MPEG4 Encoding

Single channel display

- ▶ Click a video Port icon in the Tree Browser window. A video frame will appear in the Image View window.
- ▶ Select Encoder->Encode in menu bar to bring up the Encoder Setting dialog box, then click the start button.

Note:

1. View Creator supports only one channel CIF video encoding. Ensure there is only one channel, CIF image on the screen.
2. Execute the decoder sample program in Program files->ADLINK->AngeloMPEG4->Samples to connect to the encoder (IP:127.0.0.1 for local computer)

Video image configuration

Video format

Click Format in the menu bar to select the format of the video camera. The supported video formats are NTSC, EIA, PAL, and CCIR.

Color format

The default color format setting in ViewCreator is RGB24. The color format of the application can be changed.

Video size

Click View in the menu bar and select the image size required. The supported video sizes are listed below:

- ▷ **FULL:** 640x480 for NTSC, EIA and 768x576 for PAL, CCIR
- ▷ **CIF:** 320x240 for NTSC, EIA and 384x288 for PAL, CCIR
- ▷ **QCF:** 160x120 for NTSC, EIA and 192x144 for PAL, CCIR

Video adjustments

Hue

Click and hold the left mouse button on the Hue slider of the Control Panel and drag the cursor to change its value. Values range from 0 to 255.

Contrast

Click and hold the left mouse button on the Contrast slider of the Control Panel and drag the cursor to change its value. Values range from 0 to 255.

Brightness

Click and hold the left mouse button on the Brightness slider of the Control Panel and drag the cursor to change its value. Values range from 0 to 255.

Save image file

This function can only be used in single channel display mode (select a video Port icon in the Tree Browser window).

TIF

Click Image in the menu bar and select **Save As** to bring up the Save As dialog box. Select the file location, TIF file format, enter the file name, and click **OK**.

BMP

Click **Image** in the menu bar and select **Save As** to bring up the Save As dialog box. Select the file location, BMP file format, enter the file name, and click **OK**.

Special image effect

Border

- Check the **Border** check box in the Control Panel. A red dashed border will appear around the image.
- Drag the red line to resize the border. Only the image within the border will refresh.

Cross Line

Check the Cross Line check box in the Control Panel. A crosshair will appear in the center of the rectangle.

Tools

GPIO and LED

- Click Tool in the menu bar and select **GPIO & LED** to bring up the GPIO dialog box. Select the port to access and select the digital output value. Click either the **write** or **read** button to write/read to/from the digital I/O ports.
- LED status is only supported with the cPCI Angelo series card.

EEPROM

- Click **Tool** in the menu bar and select EEPROM to bring up the EEPROM dialog box. Select the card you wish to access, enter the offset and output values, then click the **Write** button to write the value into the EEPROM. Enter the offset value and click the **Read** button to read the value from the EEPROM.
- Valid offset values are between 0 and 127. Valid output values are between 0 and 255. The value in the EEPROM will not be erased when the system is powered off.

Software trigger

- Click **Tool** in the menu bar and select **Software Trigger** to bring up the Trigger dialog box. Select the card to access and set the interval of the trigger pulse output. Check the ports you want to trigger simultaneously, and click the **Trigger** button.
- The one shot pulse output voltage goes high (from 0V to 5V).

5 Function Library

This chapter describes the API for Mpeg4 encode and decode. Users can use these functions to develop application programs under Visual C++, Visual Basic, C++ Builder, and Delphi.

5.1 List of Functions

Category	Function
Encode (Section 5.2)	AngeloMPEG4_Encode_Initial(Encoder_Index, Local_Address, Quality , Angelo_PortNo, Angelo_ChannelNo, Angelo_Color_Format, Angelo_Video_Format)
	AngeloMPEG4_Encode_InitialEx(Encoder_Index, Local_Address, Bitrate, frame_rate, Angelo_PortNo, Angelo_ChannelNo, Angelo_Color_Format, Angelo_Video_Format)
	AngeloMPEG4_Encode_Set_Callback(Encoder_Index, CallBackProc)
	AngeloMPEG4_Encode_Start(Encoder_Index)
	AngeloMPEG4_Encode_Stop(Encoder_Index)
	AngeloMPEG4_Encode_Close(Encoder_Index)
	AngeloMPEG4_Encode_Save_File_Start(Encoder_Index, n_file_name, interval_second, format)
	AngeloMPEG4_Encode_Save_File_Stop(Encoder_Index)
	AngeloMPEG4_Encode_Create_Directory(Encoder_Index, Dir)
	AngeloMPEG4_Encode_Set_Motion_Detection(Encoder_Index, Area, enable, Threshold, interval, action, X_Start, Y_Start, Width, Height)

Table 5-1: List of Functions

Category	Function
Decode (Section 5.3)	AngeloMPEG4_Decode_Set_Callback(Decoder_Index, CallBackProc)
	AngeloMPEG4_Decode_Connect(Decoder_Index,Encoder_IP, Encoder_Index)
	AngeloMPEG4_Decode_Disconnect(Decoder_Index)
	AngeloMPEG4_Decode_Set_Image_Config(Decoder_Index, ConfigIndex , Value)
	AngeloMPEG4_Decode_Set_Motion_Detection(Decoder_Index, Area, enable, Threshold, interval,action, X_Start, Y_Start, Width, Height)
	AngeloMPEG4_Decode_Get_Config(Decoder_Index,iWidth, iHeight, video_format, color_format,Bitrate, frame_rate)
	AngeloMPEG4_Decode_Start(Decoder_Index)
	AngeloMPEG4_Decode_Stop(Decoder_Index)
	AAngeloMPEG4_Decode_Get_FlowRate(Decoder_Index, Byte_Second)
	AngeloMPEG4_Decode_ReInitialEx(Decoder_Index, Bitrate, frame_rate, Angelo_Video_Format)
	AngeloMPEG4_Decode_ReInitial(Decoder_Index, Quality, Angelo_Video_Format)
	AngeloMPEG4_Decode_Save_File_Start(Decoder_Index, n_file_name, interval_second, format)
	AngeloMPEG4_Decode_Save_File_Stop(Decoder_Index)
	AngeloMPEG4_Decode_File(Decoder_Index, file_name, iWidth, iHeight, Byte_Per_Pixel, Total_Frame,Time_Seconds)
	AngeloMPEG4_Decode_File_Start(Decoder_Index, Mode)
	AngeloMPEG4_Decode_File_Set_Position(Decoder_Index, Frame_Index)
	AngeloMPEG4_Decode_File_Pause(Decoder_Index)
	AngeloMPEG4_Decode_File_Continue(Decoder_Index)
	AngeloMPEG4_Decode_File_Get_Position(Decoder_Index, Cur_Frame_Index)
	AngeloMPEG4_AVI_2_M4V(file_name, iWidth, iHeight, Byte_Per_Pixel, Total_Frame, Time_Seconds)
	AngeloMPEG4_M4V_2_AVI(file_name, iWidth, iHeight, Byte_Per_Pixel, Total_Frame, Time_Seconds)

Table 5-1: List of Functions

Category	Function
System (Section 5.4)	AngeloMPEG4_Get_Version(Mpeg4_DLLVersion, AngeloRTV_DLLVersion, Reserved)

Table 5-1: List of Functions

5.2 Encode Functions

@ Name

`AngeloMPEG4_Encode_Initial(Encoder_Index, Local_Address, Quality , Angelo_PortNo, Angelo_ChannelNo, Angelo_Color_Format, Angelo_Video_Format)`: Initialize the encoder.

`AngeloMPEG4_Encode_InitialEx(Encoder_Index, Local_Address, Bitrate, frame_rate, Angelo_PortNo, Angelo_ChannelNo, Angelo_Color_Format, Angelo_Video_Format)`: Initialize the encoder for advanced.

`AngeloMPEG4_Encode_Set_Callback(Encoder_Index, CallBackProc)`: Set up the callback function for encoder.

`AngeloMPEG4_Encode_Start(Encoder_Index)`: Start to grab image and encode.

`AngeloMPEG4_Encode_Stop(Encoder_Index)`: Stop grabbing image and encoding.

`AngeloMPEG4_Encode_Close(Encoder_Index)`: Close the encoder and network transmission.

`AngeloMPEG4_Encode_Save_File_Start(Encoder_Index, n_file_name, interval_second, format)`: Start to save compressed file in encode site.

`AngeloMPEG4_Encode_Save_File_Stop(Encoder_Index)`: Stop saving compressed file in encode site.

`AngeloMPEG4_Encode_Create_Directory(Encoder_Index, Dir)`: Create a new folder on the encode site.

AngeloMPEG4_Encode_Set_Motion_Detection(Encoder_Index, Area, enable, Threshold, interval, action, X_Start, Y_Start, Width, Height): Set the motion detection criteria, and action when motion occurs on the encode site.

@ Description

AngeloMPEG4_Encode_Initial:

This function initializes the video encoder. Its library supports 16 video encoders with the video source coming from Angelo_PortNo and Angelo_ChannelNo in the Angelo cards. Quality, and Angelo_Color_Format are parameters for encoder setting.

AngeloMPEG4_Encode_InitialEx:

This function initializes the video encoder. Its library supports 16 video encoders with the video source coming from Angelo_PortNo and Angelo_ChannelNo in the Angelo cards. Bitrate, frame_rate, and Angelo_Color_Format are parameters for encoder setting.

AngeloMPEG4_Encode_Set_Callback:

This function establishes a notification mechanism between function library and user process. Callback function is application-defined. The user passes the function pointer to function library by calling this function.

AngeloMPEG4_Encode_Start:

This function restarts encoding the video image when the encoder is paused.

AngeloMPEG4_Encode_Stop:

This function pauses encoding of the video image.

