RTV Series

Multi-Channel Real-Time Video

Frame Grabber Series

User's Manual

Manual Rev. 2.03

Revision Date: April 22, 2009

Part No: 50-1R001-1010

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

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: Chipset:	 CPU: Bios:

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

Table of Contents

List of Tables

List of Figures

1 Introduction

The RTV series acquisition board is designed without compromise for security and video surveillance applications as a PC-based multiple channel digital video recorder.

This 32-bit/64bit, 33MHz/66MHz PCI/cPCI/PMC bus frame grabber simultaneously captures four video analog streams in real-time. It accepts standard composite color (PAL, NTSC) or monochrome video formats (CCIR, EIA).

The square-pixel and broadcast resolutions are programmable (640 x 480 or 768 x 576). Before images are transferred into the PC's memory, the resolution can be scaled down using selectable ratios.

Arbitrary cropping to regions of interest is supported. The RTV series generates bitmaps in all popular color formats such as RGB.

System integrators will benefit from a watchdog timer for fault-tolerant applications and from the easy-to-use standard connectors.

1.1 Features

1.1.1 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: RTV Series Acquisition Speed

Note: The PMC-RTV21 is capable of only up to 30 frames (60 fields) in total 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.

- 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.

1.1.2 I/O Lines

The RTV series is fitted with TTL compatible I/O lines protected against overloads and electrostatic discharges. Each line may be configured as an input or output. They can be used to trigger acquisition or report alarm signals.

1.1.3 Watchdog Timer

A hardware watchdog is available on the RTV-24 that 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.

1.1.4 Supported Software

WDM driver

The drivers support VC++ / VB / Delphi / C++ Builder programming under Windows NT/98/2000/XP. DLLs and reference sample programs are provided.

ViewCreator

The package will assist in initial test and functional evaluation.

AngeloLVIEW - Angelo-LVIEW is fully compatible with LabView™ 6.0 and above and it provides a full set of VIs that can be used

with the Angelo RTV series (RTV-24, cRTV-24, cRTV-44 and PMC-RTV21/G). VIs for Windows 98/NT/2000/XP operation systems and LabView™ sample programs are provided for users' reference.

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 Requirements

The minimum system requirements for 4-CH real-time NTSC*/ PAL** color image acquisition are:

- Platform: Pentium 4, 2.4GHz CPU, 256MB DDRAM above.
- VGA display: AGP 4X or above (VIA or SiS VGA chipset **NOT** recommended).
- Display setting: 800 x 600 resolution or above, 16-bit color or above.
- OS: if using Windows 2000, please upgrade to Service Pack 4.0 or above.

Note: Lower system configurations will lower acquisition performance.

Note: Please refer to section 1.4 RTV-24 Benchmark for the performance issues due to PCI bus bandwidth limitations.

* NTSC real-time color images – Provides 640 x 480 pixel image resolution at the RGB 16-bit color format. Each channel acquires 30 frames per second with 4-CH totaling up to 120 frames per second.

** PAL real-time color images – Provides 768 x 576 pixel image resolution at the RGB 16-bit color format. Each channel acquires 25 frames per second with 4-CH totaling up to 100 frames per second.

1.4 RTV-24 Benchmarks

Motherboard: ASUS P5E64 WS EVOLUTION

CPU: Intel Core2 Duo CPU E4600 @ 2.4GHz

RAM: DDR3_SDRAM 2GB

OS: Windows XP /SP3

Image Format	**RGB16, Full(640*480)**							
Card#	Card 0				Card 1			
Port#	0	1	2	3	0	1	2	3
Real-Time	✓	✓	✓	✓	×	×	×	×
Frame Rate	29.814	29.813	29.813	29.815				

Motherboard: ASUS P5E64 WS EVOLUTION

CPU: Intel Core2 Duo CPU E4600 @ 2.4GHz

RAM: DDR3_SDRAM 2GB

OS: Windows XP /SP3

Image Format	**RGB24, Full(640*480)**							
Card#	Card 0				Card 1			
Port#	0	1	2	3	0	1	2	3
Real-Time	✓	✓	✓	×	×	×	×	×
Frame Rate	29.814	29.815	29.815					

Motherboard: NuPRO-965

CPU: Intel Core2 Quad Q6600 @ 2.4GHz

RAM: DDR2_SDRAM 2GB

OS: Windows XP /SP3

Image Format	**RGB16, CIF(320*240)**							
Card#	**Card 0**				Card 1			
Port#	**0**	1	2	3	0	1	2	3
Real-Time	✓	✓	✓	✓	✓	✓	✓	✓
Frame Rate	**29.966**	29.960	29.964	29.958	29.961	29.958	29.966	29.964
Card#	**Card 2**				Card 3			
Port#	0	1	2	3	0	1	2	3
Real-Time	✓	✓	✓	✓	×	×	×	×
Frame Rate	29.943	29.883	29.927	29.833				

Motherboard: NuPRO-965

CPU: Intel Core2 Quad Q6600 @ 2.4GHz

RAM: DDR2_SDRAM 2GB

OS: Windows XP /SP3

Image Format	**RGB24, CIF(320*240)**							
Card#	**Card 0**				Card 1			
Port#	**0**	1	2	3	0	1	2	3
Real-Time	✓	✓	✓	✓	✓	✓	✓	✓
Frame Rate	**29.966**	29.963	29.966	29.963	29.966	29.963	29.966	29.966
Card#	**Card 2**				Card 3			
Port#	0	1	2	3	0	1	2	3
Real-Time	×	×	×	×	×	×	×	×
Frame Rate								

1.5 PCIe-RTV-24 Benchmarks

Motherboard: ASUS P5E64 WS EVOLUTION

CPU: Intel Core2 Duo CPU E4600 @ 2.4GHz

RAM: DDR3_SDRAM 2GB

OS: Windows XP /SP3

Image Format	**RGB16, Full(640*480)**							
Card#	Card 0				Card 1			
Port#	0	1	2	3	0	1	2	3
Real-Time	✓	✓	✓	✓	✓	✓	✓	✓
Frame Rate	29.811	29.798	29.810	29.808	29.807	29.805	29.808	29.801

Motherboard:ASUS P5E64 WS EVOLUTION

CPU: Intel Core2 Duo CPU E4600 @ 2.4GHz

RAM: DDR3_SDRAM 2GB

OS: Windows XP /SP3

Image Format	**RGB24, Full(640*480)**							
Card#	Card 0				Card 1			
Port#	0	1	2	3	0	1	2	3
Real-Time	✓	✓	✓	×	✓	✓	✓	×
Frame Rate	29.808	29.811	29.808		29.808	29.814	29.809	

Motherboard: NuPRO-965

CPU: Intel Core2 Quad Q6600 @ 2.4GHz

RAM: DDR2_SDRAM 2GB

OS: Windows XP /SP3

Image Format	**RGB16, CIF(320*240)**							
Card#	**Card 0**				Card 1			
Port#	**0**	1	2	3	0	1	2	3
Real-Time	✓	✓	✓	✓	✓	✓	✓	✓
Frame Rate	**29.810**	29.09	29.810	29.809	29.809	29.810	29.808	29.809
Card#	**Card 2**				Card 3			
Port#	0	1	2	3	0	1	2	3
Real-Time	✓	✓	✓	✓	✓	✓	✓	✓
Frame Rate	29.810	29.809	29.809	29.809	29.809	29.810	29.809	29.810

Motherboard: NuPRO-965

CPU: Intel Core2 Quad Q6600 @ 2.4GHz

RAM: DDR2_SDRAM 2GB

OS: Windows XP /SP3

Image Format	**RGB24, CIF(320*240)**							
Card#	**Card 0**				Card 1			
Port#	**0**	1	2	3	0	1	2	3
Real-Time	✓	✓	✓	✓	✓	✓	✓	✓
Frame Rate	**29.810**	29.09	29.811	29.809	29.809	29.811	29.807	29.809
Card#	**Card 2**				Card 3			
Port#	0	1	2	3	0	1	2	3
Real-Time	✓	✓	✓	✓	✓	✓	✓	✓
Frame Rate	29.809	29.808	29.809	29.809	29.806	29.810	29.807	29.810

2 Hardware Reference

2.1 RTV Series

2.1.1 PCIe-RTV24 Specifications

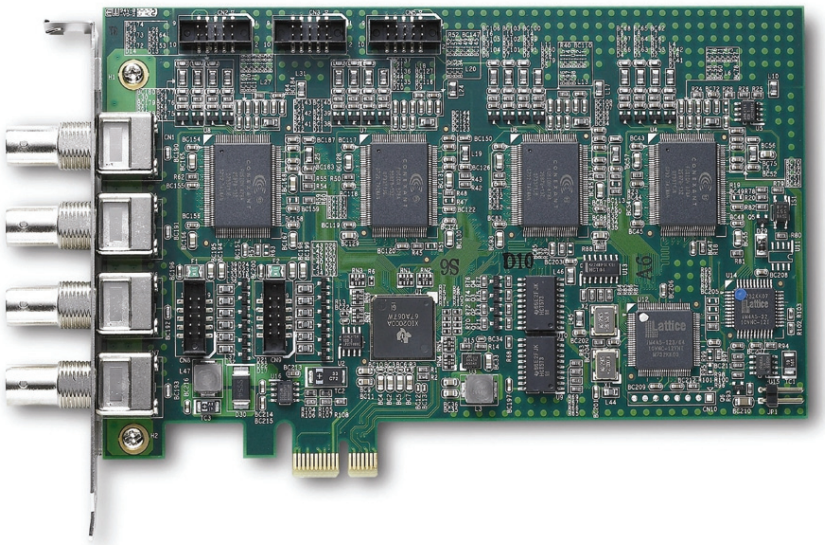


Figure 2-1: PCIe-RTV24 Appearance

Dimensions

- W x L:167.65 (mm) x 111.15 (mm)

Operating Environment

- Temperature: 0 to 55°C
- Humidity: 5 to 90% RHNC

Storage Environment

- Temperature: 0 to 70°C
- Humidity: 0 to 95% RHNC

Power Requirements

- +12 V max. 0.7A
- +3.3 V max. 0.5A
- Aux +3.3V max. 0.003A

Video Input

- Four composite video color digitizers
- Video input interface: Four composite BNC connectors
- Coaxial cable suggested

Channel Extension

- Expandable to up to 16 channels
- Channel extension interface:
  - 10-pin ribbon cable to on-board 10-pin header connector for channel extension, each header adds 4 video inputs channels
  - Three 10-pin header connectors on-board

General Purpose I/O Lines

- All I/Os are TTL compatible and support 4 inputs, 4 outputs, and 4 soft trigger lines
- GPIO interface:
  - Two 10-pin header connectors on-board
  - The I/O lines are internally pulled up and have the following characteristics:

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: GPIO Characteristics

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

4-channel software trigger output

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

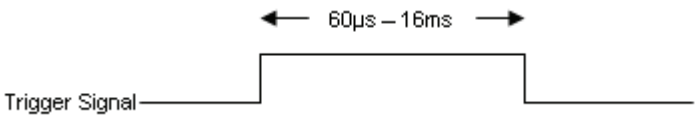


Figure 2-2: Trigger Signal Waveform

User EEPROM

- Includes 1kbit available EEPROM

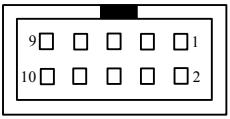
RTV-24 Standard Accessories

- Watchdog reset cable
- GPIO bracket
- User Manual
- All in One CD

RTV-24 Connectors & Pin Definitions

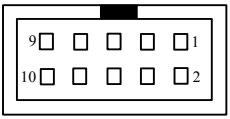
Connector	Definition
	Video IN – CH 0
	Video IN – CH 1
	Video IN – CH 2
	Video IN – CH 3

Table 2-2: RTV Video Inputs



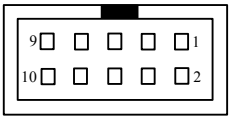
PIN	Function	PIN	Function
1	GND	2	CH4 video in
3	CH5 video in	4	GND
5	GND	6	CH6 video in
7	CH7 video in	8	GND
9	GND	10	GND

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



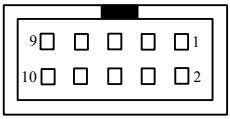
PIN	Function	PIN	Function
1	GND	2	CH8 video in
3	CH9 video in	4	GND
5	GND	6	CH10 video in
7	CH11 video in	8	GND
9	GND	10	GND

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



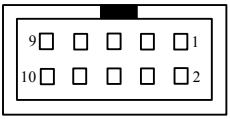
PIN	Function	PIN	Function
1	GND	2	CH12 video in
3	CH13 video in	4	GND
5	GND	6	CH14 video in
7	CH15 video in	8	GND
9	GND	10	GND

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



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)



PIN	Function	PIN	Function
1	IN2 (External interrupt)	2	GND
3	OUT0	4	Software Trigger 2
5	IN3 (External interrupt)	6	Software Trigger 3
7	OUT1	8	+5V
9	GND	10	--

Table 2-7: GPIO (CN9)

PIN	Function
1	System reset
2	GND

Table 2-8: Watchdog Timer

2.1.2 RTV-24 Specifications

Video Input

- Four composite video color digitizers
- Video input interface: Four composite BNC connectors
- Coaxial cable suggested

Channel Extension

- Expandable to up to 16 channels
- Channel extension interface:
  - 10-pin ribbon cable to on-board 10-pin header connector for channel extension, each header adds 4 video inputs channels
  - Three 10-pin header connectors on-board

General Purpose I/O Lines

- All I/Os are TTL compatible and support 4 inputs, 4 outputs, and 4 soft trigger lines
- GPIO interface:
  - Two 10-pin header connectors on-board
  - The I/O lines are internally pulled up and have the following characteristics:

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 2-9: GPIO Characteristics

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

4-channel software trigger output

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

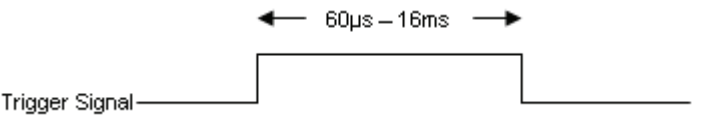


Figure 2-3: Trigger Signal Waveform

User EEPROM

- Includes 1kbit available EEPROM

Form Factor

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

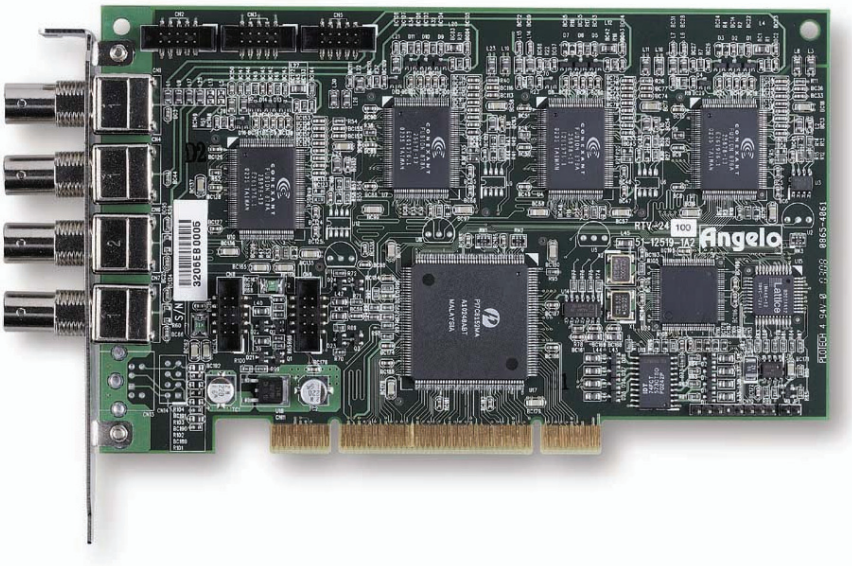


Figure 2-4: RTV-24 Appearance

Dimensions

- W x L: 106.68(mm) x 174.62(mm)

Operating Environment

- Temperature: 0 to 55°C
- Humidity: 5 to 90% RHNC

Storage Environment

- Temperature: 0 to 70°C
- Humidity: 0 to 95% RHNC

Power Requirements

- +5V max. 1.5A
- +3.3 V max. 0.5A

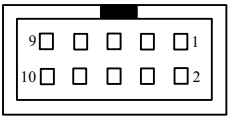
RTV-24 Standard Accessories

- Watchdog reset cable
- GPIO bracket
- User Manual
- All in One CD

RTV-24 Connectors & Pin Definitions

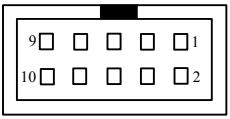
Connector	Definition
	Video IN – CH 0
	Video IN – CH 1
	Video IN – CH 2
	Video IN – CH 3

Table 2-10: RTV Video Inputs



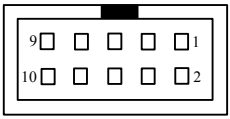
PIN	Function	PIN	Function
1	GND	2	CH4 video in
3	CH5 video in	4	GND
5	GND	6	CH6 video in
7	CH7 video in	8	GND
9	GND	10	GND

Table 2-11: Channel Extension Video Input (CN2)



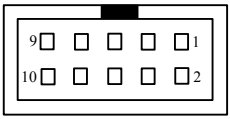
PIN	Function	PIN	Function
1	GND	2	CH8 video in
3	CH9 video in	4	GND
5	GND	6	CH10 video in
7	CH11 video in	8	GND
9	GND	10	GND

Table 2-12: Channel Extension Video Input (CN3)



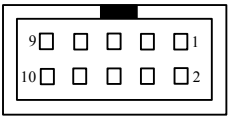
PIN	Function	PIN	Function
1	GND	2	CH12 video in
3	CH13 video in	4	GND
5	GND	6	CH14 video in
7	CH15 video in	8	GND
9	GND	10	GND

Table 2-13: Channel Extension Video Input (CN5)



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 2-14: GPIO (CN8)



PIN	Function	PIN	Function
1	IN2 (External interrupt)	2	GND
3	OUT0	4	Software Trigger 2
5	IN3 (External interrupt)	6	Software Trigger 3
7	OUT1	8	+5V
9	GND	10	--

Table 2-15: GPIO (CN9)

PIN	Function
1	System reset
2	GND

Table 2-16: Watchdog Timer

2.1.3 RTV-E4 Extension Board (Optional)

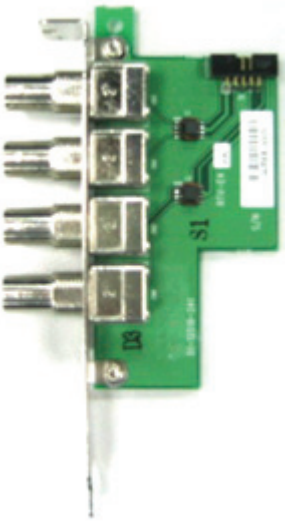
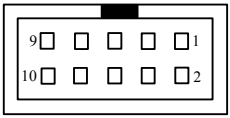


Figure 2-5: RTV-E4 Appearance

RTV-E4 Connectors & Pin Definitions



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 2-17: Channel Extension Video Input (CN11)

2.1.4 RTV-I4 Isolation GPIO Board (Optional)

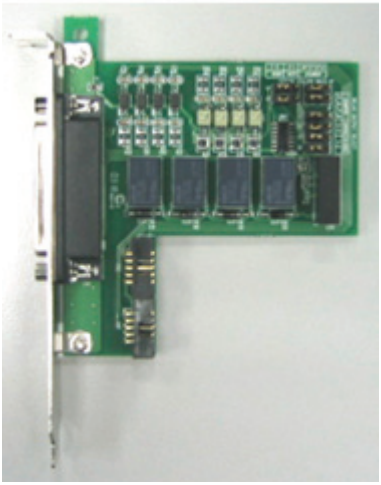


Figure 2-6: RTV-I4 Appearance

RTV-I4 Connectors & 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

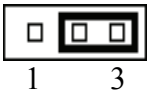
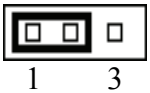
Normal Open	Normal Closed
	

Table 2-18: Relay Jumper Settings

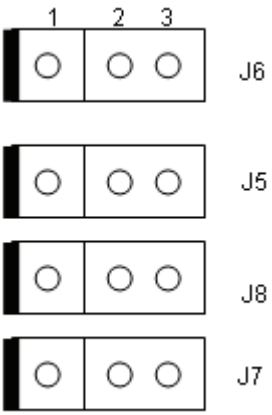


Figure 2-7: Relay Address Jumpers

Relay I/O voltage requirements

- 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
- Trigger output voltage: 0V to +5V
- Type select: Active high =>2-3, Active low =>1-2

Active High	Active Low
	

Table 2-19: STRG Jumper Settings

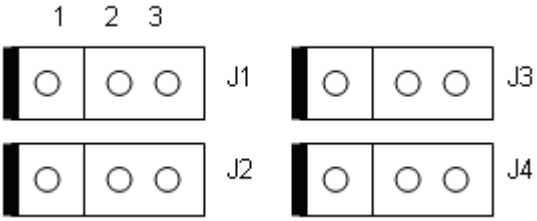
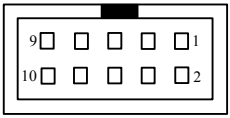


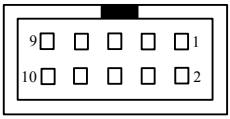
Figure 2-8: STRG Address Jumpers

2R10P Input Pin Header Definitions



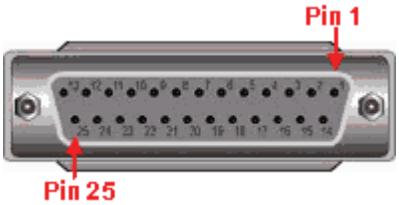
PIN	Function	PIN	Function
1	GPIO Input 1	2	GND
3	GPIO Output 1	4	PORT1 STRG Output
5	GPIO Input 2	6	PORT2 STRG Output
7	GPIO Output 2	8	VCC
9	GND	10	--

Table 2-20: RTV-I4 GPIO (CN1) <--> RTV-24 GPIO (CN8)



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	--

Table 2-21: RTV-I4 GPIO (CN2) <--> RTV-24 GPIO (CN9)



PIN	Signal	PIN	Signal
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 2-22: D-sub 25-pin Connector

2.2 cRTV Series

2.2.1 cRTV-24 Specifications

Video Input

- Four composite video color digitizers
- Video input interface: Four composite BNC connectors
- Channel status report LED
- Coaxial cable recommended

Channel Extension

- Expandable to up to 8 channels
- Channel extension interface
  - 10-pin ribbon cable to on-board 10-pin header connector for channel extension, each header adds 4 video inputs channels

User EEPROM

- Includes 1kbit usable EEPROM

Form Factor

- 32/64bit, 33/66MHz, 3U Compact PCI board

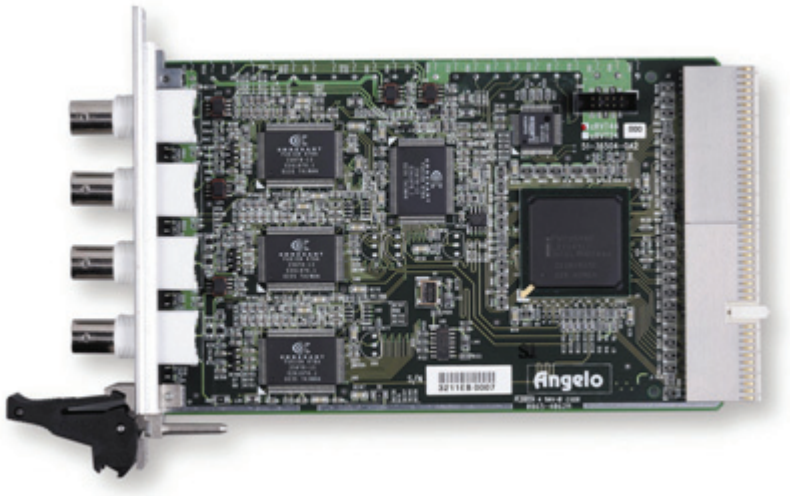


Figure 2-9: cRTV-24 Appearance

Dimensions

- W x L: 160(mm) x 100(mm)

Operating Environment

- Temperature: 0 to 55°C
- Humidity: 5 to 90% RHNC

Storage Environment

- Temperature: 0 to 70°C
- Humidity: 0 to 95% RHNC

Power Requirements

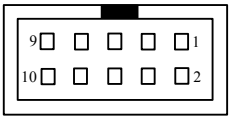
- +5V max. 1.5A
- +3.3 V max. 0.65A

cRTV-24 Standard Accessories

- User Manual
- All in One CD

Connector	Definition
	CH0 (Channel 0 BNC)
	CH1 (Channel 1 BNC)
	CH2 (Channel 2 BNC)
	CH3 (Channel 3 BNC)

Table 2-23: cRTV Video Inputs



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 2-24: Channel Extension Video Input (CN8)

2.2.2 cRTV-44 Specifications

Video Input

- Four composite video color digitizers
- Video input interface: Four composite BNC connectors
- Channel status report LED
- Coaxial cable recommended

General Purpose I/O Lines

- All I/O lines are TTL compatible with 4 input, 4 output, and 4 soft trigger lines.
- GPIO interface:
  - Two 10-pin header connectors on-board
  - The I/O lines are internally pulled up and have the following characteristics:

Voltage	MIN	MAX
Input high voltage (20μA)	2.0V	5.25V
Input low voltage (-0.2μA)	0.0V	0.80V
Output high voltage (-1.0mA)	5.0V	-
Output low voltage (100.0mA)	-	0.5V

Table 2-25: GPIO Characteristics

Channel Extension

- Expandable to up to 8 channels
- Channel extend interface
  - 10-pin ribbon cable to on-board 10-pin header connector for channel extension, each header adds 4 video inputs channels.