AngeloMPEG4_Encode_Close:

This function releases the resources of the encoder for the specified channel.

AngeloMPEG4_Encode_Save_File_Start:

Use this function to save the encoded image into an ".avi" or ".m4v" video file. The ".avi" file is the standard video format, and ".m4v" is only accessible in this function library.

Note: 1. Do not add a file extension to the file name.

2. User must install the XVID Codec in our setup disk in order to play ".avi" file in MS Media Player.

AngeloMPEG4_Encode_Save_File_Stop:

Use this function to stop saving the video file. In general, the video file will close automatically after the "Interval" parameter in AngeloMPEG4_Encode_Save_File_Start.

AngeloMPEG4_Encode_Create_Directory:

This function is used to create a new directory for saving a video file. The "filename" parameter in AngeloMPEG4_Encode_Save_File_Start contains the file path name.

AngeloMPEG4_Encode_Set_Motion_Detection:

Use this function to configure the motion detection criteria and the action when motion occurs at the encoding site.

AngeloMPEG4_Encode_Initial–

AngeloMPEG4_Encode_InitialEx–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int AngeloMPEG4_Encode_Initial(int Encoder_Index,
        char* Local_Address, int Quality ,int
        Angelo_PortNo, int Angelo_ChannelNo, int
```

```
          Angelo_Color_Format, int
          Angelo_Video_Format)
     int AngeloMPEG4_Encode_InitialEx(int
          Encoder_Index, char* Local_Address, int
          Bitrate, int frame_rate, int Angelo_PortNo,
          int Angelo_ChannelNo, int
          Angelo_Color_Format, int
          Angelo_Video_Format);
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Encode_Initial(ByVal Encoder_Index As
     Long, ByVal Local_address As String, ByVal
     Quality As Long, ByVal Angelo_PortNo As
     Long, ByVal Angelo_ChannelNo As Long, ByVal
     Angelo_Color_Format As Long, ByVal
     Angelo_Video_Format As Long) As Long
AngeloMPEG4_Encode_InitialEx (ByVal Encoder_Index
     As Long, ByVal Local_address As String,
     ByVal Bitrate As Long, ByVal frame_rate As
     Long, ByVal Angelo_PortNo As Long, ByVal
     Angelo_ChannelNo As Long, ByVal
     Angelo_Color_Format As Long, ByVal
     Angelo_Video_Format As Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Encode_Initial(Encoder_Index:Longint
     ; Local_Address:String; Quality:Longint;
     Angelo_PortNo:Longint;
     Angelo_ChannelNo:Longint;
     Angelo_Color_Format:Longint;
     Angelo_Video_Format:Longint):Longint;
AngeloMPEG4_Encode_InitialEx(Encoder_Index:Longi
     nt; Local_Address:String; Bitrate:Longint;
     frame_rate:Longint; Angelo_PortNo:Longint;
     Angelo_ChannelNo:Longint;
     Angelo_Color_Format:Longint;
     Angelo_Video_Format:Longint):Longint;
```

@ Argument

Encoder_Index: Indicates the channel index for the MPEG4 encoder. The range of channels is 0 – 15.

Local_Address: Indicates the IP Address at the encoding site. Set 0, NULL or nil for default setting.

Quality Index:

Quality Level	Value	NTSC		
		4CIF (640 x 480)	**CIF (320 x 240)**	**QCIF (160 x 120)**
Lowest	-2	Bit-rate = 400000*4 Frame rate = 5	Bit-rate = 400000 Frame rate = 5	Bit-rate = 400000/4 Frame rate = 5
Low	-1	Bit-rate = 480000*4 Frame rate = 10	Bit-rate = 480000 Frame rate = 10	Bit-rate = 480000/4 Frame rate = 10
Normal	0	Bit-rate = 560000*4 Frame rate =15	Bit-rate = 560000 Frame rate =15	Bit-rate = 560000/4 Frame rate =15
High	1	Bit-rate = 560000*4 Frame rate = 30	Bit-rate = 560000 Frame rate = 30	Bit-rate = 560000/4 Frame rate = 30
Highest	2	Bit-rate = 1024000*4 Frame rate = 30	Bit-rate = 1024000 Frame rate = 30	Bit-rate = 1024000/4 Frame rate = 30
Quality Level	**Value**	**PAL**		
		4CIF (768 x 576)	**CIF (384 x 288)**	**QCIF (192 x 144)**
Lowest	-2	Bit-rate = 400000*4 Frame rate = 4	Bit-rate = 400000 Frame rate = 4	Bit-rate = 400000/4 Frame rate = 4
Low	-1	Bit-rate = 480000*4 Frame rate = 8	Bit-rate = 480000 Frame rate = 8	Bit-rate = 480000/4 Frame rate = 8
Normal	0	Bit-rate = 560000*4 Frame rate =12	Bit-rate = 560000 Frame rate =12	Bit-rate = 560000/4 Frame rate =12
High	1	Bit-rate = 560000*4 Frame rate = 25	Bit-rate = 560000 Frame rate = 25	Bit-rate = 560000/4 Frame rate = 25
Highest	2	Bit-rate = 1024000*4 Frame rate = 25	Bit-rate = 1024000 Frame rate = 25	Bit-rate = 1024000/4 Frame rate = 25

Table 5-2: Quality Index

Bitrate: Indicates the number of bits per second.

frame_rate: Indicates the number of frames that the MPEG4 encoder will encode per second. The range of the frame_rate is 1 – 30.

Angelo_PortNo: The port number is the zero index of the Angelo series card. For example, if there are two PCI-RTV-24 Angelo cards (card 0, card 1) in the system, and each PCI-RTV-24 has four ports, the first port of card 0 is “0”, and the first port of card 1 is “4.”

Angelo_ChannelNo: Indicates the channel index of the port described above. There are four channels per port and the first channel index is 0.

Angelo_Color_Format: RGB24= 3

Angelo_Video_Format:

0: Full NTSC, with image size 640*480

1: Full PAL, with image size 768*576

2: CIF NTSC, with image size 320*240

3: CIF PAL, with image size 384*288

4: QCIF NTSC, with image size 160*120

5: QCIF PAL, with image size 192*144

@ Return Code

@ Example

<VC/BCB >

```
int Result;
int Encoder_Index = 0;
int Quality = 0;
int Angelo_PortNo = 0;
int Angelo_ChannelNo = 0;
int Angelo_Color_Format = 3; //RGB24
int Angelo_Video_Format = 2; //CIF NTSC
int Bitrate = 480000;
int frame_rate = 15;


Result = AngeloMPEG4_Encode_Initial
      (Encoder_Index, Quality, Angelo_PortNo,
      Angelo_ChannelNo, Angelo_Color_Format,
      Angelo_Video_Format);
Result = AngeloMPEG4_Encode_InitialEx
      (Encoder_Index, Bitrate, frame_rate,
      Angelo_PortNo, Angelo_ChannelNo,
      Angelo_Color_Format, Angelo_Video_Format);
```

< Visual Basic >

```
Dim result As Long
Dim Encoder_Index As Long, Quality As Long,
      Angelo_PortNo As Long, Angelo_ChannelNo As
      Long, Angelo_Color_Format As Long,
      Angelo_Video_Format As Long, Bitrate As
      Long, frame_rate As Long

Encoder_Index =  0
Quality = 0
Angelo_PortNo = 0
Angelo_ChannelNo = 0
Angelo_Color_Format = 3 "RGB24
Angelo_Video_Format = 2 "CIF NTSC
Bitrate = 480000
frame_rate = 15

Result = AngeloMPEG4_Encode_Initial
      (Encoder_Index, Quality, Angelo_PortNo,
      Angelo_ChannelNo, Angelo_Color_Format,
      Angelo_Video_Format)
Result = AngeloMPEG4_Encode_InitialEx
      (Encoder_Index, Bitrate, frame_rate,
      Angelo_PortNo, Angelo_ChannelNo,
      Angelo_Color_Format, Angelo_Video_Format)
```

< Delphi >

```
Var
Encoder_Index, Result: Longint;
Quality: Longint;
Bitrate, frame_rate: Longint;
Angelo_PortNo, Angelo_ChannelNo: Longint;
Angelo_Color_Format, Angelo_Video_Format:
     Longint;
begin
Encoder_Index:= 0;
Quality := 0; // Normal Quality
Bitrate := 480000;
frame_rate := 15;
Angelo_PortNo := 0;
Angelo_ChannelNo := 0;
Angelo_Color_Format := 3; // RGB24
Angelo_Video_Format := 2; // CIF, NTSC
Result :=
     AngeloMPEG4_Encode_Initial(Encoder_Index,
     Quality, Angelo_PortNo, Angelo_ChannelNo,
     Angelo_Color_Format, Angelo_Video_Format);
Result :=
     AngeloMPEG4_Encode_InitialEx(Encoder_Index,
     Bitrate, frame_rate, Angelo_PortNo,
     Angelo_ChannelNo, Angelo_Color_Format,
     Angelo_Video_Format);
end;
```

AngeloMPEG4_Encode_Set_Callback–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Encode_Set_Callback(int
     Encoder_Index, void ( __stdcall
     *CallBackProc)(int Encoder_Index,long
     int_status,param_str* param_struct));
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Encode_Set_Callback (ByVal
      Encoder_Index As Long, ByVal
      Encode_CallBackProcas As Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Encode_Set_Callback(Encoder_Index:Lo
      ngint;
      lpEncodeCallBackProc:EncodeCallBackProc):Lo
      ngint;
```

@ Argument

Encoder_Index: Indicates the channel Index for the MPEG4 encoder. The range of channels is 0 – 15.

@ Return Code

@ Example

< VC/BCB >

```
int Result;
int Encoder_Index = 0;

void __stdcall Encode_Callback(int Encoder_Index,
      long int_status, param_str *param_struct)
{
      if(int_status & 0x01 ==1) //Image Ready
{
      }
      if(int_status >> 4 & 0x01 ==1) //Motion
      Dection
{
      }
}
Result =
      AngeloMPEG4_Encode_Set_Callback(Encoder_Ind
      ex, Encode_Callback);
```

< Visual Basic >

```
Dim Encoder_Index As Long, Result As Long

Public Sub encode_callback(ByVal Encoder_Index As
     Long, ByVal int_status As Long, param_str As
     param_struct)
    Select Case (int_status)
    Case 1: "preview
Case 16: " motion detection
End Select
End Sub

Channel =0
Result =
     AngeloMPEG4_Encode_Set_Callback(Encoder_Ind
     ex, Encode_Callback)
```

< Delphi >

```
procedure Encode_Callback
     (Encoder_Index:Longint;int_status:Longint;v
     ar param_struct:param_str);stdcall
var
    {* add your var here *}
begin
    case int_status of
      1: begin  {********* Image Ready *********}
      end;
     2: begin  {********* Set Image Config Event
     *********}
      end;
      4: begin  {********* Connected Event
     *********}
      end;
      8: begin  {********* Disconnect Event
     *********}
      end;
     16: begin  {********* Motion Detection Event
     *********}
      end;
    end;  // end case int_status of
end;
```

```
// Main Code
var
Encoder_Index, Result: Longint;
begin
Encoder_Index:= 0;
Result :=
      AngeloMPEG4_Encode_Set_Callback(Encoder_Ind
      ex, Encode_Callback);
end;
```

AngeloMPEG4_Encode_Start–

AngeloMPEG4_Encode_Stop–

AngeloMPEG4_Encode_Close–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int AngeloMPEG4_Encode_Start(int Encoder_Index);
int AngeloMPEG4_Encode_Stop(int Encoder_Index);
int AngeloMPEG4_Encode_Close(int Encoder_Index);
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Encode_Start (ByVal Encoder_Index As
      Long) As Long
AngeloMPEG4_Encode_Stop (ByVal Encoder_Index As
      Long) As Long
AngeloMPEG4_Encode_Close (ByVal Encoder_Index As
      Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Encode_Start(Encoder_Index:Longint):
      Longint;
AngeloMPEG4_Encode_Stop(Encoder_Index:Longint):L
      ongint;
AngeloMPEG4_Encode_Close(Encoder_Index:Longint):
      Longint;
```

@ Argument

`Channel:` Indicates the channel index for the MPEG4 encoder. The range of channels is 0 – 15.

@ Return Code

@ Example

< VC/BCB >

```
 int Result;
int Encoder_Index = 0;

Result = AngeloMPEG4_Encode_Start(Encoder_Index);
Result = AngeloMPEG4_Encode_Stop(Encoder_Index);
Result = AngeloMPEG4_Encode_Close(Encoder_Index);
```

< Visual Basic >

```
Dim Result As Long, Encoder_Index As Long

Encoder_Index = 0

Result = AngeloMPEG4_Encode_Start(Encoder_Index)
Result = AngeloMPEG4_Encode_Stop(Encoder_Index)
Result = AngeloMPEG4_Encode_Close(Encoder_Index)
```

< Delphi >

```
var
Encoder_Index, Result: Longing;
begin
Result := AngeloMPEG4_Encode_Stop(Encoder_Index);
      // pause the encoder
Result :=
      AngeloMPEG4_Encode_Start(Encoder_Index); //
      restart the encoder
// close the Encoder
Result := AngeloMPEG4_Encode_Stop(Encoder_Index);
Result :=
        AngeloMPEG4_Encode_Close(Encoder_Index);
end;
```

AngeloMPEG4_Encode_Save_File_Start–

AngeloMPEG4_Encode_Save_File_Stop–

AngeloMPEG4_Encode_Create_Directory–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int   AngeloMPEG4_Encode_Save_File_Start(int
        Encoder_Index, char* n_file_name, long
        interval_second, long format);
int   AngeloMPEG4_Encode_Save_File_Stop(int
        Encoder_Index);
int AngeloMPEG4_Encode_Create_Directory(int
        Encoder_Index, char* Dir);
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Encode_Save_File_Start (ByVal
        Encoder_Index As Long, ByVal n_file_name As
        String, ByVal interval_second As Long, ByVal
        format As Long) As Long
AngeloMPEG4_Encode_Save_File_Stop (ByVal
        Encoder_Index As Long) As Long
AngeloMPEG4_Encode_Create_Directory(ByVal
        Encoder_Index As Long, ByVal Dir As String,)
        As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Encode_Save_File_Start(Encoder_Index
        :Longint; n_file_name:String;
        interval_second:Longint;
        format:Longint):Longint;
AngeloMPEG4_Encode_Save_File_Stop(Encoder_Index:
        Longint):Longint;
```

```
AngeloMPEG4_Encode_Create_Directory(Encoder_Inde
     x:Longint; Dir:String):Longint;
```

@ Argument

`Encoder_Index:` Indicates the channel index for the MPEG4 encoder. The range of channels is 0 – 15.

`n_file_name:` The argument is the path and name of the file that the encoded image will be saved to.

`interval_second:` This argument is the number of seconds of encoded video to be saved.

`Format:` The argument describes the format in which to save the file.

1. m4v file
2. avi file
3. both

Dir: The argument is the path and name of the directory that will be created.

@ Return Code

@ Example

< VC/BCB >

```
int Result;
int Encoder_Index = 0;
char* n_file_name = "test";
long interval_second = 60;
int format = 3; //save both format
char* Dir = "temp";
Result =
     AngeloMPEG4_Encode_Save_File_Start(Encoder_
     Index, n_file_name, interval_second,
     format);
Result = AngeloMPEG4_Encode_Create_Directory
     (Encoder_Index, Dir);
```

< Visual Basic >

```
Dim Result As Long, Encoder_Index As Long,
     interval_second As Long, format As Long

Encoder_Index = 0;
n_file_name = "test"
interval_second = 60
format = 3 "save both format
Dir = "temp"
Result =
     AngeloMPEG4_Encode_Save_File_Start(Encoder_
     Index, n_file_name, interval_second,
     format)
```

< Delphi >

```
Var
Encoder_Index, Result: Longint;
Dir, n_file_name: String;
interval_second, format: Longint;
begin
Encoder_Index:= 0;
Dir := "C:\VideoDir";
n_file_name := Dir + "\" + "Video0";
interval_second := 60;
format := 3; // save both format
Result :=
     AngeloMPEG4_Encode_Create_Directory(Encoder
     _Index, Dir);
Result :=
     AngeloMPEG4_Encode_Save_File_Start(Encoder_
     Index, n_file_name, interval_second,
     format);

end;
```

AngeloMPEG4_Encode_Set_Motion_Detection–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int AngeloMPEG4_Encode_Set_Motion_Detection(int
     Encoder_Index,int Area,int enable, int
     Threshold,int interval,int action,int
     X_Start,int Y_Start,int Width,int Height);
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Encode_Set_Motion_Detection(ByVal
     Encoder_Index As Long, ByVal Area As Long,
     ByVal enable As Long, ByVal Threshold As
     Long, ByVal interval As Long, ByVal action
     As Long, ByVal X_Start As Long, ByVal
     Y_Start As Long, ByVal Width As Long, ByVal
     Height As Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Encode_Set_Motion_Detection(Encoder_
     Index:Longint; Area :Longint;
     enable:Longint; Threshold:Longint;
     interval:Longint; action:Longint;
     X_Start:Longint; Y_Start:Longint;
     Width:Longint; Height:Longint):Longint;
```

@ Argument

`Encoder_Index`: Indicate the channel index for the MPEG4 encoder. The range of channels is 0 – 15.

`Area`: User can assign up to 4 motion detection areas in one frame, the valid values are from 1 - 4.

`enable`:

1: enables motion detection

0: disables motion detection

Threshold: Determines the sensitivity of motion detection measurement. The valid values are from 0 - 15, with 0 being the highest sensitivity.

Interval: The time interval between measurements of motion detection.

Action: This argument describes what actions the function will do.

bit 0: Callback,

X_Start, Y_Start, Width, Height: Sets the boundary of the motion detection area.

@ Return Code

@ Example

< VC/BCB >

```
int Result;
int Encoder_Index = 0;
int enable = 1;
int Threshold = 5;
int interval = 3;
int action = 1;
int area =1;
int X_Start = 0;
int Y_Start =0;
int Width = 160;
int Height = 120;

Result =
     AngeloMPEG4_Encode_Set_Motion_Detection(Enc
     oder_Index, area, enable, Threshold,
     interval, action, X_Start, Y_Start, Width,
     Height);
```

< Visual Basic >

```
Dim Result As Long, Encoder_Index As Long, enable
     As Long, Threshold As Long, interval As
     Long, action As Long, area As Long, X_Start
     As Long, Y_Start As Long, Width As Long,
     Height As Long


Encoder_Index = 0
enable = 1
Threshold = 5
interval = 3
action = 1
area =1
X_Start = 0
Y_Start =0
Width = 160
Height = 120

Result =
     AngeloMPEG4_Encode_Set_Motion_Detection(Enc
     oder_Index, area, enable, Threshold,
     interval, action, X_Start, Y_Start, Width,
     Height)
```

<Delphi >

```
var
Encoder_Index, Result: Longint;
enable, Threshold, interval, action: Longint,
     area:Longint, X_Star:Longint,
     Y_Start:Longint, Width: Longint,
     Height:Longint;
begin
Encoder_Index:= 0;
enable := 1;
Threshold := 5;
Interval := 3; // 3 sec
Action := 1; // callback
area =1;
X_Start = 0;
Y_Start =0;
```