User EEPROM

- Includes 1kbit usable EEPROM

Form Factor

- 32/64bit, 33/66MHz, 6U Compact PCI board

Figure 2-10: cRTV-44 Appearance

Dimensions

- W x L: 160(mm) x 233.35(mm)

Operating Environment

- Temperature: 0 to 55°C
- Humidity: 5 to 90% RHNC

Storage Environment

- Temperature: 0 to 70°C
- Humidity: 0 to 95% RHNC

Power Requirements

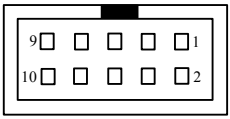
- +5V max. 1.5A
- +3.3 V max. 0.65A

cRTV-44 Standard Accessories

- User Manual
- All in One CD

Connector	Definition
	CH0 (Channel 0 BNC)
	CH1 (Channel 1 BNC)
	CH2 (Channel 2 BNC)
	CH3 (Channel 3 BNC)

Table 2-26: cRTV Video Inputs

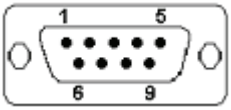


PIN	Function	PIN	Function
1	GND	2	CH4 video in
3	CH5 video in	4	GND
5	GND	6	CH6 video in
7	CH7 video in	8	GND
9	GND	10	GND

Table 2-27: Channel Extension Video Input (CN8)

GPIO 0

- Pins IN0 and OUT0 are used by channel 0
- Pins IN1 and OUT1 are used by channel 1

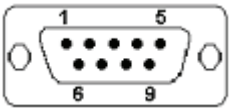


PIN	Function	PIN	Function
1	IN0 (External interrupt)	6	GND
2	OUT0	7	GND
3	IN1 (External interrupt)	8	GND
4	OUT1	9	+5V
5	GND		

Table 2-28: GPIO 0 Pinout

GPIO 1

- Pins IN2 and OUT2 are for channel 2
- Pins IN3 and OUT3 are for channel 3



PIN	Function	PIN	Function
1	IN2 (External interrupt)	6	GND
2	OUT2	7	GND
3	IN3 (External interrupt)	8	GND
4	OUT3	9	+5V
5	GND		

Table 2-29: GPIO 1 Pinout

2.3 PMC-RTV Series

2.3.1 PMC-RTV21 Specifications

Video Input

- Four composite video color digitizers
- Video input interface: DB-9 female connectors
- Coaxial cable recommended

General Purpose I/O Lines

- The I/O lines are TTL compatible with 1 input and 1 output
- GPIO interface:
  - One DB-9 male connector
  - The I/O lines are internally pulled up and have the following characteristics:

Voltage	MIN	MAX
Input high voltage (20μA)	2.0V	5.25V
Input low voltage (-0.2μA)	0.0V	0.80V
Output high voltage (-1.0mA)	5.0V	-
Output low voltage (100.0mA)	-	0.5V

Table 2-30: GPIO Characteristics

User EEPROM

- Includes 1kbit available EEPROM

Form Factor

- 32bit/33MHz PMC socket board

Figure 2-11: PMC-RTV21 Appearance

Dimensions

- W x L: 74(mm) x 149(mm)

Operating Environment

- Temperature: 0 to 55°C
- Humidity: 5 to 90% RHNC

Storage Environment

- Temperature: 0 to 70°C
- Humidity: 0 to 95% RHNC

Power Requirements

- +5V max. 0.35A

PMC-RTV21 Standard Accessories

- User Manual
- All in One CD

PMC-RTV21 Connectors & Pin Definition

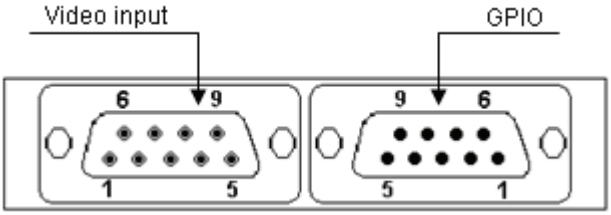
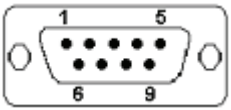
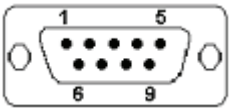


Figure 2-12: PMC-RTV21 Video Input & GPIO



PIN	Function	PIN	Function
1	GND	6	CH0 Video In
2	CH1 Video In	7	GND
3	GND	8	CH2 Video In
4	CH3 Video In	9	GND
5	--		

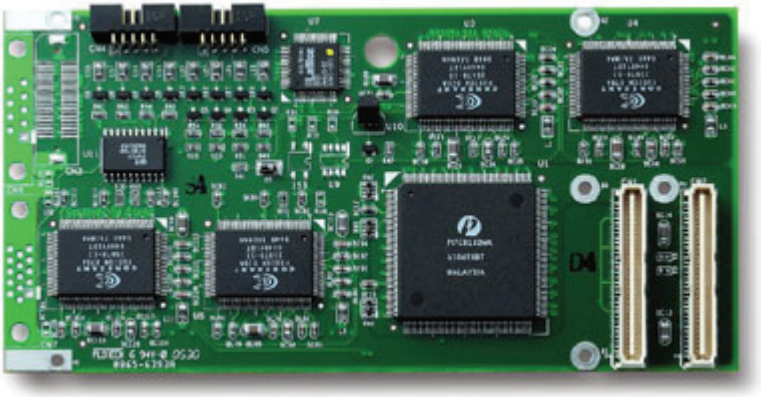
Table 2-31: Video Input



PIN	Function	PIN	Function
1	IN0 (External interrupt)	6	GND
2	OUT0	7	GND
3	--	8	GND
4	--	9	+5V
5	GND		

Table 2-32: GPIO Pinout

2.3.2 PMC-RTV24 Specifications



Dimensions

- W x L: 74(mm) x 149(mm)

Operating Environment

- Temperature: 0 to 55°C
- Humidity: 5 to 90% RHNC

Storage Environment

- Temperature: 0 to 70°C
- Humidity: 0 to 95% RHNC

Power Requirements

- +5V max. 1.5A
- +3.3 V max. 0.5A

Video Input

- Four composite video color digitizers
- Video input interface: DB-9 female connectors
- Coaxial cable recommended

General Purpose I/O Lines

- The I/O lines are TTL compatible with 1 input and 1 output
- GPIO interface:
- One DB-15 male connector
- The I/O lines are internally pulled up and have the fol-low¬ing characteristics:

Voltage	MIN	PIN
Input high voltage (20 uA)	2.0V	5.25V

Table 2-33: GPIO Characteristics

Voltage	MIN	PIN
Input low voltage (-0.2 uA)	0.0V	0.80V
Input high voltage (-1.0 mA)	5.0V	-
Output low voltage (100.0 mA)	-	0.5V

Table 2-34: GPIO Characteristics

User EEPROM

- Includes 1kbit available EEPROM

Form Factor

- 32bit/33MHz PMC socket board

Figure 2-13: PMC-RTV24 Appearance

PMC-RTV24 Standard Accessories

- User Manual
- All in One CD

PMC-RTV24 Connectors & Pin Definition

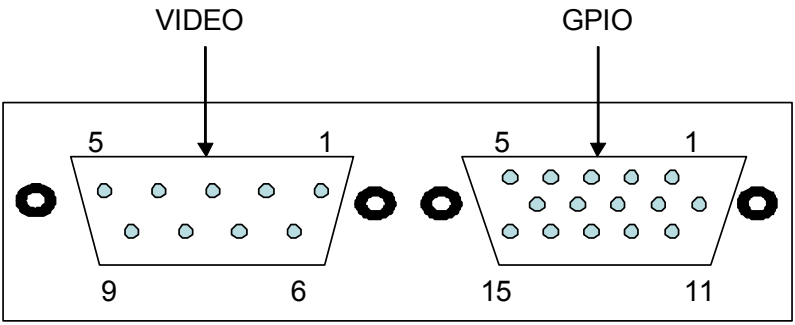
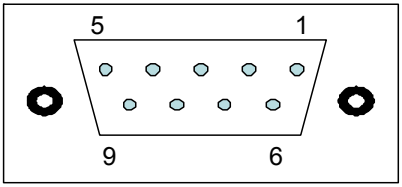
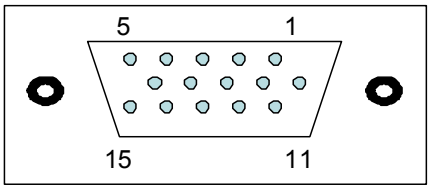


Figure 2-14: PMC-RTV24 Video Input & GPIO



PIN	Function	PIN	Function
1	GND	6	CH0 Video In
2	CH1 Video In	7	GND
3	GND	8	CH2 Video In
4	CH3 Video In	9	GND
5	--		

Table 2-35: Video Input



PIN	Function	PIN	Function	PIN	Function
1	IN0 (External interrupt)	6	+5V output(Max.1A)	11	OUT0
2	IN1 (External interrupt)	7	GND	12	OUT1
3	IN2 (External interrupt)	8	GND	13	OUT2
4	IN3 (External interrupt)	9	GND	14	OUT3
5	GND	10	GND	15	GND

Table 2-36: GPIO Pin-out

3 Installation Guide

3.1 Hardware Installation

3.1.1 RTV Series

Use the following steps to install the RTV series 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 t accommodate the board. If there is not an empty slot, remove a PCI board from the computer to make room for the RTV-24 board and take note of the chosen slot number.
3. Remove the blank metal plate located at the back of the selected slot (if any). Keep the removed screw to fasten the RTV-24 board after installation.
4. Carefully position the RTV-24 in the selected PCI slot as illustrated below. If using a tower computer, orient the board to suit the board slots.

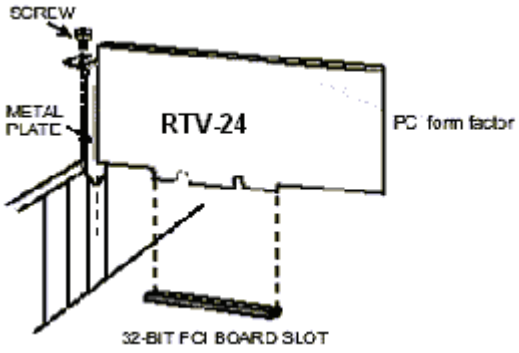


Figure 3-1: RTV-24 Installation

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 8 times (4 videos and 4 audios) and you will require drivers. For details, see the "Installation Guide."

3.1.2 cRTV Series

Use the following steps to install the cRTV series board onto the Compact PCI bus:

1. Remove the computer cover using the instructions from the computer manual.
2. Check that there is an empty cPCI (32-bit/64-bit) slot to accommodate the board. If is not an empty slot, remove a cPCI board to make room for the cRTV-24 (3U) / cRTV-44 (6U) board and take note of the chosen slot number.
3. Remove the blank metal plate located at the front of the selected slot (if present). Keep the removed screw to fasten the cRTV-24 (3U) / cRTV-44 (6U) board.
4. Carefully position the cRTV-24 or cRTV-44 in the selected cPCI slot as illustrated below.

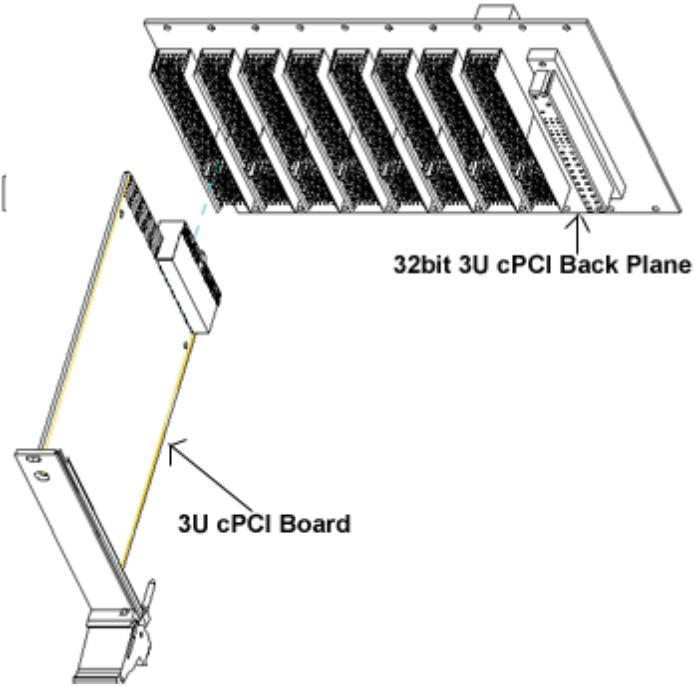


Figure 3-2: cRTV-24 (3U cPCI)

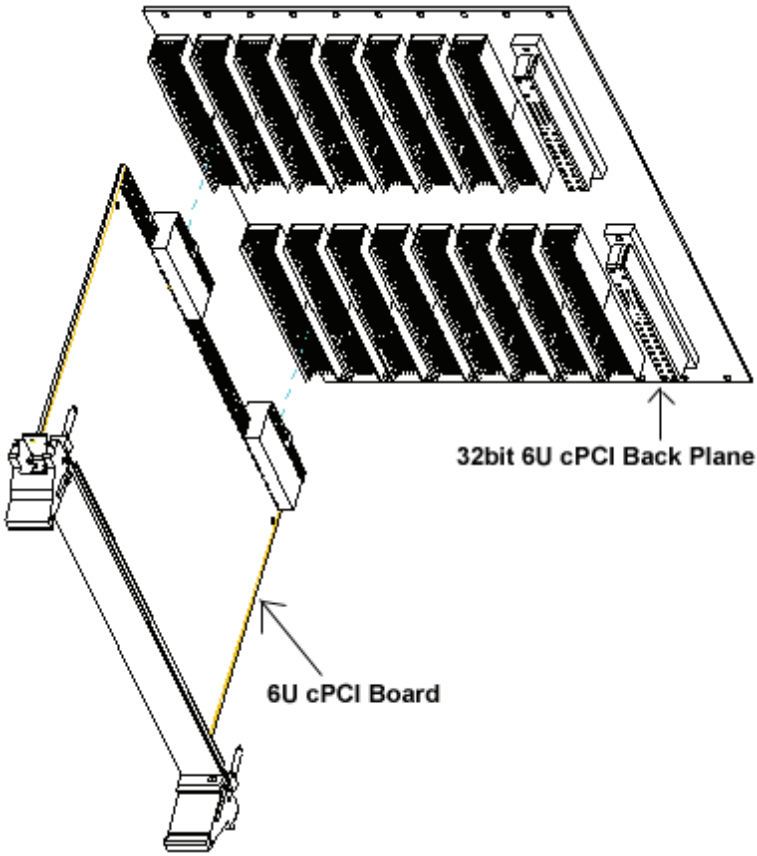


Figure 3-3: cRTV-44 (6U cPCI)

5. Carefully slide the cRTV-24 (3U)/cRTV-44 (6U) along the guide of the chosen slot to the backplane and push the board firmly but carefully into the connector, Lock the board in place by pushing the release lever outwards.
6. 6.Anchor the board by replacing the screw.
7. 7.Connect the video sources for image acquisition tests. For details, refer to the ‘ViewCreator Utility.”

8. 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 8 times (4 videos and 4 audios) and you will require drivers. For details, see the “Installation Guide.”

3.1.3 PMC-RTV Series

The PMC socket may be integrated with the cPCI CPU board or as a standalone system board for an embedded system. Use the following steps to install the PMC-RTV series board onto the PMC socket:

1. Remove the computer cover using the instructions from the computer manual.
2. Check that there is an empty PMC (32-bit) socket to accommodate the board. If there is not an empty slot, remove a PMC board from your computer to make room.
3. Carefully position PMC-RTV21 onto the PMC socket.
4. Once perfectly aligned with an empty PMC socket, press the board firmly but carefully into the connector.
5. Connect the video sources for image acquisition tests. For details, refer to the ‘ViewCreator Utility.”
6. 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 8 times (1 video and 1 audio) and you will require drivers. For details, see the “Installation Guide.”

3.1.4 RTV-E4 Extension Board (Optional)

1.For main board installation, please refer to ‘RTV series’.

2.Each RTV-E4 will attach one signal cable for connect with RTV-24 as below

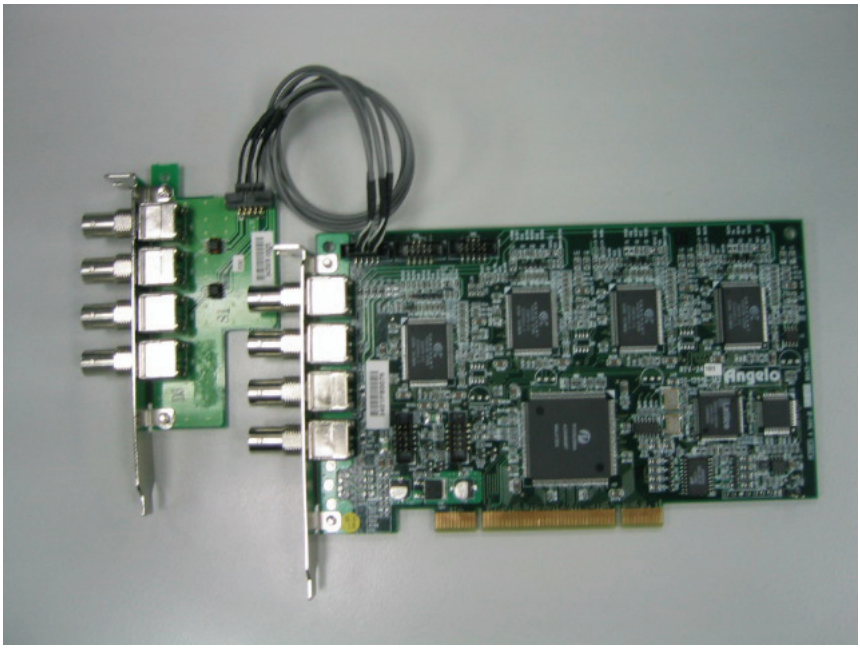


Figure 3-4: RTV-E4 Attachment

3.1.5 RTV-I4 Extension Board (Optional)

1.For main board installation, please refer to ‘RTV series’.

2.Each RTV-I4 will attach one signal cable for connect with RTV-24 as below

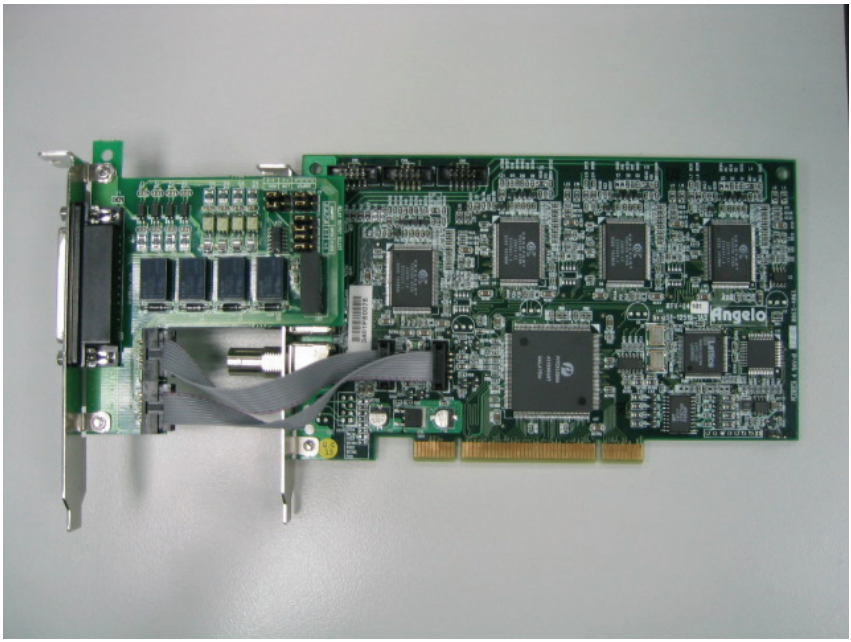
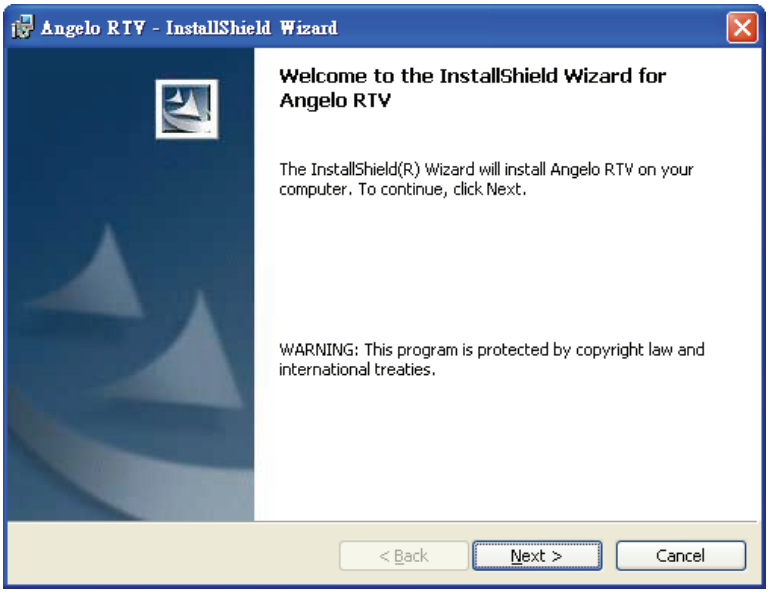


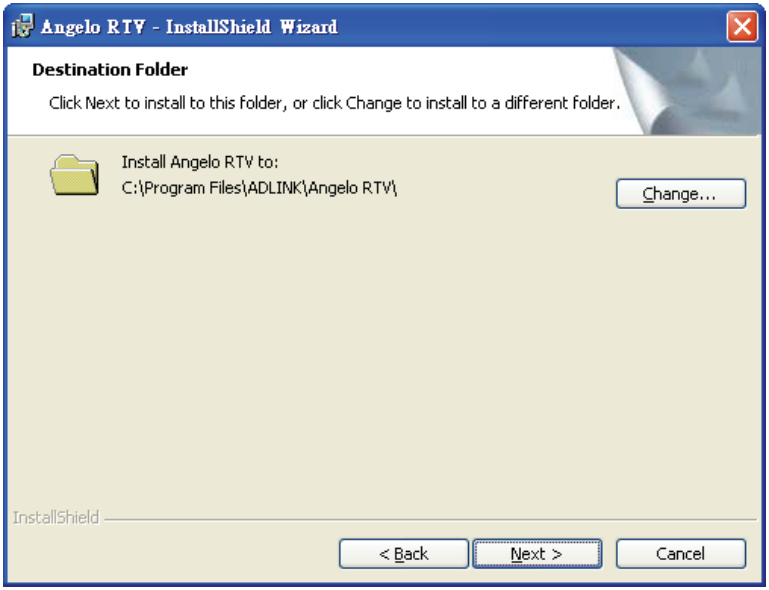
Figure 3-5: RTV-I4 Attachment

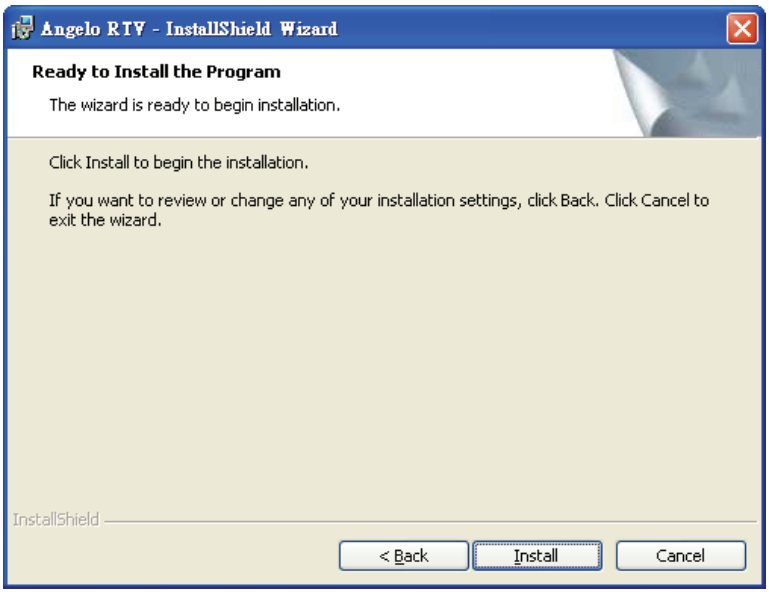
3.2 Driver Installation

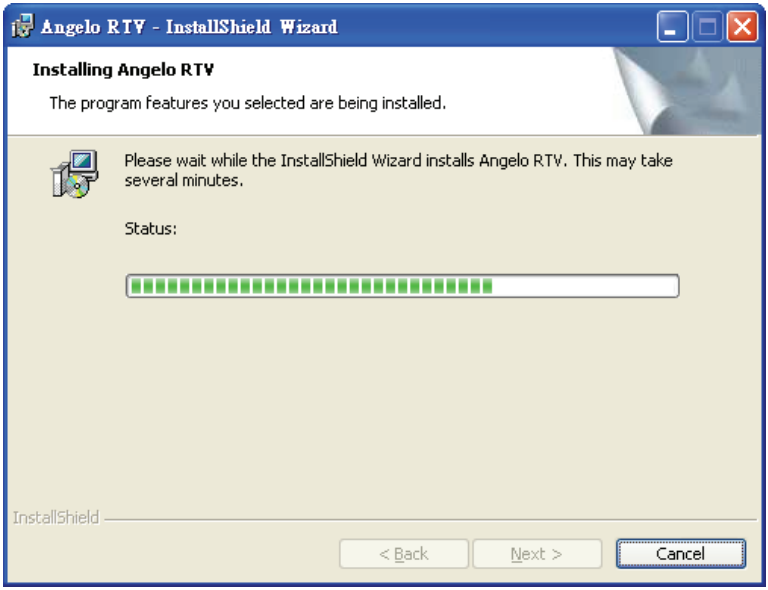
3.2.1 WDM Driver Installation

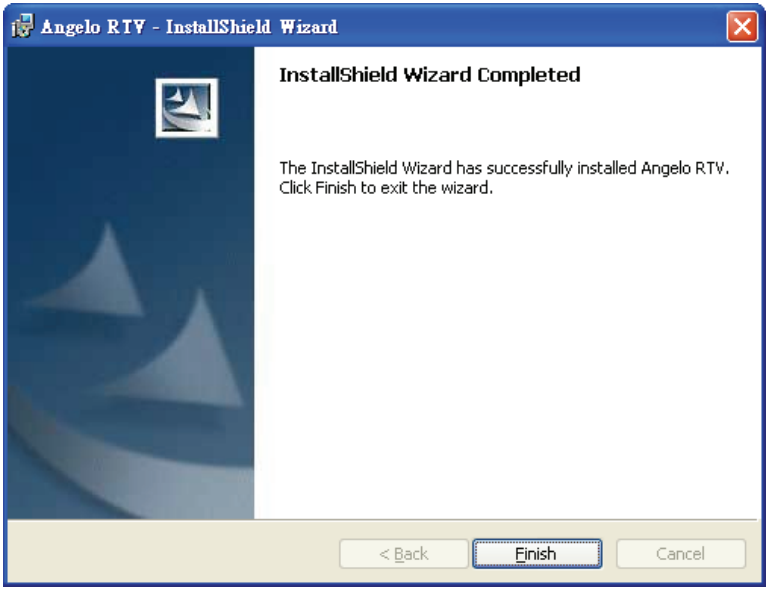
1. Run setup.
2. Click Next until the driver is completely installed.



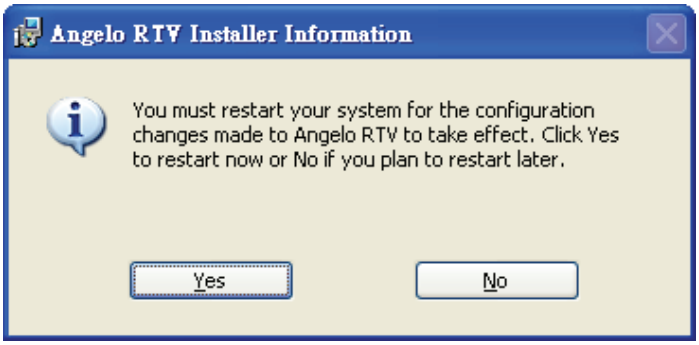

Angelo RTV - InstallShield Wizard
Destination Folder
Click Next to install to this folder, or click Change to install to a different folder.
Install Angelo RTV to:
C:\Program Files\ADLINK\Angelo RTV\
Change...
InstallShield
< Back
Next >
Cancel


Angelo RTV - InstallShield Wizard
Ready to Install the Program
The wizard is ready to begin installation.
Click Install to begin the installation.
If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.
InstallShield
< Back
Install
Cancel


Angelo RTV - InstallShield Wizard
Installing Angelo RTV
The program features you selected are being installed.
Please wait while the InstallShield Wizard installs Angelo RTV. This may take several minutes.
Status:
InstallShield
< Back
Next >
Cancel

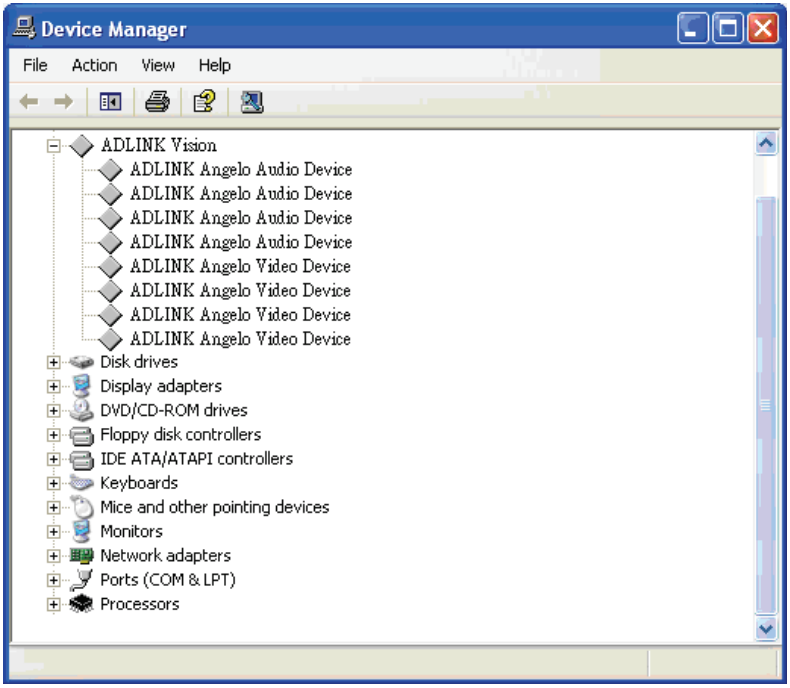


3. Click yes and restart system.

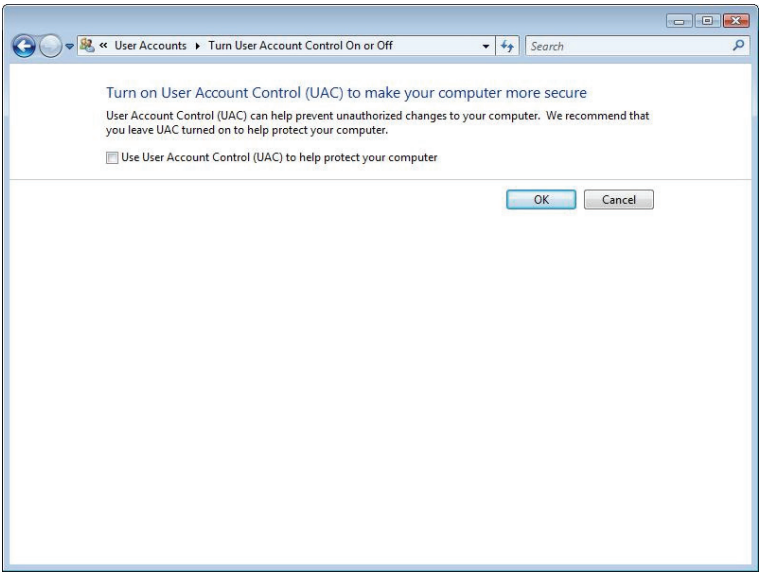


4. Open the Device Manager and check for the following 8 items:
   - ▷ ADLINK Angelo Audio Device (4 instances)
   - ▷ ADLINK Angelo Video Device (4 instances)

The Device Manager should be as follows:

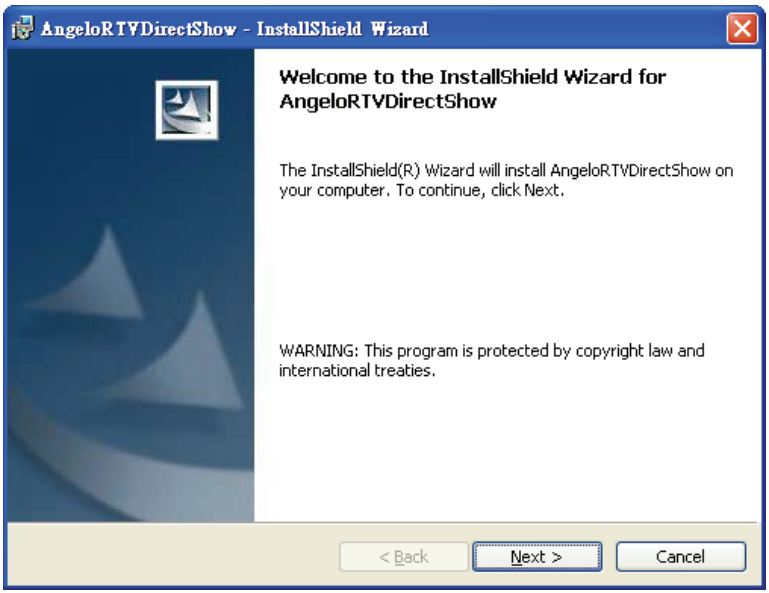


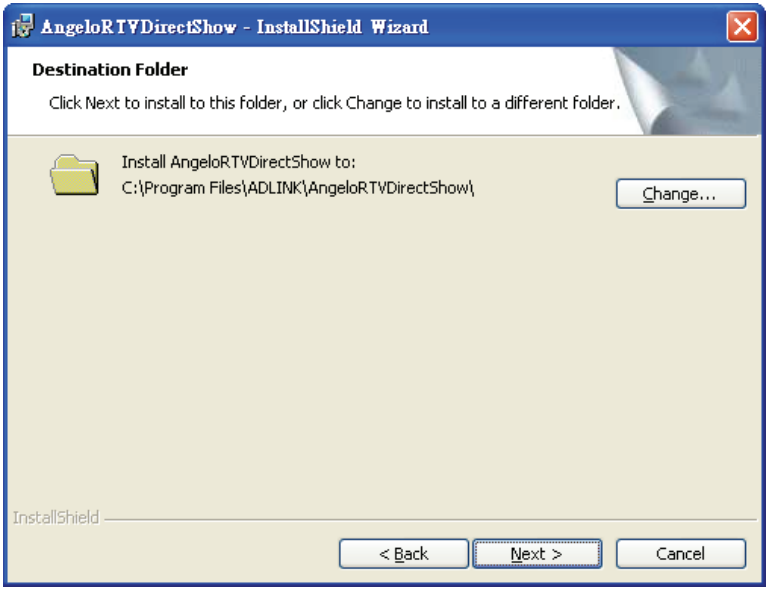
Note: If using Windows Vista, the User Account Control (UAC) needs to be turned off before using the device. To turn off the UAC, go to [Start] - [Settings] - [Control Panel] - [User Accounts] - [Turn User Account Control on or off]. Uncheck the UAC and restart the computer, then the device can work normally.

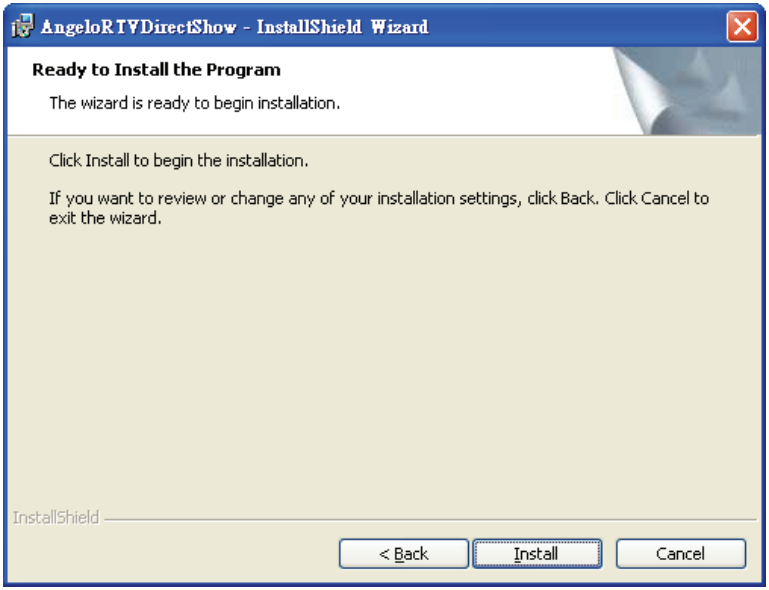


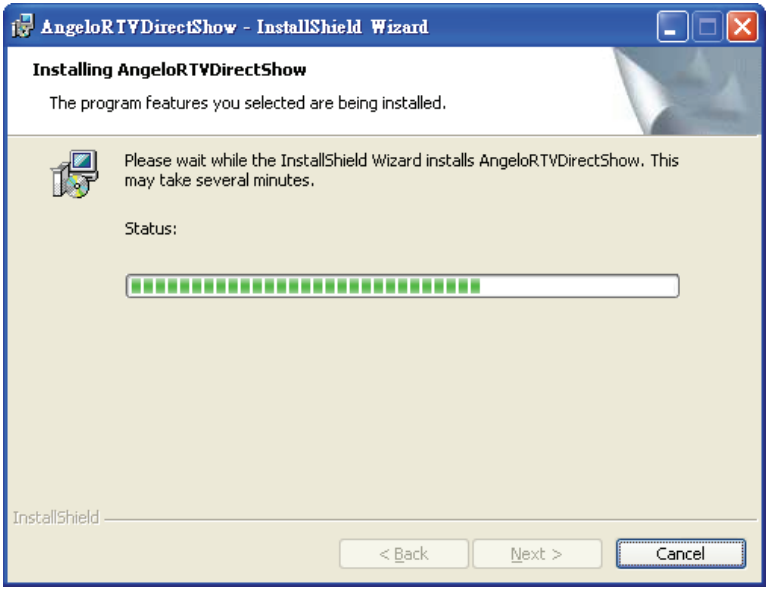
3.2.2 DirectShow Driver Installation

1. Run setup.
2. Click Next until the driver is completely installed.

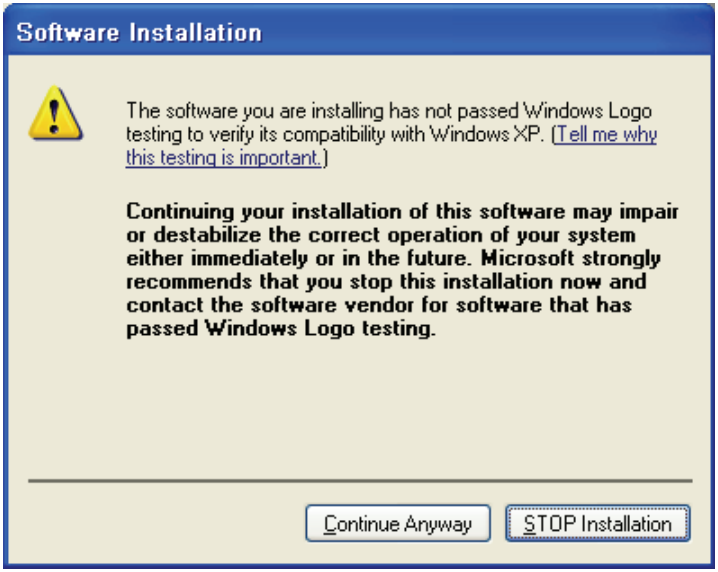


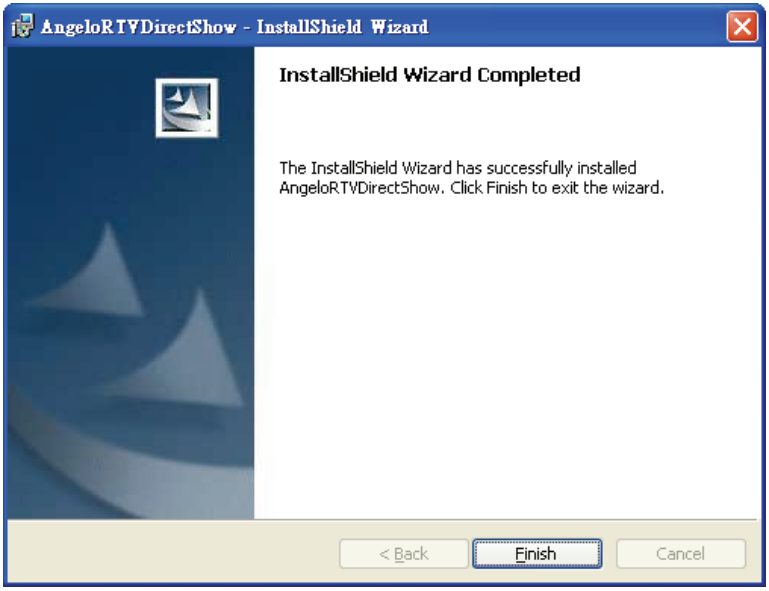

AngeloRTVDirectShow - InstallShield Wizard
Destination Folder
Click Next to install to this folder, or click Change to install to a different folder.
Install AngeloRTVDirectShow to:
C:\Program Files\ADLINK\AngeloRTVDirectShow\
Change...
InstallShield
< Back
Next >
Cancel


AngeloRTVDirectShow - InstallShield Wizard
Ready to Install the Program
The wizard is ready to begin installation.
Click Install to begin the installation.
If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.
InstallShield
< Back
Install
Cancel



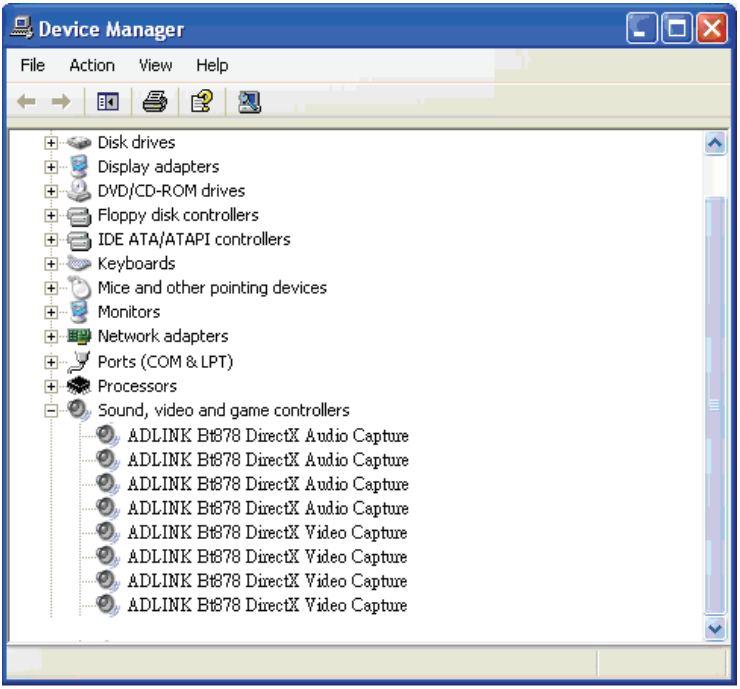
3. When the following window appears, please click “Continue Anyway”.