```
Width = 160;
Height = 120;

if (enable = 1) then
Result =
     AngeloMPEG4_Encode_Set_Motion_Detection(Enc
     oder_Index, area, enable, Threshold,
     interval, action, X_Start, Y_Start, Width,
     Height)
else // disable motion detection
Result =
     AngeloMPEG4_Encode_Set_Motion_Detection(Enc
     oder_Index, area, 0, Threshold, interval,
     action, X_Start, Y_Start, Width, Height);
end;
```

5.3 Decode Functions

@ Name

AngeloMPEG4_Decode_Set_Callback(Decoder_Index, CallBackProc) – Setup the callback function for decoder.

AngeloMPEG4_Decode_Connect(Decoder_Index, Encoder_IP, Encoder_Index) – Connect to the encoder.

AngeloMPEG4_Decode_Disconnect(Decoder_Index) – Disconnect from the encoder.

AngeloMPEG4_Decode_Set_Image_Config(Decoder_Index, ConfigIndex , Value) – Adjust the brightness, contrast, hue etc..

AngeloMPEG4_Decode_Set_Motion_Detection(Decoder_Index, Area, enable, Threshold, interval, action, X_Start, Y_Start, Width, Height) – Set the motion detection criteria, and action when motion occurs in decode site.

AngeloMPEG4_Decode_Get_Config(Decoder_Index, iWidth, iHeight, video_format, color_format, Bitrate, frame_rate) – Get the video property from encode site.

AngeloMPEG4_Decode_Start(Decoder_Index) – Start to decode the video.

AngeloMPEG4_Decode_Stop(Decoder_Index) – Stop decoding the video.

AngeloMPEG4_Decode_Get_FlowRate(Decoder_Index, Byte_Second) – Get the current data flow rate between encoder and decoder

AngeloMPEG4_Decode_ReInitialEx(Decoder_Index, Bitrate, frame_rate, Angelo_Video_Format) – Reset the video property.

AngeloMPEG4_Decode_ReInitial(Decoder_Index, Quality , Angelo_Video_Format) – Reset the video property.

AngeloMPEG4_Decode_Save_File_Start(Decoder_Index, n_file_name, interval_second, format) – Start to save compressed file in decode site.

AngeloMPEG4_Decode_Save_File_Stop(Decoder_Index) – Stop saving compressed file in decode site.

AngeloMPEG4_Decode_File(Decoder_Index, file_name, iWidth, iHeight, Byte_Per_Pixel, Total_Frame,Time_Seconds) - Decode from *.avi or *.m4v file

AngeloMPEG4_Decode_File_Start(Decoder_Index, Mode) - Start to decode from file

AngeloMPEG4_Decode_File_Set_Position(Decoder_Index, Frame_Index) – Jump to the postion

AngeloMPEG4_Decode_File_Pause(Decoder_Index) - Pauses play

AngeloMPEG4_Decode_File_Continue(Decoder_Index) - Continue the play

AngeloMPEG4_Decode_File_Get_Position(Decoder_Index, Cur_Frame_Index) - Get the current position of play

AngeloMPEG4_AVI_2_M4V(file_name, iWidth, iHeight, Byte_Per_Pixel, Total_Frame, Time_Seconds) - Translate *avi file into *.m4v file

AngeloMPEG4_M4V_2_AVI(file_name, iWidth, iHeight, Byte_Per_Pixel, Total_Frame, Time_Seconds) - Translate *m4v file into *.avi file

@ Description

AngeloMPEG4_Decode_Set_Callback:

This function establishes a notification mechanism between the function library and user process. The callback function is application-defined, users pass the function pointer to function library by calling this function. To receive notification events, users must apply this function before any decode function on the decode site.

AngeloMPEG4_Decode_ Connect:

Use this function to establish a connection between decoder and encoder. The video date will then be transferred through this connection.

AngeloMPEG4_Decode_ Disconnect:

Use this function to close the connection between decoder and encoder. After closing the connection, the decoder will not receive video data from encoder.

AngeloMPEG4_Decode_Set_Image_Config:

If the connection between encoder and decoder is established, use this function to adjust the image property such as contrast and brightness.

AngeloMPEG4_Decode_Set_Motion_Detection:

If the connection between encoder and decoder is established, use this function to configure the motion detection criteria and the action when motion occurs in decode site.

AngeloMPEG4_Decode_Get_Config:

User must define a callback function, than call "AngeloMPEG4_Decode_Set_Callback". Use "AngeloMPEG4_Decode_ Connect" to establish the connection, if connection is made, the callback function will receive a notification event. The user can then use "AngeloMPEG4_Decode_Get_Config" to retrieve the image configuration such as width, height, bitrate, framerate from the encode site.

AngeloMPEG4_Decode_Start:

If the connection between encoder and decoder is established, the video data will transfer from encoder to decoder automatically. Use this function to restart the video data transmission, if “AngeloMPEG4_Decode_Stop” has been called to stop the transmission.

`AngeloMPEG4_Decode_Stop:`

This function only stops the video data transmission between decoder and encoder, but the connection is still established.

`AngeloMPEG4_Decode_Get_FlowRate:`

If the connection between encoder and decoder is established, use this function to query the current data flow rate between encode and decode.

`AngeloMPEG4_Decode_ReInitialEx:`

Because the Bitrate, frame_rate is initialized in the encode site, the decode uses this function to reset the image quality if connection is established.

Note: If one decoder changes the quality, the others will also have a different image quality.

`AngeloMPEG4_Decode_ReInitial:`

Because the Bitrate, frame_rate is initialized in encode site, the decode use this function to reset the image quality, if the connection is established.

`AngeloMPEG4_Decode_Save_File_Start:`

If the connection between encoder and decoder is established, use this function to save the encoded image into an “.avi”, “.m4v” video file on the decode site. The .avi file is the standard video format, and .m4v is only accessible in this function library.

Note: 1. Do not add the file extension name.

2. Users must install the XVID Codec from the setup disk. The ".avi" file can be played in MS Media Player.

`AngeloMPEG4_Decode_Save_File_Stop:`

If the connection between encoder and decoder is established, use this function to stop saving video file on the decode site. In general, the video file will close automatically after the "Interval" parameter in AngeloMPEG4_Decode_Save_File_Start.

`AngeloMPEG4_Decode_File:`

If you save the video file into ".m4v" or ".avi", and the file is closed, than you can use this function to decode the ".m4v" or ".avi", and get the video image in callback function, than you can draw the image on the Windows DC. This function initialize the decode from file

`AngeloMPEG4_Decode_File_Start:`

Start decoding from file. If the callback function has been set up, a video buffer of each frame will be received.

`AngeloMPEG4_Decode_File_Set_Position:`

Skip some frames, and jump to the frame you want. You can get the total frames of the file using AngeloMPEG4_Decode_File.

AngeloMPEG4_Decode_File_Pause:

The file is paused until AngeloMPEG4_Decode_File_Continue is activated.

AngeloMPEG4_Decode_File_Get_Position:

Get the current frame index of the file.

AngeloMPEG4_AVI_2_M4V:

Use this function to translate a closed ".avi" video file into ".m4v" format.

AngeloMPEG4_ M4V_2_AVI:

Use this function to translate a closed ".m4v" video file into ".avi" format.

AngeloMPEG4_Decode_Connect –

AngeloMPEG4_Decode_Disconnect –

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int AngeloMPEG4_Decode_Connect(int Decoder_Index,
char* Encoder_IP, unsigned int Enocder_Index);
int AngeloMPEG4_Decode_Disconnect(int
Decoder_Index);
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Connect(ByVal Decoder_Index As
Long, ByVal Encoder_IP As String, ByVal
Enocder_Index As Long) As Long
AngeloMPEG4_Decode_ Disconnect (ByVal
Decoder_Index As Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Connect(Decoder_Index:Longint
; Encoder_IP:String;
Encoder_Index:Longint):Longint;
AngeloMPEG4_Decode_Disconnect(Decoder_Index:Long
int):Longint;
```

@ Argument

`Decoder_Index`: Indicates the channel number of MPEG4 Decoder. The range of channel is 0 - 15.

`Encoder_IP`: The IP address of MPEG4 Encode.

`Encoder_Index`: The channel of MPEG4 Encoder.

@ Return Code

0: ERROR_NoError

@ Example

< VC/BCB >

```
int Result;
int channel = 0;
char* Encoder_IP = "127.0.0.1"; //localhost
unsigned int Encoder_channel = 0;
Result = AngeloMPEG4_Decode_Connect(channel,
     Encoder_IP, Encoder_channel);
Result = AngeloMPEG4_Decode_Disconnect(channel);
```

< Visual Basic >

```
Dim Result As Long, channel As Long,
     Encoder_channel As Long
Dim Encoder_IP As String

channel = 0
Encoder_IP = "127.0.0.1" 'localhost
Encoder_channel = 0
Result = AngeloMPEG4_Decode_Connect(channel,
     Encoder_IP, Encoder_channel)
Result = AngeloMPEG4_Decode_Disconnect(channel)
```

<Delphi >

```
var
channel: Longint;
Encoder_IP: String;
Encoder_channel: Longint;
Result: Longint;
begin
channel := 0;
Remote_IP := ‘127.0.0.1’; //localhost
Result := AngeloMPEG4_Decode_Connect(channel,
      Encoder_IP, Encoder_channel);
Result := AngeloMPEG4_DecodeDisconnect(channel);
end;
```

AngeloMPEG4_Decode_Set_Callback–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int AngeloMPEG4_Decode_Set_Callback(int
Decoder_Index, void ( __stdcall
*CallBackProc)(int channel, long int_status, long
VideoBufferaddress));
```

Visual Basic(Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Set_Callback(ByVal
Decoder_Index As Long, ByVal CallBack As Long) As
Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Set_Callback(Decoder_Index:Lo
ngint;
lpDecodeCallBackProc:DecodeCallBackProc):Longint
;
```