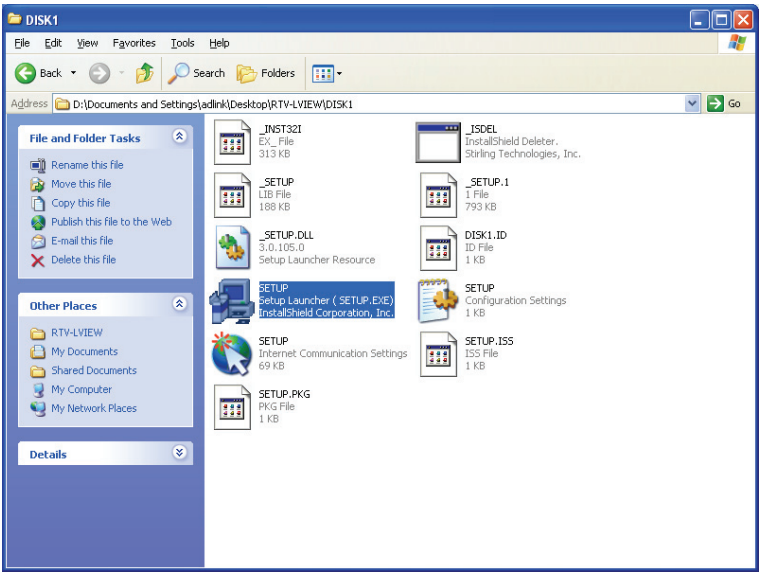
4. Open the Device Manager and check for the following 8 items:
   - ▷ ADLINK Bt878 DirectX Audio Capture (4 instances)
   - ▷ ADLINK Bt878 DirectX Video Capture (4 instances)

The Device Manager should be as follows:

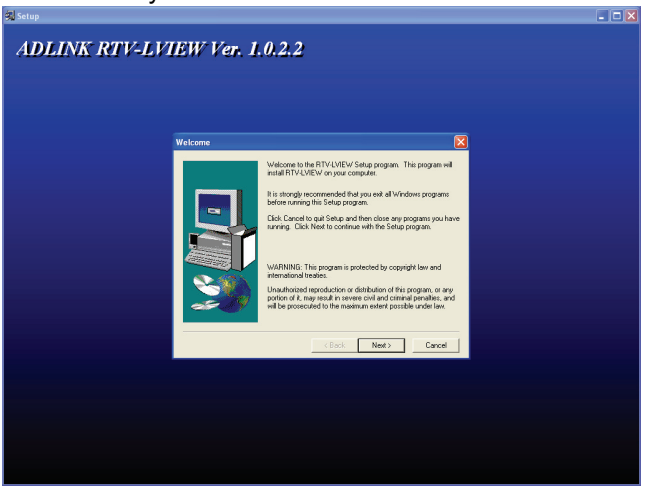


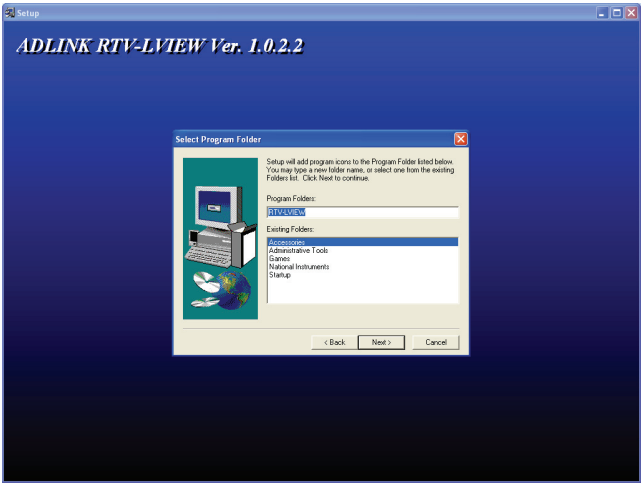
3.2.3 RTV-LVIEW Installation

1. Double-click the **setup.exe** file to start RTV-LVIEW installation.

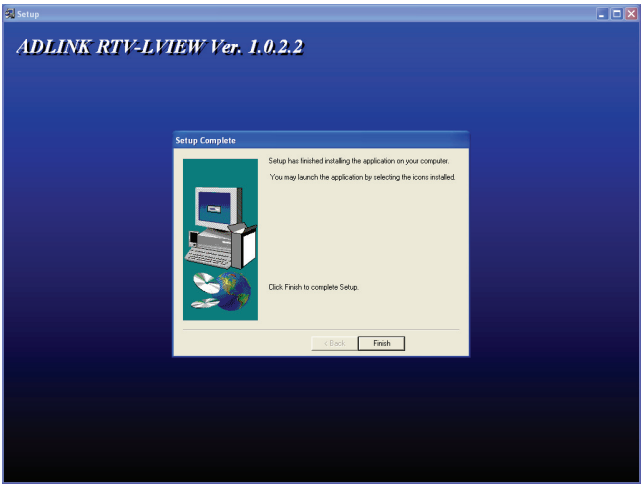


2. Continuously click the **Next** button to install RTV-LVIEW.



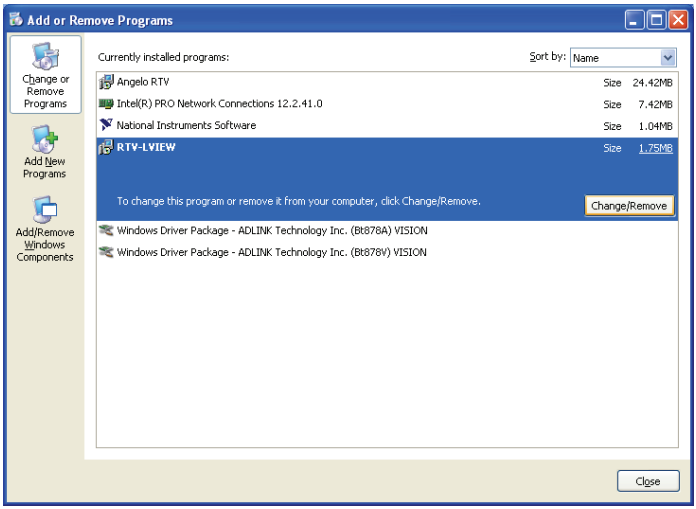


3. Click the **Finish** button to finish the installation.



3.2.4 Uninstall RTV-LVIEW

Open the **Control Panel** and double-click **Add/Remove Programs**. Select RTV-LVIEW and click the **Change/Remove** button to uninstall it.



After un-installation, all files in the directory of RTV-LVIEW will be removed, except the **ADLINK_Vision** palette. If you do not want to use it any more, you can remove the **Angelo.llb** in the **user.lib** folder and the **menus\ADLINK_Vision** folder.

3.2.5 Linux Driver Installation

The driver is compiled as a kernel module and works for kernel version 2.6.

Compile bttv for your system

BTTV is an open source driver and conforms to Video for Linux standard.

1. Open a terminal console and enter the following commands to start installation:
2. Extract the tar ball

```
# tar zxvf RTV-kernel-2.6.xx.tar.gz
```

3. Change to the driver directory which please sees README under the root directory of the RTV packet.

```
# cd xxxxx
```

4. Make and install the driver

```
# make clean
# make
# make install
```

5. Edit auto load configuration file

```
# vi /etc/modprobe.conf
```

6. Add the following lines to the file:

```
# i2c
alias char-major-89 i2c-dev
options i2c-corei2c_debug=1
options i2c-algo-bit bit_test=1

# bttv
alias char-major-81 bttv
options bttv card=134,134,134,134
```

In this example, the 134 depends on how many ports the system has. For example, two PCIe-RTV24 cards have 8 ports total. The text will thus be:

```
options bttv card=134,134,134,134,134,134,134,134
```

7. Restart the computer. The driver should be loaded automatically while booting. Enter the following command to see if the driver was loaded:

```
# lsmod | grep bttv
```

8. If there is a bttv module, the driver is loaded successfully. If not, enter the command to load manually:

```
# modprobe bttv
```

Note: The linux kernel need at least these config options for Video4Linux:

```
CONFIG_I2C=m
CONFIG_I2C_ALGOBIT=m
CONFIG_VIDEO_DEV=m
```

If these config options are not set as module, you need recompile kernel.

Run a test program

1. Open a terminal console and enter the following commands
2. Change to the sample directory which please sees README under the root directory of the RTV packet.

```
# cd xxxxx
# cd libfg-x.x.x
# make clean
# make
# ./camview
```

3. Select the video format and preview channel. You will see a new opened window and show the life image.

The samples are based on Video4 Linux API that the document can found at http://www.linuxtv.org/downloads/video4linux/API/V4L2_API/.

4 ViewCreatorPro Utility

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

Note: ViewCreatorPro is only available for Windows /XP/Vista with a recommended screen resolution higher than 800x600.

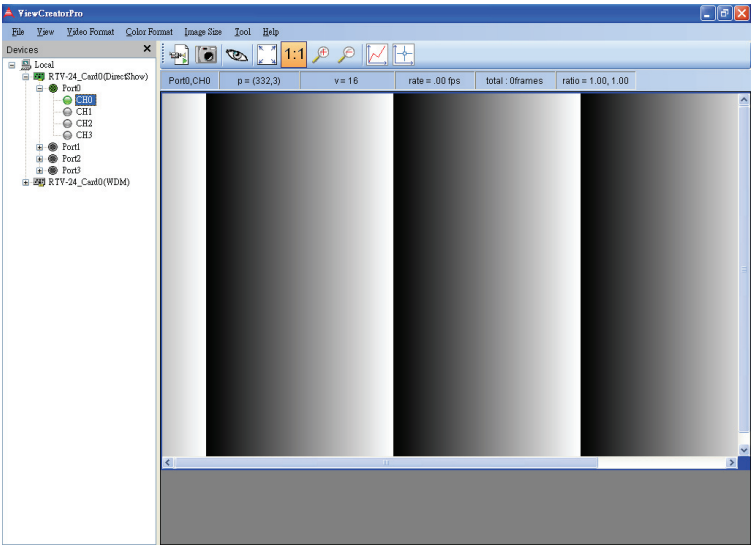
4.1 Overview

ViewCreatorPro offers the following features:

- 32-bit.64-bit operation under Windows XP/Vista WDM or DirectShow driver
- Angelo series cards access and configuration
- Video picture adjustments
- Image file saving (BMP or JPG)
- Direct access to general purpose I/Os
- FULL, CIF, or QCIF Image size, 2x2 or 4x4 display
- Software triggering

4.2 Component Description

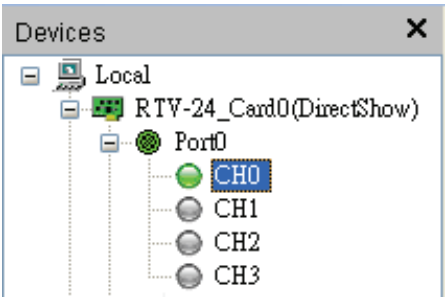
Start the utility and the view should like below:



4.3 Operation Theory

ViewCreatorPro provides many functions for the Angelo RTV series cards as described below:

4.3.1 Devices Panel



Local

Current active Device

All operations will apply to this device.

Inactive Device

Click the port after this icon to activate this device.

Current active port

All operations will apply to this port.

Inactive port

Click the port after this icon to activate this port.

Current active channel

All operations will apply to this channel.

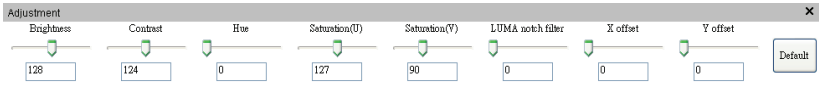
Inactive channel

Click the port after this icon to activate this channel.

Close this panel

4.3.2 Adjustment Panel

A panel allows user adjusting video images. Click and hold the left mouse button on the slider of the Adjustment Panel and drag the cursor to change its value. Or type value into the edit tool to change its value directly.



Default Button

Press Default Button resetting all values to default value.

Close this panel

4.3.3 Toolbar

Continue Grab

Start to grab images and display the images on display panel. Click it again to stop the grab. This is a toggle button.

Stop Grab

Stop grabbing.

Snap Shot

Capture an image and display the image on display panel.

Hind Image

Hide or unhide displaying image. This is a toggle button.

Fit Size

Fit the images which are selected to whole display panel. The images which are selected will have a blue frame.

Original Size

Restore the images which are selected to original size. The images which are selected will have a blue frame.

Zoom In

Zoom in the images which are selected. The images which are selected will have a blue frame.

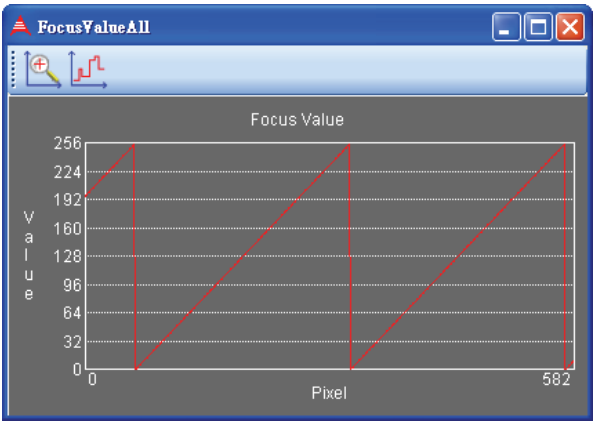
Zoom Out

Zoom in the images which are selected. The images which are selected will have a blue frame.

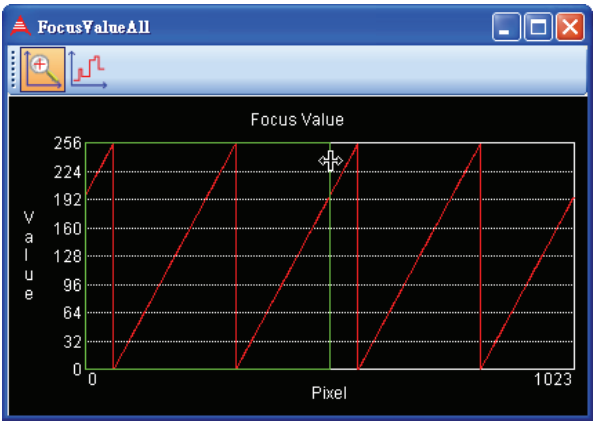
Focus Value

Open a chart to see pixel values of the selected horizontal line of the image which is selected first. The display image shows a red horizontal line on it. Click mouse on the display image to move the selected line.

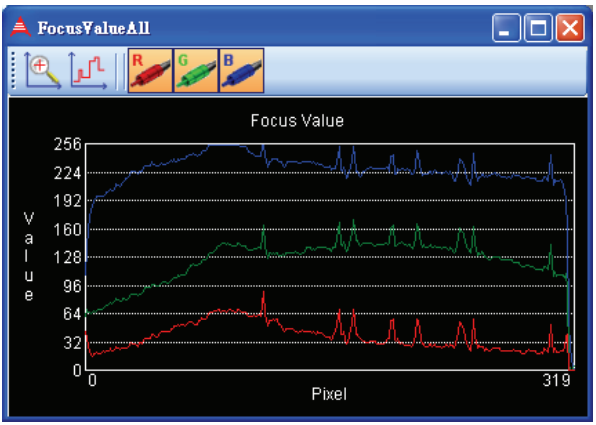
If it is grabbing image, the background color of focus value window is gray. The chart will update immediately by acquired image and the x-axis region depends on which horizontal pixels shown in display panel. The window is shown below:



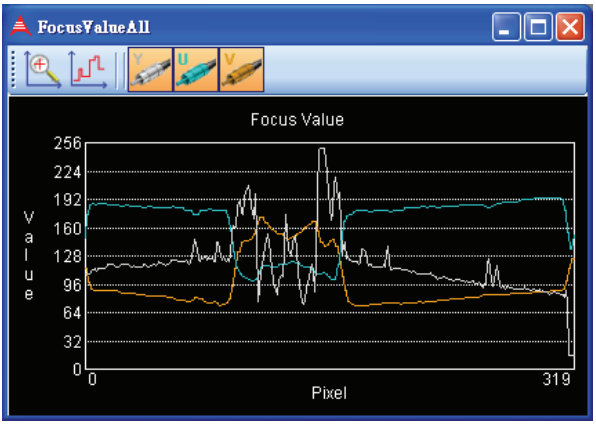
After stopping grabbing, the background color of focus value window is black. The x-axis size is the width of the whole image. The window is shown below:



If the image is chromatic and is RGB type, there are three curves represented red, green, and blue individual in the chart. The window is shown below:



If the image is chromatic and is YUV type, there are three curves represented y, u, and v individual in the chart. The window is shown below:



Zoom In

Open a window to zoom in the green rectangle region.

Differential

Open a window to show the slop of the line for the green rectangle region.

Drag the vertical green line to resize the green rectangle.

Show/Hide Red Values

Show or hide the red value of the pixels.

Show/Hide Green Values

Show or hide the green value of the pixels.

Show/Hide Blue Values

Show or hide the blue value of the pixels.

Show/Hide Y Values

Show or hide the y value of the pixels.

Show/Hide U Values

Show or hide the u value of the pixels.

Show/Hide V Values

Show or hide the v value of the pixels.

Focus Cross

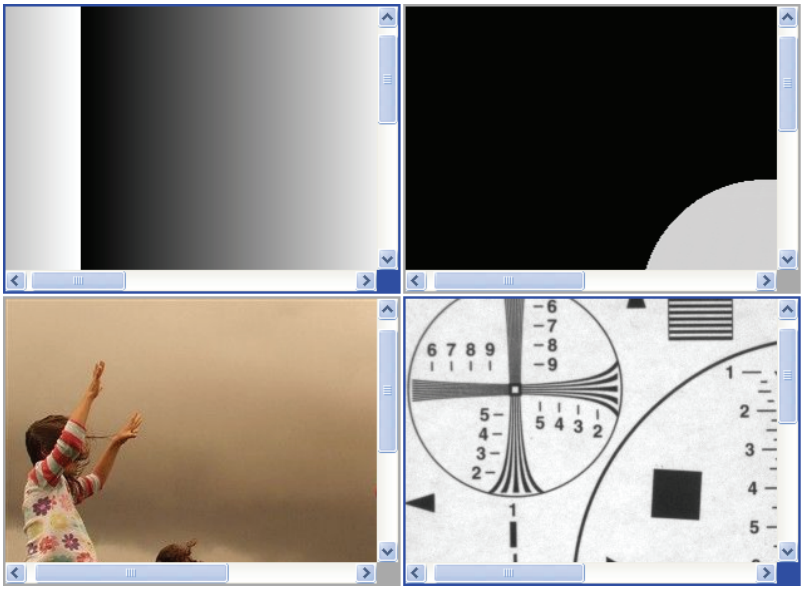
See pixel values of the selected point of the image on toolbar. The display image shows a blue cross line on it. Click mouse on the display image to move the selected point.

4.3.4 Status Bar

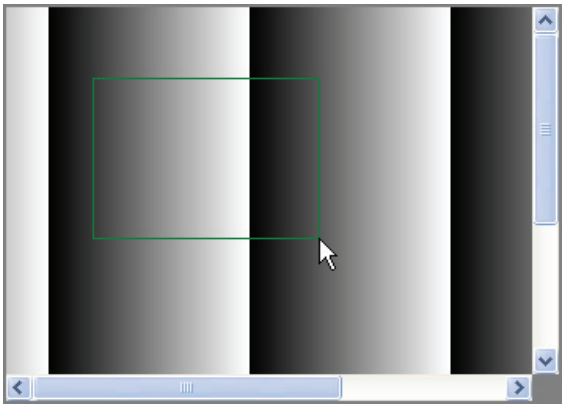
From left to right, the panel items are status host, cursor position, pixel value, frame rate, total captured frames, and magnification (horizontal ratio, vertical ratio).

4.3.5 Display Panel

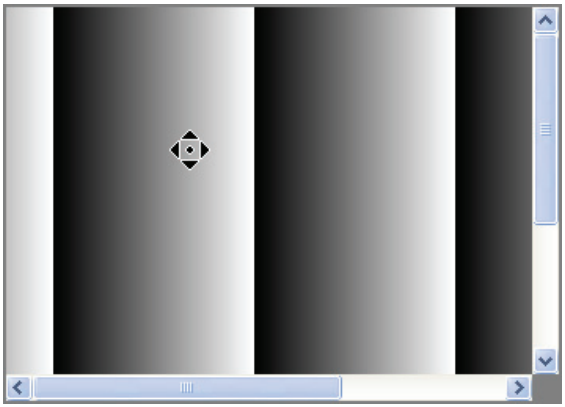
Press left mouse button on the image and then there will be a blue frame around the image. It means the image is selected. If user wants select more images, please keep pressing Ctrl and click the other images. Like the picture below, the up-left channel and down-right channel are selected. Then user can adjust these images' size by "Fit Size", "Original Size", "Zoom In", and "Zoom out" button.



Press left mouse button and then drag it, display panel will appear a green rectangle region which will be zoomed in. Keep pressing Shift during dragging, the image will be zoomed in at the same proportion of width and height. Shown below:



Press right mouse button, the cursor will become a move2D icon. Then user can drag the image. Shown below:



4.3.6 Main Menu

File menu

- **Open Image**
  Open an image from a file and display it to the display panel.
- **Save Image**
  Save current displaying image to a bitmap file.
- **Exit**
  Terminate ViewCreatorPro.

View menu

- **Devices**
  Hide or unhide Devices panel.
- **Adjustment**
  Hide or unhide Adjustment panel.
- **ChannelExtensionEnable**
  Determine if let user select channel node.

Video Format menu

- **NTSC**
  Set the channels showed on display panel to NTSC format.
- **PAL**
  Set the channels showed on display panel to PAL format.

Color Format menu

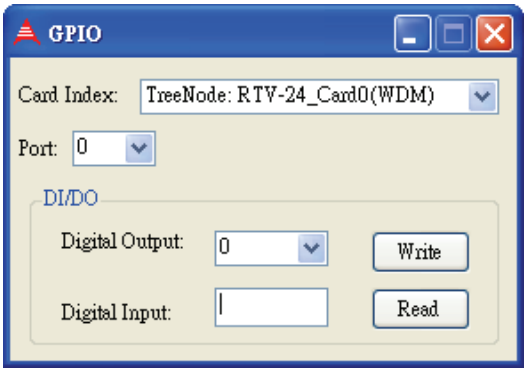
- **Gray**
  Set the channels showed on display panel to gray format.
- **RGB32**
  Set the channels showed on display panel to rgb32 format.
- **RGB24**
  Set the channels showed on display panel to rgb24 format.
- **RGB16**
  Set the channels showed on display panel to rgb16 format.
- **RGB15**
  Set the channels showed on display panel to rgb15 format.
- **YUV**
  Set the channels showed on display panel to yuv format.

Image Size menu

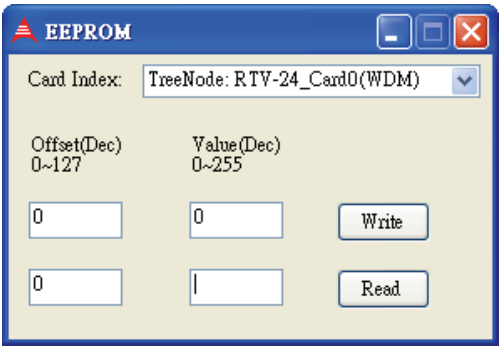
- **Full Image**
  Set buffer sizes of the channels showed on display panel to full image size.
- **Cif Image**
  Set buffer sizes the channels showed on display panel to cif image size.
- **Qcif Image**
  Set buffer sizes the channels showed on display panel to qcif image size.

Tool menu

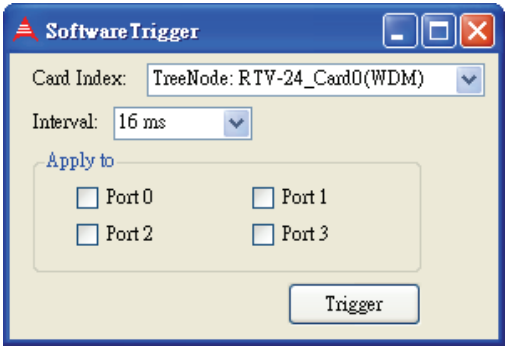
- **GPIO**
  Click Tool in the menu bar and select GPIO item to bring up the GPIO dialog box. Select the card and port to access and select the digital output value. Click the write or read button to write/read to/from the digital I/O ports.



- **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, and 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 0-127. Valid output values are 0-255. 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).



Help menu

- **About**
  Click Help in the menu bar and select About ViewCreator-Pro to bring up the About ViewCreatorPro box. This window will show ViewCreatorPro version.

- **AboutDevice**
  Click Help in the menu bar and select About Device to bring up the About Device box. This window will show the driver version and dll version.

5 Function Library

This chapter describes the API for Angelo RTV series cards. Users can use these functions to develop application programs under Visual C++, Visual Basic, C++ Builder, C#, Visual Basic .Net, and Delphi.

5.1 List of Functions

Category	Section	Function
System	5.3	AngeloRTV_Initial(PortNo)
		AngeloRTV_Close(PortNo)
		AngeloRTV_Software_Reset(PortNo)
		AngeloRTV_Read_Serial(CardNo, HighByte, LowByte)
		AngeloRTV_Get_Version(DriverVersion, DLLVersion, Reserved)
Configuration	5.4	AngeloRTV_Set_Image_Config(PortNo, ConfigIndex,Value)
		AngeloRTV_Get_Image_Config(PortNo, ConfigIndex, Value)
		AngeloRTV_Set_Color_Format(PortNo, ColorFormat)
		AngeloRTV_Get_Color_Format(PortNo, ColorFormat)
		AngeloRTV_Set_Video_Format(PortNo, Value)
		AngeloRTV_Get_Video_Format(PortNo, Value)
		AngeloRTV_Set_Image_Geometric(PortNo, X_Offset, Y_Offset, X_Active, Y_Active, X_Scale, Y_Scale)
		AngeloRTV_Detect_Video_Format(PortNo, FormatValue)
Image Grabbing	5.5	AngeloRTV_Capture_Start(PortNo, CaptureNo)
		AngeloRTV_Select_Channel(PortNo, Multiplex)
		AngeloRTV_Capture_Stop (PortNo)
		AngeloRTV_Capture_Config(PortNo, Start_Field)
		AngeloRTV_Sync_Grab(PortNo, Start_Address, Width, Height, Size_Byte)
GPIO & EPROM	5.6	AngeloRTV_Set_GPIO_Sts(PortNo, Status)
		AngeloRTV_Get_GPIO_Sts(PortNo, Status)
		AngeloRTV_Set_GPIO_Int_Logic (PortNo, Logic)
		AngeloRTV_Write_EEPROM(PortNo, Offset, Value)
		AngeloRTV_Read_EEPROM(PortNo, Offset, Value)
		AngeloRTV_Set_LED_Sts (PortNo, LEDStatus)
Callback & Thread	5.7	AngeloRTV_Set_Int_Event(PortNo,hEvent)
		AngeloRTV_Set_Callback(PortNo, CallBackProc)
		AngeloRTV_Get_Int_Status(PortNo,IntStatus)
Software Trigger	5.8	AngeloRTV_Trigger_Config(PortNo, Interval)
		AngeloRTV_Trigger_Start(CardNo, Multiplex)
Frame Buffer	5.9	AngeloRTV_Get_frame(PortNo, Start_Address, Width, Height, Size_Byte)
		AngeloRTV_Save_File(PortNo, FileName, FileFormat, nQuality)
		AngeloRTV_Copy_frame(PortNo, Dest_Address, Size_Byte)

Table 5-1: List of Functions

5.2 C/C++ Programming Library

Function prototypes and common data types are defined in Angelo.h. The Angelo series library uses these data types. We suggest that these data types be used in your application programs. The following table shows the data types and their range:

Type Name	Description	Range
U8	8-bit ASCII character	0 to 255
I16	16-bit integer	-32768 to 32767
U16	16-bit unsigned integer	0 to 65535
I32	32-bit long integer	-2147483648 to 2147483647
U32	32-bit unsigned long integer	0 to 4294967295
F32	32-bit float	-3.402823E38 to 3.402823E38
F64	64-bit double float	-1.797683134862315E308 to 1.797683134862315E309
Boolean	Boolean logic	TRUE, FALSE

Table 5-2: C/C++ Data Types

5.3 System Functions

@ Name

AngeloRTV_Initial(PortNo)

Initialize the port in Angelo series card.

AngeloRTV_Close(PortNo)

Close the port in Angelo series card.

AngeloRTV_Software_Reset(PortNo)

Reset the port in Angelo series card.

AngeloRTV_Read_Serial(CardNo, HighByte, LowByte)

Read the unique 48-Bit Serial Number of Angelo Series Card (Only for RTV-24 Rev.B1 above, PCI-2100 Rev.A2 above)

AngeloRTV_Get_Version(DriverVersion, DLLVersion, Reserved)

Get the version of driver of AngeloRTV card and AngeloRTV.dll.

@ Description

AngeloRTV_Initial:

This function initializes the ports of the Angelo Series card. Each application program must call this function before any other functions can be used. If the initialization is executed successfully, it returns a value of 0.

Note: There are four ports on the RTV-24, cRTV-24, and cRTV-44 series cards, and one port on the PMC- RTV21.

AngeloRTV_Close:

Releases all resources from the ports.

AngeloRTV_Software_Reset:

Resets the port to its initial state.

AngeloRTV_Read_Serial:

This function can read a 48-bit unique ID and store in 2 Long interger.

AngeloRTV_Get_Version:

Used to get the current version of AngeloRTV card driver and AngeloRTV.dll file.

@ Syntax

C/C++ (Windows/CE.NET)

```
I16 AngeloRTV_Initial(U16 PortNo)
```

```
I16 AngeloRTV_Close(U16 PortNo)
I16 AngeloRTV_Software_Reset(U16 PortNo)
U16 AngeloRTV_Read_Serial(U16 CardNo, U32*
    HighByte, U32* LowByte);
I16 AngeloRTV_Get_Version(U32 *DriverVersion, U32
    *DLLVersion, U32 *Reserved)
```

Visual Basic (Windows/CE.NET)

```
AngeloRTV_Initial (ByVal PortNo As Integer) As
    Integer
AngeloRTV_Close(ByVal PortNo As Integer) As
    Integer
AngeloRTV_Software_Reset (ByVal PortNo As
    Integer) As Integer
AngeloRTV_Read_Serial(Byval CardNo as Integer,
    ByRef HighByte As Long, ByRef LowByte As
    Long) As Integer
AngeloRTV_Get_Version (ByRef DriverVersion As
    Long, ByRef DLLVersion As Long, ByRef
    Reserved As Long) As Integer
```

Delphi (Windows)

```
AngeloRTV_Initial(PortNo:Smallint):Smallint
AngeloRTV_ Close (PortNo:Smallint):Smallint
AngeloRTV_Software_Reset
    (PortNo:Smallint):Smallint
AngeloRTV_Read_Serial(CardNo:Smallnt; Var
    HighByte: Longint; Var
    LowBytet:Longint):Smallint;
AngeloRTV_Get_Version (var DriverVersion:Longint;
    var DLLVersion:Longint; var
    Reserved:Longint):Smallint
```

@ Arguments

PortNo:

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

HighByte:

HighByte stores the upper 16Bit of Serial No..

LowByte:

LowByte stores the lower 32Bit of Serial No.

DriverVersion:

Indicate the current version of AngeloRTV driver. This parameter is a pointer to an integer array with length 4.

DLLVersion:

Indicate the current version of AngeloRTV.dll file. This parameter is a pointer to an integer array with length 4.

@ Return Code

- 0: ERROR_NoError
- -2: ERROR_Card_Not_Exist – make sure the Angelo series card is plugged into the system, check the device manager to make sure the device is loaded, and the “PortNo” parameter is valid.
- -3: ERROR_Card_Not_Accessible – make sure the Angelo series card is plugged into the system, check the device manager to make sure the device is loaded, and the “PortNo” parameter is valid.
- -12: ERROR_CPLD_Check_Failed – Power off the computer and power on again.

@ Example

<VC/BCB >

AngeloRTV_Initial –

```
I16 Result;
for(int PortNo= 0 ; PortNo <4;PortNo++)
Result = AngeloRTV_Initial (PortNo);
```

AngeloRTV_Cose –

```
I16 Result;
for(int PortNo= 0 ; PortNo <4;PortNo++)
Result = AngeloRTV_Cose (PortNo);
```

AngeloRTV_Software_Reset–

```
I16 Result;
for(int PortNo= 0 ; PortNo <4;PortNo++)
Result = AngeloRTV_Software_Reset (PortNo);
```

AngeloRTV_Read_Serial–

```
int Result;
int CardNo = 0;
unsigned long HighByte = 0, LowByte = 0;
Result = AngeloRTV_Read_Serial(CardNo, &HighByte,
     &LowByte);
```

AngeloRTV_Get_Version –

```
I16 Result;
U32 DriverVersion[4] = {0}, DLLVersion[4] = {0},
     Reserved[4] = {0};
char strDriverVersion[20], strDLLVersion[20];
Result = AngeloRTV_Get_Version (DriverVersion,
     DLLVersion, Reserved);
sprintf(strDriverVersion, "%d.%d.%d.%d",
     DriverVersion[0], DriverVersion[1],
     DriverVersion[2], DriverVersion[3]);
sprintf(strDLLVersion, "%d.%d.%d.%d",
     DLLVersion[0], DLLVersion[1],
     DLLVersion[2], DLLVersion[3]);
```

< Visual Basic >

AngeloRTV_Initial –

```
Dim Result As Integer
Dim PortNo As Integer
For PortNo= 0 To 3
Result = AngeloRTV_Initial (ByVal PortNo)
```

AngeloRTV_Cose –

```
Dim Result As Integer
Dim PortNo As Integer
For PortNo= 0 To 3
Result = AngeloRTV_Close (ByVal PortNo)
```

AngeloRTV_Read_Serial–

```
Dim Result As Integer
Dim CardNo As Integer
Dim HighByte As Long, LowByte As Long
CardNo=0
HighByte=0
LowByte=0
Result = AngeloRTV_Read_Serial(CardNo, HighByte,
     LowByte)
```

AngeloRTV_Software_Reset–

```
Dim Result As Integer
Dim PortNo As Integer
For PortNo= 0 To 3
Result = AngeloRTV_Software_Reset (ByVal PortNo)
```

AngeloRTV_Get_Version –

```
Dim Result As Integer
Dim DriverVersion(3) As Long, DLLVersion(3) As
     Long, Reserved(3) As Long
Dim strDriverVersion, strDLLVersion As String
```

```
Result = AngeloRTV_Get_Version (DriverVersion(0),
     DLLVersion(0), Reserved(0))
strDriverVersion = CStr(DriverVersion(0)) + "." +
     CStr(DriverVersion(1)) + "." +
     CStr(DriverVersion(2)) + "." +
     CStr(DriverVersion(3))
strDLLVersion = CStr(DLLVersion(0)) + "." +
     CStr(DLLVersion(1)) + "." +
     CStr(DLLVersion(2)) + "." +
     CStr(DLLVersion(3))
```

<Delphi >

AngeloRTV_Initial –

```
var PortNo,Result:SmallInt;
for i:= 0 to 3 do
begin
Result := AngeloRTV_Initial (PortNo);
End;
```

AngeloRTV_Cose –

```
var PortNo,Result:SmallInt;
for i:= 0 to 3 do
begin
Result := AngeloRTV_Close (PortNo);
End;
```

AngeloRTV_Software_Reset–

```
var PortNo,Result:SmallInt;
for i:= 0 to 3 do
begin
Result := AngeloRTV_Software_Reset (PortNo);
End;
```

AngeloRTV_Read_Serial–

```
var
CardNo,Result:SmallInt;
HighByte, LowByte:SmallInt;
Result := AngeloRTV_Read_Serial(CardNo, HighByte,
LowByte)
```

AngeloRTV_Get_Version –

```
var
Result: Smallint;
DriverVersion: array[1..4] of Longint;
DLLVersion: array[1..4] of Longint;
Reserved: array[1..4] of Longint;
strDriverVersion, strDLLVersion: String;
```

```
Result := AngeloRTV_Get_Version
      (DriverVersion[1], DLLVersion[1],
      Reserved[1]);
strDriverVersion := IntToStr(DriverVersion[1]);
strDriverVersion := strDriverVersion + '.' +
      IntToStr(DriverVersion[2]);
strDriverVersion := strDriverVersion + '.' +
      IntToStr(DriverVersion[3]);
strDriverVersion := strDriverVersion + '.' +
      IntToStr(DriverVersion[4]);
strDLLVersion := IntToStr(DLLVersion[1]);
strDLLVersion := strDLLVersion + '.' +
      IntToStr(DLLVersion[2]);
strDLLVersion := strDLLVersion + '.' +
      IntToStr(DLLVersion[3]);
strDLLVersion := strDLLVersion + '.' +
      IntToStr(DLLVersion[4]);
```

5.4 Configuration Functions

@ Name

AngeloRTV_Set_Image_Config(PortNo, ConfigIndex ,Value)

Set the video adjustments.

AngeloRTV_Get_Image_Config(PortNo, ConfigIndex, Value)

Get the video adjustments.

AngeloRTV_Set_Color_Format(PortNo, ColorFormat)

Set the color format.

AngeloRTV_Get_Color_Format(PortNo, ColorFormat)

Get the color format.

AngeloRTV_Set_Video_Format(PortNo, Value)

Set the video format.

AngeloRTV_Get_Video_Format(PortNo, Value)

Set the video format.

AngeloRTV_Set_Image_Geometric(PortNo, X_Offset, Y_Offset, X_Active, Y_Active, X_Scale, Y_Scale)

Advanced image processing.

AngeloRTV_Detect_Video_Format(PortNo, FormatValue)

Detect the video format and if there is signal input.

@ Description

AngeloRTV_Set_Image_Config:

Adjusts the hue, contrast, Saturation and brightness of the port for the Angelo series card.

AngeloRTV_Get_Image_Config:

Retrieves the current hue, contrast, Saturation and brightness setting of the port for the Angelo series card.

AngeloRTV_Set_ Color _Format:

Sets the color format of the port for the Angelo series card. Valid color formats are: gray scale, RGB.

AngeloRTV_Get_ Color _Format:

Retrieves the color format of the port for the Angelo series card.

AngeloRTV_Set_ Video_Format:

Sets the Video format of the port for the Angelo series card. Valid color formats are: NTSC, EIA, PAL, CCIR.

AngeloRTV_Get_ Video _Format:
Retrieves the video format of the port for the Angelo series card.

AngeloRTV_Set_Image_Geometric:
This function is used for image cropping and scaling.

AngeloRTV_Detect_Video_Format:
Use the function to retrieve the video format. And if the return value of the 2nd parameter is 0 that means there is no signal input.

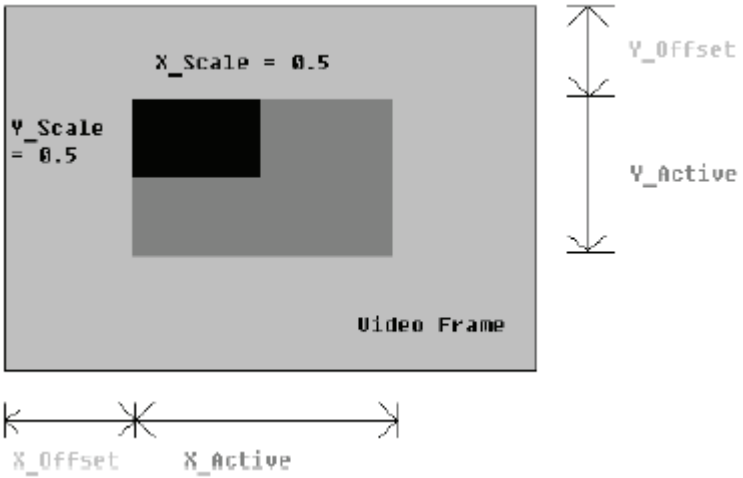


Figure 5-1: Video Frame

@ Syntax

C/C++ (Windows/CE.NET)

```
I16 AngeloRTV_Set_Image_Config(U16 PortNo,U8
     ConfigIndex , U8 Value);
I16 AngeloRTV_Get_Image_Config(U16 PortNo,U8
     ConfigIndex , U8* Value);
I16 AngeloRTV_Set_ Color _Format (U16 PortNo, U8
     ColorFormat);
I16 AngeloRTV_Get_ Color _Format (U16 PortNo, U8*
     ColorFormat);
```

```
I16 AngeloRTV_Set_Video _Format (U16 PortNo, U8
     VideoFormat);
I16 AngeloRTV_Set_Video _Format (U16 PortNo, U8*
     VideoFormat);
I16 AngeloRTV_Set_Image_Geometric(U16 PortNo, U32
     X_Offset, U32 Y_Offset, U32 X_Active, U32
     Y_Active,double X_Scale,double Y_Scale);
I16 AngeloRTV_Detect_Video_Format (U16 PortNo, U8
     *FormatValue);
```

Visual Basic (Windows/CE.NET)

```
AngeloRTV_Set_Image_Config(ByVal PortNo As
     Integer, ByVal ConfigIndex As Byte,ByVal
     Value As Byte) As Integer
AngeloRTV_Get_Image_Config(ByVal PortNo As
     Integer, ByVal ConfigIndex As Byte, ByRef
     Value As Byte) As Integer
AngeloRTV_Set_ Color _Format (ByVal PortNo As
     Integer, ByVal ColorFormat As Byte) As
     Integer
AngeloRTV_Get_ Color _Format (ByVal PortNo As
     Integer, ByRef ColorFormat As Byte) As
     Integer
AngeloRTV_Set_Video _Format (ByVal PortNo As
     Integer, ByVal VideoFormat As Byte) As
     Integer
AngeloRTV_Set_Video _Format (ByVal PortNo As
     Integer, ByRef VideoFormat As Byte) As
     Integer
AngeloRTV_Set_Image_Geometric(ByVal PortNo As
     Integer, ByVal X_Offset As Long, ByVal
     Y_Offset As Long, ByVal X_Active As Long,
     ByVal Y_Active As Long, ByVal X_Scale As
     Double, ByVal Y_Scale As Double)  As Integer
AngeloRTV_Detect_Video_Format (ByVal PortNo,
     ByRef FormatValue As Byte) As Integer
```

Delphi (Windows)

```
AngeloRTV_Set_Image_Config(PortNo:Smallint;Confi
     gIndex:Byte;Value:Byte): Smallint;
     AngeloRTV_Get_Image_Config(PortNo:Smallint;
     ConfigIndex:Byte;var Value:Byte):Smallint;
AngeloRTV_Set_Color_Format(PortNo:Smallint;Color
     Format:Byte):Smallint;
```

```
AngeloRTV_Get_Color_Format(PortNo:Smallint;var
      ColorFormat:Byte):Smallint;
AngeloRTV_Set_Video_Format(PortNo:Smallint;Video
      Format:Byte):Smallin;
AngeloRTV_Get_Video_Format(PortNo:Smallint;var
      VideoFormat:Byte):Smallint;
AngeloRTV_Set_Image_Geometric (PortNo:Smallint;
      X_Offset:LongInt; Y_Offset:LongInt;
      X_Active:LongInt; Y_Active:LongInt;
      X_Scale:Double; Y_Scale:Double):Smallint;
AngeloRTV_Detect_Video_Format(PortNo:Smallint;
      var FormatValue:Byte):Smallint;
```

@ Arguments

PortNo:

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.”

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 90
- LUMA 0 ---- 255 124
- LUMA notch filter 0(Enable) or 1(Disable)

Color Format:

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

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,

Note: Please do not use Full NTSC and Full PAL format to acquire dynamic object image, because the interlaced scanning may not be able to present clear image for it.

X_Scale:

This parameter is the scaling factor applied to the Angelo sampled line to obtain pixels according to the resolution.

X_Active

This parameter value is the length of the active video line

X_Offset

This parameter value is the number of scaled pixels to skip before the start of the active video line.

Y_Scale:

This parameter is the scaling factor applied to the Angelo sampled data lines in the vertical direction.

Y_Active

This parameter value is the height (in lines) of the active video image.

Y_Offset

This parameter value is the number of lines to skip before the first line of the active video image.

FormatValue:

If the return value of this parameter is 0 that means there is no video signal input. And if the value is 1 or 2, the video format of the port is NTSC. Otherwise, if the value is 3, 4 or 5, the video format of the port is PAL.