@ Argument

`Decoder_Index:` Indicates the channel number of Decoder. The range of channel is 0 - 15.

int_status:

Interrupt status:

Bit 0: Image ready

Bit 1: Motion Detection occur

Bit 2: Connection establish

@ Return Code

0: ERROR_NoError

@ Example

< VC/BCB >

```
int Result;
int channel = 0;
void __stdcall Decode_Callback(int channel, long
     int_status, long VideoBufferaddress)
{
     if((int_status & 0x01) == 1) //Image Ready
     {
          //Start Drawing

     memcpy(Temp,(PVOID)VideoBufferaddress,iWidt
     h*iHeight*3);
          gpDC-
     >BitBlt(10,10,iWidth,iHeight,MemDC,0,0,SRCC
     OPY);
     }
     if((int_status>>1 & 0x01) == 1) //
     MotionDetection Occur
     {
          //Deal with MotionDetection
          Beep(1024, 100);
     }
     if((int_status>>2 & 0x01) == 1) //Connection
     establish
     {
          //Prepare DC for Preview
```

```
            int Bitrate = 0, frame_rate = 0,
      colorspace = 0;

            AngeloMPEG4_Decode_Get_Config(channel,
      &iWidth, &iHeight, &videoformat,
      &colorspace, &Bitrate, &frame_rate);
            }
}

Result = AngeloMPEG4_Decode_Set_Callback(channel,
      Decode_Callback);
```

< Visual Basic >

```
Dim Result As Long, channel As Long

      Public Sub lpcallback(ByVal channel As Long,
      ByVal int_status As Long, ByVal
      VideoBufferaddress As Long)
                  If int_status And &H2 Then
      'detected motion

                  ElseIf int_status And &H4 Then '
      connect to encoder

                  ElseIf int_status And &H1 Then '
      image ready
                  End If
      End Sub

      Result =
      AngeloMPEG4_Decode_Set_Callback(channel,
      AddressOf lpcallback)
```

< Delphi >

```
procedure DecoderCallbackProc(channel:Longint;
      int_status:Longint;
      VideoBufferaddress:Longint); stdcall
var
     Str_Addr: Pointer;
```

```
          Bitrate, Framerate, colorspace, videoformat:
            Longint;
      begin
          case int_status of
            1: begin  {********* image buffer OK
           *********}
               // draw image here
            end;
            2: begin  {********* Motion Detected
           *********}
            end;
           4: begin  {********* Connect Ready Interrupt
           *********}
               // You can get image config here and do
           somthing
            end;
          end;  // end case int_status of
      end;

      // Main Code
      var
      channel: Longint;
      Result: Longint;
      begin
      channel := 0;
      Result :=
           AngeloMPEG4_Decode_Set_Callback(channel,
           DecoderCallbackProc);
      end;
```

AngeloMPEG4_Decode_Set_Image_Config–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int AngeloMPEG4_Decode_Set_Image_Config(int
     Decoder_Index, int ConfigIndex , int Value);
```

Visual Basic(Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Set_Image_Config(ByVal
     channel As Long, ByVal Decoder_Index As
     Long, ByVal Value As Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Set_Image_Config(Decoder_Inde
     x:Longint; ConfigIndex:Longint;
     Value:Longint):Longint;
```

@ Argument

`Decoder_Index:` Indicate the channel number of Decoder. The range of channel is 0 ~ 15.

`ConfigIndex:`

0 for BRIGHTNESS

1 for HUE

2 for SATURATION (U)

3 for SATURATION (V)

4 for CONTRAST (LUMA)

5 for luma notch filter (for monochrome video, the notch filter should not be used)

value: (0-255):

	Range	Default value
BRIGHTNESS	0 - 255	128
HUE	0 - 255	0
CHROMA (U)	0 - 255	127
CHROMA (V)	0 - 255	127
LUMA	0 - 255	108
LUMA notch filter	0 (Enable) or 1 (Disable)	

Table 5-3: Video adjustments table

@ Return Code

0: ERROR_NoError

@ Example

< VC/BCB >

```
int Result;
int channel = 0;
int ConfigIndex = 0;
int value = 128;
Result =
     AngeloMPEG4_Decode_Set_Image_Config(channel
     , ConfigIndex, value);
```

< Visual Basic >

```
Dim Result As Long, channel As Long, ConfigIndex
     As Long, value As Long
channel = 0
ConfigIndex = 0
value = 128
Result =
     AngeloMPEG4_Decode_Set_Image_Config(channel
     , ConfigIndex, value)
```

<Delphi >

```
var
channel: Longint;
ConfigIndex: Longint;
Value: Longint;
Result: Longint;
begin
channel := 0;
ConfigIndex := 0;
Value := 128;
Result :=
     AngeloMPEG4_Decode_Set_Image_Config(channel
     , ConfigIndex, Value);
end;
```

AngeloMPEG4_Decode_Set_Motion_Detection–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int AngeloMPEG4_Decode_Set_Motion_Detection(int
     Decoder_Index,int Area,int enable, int
     Threshold,int interval,int action,int
     X_Start,int Y_Start,int Width,int Height);
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Set_Motion_Detection(ByVal
     Decoder_Index As Long, ByVal enable As Long,
     ByVal Threshold As Long, ByVal interval As
     Long, ByVal action As Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Set_Motion_Detection(Decoder_
     Index:Longint; Area :Longint;
     enable:Longint; Threshold:Longint;
     interval:Longint; action:Longint;
     X_Start:Longint; Y_Start:Longint;
     Width:Longint; Height:Longint):Longint;
```

@ Argument

Decoder_Index: Indicates the channel number of Decoder. The range of channel is 0 - 15.

Area: User can assign up to four motion detection areas in one frame, the valid value range is 1 - 4.

Enable:

1: enable Motion Detection

0: disable Motion Detection

Threshold: The threshold senses motion detection occurrence. The value range is 0 - 15, with 0 being the highest sensitivity.

Interval: Time interval measures motion detection occurrence.

Action: The argument descript what actions the function will do.

bit 0: Callback,

bit 1: Reserved,

bit 2: Send motion frame

Example: when action = 1 + 4, the function will perform callback and send the motion image.

X_Start, Y_Start, Width, Height: Set the boundary of motion detection area.

@ Return Code

@ Example

< VC/BCB >

```
int Result;
int channel = 0;
int enable = 1;
int Threshold = 5;
int interval = 3;
int action = 1 + 4;
int area =1;
int X_Start = 0;
int Y_Start =0;
int Width = 160;
int Height = 120;
Result =
     AngeloMPEG4_Decode_Set_Motion_Detection(cha
     nnel, area, enable, Threshold, interval,
     action, X_Start, Y_Start, Width, Height);
```

< Visual Basic >

```
Dim Result As Long, channel As Long, enable As
     Long, Threshold As Long, interval As Long,
     action As Long, area As Long, X_Start As
     Long, Y_Start As Long, Width As Long, Height
     As Long
```

```
channel = 0
enable = 1
Threshold = 5
interval = 3
action = 1 + 4
area =1
X_Start = 0
Y_Start =0
Width = 160
Height = 120

Result =
     AngeloMPEG4_Decode_Set_Motion_Detection(cha
     nnel, area, enable, Threshold, interval,
     action, X_Start, Y_Start, Width, Height)
```

< Delphi >

```
var
channel, Result: Longint;
enable, Threshold, interval, action: Longint,
     area:Longint, X_Star:Longint,
     Y_Start:Longint, Width: Longint,
     Height:Longint;
begin
channel := 0;
enable := 1;
Threshold := 5;
Interval := 3; // 3 sec
Action := 1+4; // callback & send motion image
area =1;
X_Start = 0;
Y_Start =0;
Width = 160;
Height = 120;


if (enable = 1) then
Result =
     AngeloMPEG4_Decode_Set_Motion_Detection(cha
     nnel, area, enable, Threshold, interval,
     action, X_Start, Y_Start, Width, Height)
else // disable motion detection
```

```
Result =
      AngeloMPEG4_Decode_Set_Motion_Detection(cha
      nnel, area, 0, Threshold, interval, action,
      X_Start, Y_Start, Width, Height);
end;
```

AngeloMPEG4_Decode_Get_Config–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int AngeloMPEG4_Decode_Get_Config(int
      Decoder_Index, int* iWidth, int* iHeight,
      int* video_format, int* color_format, int*
      Bitrate, int* frame_rate);
```

Visual Basic(Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Get_Config (ByVal
      Decoder_Index As Long, ByRef iWidth As Long,
      ByRef iHeight As Long, ByRef video_format As
      Long, ByRef color_format As Long, ByRef
      Bitrate As Long, ByRef frame_rate As Long)
      As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Get_Config(Decoder_Index:Long
      int; var iWidth:Longint; var
      iHeight:Longint; var video_format:Longint;
      var color_format:Longint; var
      Bitrate:Longint; var
      frame_rate:Longint):Longint;
```

@ Argument

`Decoder_Index`: Indicates the channel number of Decoder. The range of channel is 0 - 15.

`iWidth`: Indicates the width of the MPEG4 image size.

`iHeight`: Indicates the height of the MPEG4 image size.

`video_format:`

Full NTSC (640*480)	= 0,
Full PAL (768*576)	= 1,
CIF NTSC (320*240)	= 2,
CIF PAL (384*288)	= 3,
QCIF NTSC (160*120)	= 4,
QCIF PAL (192*144)	= 5,

`color_format:`

RGB16	= 0,
GRAY	= 1,
RGB15	= 2,
RGB24	= 3,
RGB32	= 4,
RGB8	= 5,
RAW8X	= 6,
YUY24:2:2	= 7,
BtYUV 4:1:1	= 8

At present, we only provide RGB24 color format, hence the value should always be set at 3.

`Bitrate:` Indicates the bitrate of MPEG4 stream from the encode server.

`frame_rate:` Indicates the frame rate of MPEG4 stream from the encode server.

@ Return Code

0: ERROR_NoError

@ Example

< VC/BCB >