@ Example

<VC/BCB >

AngeloRTV_Set_Image_Config –

AngeloRTV_Get_Image_Config –

```
I16 Result;
I16 PortNo = 0;
U8 ConfigIndex = 0;
U8 Value = 128;
Result = AngeloRTV_Set_Image_Config (PortNo,
ConfigIndex, Value);
Result = AngeloRTV_Get_Image_Config (PortNo,
ConfigIndex, &Value);
```

AngeloRTV_Set_Color_Format –

AngeloRTV_Get_Color_Format –

AngeloRTV_Set_Video_Format –

AngeloRTV_Get_Video_Format –

```
I16 Result;
I16 PortNo = 0;
U8 VideoFormat = 0;
U8 ColorFormat = 3;
Result = AngeloRTV_Set_Color_Format(PortNo,
     ColorFormat);
Result = AngeloRTV_Get_Color_Format(PortNo,
     &ColorFormat);
Result = AngeloRTV_Set_Video_Format(PortNo,
     VideoFormat);
Result = AngeloRTV_Get_Video_Format(PortNo,
     &VideoFormat);
```

AngeloRTV_Set_Image_Geometric –

```
I16 Result;
I16 PortNo = 0;
U32 X_Active = 600;
U32 Y_Active = 400;
U32 X_Offset = 40;
```

```
U32 Y_Offset = 80;
Double X_Scale = 1.0;
Double Y_Scale = 1.0;
Result = AngeloRTV_Set_Image_Geometric (PortNo,
     X_Offset, Y_Offset, X_Active, Y_Active,
     X_Scale, Y_Scale);
```

AngeloRTV_Detect_Video_Format –

```
I16 Result;
U16 PortNo;
U8   FormatValue;
PortNo = 0;
Result = AngeloRTV_Detect_Video_Format (PortNo,
     &FormatValue);
```

< Visual Basic >

AngeloRTV_Set_Image_Config –
AngeloRTV_Get_Image_Config –

```
Dim Result As Integer
Dim PortNo As Integer
Dim ConfigIndex As Byte
Dim Value As Byte
PortNo = 0
ConfigIndex =0
Value = 128
Result = AngeloRTV_Set_Image_Config (ByVal
     PortNo, ByVal ConfigIndex, ByVal Value)
Result = AngeloRTV_Get_Image_Config (ByVal
     PortNo, ByVal ConfigIndex, ByRef Value)
```

AngeloRTV_Set_Color_Format –
AngeloRTV_Get_Color_Format –
AngeloRTV_Set_Video_Format –
AngeloRTV_Get_Video_Format –

```
Dim Result As Integer
Dim PortNo As Integer
Dim ColorFormat As Byte
Dim VideoFormat As Byte
PortNo = 0
ColorFormat =3
VideoFormat = 0
Result = AngeloRTV_Set_Color_Format(ByVal PortNo,
     ByVal ColorFormat)
Result = AngeloRTV_Get_Color_Format(ByVal PortNo,
     ByRef ColorFormat)
```

```
Result = AngeloRTV_Set_Video_Format(ByVal PortNo,
     ByVal VideoFormat)
Result = AngeloRTV_Get_Video_Format(ByVal PortNo,
     ByRef VideoFormat)
```

AngeloRTV_Set_Image_Geometric –

```
Dim Result As Integer
Dim PortNo As Integer
Dim X_Active As Long
Dim Y_Active As Long
Dim X_Offset As Long
Dim Y_Offset As Long
Dim X_Scale As Double
Dim Y_Scale As Double
PortNo = 0
X_Active = 600
Y_Active = 400
X_Offset = 40
Y_Offset = 80
X_Scale = 1.0
Y_Scale = 1.0
Result = AngeloRTV_Set_Image_Geometric (PortNo,
     X_Offset, Y_Offset, X_Active, Y_Active,
     X_Scale, Y_Scale)
```

AngeloRTV_Detect_Video_Format –

```
Dim Result As Integer
Dim PortNo As Integer
Dim FormatValue As Byte
PortNo = 0
Result = AngeloRTV_Detect_Video_Format (ByVal
     PortNo, ByRef FormatValue)
```

<Delphi >

AngeloRTV_Set_Image_Config –

AngeloRTV_Get_Image_Config –

```
Var
Result : SmallInt;
PortNo : SmallInt;
ConfigIndex: Byte;
Value: Byte;
PortNo:=0;
ConfigIndex:=0;
Value:=0;
Result := AngeloRTV_Set_Image_Config
     (PortNo,ConfigIndex, Value);
```

```
Result := AngeloRTV_Get_Image_Config (PortNo,
     ConfigIndex, Value);
```

AngeloRTV_Set_Color_Format –
AngeloRTV_Get_Color_Format –
AngeloRTV_Set_Video_Format –
AngeloRTV_Get_Video_Format –

```
Var
Result : SmallInt;
PortNo : SmallInt;
VideoFormat: Byte;
ColorFormat: Byte;
PortNo:=0;
VideoFormat:=0;
ColorFormat:=3;
Result :=
     AngeloRTV_Set_Color_Format(PortNo,ColorForm
     at);
Result :=
     AngeloRTV_Get_Color_Format(PortNo,ColorForm
     at);
Result :=
     AngeloRTV_Set_Video_Format(PortNo,VideoForm
     at);
Result := AngeloRTV_Get_Video_Format(PortNo,
     VideoFormat);
```

AngeloRTV_Set_Image_Geometric –

```
Var
Result : SmallInt;
PortNo : SmallInt;
X_Active : LongInt;
Y_Active : LongInt;
X_Offset : LongInt;
Y_Offset : LongInt;
X_Scale : Double;
Y_Scale : Double;
PortNo := 0;
X_Active := 600;
Y_Active := 400;
X_Offset := 40;
Y_Offset := 80;
X_Scale := 1.0;
Y_Scale := 1.0;
```

```
Result := AngeloRTV_Set_Image_Geometric(PortNo,
     X_Offset, Y_Offset, X_Active, Y_Active,
     X_Scale, Y_Scale);
```

AngeloRTV_Detect_Video_Format –

```
var
Result : SmallInt;
PortNo : SmallInt;
FormatValue : Byte;
PortNo := 0;
Result := AngeloRTV_Detect_Video_Format (PortNo,
     FormatValue);
```

5.5 Image Grabbing

@ Name

AngeloRTV_Capture_Start(PortNo, CaptureNo)

Start to grab the video image

AngeloRTV_Select_Channel(PortNo, Multiplex)

Channel extension of video signal, for advanced only

AngeloRTV_Capture_Stop(PortNo)

Stop to grab the video image

AngeloRTV_Capture_Config(PortNo, Start_Field)

Set the starting field of image

AngeloRTV_Sync_Grab(PortNo, Start_Address, Width, Height, Size_Byte)

Get an image frame with start address of memory

@ Description

AngeloRTV_Capture_Start:

Continuously captures video frames and stops when the total frame number equals the “CaptureNo” parameter. The frame update rate is 30 frames/sec. If the “CaptureNo” is 0xFFFFFFFF, the frame grabbing will not stop until the “AngeloRTV_Capture_Stop” function is called.

AngeloRTV_Capture_Stop:

Stop grabbing video frames.

AngeloRTV_Select_Channel:

Angelo series cards are capable of channel extension. This function is used to multiplex video signals for the ports. In most cases using this function should not be required because the default setting is one port is dedicated to one channel.

Note: Do not call this function if there is no channel extension board in the system.

AngeloRTV_Capture_Config:

Chooses the starting field of image.

AngeloRTV_Sync_Grab:

This is a synchronous image grabbing function to get an image frame. Retrieve the memory start address from the frame data, width, height, and size in bytes of the image.

@ Syntax

C/C++ (Windows/CE.NET)

```
I16 AngeloRTV_Capture_Start (U16 PortNo, U32
      CaptureNo)
I16 AngeloRTV_Select_Channel (U16 PortNo, U16
      Multiplex)
I16 AngeloRTV_Capture_Stop (U16 PortNo)
I16 AngeloRTV_Capture_Config (U16 PortNo, U32
      Start_Field)
I16 AngeloRTV_Sync_Grab(U16 PortNo, U32*
      Start_Address, U32* Width, U32* Height, U32*
      Size_Byte)
```

Visual Basic (Windows/CE.NET)

```
AngeloRTV_Capture_Start (ByVal PortNo As Integer,
      ByVal CaptureNo As Long) As Integer
AngeloRTV_Select_Channel (ByVal PortNo As
      Integer, ByVal Multiplex As Integer) As
      Integer
AngeloRTV_Capture_Stop (ByVak PortNo As Integer)
      As Integer
AngeloRTV_Capture_Config (ByVal PortNo As
      Integer, ByVal Start_Field As Long) As
      Integer
AngeloRTV_Sync_Grab(ByVal PortNo As Integer,
      ByRef Start_Address As Long, ByRef Width as
      Long, ByRef Height As Long, ByRef Size_byte
      As Long) As Integer
```

Delphi (Windows)

```
AngeloRTV_Capture_Start (PortNo:Smallint;
      CaptureNo:LongInt):Smallint
AngeloRTV_Select_Channel (PortNo:Smallint;
      Multiplex:SmallInt):Smallint
AngeloRTV_Capture_Stop
      (PortNo:Smallint):Smallint
AngeloRTV_Capture_Config (PortNo:Smallint;
      Start_Field:LongInt):Smallint
AngeloRTV_Sync_Grab(PortNo:Smallint; var
      Start_Address:Pointer; var Width:Longint;
      var Height:Longint; var
      Size_byte:Longint):Smallint
```

@ Argument

PortNo:

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.”

CaptureNo:

Total number of frames to capture. If the “CaptureNo” is 0xFFFFFFFF, the frame grabbing will not stop until the “AngeloRTV_Capture_Stop” function is called.

Multiplex:

Indicates the multiplex channels.

- Bit 0 : Channel 0, 0 for disable ; 1 for enable.
- Bit 1 : Channel 1, 0 for disable ; 1 for enable.
- Bit 2 : Channel 2, 0 for disable ; 1 for enable.
- Bit 3 : Channel 3, 0 for disable ; 1 for enable.

For example:

- Multiplex = 1, only channel 0 is enable
- Multiplex = 2, only channel 1 is enable
- Multiplex = 15, four channels are enable

Start_Filed:

Indicates the first field of image.

- 0: first field is Odd, so the image will be Odd field + Even field.
- 1: first field is Even, so the image will be Even field + Odd field.
- 2: first field depends on the current field, so the image will be Even field + Odd field, or Odd field + Even field.

Start_Address:

Memory start address of the video frame.

Width:

Image width.

Height:

Image height.

Size_Byte:

Memory size in bytes.

@ Return Code

- 0: ERROR_NoError
- -7: ERROR_Not_Initialized – Make sure the port has been initialized by “AngeloRTV_Initial”.
- -9: ERROR_Invalid_PortNo – Please input the correct “PortNo” parameter.

@ Example

<VC/BCB >

AngeloRTV_Capture_Config –
AngeloRTV_Capture_Start –
AngeloRTV_Sync_Grab –
AngeloRTV_Capture_Stop –

```
I16 Result;
U16 PortNo = 0;
U32 CaptureNo = 0xFFFFFFFF;
U32 Start_Field = 0;
U32 StrAddr;
U32 Width, Height, Size_Byte;
Result = AngeloRTV_Capture_Config (PortNo,
     Start_Field);
Result = AngeloRTV_Capture_Start (PortNo,
     CaptureNo);
Result = AngeloRTV_Sync_Grab (PortNo, &StrAddr,
     &Width, &Height, &Size_Byte);
Result = AngeloRTV_Capture_Stop (PortNo);
```

< Visual Basic >

AngeloRTV_Capture_Config –
AngeloRTV_Capture_Start –
AngeloRTV_Sync_Grab –

```
AngeloRTV_Capture_Stop –
Dim Result As Integer
Dim PortNo As Integer
Dim CaptureNo As Long
Dim Start_Field As Long
Dim StrAddr As Long
Dim Width as Long, Height As Long, Size_Byte As
     Long
PortNo = 0
CaptureNo = &HFFFFFFFF
Start_Field = 0
```

```
Result = AngeloRTV_Capture_Config (ByVal PortNo,
     ByVal Start_Field)
Result = AngeloRTV_Capture_Start (ByVal PortNo,
     ByVal CaptureNo)
Result = AngeloRTV_Sync_Grab (ByVal PortNo,
     StrAddr, Width, Height, Size_Byte)
Result = AngeloRTV_Capture_Stop (ByVal PortNo)
```

<Delphi >

AngeloRTV_Capture_Config –
AngeloRTV_Capture_Start –
AngeloRTV_Sync_Grab –
AngeloRTV_Capture_Stop –

```
Var
Result : SmallInt;
PortNo: SmallInt;
CaptureNo: LontInt;
Start_Field: LontInt;
StrAddr: Pointer;
Width, Height, Size_Byte: LongInt;
begin
PortNo:=0;
Start_Field :=0;
CaptureNo:= INFINITE;
Result := AngeloRTV_Capture_Config (PortNo,
     Start_Field);
Result := AngeloRTV_Capture_Start (PortNo,
     CaptureNo);
Result := AngeloRTV_Sync_Grab (PortNo, StrAddr,
     Width, Height, Size_Byte);
Result: = AngeloRTV_Capture_Stop (PortNo);
end;
```

5.6 GPIO & EEPROM Functions

@ Name

AngeloRTV_Set_GPIO_Sts (PortNo, Status)

Set Digital Output status.

AngeloRTV_Get_GPIO_Sts (PortNo, Status)

Get Digital Input status.

AngeloRTV_Set_GPIO_Int_Logic (PortNo, Logic)

Configure the Digital Input Interrupt condition

AngeloRTV_Write_EEPROM (PortNo, Offset, Value)

Write data into EEPROM

AngeloRTV_Read_EEPROM (PortNo, Offset, Value)

Read data from EEPROM

AngeloRTV_Set_LED_Sts (PortNo, LEDStatus)

Set LED status for cPci RTV24 card.

@ Description

AngeloRTV_Set_GPIO_Sts:

There is one digital output channel in each port of the Angelo series card, use this function to set the digital output status.

AngeloRTV_Get_GPIO_Sts:

There is one digital input channel in each port of Angelo series card, use this function to get the digital input status.

AngeloRTV_Set_GPIO_Int_Logic:

This function used to configure the Digital Input Interrupt condition.

AngeloRTV_Write_EEPROM:

Writes data into the EEPROM. Data in EEPROM will not be lost even when powered off.

AngeloRTV_Read_EEPROM:

Reads data from the EEPROM. Data in EEPROM will not be lost even when powered off.

AngeloRTV_Set_LED_Sts:

Use the function to set LED status. The function is for cPci RTV24 card only.

@ Syntax

C/C++ (Windows/CE.NET)

```
I16 AngeloRTV_Set_GPIO_Sts(U16 PortNo,U8 Status);
I16 AngeloRTV_Get_GPIO_Sts(U16 PortNo,U8*
    Status);
I16 AngeloRTV_Set_GPIO_Int_Logic(U16 PortNo, U16
    Logic);
I16 AngeloRTV_Write_EEPROM(U16 CardNo, U8 Offset,
    U8 Value);
I16 AngeloRTV_Read_EEPROM(U16 CardNo, U8 Offset,
    U8* Value);
I16 AngeloRTV_Set_LED_Sts (U16 PortNo, U8
    LEDStatus);
```

Visual Basic (Windows/CE.NET)

```
AngeloRTV_Set_GPIO_Sts (ByVal PortNo As Integer,
    ByVal Status As Byte) As Integer
AngeloRTV_Get_GPIO_Sts (ByVal PortNo As Integer,
    ByRef Status As Byte) As Integer
AngeloRTV_Set_GPIO_Int_Logic(ByVal PortNo As
    Integer, ByVal Logic As Integer) As Integer
AngeloRTV_Write_EEPROM (ByVal PortNo As Integer,
    ByVal Offset As Byte, ByVal Value As Byte)
    As Integer
AngeloRTV_Read_EEPROM (ByVal PortNo As Integer,
    ByVal Offset As Byte, ByRef Value As Byte)
    As Integer
AngeloRTV_Set_LED_Sts (ByVal PortNo As Integer,
    ByVal LEDStatus As Byte) As Integer
```

Delphi (Windows)

```
AngeloRTV_Set_GPIO_Sts
    (PortNo:Smallint;status:Byte):Smallint;
AngeloRTV_Get_GPIO_Sts (PortNo:Smallint;var
    status:Byte):Smallint;
AngeloRTV_Set_GPIO_Int_Logic(PortNo:Smallint;
    Logic:Smallint):Smallint;
AngeloRTV_Write_EEPROM (
    PortNo:Smallint;Offset:Byte;Value:Byte):Sma
    llint;
AngeloRTV_Read_EEPROM ( PortNo:Smallint;
    Offset:Byte;var Value:Byte):Smallint;
AngeloRTV_Set_LED_Sts (PortNo:Smallint;
    LEDStatus:Byte):Smallint;
```

@ Argument

PortNo:

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.”

Status:

The digital input or digital output status

- 0 Low
- 1 High

Logic:

The digital input interrupt condition

- 0: Active Low
- 1: Active High

Offset:

The offset address of the EEPROM. This parameter is valid between 0 and 127

Value: The value in Byte data type, this parameter is valid between 0 and 255.

LEDStatus:

Use the parameter to set the LED status.

- LEDStatus = 1: High
- LEDStatus = 0: Low

@ Return Code

- 0: ERROR_NoError
- -7: ERROR_Not_Initialized – Make sure the port has been initialized by “AngeloRTV_Initial”.
- -9: ERROR_Invalid_PortNo – Please input the correct “PortNo” parameter.
- -15: ERROR_Invalid_Address – a valid offset address is between 0 and 127

@ Example

<VC/BCB >

AngeloRTV_Set_GPIO_Sts –

AngeloRTV_Get_GPIO_Sts –

```
I16 Result;
I16 PortNo = 0;
U8    Status = 1;
Result = AngeloRTV_Set_GPIO_Sts (PortNo, Status);
Result = AngeloRTV_Get_GPIO_Sts (PortNo, &
     Status);
```

AngeloRTV_Set_GPIO_Int_Logic –

```
I16 Result;
U16 PortNo = 0;
U16 Logic = 0;
Result = AngeloRTV_Set_GPIO_Int_Logic (PortNo,
     Logic);
```

AngeloRTV_Write_EEPROM
AngeloRTV_Read_EEPROM

```
I16 Result;
I16 PortNo = 0;
U8    Offset = 0;
U8    Value = 128;
Result = AngeloRTV_Write_EEPROM (PortNo, Offset,
     Value);
Result = AngeloRTV_Read_EEPROM (PortNo, Offset,
     &Value);
```

AngeloRTV_Set_LED_Sts –

```
I16 Result;
U16 PortNo;
U8  LEDStatus;
PortNo = 0;
LEDStatus = 1;
Result = AngeloRTV_Set_LED_Sts (PortNo,
     LEDStatus);
```

< Visual Basic >

AngeloRTV_Set_GPIO_Sts –
AngeloRTV_Get_GPIO_Sts –

```
Dim Result As Integer
Dim PortNo As Integer
Dim Status As Byte
PortNo = 0
Status = 1
Result = AngeloRTV_Set_GPIO_Sts (ByVal PortNo,
     ByVal Status)
Result = AngeloRTV_Get_GPIO_Sts (ByVal PortNo,
     ByRef Status)
```

AngeloRTV_Set_GPIO_Int_Logic –

```
Dim Result As Integer
Dim PortNo As Integer
Dim  Logic As Integer
PortNo = 0
Logic = 0
Result = AngeloRTV_Set_GPIO_Int_Logic (ByVal
     PortNo, ByVal Logic)
```

AngeloRTV_Write_EEPROM

AngeloRTV_Read_EEPROM

```
Dim Result As Integer
Dim PortNo As Integer
Dim Offset As Byte
Dim Value As Byte
PortNo = 0
Offset =0
Value = 128
Result = AngeloRTV_Write_EEPROM(ByVal PortNo,
     ByVal Offset, ByVal Value)
Result = AngeloRTV_Read_EEPROM(ByVal PortNo,
     ByVal Offset, ByRef Value)
```

AngeloRTV_Set_LED_Sts –

```
Dim Result As Integer
Dim PortNo As Integer
Dim LEDStatus As Byte
PortNo = 0
LEDStatus = 1
Result = AngeloRTV_Set_LED_Sts (ByVal PortNo,
     ByVal LEDStatus)
```

<Delphi >

AngeloRTV_Set_GPIO_Sts –

AngeloRTV_Get_GPIO_Sts –

```
Var
Result : SmallInt;
PortNo : SmallInt;
Status: Byte;
PortNo:=0;
Status:=1;
Result := AngeloRTV_Set_GPIO_Sts (PortNo,
     Status);
Result := AngeloRTV_Get_GPIO_Sts (PortNo,
     Status);
```

AngeloRTV_Set_GPIO_Int_Logic –

```
var
Result: SmallInt;
PortNo: SmallInt;
Logic: SmallInt;
PortNo := 0;
Logic := 0;
Result := AngeloRTV_Set_GPIO_Int_Logic (PortNo,
     Logic);
```

AngeloRTV_Write_EEPROM
AngeloRTV_Read_EEPROM

```
Var
Result : SmallInt;
PortNo : SmallInt;
Offset: Byte;
Value: Byte;
PortNo:=0;
Offset:=0;
Value:=128;
Result := AngeloRTV_Write_EEPROM (PortNo, Offset,
     Value);
Result := AngeloRTV_Read_EEPROM (PortNo, Offset,
     Value);
```

AngeloRTV_Set_LED_Sts –

```
var
Result: Smallint;
PortNo: Smallint;
LEDStatus: Byte;
PortNo := 0;
LEDStatus := 1;
Result := AngeloRTV_Set_LED_Sts (PortNo,
     LEDStatus);
```

5.7 Callback & Thread Functions

@ Name

AngeloRTV_Get_Int_Status (PortNo, IntStatus)

Gets the current interrupt status

AngeloRTV_Set_Int_Event (PortNo,hEvent)

Assigns the windows interrupt event

AngeloRTV_Set_Callback(PortNo, CallBackProc)

Sets the callback function when an interrupt is generated

@ Description

AngeloRTV_Get_Int_Status:

Allows users to identify what caused an interrupt signal.

- Bit 0: GPIO interrupt, when Digital input channel is changed.
- Bit 1: Channel 0 Image ready
- Bit 2: Channel 1 Image ready
- Bit 3: Channel 2 Image ready
- Bit 4: Channel 3 Image ready

Note: There are four channels in each port, the default channel is channel 0.

AngeloRTV_Set_Int_Event:

Links interrupt events. Users only have to declare the “hEvent” variable and call this function to DLL, the DLL will link the event and interrupt automatically.

AngeloRTV_Set_Callback:

Links the callback function when an interrupt is generated to host pc.

Note: There are two ways to use the synchronization mechanism, one is the callback function, and the other is the thread function.

@ Syntax

C/C++ (Windows/CE.NET)

```
I16 AngeloRTV_Get_Int_Status(U16 PortNo,U32
     *IntStatus);
I16 AngeloRTV_Set_Int_Event(U16 PortNo,HANDLE*
     hEvent);
```

```
I16 AngeloRTV_Set_Callback (U16 PortNo, void (
     __stdcall *CallBackProc)(U32
     VideoBufferaddress ,U16 PortNo));
```

Visual Basic (Windows/CE.NET)

```
AngeloRTV_Set_Int_Event (ByVal PortNo As Integer,
     ByRef hEvent As Long) As Integer
AngeloRTV_Get_Int_Status(ByVal PortNo As Integer,
     ByRef IntStatus As Long) As Integer
AngeloRTV_Set_Callback(ByVal PortNo As Integer,
     ByVal CallBack As Long) As Integer
```

Delphi (Windows)

```
AngeloRTV_Set_Int_Event(PortNo:Smallint;var
     hEvent:Integer):Smallint;
AngeloRTV_Get_Int_Status(PortNo:Smallint;var
     IntStatus:Longint):Smallint;
AngeloRTV_Set_Callback(PortNo:Smallint;lpCallBac
     kProc:CallbackFunc):Smallint;
```

@ Argument

PortNo:

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.”

IntStatus:

Interrupt status

- Bit 0:GPIO interrupt, when Digital input channel is changed.
- Bit 1:Channel 0 Image ready
- Bit 2:Channel 1 Image ready
- Bit 3:Channel 2 Image ready
- Bit 4:Channel 3 Image ready

hEvent:

Interrupt event handle.

@ Return Code

- 0: ERROR_NoError
- -7: ERROR_Not_Initialized – Make sure the port has been initialized by “AngeloRTV_Initial”.
- -9: ERROR_Invalid_PortNo – Please input the correct “PortNo” parameter.

@ Example

< VC/BCB >

Use Thread:

```
HANDLE hEvent=NULL;
void *pThread=NULL;
U32 threadID;
U16 PortNo = 0;

DWORD nObj;
U32 Size_Byte;
U32  Status =0;
I16 ISR_ON=0;
DWORD WINAPI IntThreadProc( LPVOID lpParam )
{
      while( ISR_ON )
      {
      nObj = WaitForSingleObject(hEvent,
      INFINITE);
      AngeloRTV_Get_Int_Status(PortNo,&Status);
      if((Status&0x01)==1)//GPIO
      {
      }
      if((Status>>1&0x01)==1)//Channel 0 of the
      nPort
      {
      }
      else if((Status>>2&0x01)==1)//Channel 1 of
      the nPort
      {
      }
      else if((Status>>3&0x01)==1)//Channel 2 of
      the nPort
      {
      }
      else if((Status>>4&0x01)==1)//Channel 3 of
      the nPort
```

```
        {
        }
        ResetEvent(hEvent);
        }
        Return TRUE;
    }
    AngeloRTV_Set_Int_Event(PortNo,&hEvent);
    pThread =CreateThread(NULL, 0, IntThreadProc, 0,
        0, &threadID);
```

Use Callback Function:

```
    U16 PortNo = 0;
    void __stdcall MediaStreamProc( U32
        VideoBufferaddress ,U16 PortNo)
    {
        U32 Status;
        AngeloRTV_Get_Int_Status(PortNo,&Status);
        if((Status&0x01)==1)//GPIO
        {
        }
        if((Status>>1&0x01)==1)//Channel 0 of the
        nPort
        {
        }
        else if((Status>>2&0x01)==1)//Channel 1 of
        the nPort
        {
        }
        else if((Status>>3&0x01)==1)//Channel 2 of
        the nPort
        {
        }
        else if((Status>>4&0x01)==1)//Channel 3 of
        the nPort
        {
        }
     }
    AngeloRTV_Set_Callback(PortNo,MediaStreamProc);
```

< Visual Basic >

Use Callback Function

```
    Dim Result As Integer
    Dim PortNo As Integer
    Public Sub lpcallback(ByVal VideoBufferaddress As
        Long, ByVal PortNo As Integer)
```

```
Dim Status As Long
Result  = AngeloRTV_Get_Int_Status(PortNo,
     Status)
End Sub
PortNo = 0
Result = AngeloRTV_Set_Callback(PortNo, AddressOf
     lpcallback)
```

<Delphi >

Use Thread

```
Var
ISR_ON : SmallInt;
Event_Angelo:Integer;
ThreadId : LongInt;
PortNo: SmallInt;
PortNo:=0;
     function ThreadFunc(Parameter: Pointer):
     Integer ;
          var
        Str_Add :Pointer;
        Size_Byte :Longint;
        intstatus : LongInt;
begin
        while(ISR_ON=1) do
        begin

     WaitForSingleObject(Event_Angelo,INFINITE);
               ResetEvent(Event_Angelo);

     AngeloRTV_Get_Int_Status(PortNo,intstatus);
               if intstatus = 2 then //image
     ready for channel 0 of port
               begin

               end;
        end;
end;

     AngeloRTV_Set_Int_Event(PortNo,Event_Angelo
     );
     ISR_ON :=1;
       Mythread :=
     BeginThread(nil,0,ThreadFunc,nil,0,ThreadId
     );
```

Use Callback function

```
var
PortNo: SmallInt;
PortNo:=0;
procedure MyCallback(VideoBufferAddress :
      LongInt;PortNo : SmallInt);stdcall
var
      Str_Add :Pointer;
      Result :Smallint;
      Size_Byte :LongInt;
      intstatus :LongInt;
begin
      AngeloRTV_Get_Int_Status(PortNo,intstatus);
         if intstatus = 2 then
         begin
         end;
      end;
AngeloRTV_Set_Callback(Cur_Port,MyCallback);
```

5.8 Watchdog Timer

Note: This function is only available for RTV-24.

@ Name

AngeloRTV_Set_WDT(CardNo, Enable, Interval)

Sets the watch dog status(Only for PCI-RTV24)

@ Description

AngeloRTV_Set_WDT:

Enables or disables the watch dog timer in the Angelo series cards, and set the interval of timer. When users have enabled the watch dog timer and selected a 16 seconds interval, a system reset signal will be triggered if this function is not called after 16 seconds.

@ Syntax

C/C++ (Windows/CE.NET)

```
I16 AngeloRTV_Set_WDT (U16 CardNo,U16 Enable,U16
      Interval)
```

Visual Basic (Windows/CE.NET)

```
AngeloRTV_Set_WDT (ByVal PortNo As Integer, ByVal
      Enable As Integer, ByVal Interval As
      Integer) As Integer
```

Delphi (Windows)

```
AngeloRTV_Set_WDT(CardNo:Smallint;enable:Smallin
      t;interval:Smallint):Smallint;
```

@ Argument

CardNo:

Card number is the zero index in Angelo series card. For example, if there are two Pci-RTV-24 Angelo cards (card 0, card 1) in the system, “CardNo” of card 0 is 0, and 1 for card 1.

Enable:

Enables or disables the watch dog timer. 0 for disable, 1 for enable.

Interval:

Indicates the watch dog timer interval.

- 1: 8 seconds
- 2: 16 seconds
- 3: 32 seconds

@ Return Code

- 0 : ERROR_NoError
- -7: ERROR_Not_Initialized – Make sure the port has been initialized by “AngeloRTV_Initial”.
- -9 : ERROR_Invalid_PortNo – Please input the correct “PortNo” parameter.

@ Example

<VC/BCB >

AngeloRTV_Set_WDT

```
I16 Result;
U16 CardNo = 0;
U16 Enable = 1;
U16 Interval = 1;
Result =
     AngeloRTV_Set_WDT(CardNo,Enable,Interval);
```

< Visual Basic >

AngeloRTV_Set_WDT

```
Dim Result As Integer
Dim CardNo As Integer
Dim Enable As Integer
Dim Interval As Integer
CardNo = 0
Enable = 1
Interval = 1
Result =
     AngeloRTV_Set_WDT(CardNo,Enable,Interval)
```

<Delphi >

AngeloRTV_Set_WDT

```
Var
Result : SmallInt;
CardNo: SmallInt;
Enable: SmallInt;
Interval: SmallInt;
CardNo :=0;
Enable:=1;
Interval:=1;
Result :=
     AngeloRTV_Set_WDT(CardNo,Enable,Interval);
```

5.9 Software Trigger

@ Name

AngeloRTV_Trigger_Config (PortNo,Interval)

Sets software trigger configuration(Only for PCI-RTV24, cPCI-RTV-24, cPCI-RTV44)

AngeloRTV_Trigger_Start (CardNo, Multiplex)

Generates single or multiple trigger output simultaneously(Only for PCI-RTV24, cPCI-RTV-24, cPCI-RTV44)

@ Description

AngeloRTV_Trigger_Config:

Configures the pulse output interval.

AngeloRTV_Trigger_Start:

Generates a one shot pulse output for single or multiple ports.

@ Syntax

C/C++ (Windows/CE.NET)

```
I16 AngeloRTV_Trigger_Config(U16 PortNo,U16
      Interval);
I16 AngeloRTV_Trigger_Start(U16 CardNo,U16
      Multiplex);
```

Visual Basic (Windows/CE.NET)

```
AngeloRTV_Trigger_Config (ByVal PortNo As
      Integer, ByVal Interval As Integer) As
      Integer
AngeloRTV_Trigger_Start (ByVal CardNo As Integer,
      ByVal Multiplex As Integer) As Integer
```

Delphi (Windows)

```
AngeloRTV_Trigger_Config (PortNo:Smallint;
      Interval:Smallint):Smallint;
AngeloRTV_Trigger_Start (CardNo:Smallint;
      Multiplex:Smallint):Smallint;
```

@ Argument

CardNo:

Card number is the zero index in Angelo series card. For example, if there are two Pci-RTV-24 Angelo cards (card 0, card 1) in the system, “CardNo” of card 0 is 0, and 1 for card 1.

PortNo:

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.”

Interval:

Indicates the trigger output interval, the valid range is from 0 to 253, the definition is as following

- 0: 16ms
- 32: 12ms
- 128: 8ms
- 253: 60µs

Multiplex:

Indicates the trigger output ports in Angelo series cards.

- Bit 0: Port 0 on each card. 0 for disable, 1 for enable.
- Bit 1: Port 1 on each card. 0 for disable, 1 for enable.
- Bit 2: Port 2 on each card. 0 for disable, 1 for enable.
- Bit 3: Port 3 on each card. 0 for disable, 1 for enable.

For example:

- Multiplex = 1, only port 0 in each Angelo series card generates a trigger output.
- Multiplex = 2, only port 1 in each Angelo series card generates a trigger output.
- Multiplex = 15, four ports in each Angelo series card generates a trigger output.

@ Return Code

- 0: ERROR_NoError
- -7: ERROR_Not_Initialized – Make sure the port has been initialized by “AngeloRTV_Initial”.
- -9: ERROR_Invalid_PortNo – Please input the correct “PortNo” parameter.

@ Example

<VC/BCB >

AngeloRTV_Trigger_Config
AngeloRTV_Trigger_Start

```
I16 Result;
U16 CardNo = 0;
U16 PortNo = 0;
U16 Multiplex = 1;
U16 Interval = 32;
Result =
     AngeloRTV_Trigger_Config(PortNo,Interval);
Result = AngeloRTV_Trigger_Start(CardNo,
     Multiplex);
```

< Visual Basic >

AngeloRTV_Trigger_Config

AngeloRTV_Trigger_Start

```
Dim Result As Integer
Dim CardNo As Integer
Dim PortNo As Integer
Dim Multiplex As Integer
Dim Interval As Integer
CardNo = 0
PortNo = 0
Multiplex = 1
Interval = 32
Result = AngeloRTV_Trigger_Config
     (PortNo,Interval)
Result = AngeloRTV_Trigger_Start (CardNo,
     Multiplex)
```

<Delphi >

AngeloRTV_Trigger_Config

AngeloRTV_Trigger_Start

```
Var
Result : SmallInt;
CardNo: SmallInt;
PortNo: SmallInt;
Multiplex: SmallInt;
Interval: SmallInt;
CardNo :=0;
PortNo:=0;
Multiplex:=1;
Interval:=32;
Result := AngeloRTV_Trigger_Config
     (PortNo,Interval);
Result := AngeloRTV_Trigger_Start (CardNo,
     Multiplex);
```

5.10 Frame Buffer

@ Name

AngeloRTV_Copy_frame (PortNo, Dest_Address, Size_Byte)

Copies the frame date to the user allocated destination memory (bytes).

AngeloRTV_Get_frame(PortNo, Start_Address,Width, Height, Size_Byte)

Gets the frame memory start address and size of frame (bytes).

AngeloRTV_Save_File(PortNo, FileName, FileFormat, nQuality)

Save the video frame into an image file.

@ Description

AngeloRTV_Copy_frame:

Copies frame data to memory or an array that the user has allocated. Before using this function, remember to allocate enough memory address space or array elements.

AngeloRTV_Save_File:

Saves the current video frame into an image file (TIF, BMP, or JPEG). nQuality is only used JPEGs.

AngeloRTV_Get_frame:

Retrieves the memory start address from the frame data, width, height, and size in bytes of the image. For example a FULL NTSC RGB24 video frame will occupy 900K Byte (640*480*3) memory address space.

Format	DWORD(32Bit)	Pixel Data			
		Byte 3 Bit [31:24]	Byte 2 Bit[23:16]	Byte 1 Bit[15:8]	Byte 0 Bit[7:0]
RGB32	Dw0	Appha	R	G	B
RGB24	Dw0	B1	R0	G0	B0
	Dw1	G2	B2	R1	G1
	Dw2	R3	G3	B3	R2
RGB16	Dw0	{R0[31:27], G0[26:21], B0[20:16]}	{R0[15:11], G0[10:5], B0[4:0]}		

Table 5-3: Pixel Data

Format	DWORD(32Bit)	Pixel Data			
RGB15	Dw0	{0,R0[30:26], G0[25:21], B0[20:16]}	{0,R0[14:10], G0[9:5], B0[4:0]}		
Gray Scale(Y8)	Dw0	Y3	Y2	Y1	Y0

Table 5-3: Pixel Data

@ Syntax

C/C++ (Windows/CE.NET)

```
I16 AngeloRTV_Copy_Frame(U16 PortNo,U8
    *Dest_Address,U32 Size_Byte);
I16 AngeloRTV_Get_Frame(U16 PortNo,U32*
    Start_Address, U32* Width, U32* Height, U32*
    Size_Byte);
16 AngeloRTV_Save_File(U16 PortNo, char*
    FileName,U8 FileFormat,U32 nQuality);
```

Visual Basic (Windows/CE.NET)

```
AngeloRTV_Copy_Frame (ByVal PortNo As Integer,
    Dest_Address As Byte, ByVal Size_byte As
    Long) As Integer
AngeloRTV_Get_Frame (ByVal PortNo As Integer,
    ByRef Start_Address As Long, ByRef Width as
    Long, ByRef Height As Long, ByRef Size_byte
    As Long) As Integer
AngeloRTV_Save_File (ByVal PortNo As Integer,
    ByVal FileName As String, ByVal FileFormat
    As Byte, ByVal nQuality As Long) As Integer
```

Delphi (Windows)

```
AngeloRTV_Copy_Frame(PortNo:Smallint;var
    Dest_Address:Byte;Size_byte:Longint):Smalli
    nt;
AngeloRTV_Get_Frame(PortNo:Smallint;var
    Start_Address:Pointer; var Width:Longint ,
    var Height:Longint ,var
    Size_byte:Longint):Smallint;
AngeloRTV_Save_File(PortNo:Smallint;FileName:Str
    ing;FileFormat:Byte;nQuality
    :LongIng):Smallint;
```

@ Argument

PortNo:

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.”

Dest_Address:
User allocated destination memory address or array.

Start_Address:
Memory start address of the video frame.

Width:
Image width.

Height:
Image height.

Size_Byte:
Memory size in bytes.

FileName:
File name to save to. Remember to add the file extension name.

FileFormat:
File format to save to.

- 0: TIF
- 1: BMP
- 2: JPEG

nQuality:
This parameter in used only for the JPEG file format.

@ Return Code

- 0: ERROR_NoError
- -7: ERROR_Not_Initialized – Make sure the port has been initialized by “AngeloRTV_Initial”.
- -9: ERROR_Invalid_PortNo – Please input a correct “PortNo” parameter.

@ Example

<VC/BCB >

AngeloRTV_Copy_Frame

```
I16 Result;
U16 PortNo = 0;
```

```
U32 Size_Byte = 640*480*3;
U8* Dest_Address =NULL;
Dest_Address = (U8*)malloc(Size_Byte );
Result = AngeloRTV_Copy_Frame (PortNo,
     Dest_Address, Size_Byte);
```

AngeloRTV_Get_Frame

```
I16 Result;
U16 PortNo = 0;
U32 Size_Byte,Width,Height ;
U32 StrAddr ;
Result = AngeloRTV_Get_Frame(PortNo,&StrAddr,
     &Width, &Height,&Size_Byte);
```

AngeloRTV_Save_File

```
I16 Result;
U16 PortNo = 0;
U8 File_Format = 2;
U32 nQuality = 25;
Result = AngeloRTV_Save_File(PortNo,"Image.jpg",
     File_Format, nQuality);
```

< Visual Basic >

AngeloRTV_Copy_Frame

```
Dim Result As Integer
Dim PortNo As Integer
Dim Size_Byte As Long
Dest_Address( ) As Byte
PortNo = 0
Size_Byte =640*480*3
ReDim Dest_Address(0 To Size_Byte - 1) As Byte
Result = AngeloRTV_Copy_Frame (PortNo,
     Dest_Address(0), Size_Byte);
```

AngeloRTV_Get_Frame

```
Dim Result As Integer
Dim PortNo As Integer
Dim Size_Byte As Long
Dim StrAddr As Long
Dim Width as Long,Height As Long
PortNo = 0
Result = AngeloRTV_Get_Frame( ByVal PortNo,
     Str_Add, Width, Height, Size_Byte)
```

AngeloRTV_Save_File

```
Dim Result As Integer
Dim File_Format as Byte
Dim nQuality as Long
```

```
PortNo = 0
File_Format = 2
NQuality = 25
Result = AngeloRTV_Save_File (PortNo,
     "Image.jpg", File_Format, NQuality)
```

<Delphi >

AngeloRTV_Copy_Frame

```
Var
Result : SmallInt;
PortNo: SmallInt;
Size_Byte :Longint;
Dest_Add : array of Byte;
PortNo := 0;
Size_Byte := 640*480*3;
SetLength(Dest_Add, Size_Byte);
Result := AngeloRTV_Copy_Frame (PortNo,
     Dest_Add[0], Size_Byte);
```

AngeloRTV_Get_Frame

```
Var
Result : SmallInt;
PortNo: SmallInt;
Size_Byte : LongIng;
Width :LongIng;
Height :LongIng;
Str_Add :Pointer;
PortNo:=0;
Result :=AngeloRTV_Get_Frame(PortNo,
     Str_Add,Width, Height, Size_Byte);
```

AngeloRTV_Save_File

```
Var
Result : SmallInt;
PortNo: SmallInt;
File_Format : Byte;
NQuality :LongInt;
PortNo:=0;
File_Format:=2;
Nquality := 25;
Result := AngeloRTV_Save_File (PortNo,
     'Image.jpg', File_Format, Nquality)
```

5.11 Angel RTV LabVIEW Function Library

AngeloRTV_Init.vi

This VI initializes the port of RTV card. Set video format and color format for the port of RTV card. Call this VI before AngeloRTV_Snap.vi.

Video Format

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

Color Format

- 0: RGB16
- 1: GRAY
- 2: RGB15
- 3: RGB24
- 4: RGB32
- 5: RGB8
- 6: RAW8X
- 7: YUY2 4:2:2
- 8: BtYUV 4:1:1

AngeloRTV_Snap.vi

Obtain an image and output the image data for picture control.

AngeloRTV_Close.vi

Release resources of all ports.

AngeloRTV_Software_Reset.vi

Reset the port to its initial state.

AngeloRTV_Hardware_Initial.vi
This VI initializes the port of RTV card. Each application program must call this function before any other function. If the initialization succeeds, it returns a value 0.

AngeloRTV_Hardware_Close.vi
Release resources of all ports.

AngeloRTV_Int_Enable.vi
This VI links the event and the interrupt automatically.

AngeloRTV_Wait_Int.vi
Wait for interrupt events. You can get a complete image data from the image buffer after this VI returns correctly.

AngeloRTV_Set_Video_Format.vi
Set the Video format for the port of RTV card. Valid color formats are: NTSC, EIA, PAL, CCIR.

AngeloRTV_Get_Video_Format.vi
Retrieve the video format of the port.

AngeloRTV_Set_Color_Format.vi
Set the color format for the port of RTV card. Valid color format are: gray scale, RGB, YUV.

Color Format:

- RGB16 = 0
- GRAY = 1
- RGB15 = 2
- RGB24 = 3
- RGB32 = 4
- RGB8 = 5

AngeloRTV_Get_Color_Format.vi
Retrieve the color format of the port

AngeloRTV_Set_Image_Config.vi
Adjust hue, contrast, saturation and brightness for the port of RTV card.

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)
- ▷ 6 for Gamma Correction Removal

	Range	Default Value
BRIGHTNESS	0 to 255	128
HUE	0 to 255	0
CHROMA (U)	0 to 255	127
CHROMA (V)	0 to 255	127
LUMA	0 to 255	112
LUMA notch filter	0(Enable) or 1(Disable)	

AngeloRTV_Get_Image_Config.vi
Retrieve current hue, contrast, saturation and brightness of the port.

AngeloRTV_Set_Image_Geometric.vi
This VI is used for image cropping and scaling.

X_Scale

This parameter is the scaling factor applied to the RTV sampled line to obtain pixels according to the resolution.

X_Active

This parameter is the length of the active video line.

X_Offset

This parameter is the number of scaled pixels to skip before the start of the active video line.

Y_Scale

This parameter is the scaling factor applied to the RTV sampled data lines in the vertical direction. It must be the following values: 1.0, 0.5, 0.25.

Y_Active

This parameter is the height (in lines) of the active video image.

Y_Offset

This parameter is the number of lines to skip before the first line of the active video image.

AngeloRTV_Select_Channel.vi

RTV card is capable of channel extension. This VI is used to multiplex video signals for ports. In most cases, this VI is not required because the default setting is one port dedicated to one channel.

Note: Do not call this VI if there is no channel extension board in the system.

AngeloRTV_Capture_Config.vi

Set the starting field of image, only for Full size image (Video format = 0 or 1)

Start_Filed

Indicate the first field of image.

- ▷ 0: First field is odd, so the image will be odd field + even field.
- ▷ 1: First field is even, so the image will be even field + odd field.
- ▷ 2: First field depends on the current field, so the image will be even field + odd field or odd field + even field.
- ▷ 3: Single field frame, used for moving object inspection.

AngeloRTV_Capture_Start.vi

Start to grab video images. If the "CaptureNumber" is 0xFFFFFFFF, the frame grabbing will not stop until the "AngeloRTV_Capture_Stop.vi" is called.

AngeloRTV_Capture_Stop.vi

Stop grabbing video images.

AngeloRTV_Trigger_Start.vi

Generate a one shot pulse output for single or multiple ports.

Multiplex

Indicate the trigger output ports on the RTV card.

- ▷ Bit 0: Port 0 on each card. 0 for disable, 1 for enable.
- ▷ Bit 1: Port 1 on each card. 0 for disable, 1 for enable.
- ▷ Bit 2: Port 2 on each card. 0 for disable, 1 for enable.
- ▷ Bit 3: Port 3 on each card. 0 for disable, 1 for enable.

AngeloRTV_Trigger_Config.vi

Configure the pulse output interval.

Interval

Indicates the trigger output interval. The valid range is from 0 to 253. The definition is as follows:

- ▷ 0: 16 ms
- ▷ 32: 12 ms
- ▷ 128: 8 ms
- ▷ 253: 60μs

AngeloRTV_Sync_Grab.vi

Use this VI to obtain an image frame. Retrieve the memory start address from the frame data, width, height, and size in bytes of the image.

AngeloRTV_Get_Frame.vi

Retrieve the memory start address from the frame data, width, height, and size in bytes of the image.

AngeloRTV_Copy_Frame.vi

Copy frame data to memory or an array that the user allocates. Before using this VI, remember to allocate enough memory space or array elements.

AngeloRTV_Set_GPIO_Sts.vi

There is one digital output channel in each port of RTV card. Use this VI to set digital output status.

1.

AngeloRTV_Get_GPIO_Sts.vi

There is one digital input channel in each port of RTV card. Use this VI to get the digital input status.

AngeloRTV_Write_EEPROM.vi

Write data into EEPROM. Data in EEPROM will not be lost when power off.

Offset

This parameter is valid between 0 and 127

Value

Value in byte. This parameter is valid between 0 and 255.

AngeloRTV_Read_EEPROM.vi

Read data from EEPROM. Data in EEPROM will not be lost when power off.

Offset:

This parameter is valid between 0 and 127

Value:

Value in byte. This parameter is valid between 0 and 255.

AngeloRTV_Read_Serial.vi

This VI can read a 48-bit unique ID and store in 2 long integers.