```
int Result;
int channel = 0;
int iWidth = 0;
int iHeight = 0;
int video_format = 0;
int color_format = 0;
int Bitrate = 0;
int frame_rate = 0;


Result = AngeloMPEG4_Decode_Get_Config(channel,
     &iWidth, &iHeight, &videoformat,
     &color_format, &Bitrate, &frame_rate);
```

< Visual Basic >

```
     Dim Result As Long, channel As Long, iWidth
     As Long, iHeight As Long, video_format As
     Long, color_format As Long, Bitrate As Long,
     frame_rate As Long

Channel = 0
Result = AngeloMPEG4_Decode_Get_Config(channel,
     iWidth, iHeight, videoformat, colorformat,
     Bitrate, frame_rate)
```

< Delphi >

```
var
channel: Longint;
iWidth, iHeight: Longint;
videoformat, colorspace, Bitrate, frame_rate:
     Longint;
Result: Longint;
begin
channel := 0;
```

```
Result := AngeloMPEG4_Decode_Get_Config(channel,
      iWidth, iHeight, videoformat, colorspace,
      Bitrate, frame_rate);
end;
```

AngeloMPEG4_Decode_Start–

AngeloMPEG4_Decode_Stop–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int AngeloMPEG4_Decode_Start(int Decoder_Index);
int AngeloMPEG4_Decode_Stop(int Decoder_Index);
```

Visual Basic(Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Start(ByVal Decoder_Index As
      Long) As Long
AngeloMPEG4_Decode_Stop(ByVal Decoder_Index As
      Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Start(Decoder_Index:Longint):
      Longint;
AngeloMPEG4_Decode_Stop(Decoder_Index:Longint):L
      ongint;
```

@ Argument

`Decoder_Index:` Indicates the channel number of Decoder. The range of channel is 0 - 15.

@ Return Code

0: ERROR_NoError

@ Example

< VC/BCB >

```
int Result;
int channel = 0;
Result = AngeloMPEG4_Decode_Start(channel);
Result = AngeloMPEG4_Decode_Stop(channel);
```

< Visual Basic >

```
Dim Result As Long, channel As Long
channel = 0
Result = AngeloMPEG4_Decode_Start(channel)
Result = AngeloMPEG4_Decode_Stop(channel)
```

< Delphi >

```
var
channel: Longint;
Result: Longint;
begin
channel := 0;
Result := AngeloMPEG4_Decode_Start(channel);
Result := AngeloMPEG4_Decode_Stop(channel);
end;
```

AngeloMPEG4_Decode_Get_FlowRate–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int AngeloMPEG4_Decode_Get_FlowRate(int
     Decoder_Index, long* Byte_Second);
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Get_FlowRate(ByVal
     Decoder_Index As Long, ByRef flow_rate As
     Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Get_FlowRate(Decoder_Index:Lo
     ngint; var Byte_Second:Longint):Longint;
```

@ Argument

`Decoder_Index:` Indicates the channel number of MPEG4 Decoder. The range of channel is 0 - 15.

`Byte_Second:` The current flow rate of MPEG4 streaming measured in Byte/sec.

@ Return Code

0: ERROR_NoError

@ Example

< VC/BCB >

```
int Result;
int channel = 0;
long Byte_Second;
Result = AngeloMPEG4_Decode_Get_FlowRate(channel,
     &Byte_Second);
```

< Visual Basic >

```
Dim Result As Long, channel As Long, Byte_Second
     As Long
Result = AngeloMPEG4_Decode_Get_FlowRate(channel,
     Byte_Second)
```

<Delphi >

```
AngeloMPEG4_Decode_Get_FlowRate –
var
channel: Longint;
Byte_Second: Longint;
Result: Longint;
begin
channel := 0;
Result :=
     AngeloMPEG4_Decode_Get_FlowRate(channel,
     Byte_Second);
end;
```

AngeloMPEG4_Decode_ ReInitial–

AngeloMPEG4_Decode_ ReInitialEx–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int AngeloMPEG4_Decode_ReInitial(int
     Decoder_Index, int Quality, int
     Angelo_Video_Format);
int AngeloMPEG4_Decode_ReInitialEx(int
     Decoder_Index, int Bitrate, int frame_rate,
     int Angelo_Video_Format);
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_ReInitial (ByVal Decoder_Index
     As Long, ByVal Quality As Long, ByVal
     Video_Format As Long) As Long
AngeloMPEG4_Decode_ReInitialEx (ByVal
     Decoder_Index As Long, ByVal Bitrate As
     Long, ByVal frame_rate As Long, ByVal
     Video_Format As Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_ReInitial(Decoder_Index:Longi
     nt; Quality:Longint;
     Angelo_Video_Format:Longint):Longint;
AngeloMPEG4_Decode_ReInitialEx(Decoder_Index:Lon
     gint; Bitrate:Longint; frame_rate:Longint;
     Angelo_Video_Format:Longint):Longint;
```

@ Argument

`Decoder_Index`: Indicates the channel number of MPEG4 Decoder. The range of channel is 0 - 15.

`Quality:`

Quality	value	image 640*480	image 320*240	image 160*120

Lowest	-2	Bitrate = 320000*4
		frame_rate = 3
			Bitrate = 320000
			frame_rate = 3
				Bitrate = 240000/4
				frame_rate = 3

Low	-1	Bitrate = 400000*2
		frame_rate = 6
			Bitrate = 400000
			frame_rate = 6
				Bitrate = 400000/4
				frame_rate = 6

Normal	0	Bitrate = 480000*4
		frame_rate = 15
			Bitrate = 480000
			frame_rate = 15
				Bitrate = 480000/4
				frame_rate = 15

High	1	Bitrate = 512000*4
		frame_rate = 30
			Bitrate = 512000
			frame_rate = 30
				Bitrate = 512000/4
				frame_rate = 30

Highest	2	Bitrate = 1024000*4
		frame_rate = 30
			Bitrate = 1024000
			frame_rate = 30
				Bitrate = 1024000/4
				frame_rate = 30

Table 5-4: Video quality table

`Bitrate:` Indicates the bitrate of MPEG4 stream from encode server.

`Frame_rate:` Indicates the frame rate of MPEG4 stream from encode server. The values range is 0 - 30.

`Angelo_Video_Format:`

Full NTSC (640*480) = 0,

Full PAL (768*576) = 1,

CIF NTSC (320*240) = 2,

CIF PAL (384*288) = 3,

QCIF NTSC (160*120) = 4,

QCIF PAL (192*144) = 5,

@ Return Code

0: ERROR_NoError

@ Example

< VC/BCB >

```
int Result;
int channel = 0;
int Quality =0
int Bitrate = 480000;
int frame_rate = 15;
int Angelo_Video_Format = 2;
Result = AngeloMPEG4_Decode_ReInitia(channel,
      Quality, Angelo_Video_Format);
Result = AngeloMPEG4_Decode_ReInitialEx(channel,
      Bitrate, frame_rate, Angelo_Video_Format);
```

< Visual Basic >

```
Dim Result As Long, channel As Long, Quality As
      Long, Bitrate As Long, frame_rate As Long,
      Angelo_Video_Format As Long
channel = 0
Quality =0
Bitrate = 480000
frame_rate = 15
Angelo_Video_Format = 2
Result = AngeloMPEG4_Decode_ReInitia(channel,
      Quality, Angelo_Video_Format)
Result = AngeloMPEG4_Decode_ReInitiaEx(channel,
      Bitrate, frame_rate, Angelo_Video_Format)
```

< Delphi >

```
var
channel: Longint;
Quality, Bitrate, frame_rate,
      Angelo_Video_Format: Longint;
Result: Longint;
begin
channel := 0;
Quality :=0;
Bitrate := 480000;
frame_rate := 15;
Angelo_Video_Format = 2;
Result = AngeloMPEG4_Decode_ReInitial(channel,
      Quality, Angelo_Video_Format);
Result = AngeloMPEG4_Decode_ReInitialEx(channel,
      Bitrate, frame_rate, Angelo_Video_Format);
end;
```

AngeloMPEG4_Decode_ Save_File_Start–

AngeloMPEG4_Decode_ Save_File_Stop–

@ Syntax

C/C++ (Windows 98/NT/2000/XP/CE.NET)

```
int AngeloMPEG4_Decode_Save_File_Start(int
      Decoder_Index, char* n_file_name, long
      interval_second, long format);
int AngeloMPEG4_Decode_Save_File_Stop(int
      Decoder_Index);
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Save_File_Start (ByVal
      Decoder_Index As Long, ByVal n_file_name As
      String, ByVal interval_second As Long, ByVal
      format As Long) As Long
AngeloMPEG4_Decode_Save_File_Stop (ByVal
      Decoder_Index As Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Save_File_Start(Decoder_Index
      :Longint; n_file_name:String;
      interval_second:Longint;
      format:Longint):Longint;
AngeloMPEG4_Decode_Save_File_Stop(Decoder_Index:
      Longint):Longint;
```

@ Argument

`Decoder_Index`: Indicates the channel number of MPEG4 Decoder. The range of channel is 0 - 15.

`n_file_name`: The name to save the file to, excludes the extension of file name.

`interval_second`: Specify the save time for MPEG4 streaming.

format:

1: m4v,

2: avi.

3: both.

@ Return Code

0: ERROR_NoError

@ Example

< VC/BCB >

```
int Result;
int channel = 0;
char* n_file_name = “test”;
int interval_second = 10; //10 seconds
long format = 1 + 2; //save both file format
Result =
     AngeloMPEG4_Decode_Save_File_Start(channel,
     n_file_name, interval_second, format);
```

< Visual Basic >

```
Dim Result As Long, channel As Long,
     interval_second As Long, format As Long

channel = 0;
n_file_name = “test”
interval_second = 60
format = 3 ‘save both format
Result =
     AngeloMPEG4_Decode_Save_File_Start(channel,
     n_file_name, interval_second, format)
```

<Delphi >

```
var
channel: Longint;
n_file_name: String;
```

```
interval_second, format: Longint;
Result: Longint;
begin
channel := 0;
n_file_name := 'Video0';
interval_second := 10;
format := 3; // Save both format
Result :=
     AngeloMPEG4_Decode_Save_File_Start(channel,
     n_file_name, interval_second, format);
end;
```

AngeloMPEG4_Decode_File–

AngeloMPEG4_Decode_File_Start–

AngeloMPEG4_Decode_File_Set_Position–

AngeloMPEG4_Decode_File_Pause–

AngeloMPEG4_Decode_File_Continue–

AngeloMPEG4_Decode_File_Get_Position–

AngeloMPEG4_AVI_2_M4V–

AngeloMPEG4_M4V_2_AVI–

@ Syntax

C/C++ (Windows 98/NT/2000/XP/CE.NET)

```
int AngeloMPEG4_Decode_File(int
     Decoder_Index,char* file_name,unsigned
     long* iWidth,unsigned long*
     iHeight,unsigned long*
     Byte_Per_Pixel,unsigned long*
     Total_Frame,unsigned long* Time_Seconds);
```

```
int AngeloMPEG4_Decode_File_Start(int
      Decoder_Index,int Mode);
int AngeloMPEG4_Decode_File_Set_Position(int
      Decoder_Index, long* Frame_Index);
int AngeloMPEG4_Decode_File_Pause(int
      Decoder_Index);
int AngeloMPEG4_Decode_File_Continue(int
      Decoder_Index);
int AngeloMPEG4_Decode_File_Get_Position(int
      Decoder_Index, long* Cur_Frame_Index);
int AngeloMPEG4_AVI_2_M4V(char*
      file_name,unsigned long* iWidth,unsigned
      long* iHeight,unsigned long*
      Byte_Per_Pixel,unsigned long*
      Total_Frame,unsigned long* Time_Seconds);
int AngeloMPEG4_M4V_2_AVI(char*
      file_name,unsigned long* iWidth,unsigned
      long* iHeight,unsigned long*
      Byte_Per_Pixel,unsigned long*
      Total_Frame,unsigned long* Time_Seconds);
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_File (ByVal Decoder_Index As
      Long, ByVal file_name As String, iWidth As
      Long, iHeight As Long, Byte_Per_Pixel As
      Long, Total_Frame As Long, Time_Seconds As
      Long) As Long
AngeloMPEG4_Decode_File_Start (ByVal
      Decoder_Index As Long, ByVal Mode As Long)
      As Long
AngeloMPEG4_Decode_File_Set_Position (ByVal
      Decoder_Index As Long, Frame_Index As Long)
      As Long
AngeloMPEG4_Decode_File_Pause (ByVal
      Decoder_Index As Long) As Long
AngeloMPEG4_Decode_File_Continue (ByVal
      Decoder_Index As Long) As Long
AngeloMPEG4_Decode_File_Get_Position (ByVal
      Decoder_Index As Long, Cur_Frame_Index As
      Long) As Long
AngeloMPEG4_AVI_2_M4V (ByVal file_name As String,
      iWidth As Long, iHeight As Long,
          Byte_Per_Pixel As Long, Total_Frame As Long,
          Time_Seconds As Long) As Long
     AngeloMPEG4_M4V _2_ AVI (ByVal file_name As
          String, iWidth As Long, iHeight As Long,
          Byte_Per_Pixel As Long, Total_Frame As Long,
          Time_Seconds As Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_File(Decoder_Index:Longint;
     file_name:String; var iWidth:Longint; var
     iHeight:Longint; var
     Byte_Per_Pixel:Longint; var
     Total_Frame:Longint; var
     Time_Seconds:Longint):Longint;
AngeloMPEG4_Decode_File_Start(Decoder_Index:Long
     int; Mode:Longint):Longint;
AngeloMPEG4_Decode_File_Set_Position(Decoder_Ind
     ex:Longint; var
     Frame_Index:Longint):Longint;
AngeloMPEG4_Decode_File_Pause(Decoder_Index:Long
     int):Longint;
AngeloMPEG4_Decode_File_Continue(Decoder_Index:L
     ongint):Longint;
AngeloMPEG4_Decode_File_Get_Position(Decoder_Ind
     ex:Longint; var
     Cur_Frame_Index:Longint):Longint;
AngeloMPEG4_AVI_2_M4V(file_name:String; var
     iWidth:Longint; var iHeight:Longint; var
     Byte_Per_Pixel:Longint; var
     Total_Frame:Longint; var
     Time_Seconds:Longint):Longint;
AngeloMPEG4_M4V_2_AVI(file_name:String; var
     iWidth:Longint; var iHeight:Longint; var
     Byte_Per_Pixel:Longint; var
     Total_Frame:Longint; var
     Time_Seconds:Longint):Longint;
```

@ Argument

`Decoder_Index`: Indicates the channel number of MPEG4 Decoder. The range of channel is 0 - 15.

file_name: The name of file to save to, includes the path and extension of file name.

iWidth: Indicate the width of the MPEG4 image size.

iHeight: Indicates the height of the MPEG4 image size.

Byte_Per_Pixel: Number of Bytes per Pixel

Total_Frame: Number of frames in the MPEG4 file.

Time_Seconds: The total time of the MPEG4 file in seconds.

Mode: The play mode of the Mpeg4 file

0: Play once

1: Repeat

Frame_Index: Zero index of the frame

Cur_Frame_Index: Current frame index

PlayFactor: The speed to play the MPEG4 file

1: Normal

2: 2x faster

-2: 2x slower

@ Return Code

0: ERROR_NoError

@ Example

< VC/BCB >

```
int Result;
int m_Decoder_Channel = 0;
long Width=0;
long Height=0;
long Byte_Pixel=0;
long m_total_frame=0;
long m_Time_Seconds=0;
long m_pos=0;
long Mode = 0; //play once
char* m_filename = “test1.m4v”;
```

```
char* m4v_filename = “test2.m4v”;
char* avi_filename = “test3.avi”;

void CM4VPlayerView::MediaStreamProc( int
      Decoder_Channel ,long int_status,long
      VideoBufferaddress )
{
      …
      …
}

AngeloMPEG4_Decode_Set_Callback(m_Decoder_Channe
      l,MediaStreamProc);
AngeloMPEG4_Decode_File(m_Decoder_Channel,m_file
      name,&Width,&Height,&Byte_Pixel,&m_total_fr
      ame,&m_Time_Seconds);
AngeloMPEG4_Decode_File_Start(m_Decoder_Channel,
      Mode);
AngeloMPEG4_Decode_File_Set_Position(m_Decoder_C
      hannel,& m_total_frame/2);
AngeloMPEG4_Decode_File_Pause(m_Decoder_Channel)
      ;
AngeloMPEG4_Decode_File_Continue(m_Decoder_Chann
      el);
AngeloMPEG4_Decode_File_Get_Position(m_Decoder_C
      hannel,&m_pos);
AngeloMPEG4_Decode_Stop(m_Decoder_Channel);
AngeloMPEG4_Decode_M4V_2_AVI(m4v_filename,&Width
      ,&Height,&Byte_Pixel,&m_total_frame,&m_Time
      _Seconds);
AngeloMPEG4_Decode_AVI_2_M4V(avi_filename,&Width
      ,&Height,&Byte_Pixel,&m_total_frame,&m_Time
      _Seconds);
```

< Visual Basic >

```
Dim Result As Long, m_Decoder_Channel As Long,
      Width As Long, Height As Long, Byte_Pixel As
      Long, m_total_frame As Long, m_Time_Seconds
      As Long, m_pos As Long
Dim m_filename As String, m4v_filename As String,
      avi_filename As String,
m_filename = "test1.m4v"
m4v_filename = "test2.m4v"
avi_filename = "test3.avi"
m_Decoder_Channel = 0
Mode = 0 'play once

Public Sub lpcallback(ByVal Decoder_Index As
     Long, ByVal int_status As Long, ByVal
     VideoBufferaddress As Long)
…
…
End Sub

Result =
     AngeloMPEG4_Decode_Set_Callback(m_Decoder_C
     hannel, AddressOf lpcallback)

Result=AngeloMPEG4_Decode_File(m_Decoder_Channel
     ,m_filename,Width,Height,Byte_Pixel,m_total
     _frame,m_Time_Seconds)
Result =
     AngeloMPEG4_Decode_File_Start(m_Decoder_Cha
     nnel,Mode)
Result =
     AngeloMPEG4_Decode_File_Set_Position(m_Deco
     der_Channel,m_total_frame/2)
Result =
     AngeloMPEG4_Decode_File_Pause(m_Decoder_Cha
     nnel)
Result =
     AngeloMPEG4_Decode_File_Continue(m_Decoder_
     Channel)
Result =
     AngeloMPEG4_Decode_File_Get_Position(m_Deco
     der_Channel,m_pos)
Result =
     AngeloMPEG4_Decode_Stop(m_Decoder_Channel)
Result=AngeloMPEG4_Decode_M4V_2_AVI(m4v_filename
     ,Width,Height,Byte_Pixel,m_total_frame,m_Ti
     me_Seconds)
Result=AngeloMPEG4_Decode_AVI_2_M4V(avi_filename
     ,Width,Height,Byte_Pixel,m_total_frame,m_Ti
     me_Seconds)
```

< Delphi >

```
procedure
     DecoderCallbackProc(Decoder_Index:Longint;i
     nt_status:Longint;VideoBufferaddress:Longin
     t); stdcall
var
    Str_Addr : Pointer;
    Bitrate, Framerate, colorspace,videoformat :
     Longint;
begin
          …
          …

end;
…
…


var
m_filename, m4v_filename, avi_filename: String;
Result, m_Decoder_Channel, Width, Height,
     Byte_Pixel, m_total_frame, m_Time_Seconds,
     m_pos, Mode: Longint;

begin
m_Decoder_Channel:= 0;
Mode := 0; //play once
m_filenam := 'test1.m4v';
m4v_filename:= 'test2.m4v';
avi_filename := 'test3.avi';

Result :=
     AngeloMPEG4_Decode_Set_Callback(m_Decoder_C
     hannel, DecoderCallbackProc);

Result
     :=AngeloMPEG4_Decode_File(m_Decoder_Channel
     ,m_filename,Width,Height,Byte_Pixel,m_total
     _frame,m_Time_Seconds);
Result :=
     AngeloMPEG4_Decode_File_Start(m_Decoder_Cha
     nnel,Mode);
```

```
Result :=
      AngeloMPEG4_Decode_File_Set_Position(m_Deco
      der_Channel,m_total_frame div 2);
Result :=
      AngeloMPEG4_Decode_File_Pause(m_Decoder_Cha
      nnel);
Result :=
      AngeloMPEG4_Decode_File_Continue(m_Decoder_
      Channel);
Result :=
      AngeloMPEG4_Decode_File_Get_Position(m_Deco
      der_Channel,m_pos);
Result :=
      AngeloMPEG4_Decode_Stop(m_Decoder_Channel);
Result:=AngeloMPEG4_Decode_M4V_2_AVI(m4v_filenam
      e,Width,Height,Byte_Pixel,m_total_frame,m_T
      ime_Seconds);
Result:=AngeloMPEG4_Decode_AVI_2_M4V(avi_filenam
      e,Width,Height,Byte_Pixel,m_total_frame,m_T
      ime_Seconds);
end;
```

5.4 System Functions

@ Name

AngeloMPEG4_Get_Version(lMpeg4_DLLVersion, AngeloRTV_DLLVersion, Reserved)

@ Description

AngeloMPEG4_Get_Version: Use this function to get the software information.

AngeloMPEG4_Get_Version –

@ Syntax

C/C++ (Windows 98/NT/2000/XP/CE.NET)

```
int AngeloMPEG4_Get_Version(long
      *Mpeg4_DLLVersion, long
      *AngeloRTV_DLLVersion, long *Reserved);
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Get_Version(var
      Mpeg4_DLLVersion:Longint; var
      AngeloRTV_DLLVersion:Longint; var
      Reserved:Longint):Longint;
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Get_Version(ByRef
      AngeloMpeg4_DLLVersion As Long, ByRef
      AngeloRTV_DLLVersion As Long, ByRef Reserved
      As Long) As Long
```

@ Argument

`Mpeg4_DLLVersion:` Indicates the current version of the MPEG4 DLL. It is of 4 rows in length.

`AngeloRTV_DLLVersion:` Indicates the current version of AngeloRTV DLL. It is of 4 rows in length.

@ Return Code

0: ERROR_NoError

@ Example

< VC/BCB >

```
int Result;
long Mp4Version[4] = {0}, DLLVersion[4] = {0},
     VersionReserved[4] = {0};
CString str1, str2;
Result = AngeloMPEG4_Get_Version(Mp4Version,
     DLLVersion, VersionReserved);
str1.Format("%d.%d.%d.%d", DLLVersion[0],
     DLLVersion[1], DLLVersion[2],
     DLLVersion[3]);
str2.Format("%d.%d.%d.%d", Mp4Version[0],
     Mp4Version[1], Mp4Version[2],
     Mp4Version[3]);
```

< Visual Basic >

```
Dim Result As long, Mp4Version(0 to 3) As Long,
     DLLVersion(0 to 3) As Long,
     VersionReserved(0 to 3) As Long
Result = AngeloMPEG4_Get_Version(Mp4Version(0),
     DLLVersion(0), VersionReserved(0))
```

< Delphi >

```
var
Mpeg4_DLLVersion : array[0..3] of Longint;
AngeloRTV_DLLVersion : array[0..3] of Longint;
Reserved : array[0..3] of Longint;
```

```
Result: Longint;
Str_AngeloMPEG4_Version, Str_AngeloRTV_Version:
     String;
begin
Result :=
     AngeloMPEG4_Get_Version(Mpeg4_DLLVersion[0]
     , AngeloRTV_DLLVersion[0], Reserved[0]);
Str_AngeloMPEG4_Version :=
     IntToStr(Mpeg4_DLLVersion[0]);
Str_AngeloMPEG4_Version :=
     Str_AngeloMPEG4_Version + “.” +
     IntToStr(Mpeg4_DLLVersion[1]);
Str_AngeloMPEG4_Version :=
     Str_AngeloMPEG4_Version + “.” +
     IntToStr(Mpeg4_DLLVersion[2]);
Str_AngeloMPEG4_Version :=
     Str_AngeloMPEG4_Version + “.” +
     IntToStr(Mpeg4_DLLVersion[3]);
Str_AngeloRTV_Version :=
     IntToStr(AngeloRTV_DLLVersion[0]);
Str_AngeloRTV_Version := Str_AngeloRTV_Version +
     “.” + IntToStr(AngeloRTV_DLLVersion[1]);
Str_AngeloRTV_Version := Str_AngeloRTV_Version +
     “.” + IntToStr(AngeloRTV_DLLVersion[2]);
Str_AngeloRTV_Version := Str_AngeloRTV_Version +
     “.” + IntToStr(AngeloRTV_DLLVersion[3]);
end;
```

Appendix

Appendix A: Glossary

Brightness:

Attribute of a visual sensation according to which an area appears to exhibit more or less light

CCIR:

Committee Consulat International Radiotelegraphique. This is a standards committee of the International Telecommunications Union, which made the technical recommendation for European 625 line standard for video signals.

Composite Video:

Composite video (CVS/CVBS) signal carries video picture information for color, brightness, and synchronizing signals for both horizontal and vertical scans.

CIF:

CIF has 352(H) x 288(V) luminance pixels, and 176(H) x 144(V) chrominance pixels. QCIF is a similar picture format with one-quarter the size of CIF.

EIA:

Electronic Industry Association. An industry lobbying group; it collects statistics and establishes testing standards for many types of home electronics.

Field:

For interlaced video the total picture is divided into two fields, one even and one odd, each containing one half of the total vertical information. Each field takes one sixtieth of a second (one fiftieth for PAL) to complete. Two fields make a complete frame of video.

Frame:

One frame (two fields) of video contains the full vertical interlaced information content of the picture. For NTSC this consists of 525 lines and PAL a frame is consisted of 625 lines.

Gamma:

Cathode ray tubes (CRTs) do not have a linear relationship between brightness and the input voltage applied. To compensate for this non-linearity, a pre distortion or gamma correction is applied, generally at the camera source. A value of gamma equal to 2.2 is typical, but can vary for different CRT phosphors.

Hue:

Attribution of visual sensation according to which area appears to be similar to one, or proportions of two, of the perceived colors red, yellow, green, and blue.

NTSC:

Color TV standard developed in the U.S. in 1953 by National Television System Committee. NTSC is used in United States, Canada, Japan, in most of the American continent countries and in various Asian countries. The rest of the world uses either some variety of PAL or SECAM standards.

NTSC runs on 525 lines/frame and it's vertical frequency is 60Hz. NTSC's framerate is 29, 97 frames/sec.

PAL:

PAL (Phase Alternating Line) TV standard was introduced in the early 1960's in Europe. It has better resolution than in NTSC, having 625 lines/frame, but the frame rate is slightly lower - 25 frames/sec. PAL is used in most of the western European countries (except France, where SECAM is used instead), Australia, various countries in Africa and in South America and in some Asian countries. There are various versions of PAL, the most commonly used method is PAL B/G, but others include PAL I (used in the UK and in

Ireland) and PAL M (hybrid standard, having the same resolution as NTSC, but uses PAL transmission and color coding technology).

Saturation:

A characteristic describing color amplitude or intensity. A color of a given hue may consist of low or high saturation value, which relates to the vividness of color.

AGC

Abbreviation for automatic gain control. On a TV or VCR, AGC is a circuit that automatically adjusts the incoming signal to the proper levels for display or recording. On a video camera, AGC is a circuit that automatically adjusts the sensitivity of the pickup tube to render the most pleasing image.

Appendix B: Standard Compliance

Notice for USA

Compliance Information Statement (Declaration of Conformity Procedure) DoC FCC Part 15

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules.

These limits are designed to provide reasonable protection against harmful interference in a residential installation or when the equipment is operated in a commercial environment.

This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

Notice for Europe

This product is in conformity with the Council Directive 89/336/EEC amended by 92/31/EEC and 93/68/EEC

This equipment has been tested and found to comply with EN55022/CISPR22 and EN55024/CISPR24. To meet EC requirements, shielded cables must be used to connect a peripheral to the card. This product has been tested in a typical class B compliant host system. It is assumed that this product will also achieve compliance in any class A compliant unit.

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 two-year guarantee:

    - The warranty period starts from the product's shipment date 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 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 two-year 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 technicians.
   - Products with altered and/or damaged serial numbers are not entitled to our service.
   - 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.