HighByte

HighByte stores the upper 16-bit of Serial No.

LowByte

LowByte stores the lower 32-bit of Serial No.

AngeloRTV_Save_File.vi

Save the current video frame into an image file (TIF, BMP, or JPEG). Quality is used only for JPEGs.

FileFormat

- ▷ 0: TIF
- ▷ 1: BMP
- ▷ 2: JPEG

6 Programming Guide

6.1 DirectShow Programming Guide

Introduction

A complete documentation on DirectShow application programming can be found at:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directx9_c/directX/htm/introductiontodirectshow.asp.

If a DirectX 9.0 is installed, this documentation is also available from DirectX SDK Help.

The main goal of writing a DirectShow Application is to build a filter graph by connecting several filters together to perform a given task such as previewing video/audio, capturing video/audio and multiplexing them to write into a file. Each filter performs a single operation and pass data from its output pin to the input pin of the next filter in the graph.

To build a capture graph using a program, the first thing is to obtain the interface pointer of the capture filter. The ADLink Bt878 Video Capture filter can be obtained through **system device enumerator**. After holding an interface pointer to the capture filter object, use method **IGraphBuilder::AddSourceFilter** to add the source filter object to the filter graph. Use **IFilterGraph::AddFilter** to add other downstream filters to the filter graph. After filters are added, call **IFilterGraph::ConnectDirect** or **IGraphBuilder::Connect** methods to connect output pins from upstream filters to the input pins of the downstream filters. Calling methods **IMediaControl::Run, IMediaControl::Pause** or **IMediaControl::Stop** will change filter state to running, paused or stopped.

The filters that are needed for capturing video streams are listed in next section, with detailed information for each filter and its pins. Example filter graphs for previewing/capturing video streams are also illustrated in this chapter and gives examples of two ways of controlling device driver.

Descriptions of Filters

This section lists filters needed to build a filter graph for capturing video stream and previewing video stream.

Source Filter

ADLink Bt878 Video Capture

ADLink Bt878 Video Capture Filter belongs to the category of WDM Streaming Capture Devices. It is actually a kernel-mode KsProxy plug-in. An application can treat it simply as a filter. Use System Device Enumerator to add this filter to a filter graph.

Filter Name	ADLink Bt878 Video Capture
Filter CLSID	Not applicable.
Filter Category Name	WDM Streaming Capture Devices
Filter Category	AM_KSCATEGORY_CAPTURE
Video Capture Pin Supported Media Types	MEDIATYPE_Video Subtypes: ▶ MEDIASUBTYPE_YUY2 ▶ MEDIASUBTYPE_YVU9 ▶ MEDIASUBTYPE_UYVY ▶ MEDIASUBTYPE_YV12 ▶ MEDIASUBTYPE_I420 ▶ MEDIASUBTYPE_Y41P ▶ MEDIASUBTYPE_RGB24 ▶ MEDIASUBTYPE_RBG32 ▶ MEDIASUBTYPE_RBG565 ▶ MEDIASUBTYPE_RBG555
Video Preview Pin Supported Media Types	MEDIATYPE_Video Subtypes: ▶ MEDIASUBTYPE_YUY2 ▶ MEDIASUBTYPE_YVU9 ▶ MEDIASUBTYPE_UYVY ▶ MEDIASUBTYPE_YV12 ▶ MEDIASUBTYPE_I420 ▶ MEDIASUBTYPE_Y41P ▶ MEDIASUBTYPE_RGB24 ▶ MEDIASUBTYPE_RBG32 ▶ MEDIASUBTYPE_RBG565 ▶ MEDIASUBTYPE_RBG555
Merit	MERIT_DO_NOT_USE

CrossBar Filter

If the device is a capture board, a CrossBar filter is needed for switching video source. In hardware design, crossbar can switch channel input of same port.

Filter Name	ADLink Bt878 CrossBar
Filter Category Name	WDM_Streaming Crossbar Devices

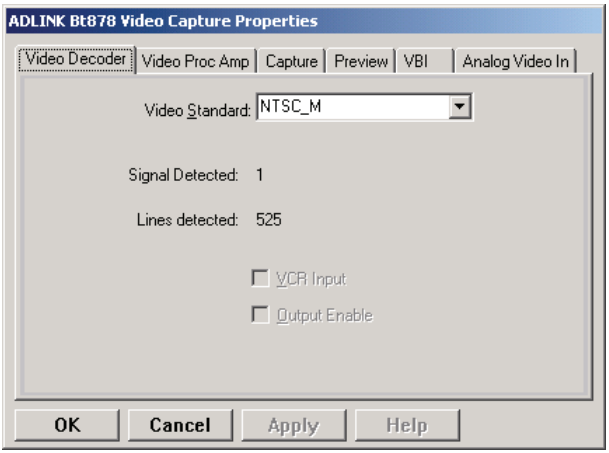
Example Graphs

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

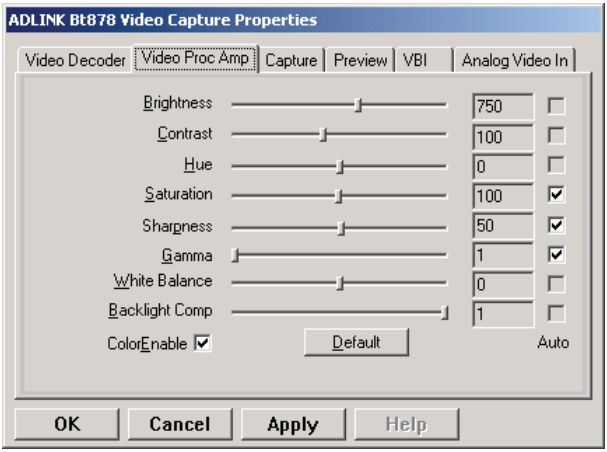
After inserting ADLink Bt878 Video Capture filter and ADLink Bt878 Crossbar filter, right click on the rectangle and click Filter Properties…. The **filter properties** dialogue will appear. Use the property pages to set video settings before connecting video pins to other filters. The property pages are shown below:

ADLink Bt878 Video Capture filter:

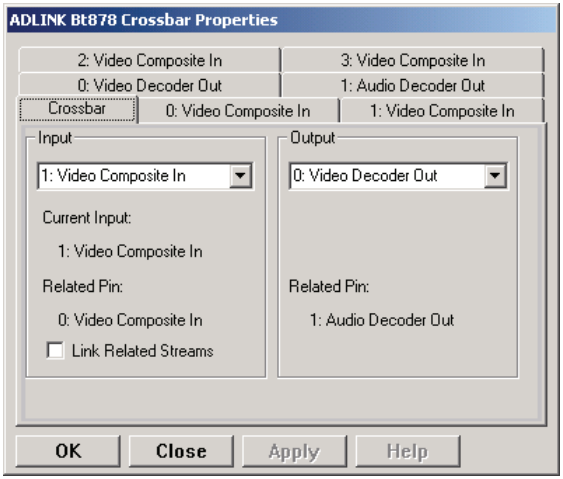
Video Decoder:



Video Proc Amp:

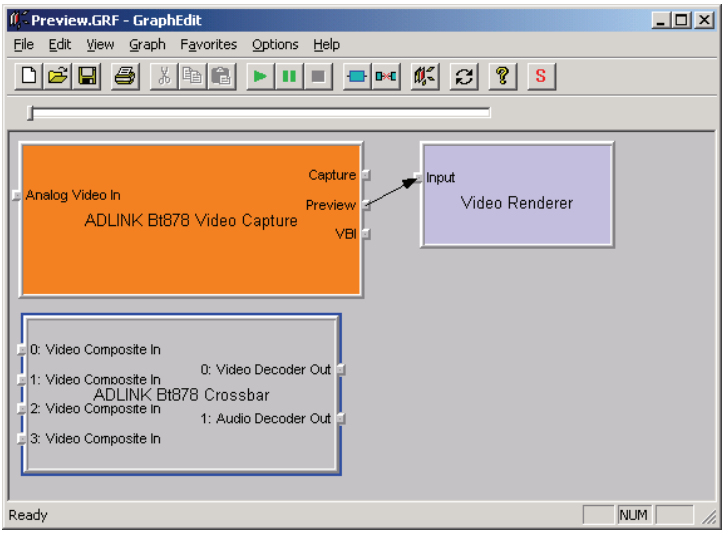


ADLink Bt878 Crossbar filter:



Select video input before or during video previewing.

Example Graph



Controlling Driver

The ADLink Bt878 Video Capture Filter provides property pages and exposes COM interfaces to control video. So an application can have two ways to control video configurations: using the property pages and using the COM interfaces.

Use Property Pages

There are two embedded property pages in the driver. To show these property pages, use Windows API: **OleCreatePropertyFrame**.

Documentation about Displaying a Filter's Property Page can be found on Microsoft MSDN homepage.

Below is the example code for adding property pages:

```
// pFilter points to the capture filter
ISpecifyPropertyPages *pSpecify;
HRESULT hr;
hr = pFilter-
     >QueryInterface(IID_ISpecifyPropertyPages,
     (void **)&pSpecify);
if (SUCCEEDED(hr))
{
     FILTER_INFO FilterInfo;
     pFilter->QueryFilterInfo(&FilterInfo);
     FilterInfo.pGraph->Release();
     CAUUID caGUID;
     pSpecify->GetPages(&caGUID);
     pSpecify->Release();
     OleCreatePropertyFrame(
          NULL,// Parent window
          0,   // x (Reserved)
          0,   // y (Reserved)
          FilterInfo.achName,// Caption for the
          dialog box
          1,   // Number of filters
          (IUnknown **)&m_pFilter,// Pointer to
          the filter
          caGUID.cElems,// Number of property
          pages
          caGUID.pElems,// Pointer to property
          page CLSIDs
          0,   // Locale identifier
          0,   // Reserved
          NULL // Reserved
          );
     CoTaskMemFree(caGUID.pElems);
}
```

Use COM interfaces

Use the methods of **IAMVideoProvAmp** interface of standard DirectShow Interface to get or set the qualities of an incoming video signal.

ADLink Bt878 Crossbar

The ADLink Bt878 Crossbar filter implements an **IAMCrossbar** interface. It routes signals from an analog or digital source to a video capture filter.

Proprietary Interface

GPIO Access

The GPIO provides a method to read board information, select input channel, and control digital inputs/digital outputs.

Sample:

#define INSTANCE_DATA_OF_PROPERTY_PTR(x)
((PKSPROPERTY((x))) + 1)

#define INSTANCE_DATA_OF_PROPERTY_SIZE(x)
(sizeof((x)) - sizeof(KSPROPERTY))

```
void GPIOWrite(IBaseFilter* pFilter, DWORD value)
/*
Purpose:
      Set the electronic level of the gpio pin.
Parameters:
      pFilter: Interface of BT878 filter
      value: 1 for high level, and 0 for low level
*/
{
      IKsPropertySet *pKs = NULL;
      DWORD TypeSupport = 0;
      KSPROPERTY_CUSTOMBT848_GPIO_S rc;
      HRESULT hr;
      ULONG ret=0;
      DWORD bit = 6;// Offset of GPIO pin
      if (pFilter == NULL)
            return;
value = value ? 0 : 1; // its phase is inverse
```

```
        if (pFilter-
        >QueryInterface(IID_IKsPropertySet, (void
        **)&pKs) == S_OK)
        {
              hr = pKs-
        >QuerySupported(PROPSETID_CUSTOMBT848,
KSPROPERTY_CUSTOMBT848_GPIO,
&TypeSupport);
        if(TypeSupport & KSPROPERTY_SUPPORT_GET)
              {
              ZeroMemory(&rc,sizeof(rc));

        rc.dwOperation=BT848_CUSTPROP_GPIO_SETGPDAT
        ABITS;
              rc.dwFromBit = bit;
              rc.dwToBit = bit;
              rc.dwValue = value;
              rc.dwOffset =0;
              hr = pKs->Get(
                    PROPSETID_CUSTOMBT848,
                    KSPROPERTY_CUSTOMBT848_GPIO,
              INSTANCE_DATA_OF_PROPERTY_PTR(&rc),
              INSTANCE_DATA_OF_PROPERTY_SIZE(rc),
              &rc,
              sizeof(rc),
              &ret);
        }
pKs->Release();
        }
}
DWORD GPIORead(IBaseFilter* pFilter)
/*
Purpose:
        Get the electronic level of the gpio pin.
Parameters:
        pFilter: Interface of BT878 filter
*/
{
        IKsPropertySet *pKs = NULL;
        DWORD TypeSupport = 0;
        KSPROPERTY_CUSTOMBT848_GPIO_S rc;
        HRESULT hr;
        ULONG ret=0;
```

```
      DWORD ReturnValue=0;
DWORD bit = 6;// Offset of GPIO pin
      if (pFilter == NULL)
            return 0;
      if (pFilter-
      >QueryInterface(IID_IKsPropertySet, (void
      **)&pKs) == S_OK)
      {
            hr = pKs-
      >QuerySupported(PROPSETID_CUSTOMBT848,
KSPROPERTY_CUSTOMBT848_GPIO,
&TypeSupport);
            if(TypeSupport &
      KSPROPERTY_SUPPORT_GET)
            {
                  ZeroMemory(&rc,sizeof(rc));
                  rc.dwOperation =
      BT848_CUSTPROP_GPIO_GETGPDATABITS;
            rc.dwFromBit = bit;
            rc.dwToBit = bit;
            rc.dwOffset =0;
            hr = pKs->Get(
                  PROPSETID_CUSTOMBT848,
                  SPROPERTY_CUSTOMBT848_GPIO,
            INSTANCE_DATA_OF_PROPERTY_PTR(&rc),
            INSTANCE_DATA_OF_PROPERTY_SIZE(rc),
            &rc,
            sizeof(rc),
            &ret);
      ReturnValue = rc.dwValue;
      }
pKs->Release();
}
return ReturnValue;
}
```

EEPROM Access

ADLink Bt878 Video Capture provides a method for accessing I2C register. The interface can store a few data, for example, board identification.

Sample:

```
#define INSTANCE_DATA_OF_PROPERTY_PTR(x)
     ( (PKSPROPERTY((x)) ) + 1 )
#define INSTANCE_DATA_OF_PROPERTY_SIZE(x)
     ( sizeof((x)) - sizeof(KSPROPERTY) )
BYTE EEPROMRead(IBaseFilter *pFilter, BYTE
     offset)
/*
Purpose:
     Read.the value stored in EEPROM
Parameters:
     pFilter: Interface of BT878 filter
     offset: the offset (0~127) based on starting
     address of EEPROM
*/
{
     IKsPropertySet *pKs = NULL;
     DWORD TypeSupport = 0;
     KSPROPERTY_CUSTOMBT848_I2C_S I2C;
     BYTE uAddress;
     HRESULT hr;
     ULONG ret=0;

     if(pFilter == NULL)
          return 0;

     if((hr=pFilter-
     >QueryInterface(IID_IKsPropertySet, (void
     **)&pKs)) == S_OK)
     {
          hr = pKs-
     >QuerySupported(PROPSETID_CUSTOMBT848,

     KSPROPERTY_CUSTOMBT848_I2C,

     &TypeSupport);
          if(TypeSupport &
     KSPROPERTY_SUPPORT_GET)
```

```
        {
                uAddress = 0xa0; // address for
    the EEPROM device
                // Set frequency first
                ZeroMemory(&I2C,sizeof(I2C));
                I2C.bDontWaitACK = true;
                I2C.dwOperation =
    BT848_CUSTPROP_I2C_SETFREQ;
                I2C.dwFreq = 100000;
                hr = pKs->Get(
                      PROPSETID_CUSTOMBT848,
                      KSPROPERTY_CUSTOMBT848_I2C,

    INSTANCE_DATA_OF_PROPERTY_PTR(&I2C),

    INSTANCE_DATA_OF_PROPERTY_SIZE(I2C),
                      &I2C,
                      sizeof(I2C),
                      &ret);
                // Read value then

    I2C.dwOperation=BT848_CUSTPROP_I2C_R3;
                I2C.ucAddress= uAddress;
                I2C.ucInBuf[0] = offset;
                I2C.dwOutLen = 0;
                I2C.dwInLen  =  1;
                I2C.bDontWaitACK = TRUE;
                hr = pKs->Get(
                      PROPSETID_CUSTOMBT848,
                      KSPROPERTY_CUSTOMBT848_I2C,

    INSTANCE_DATA_OF_PROPERTY_PTR(&I2C),

    INSTANCE_DATA_OF_PROPERTY_SIZE(I2C),
                      &I2C,
                      sizeof(I2C),
                      &ret);
        }
        pKs->Release();
    }
    return I2C.ucInBuf[1];
}
```

```
void EEPROMWrite(IBaseFilter *pFilter, BYTE
     offset, BYTE value)
/*
Purpose:
     Write.the value to EEPROM
Parameters:
     pFilter: Interface of BT878 filter
     offset: the offset (0~127) based on starting
     address of EEPROM
     value: the data to EEPROM
*/
{
     IKsPropertySet *pKs = NULL;
     DWORD TypeSupport = 0;
     KSPROPERTY_CUSTOMBT848_I2C_S I2C;
     BYTE uAddress;
     HRESULT hr;
     ULONG ret=0;

     if(pFilter == NULL)
          return;

     if((hr=pFilter-
     >QueryInterface(IID_IKsPropertySet, (void
     **)&pKs)) == S_OK)
     {
          hr = pKs-
     >QuerySupported(PROPSETID_CUSTOMBT848,

     KSPROPERTY_CUSTOMBT848_I2C,

     &TypeSupport);
          if(TypeSupport &
     KSPROPERTY_SUPPORT_GET)
          {
               uAddress = 0xa0;// address for
               the EEPROM device
               // Set frequency first
               ZeroMemory(&I2C,sizeof(I2C));
               I2C.bDontWaitACK = true;
               I2C.dwOperation =
               BT848_CUSTPROP_I2C_SETFREQ;
               I2C.dwFreq = 100000;
```

```
                hr = pKs->Get(
                      PROPSETID_CUSTOMBT848,
                      KSPROPERTY_CUSTOMBT848_I2C,

        INSTANCE_DATA_OF_PROPERTY_PTR(&I2C),

        INSTANCE_DATA_OF_PROPERTY_SIZE(I2C),
                      &I2C,
                      sizeof(I2C),
                      &ret);
                // Write value then

        I2C.dwOperation=BT848_CUSTPROP_I2C_WR;
                I2C.ucAddress= uAddress;
                I2C.ucOutBuf[0] = offset;
                I2C.ucOutBuf[1] = value;
                I2C.dwOutLen = 2;
                I2C.dwInLen  =  0;
                I2C.bDontWaitACK = TRUE;
                hr = pKs->Get(
                      PROPSETID_CUSTOMBT848,
                      KSPROPERTY_CUSTOMBT848_I2C,

        INSTANCE_DATA_OF_PROPERTY_PTR(&I2C),

        INSTANCE_DATA_OF_PROPERTY_SIZE(I2C),
                      &I2C,
                      sizeof(I2C),
                      &ret);
           }
           pKs->Release();
        }
}
```

Build Environment Settings

Include Files

All applications need include the file shown in the following table.

Include File	Description
DShow.h	The header file is required for all C++ applications.
Custprop.h	The header file is required for all C++ applications.
Bt848guid.h	The header file is required for all C++ applications which need access BT878 proprietary interfaces, for instance, EEPROM and GPIO.
Bt878.cs	The class definition is required for all C# applications.

Library File

All applications need the library file shown in the following table.

Library File	Description
Strmiids.lib	Exports class identifiers (CLSIDs) and interface identifiers (IIDs). All C++ applications require this library.
Quartz.lib	Exports the AMGetErrorText function. If you do not call this function, this library is not required.
DirectShowLib-2005.dll	The class library is required for all Microsoft .Net applications.

Microsoft Visual C++ Users

VC++ users need to setup the builder environment prior to start to build your program. There are few steps you need to follow as below:

1. Open the solution file (baseclasses.sln) or the project file (baseclasses.dsw) under %DXSDK%\Samples\C++\DirectShow\BaseClasses and build it.

In above, %DXSDK% is the path of DirectX SDK.

2. Add the paths to the include directory in the settings of your project:

%DXSDK%\include

%DXSDK%\Samples\C++\DirectShow\BaseClasses

3. Add the paths to the additional library directory in the settings of your project:

%DXSDK%\Lib

%DXSDK%\Samples\C++\DirectShow\BaseClasses\Release

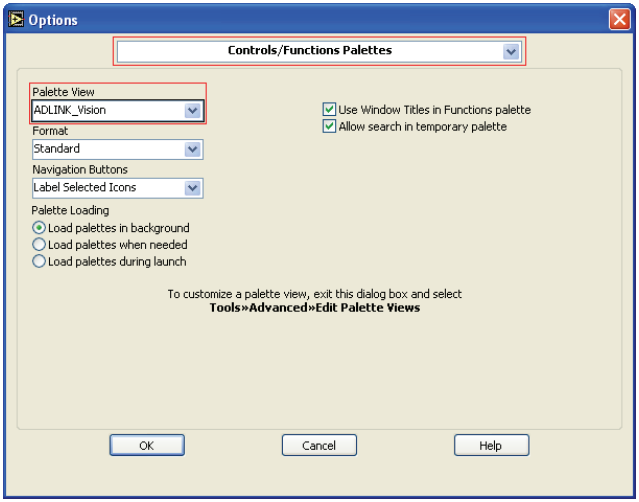
.Net Programming Users

Microsoft DirectShow only provides C++ programming. As for .net users, they need convert DirectShow COM objects to .net class. Fortunately, the work had been done as a sourceforge project. Download the source codes and samples from http://sourceforge.net/projects/directshownet/. It is a good start to program your DirectShow codes by .net languages. We also provided samples dedicated to RTV cards in the installation directory.

6.2 LabVIEW Programming Guide

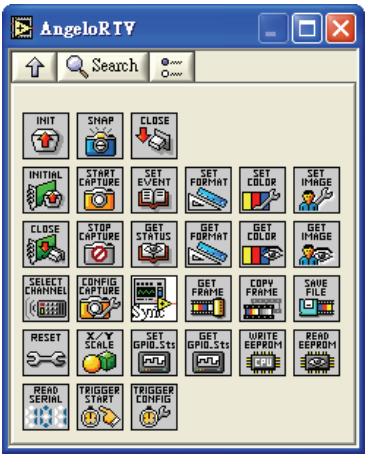
ADLINK_Vision Controls/Functions Palettes

To use RTV-LVIEW VIs, you have to switch the **Controls/Functions palettes** to the **ADLINK_Vision** palette view first. In LabVIEW 7.0, select **Tools>>Options** to display the **Options** dialog box. Select **Controls/Functions Palettes** from the top pull-down menu in the **Options** dialog box, and select **ADLINK_Vision** from the **Palette View** pull-down menu.

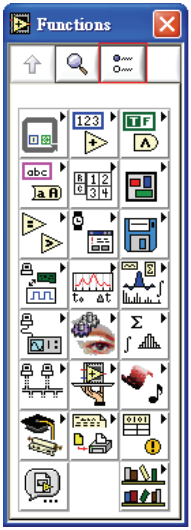


Click OK button. Then **ADLINK Vision** icon is shown in the **Functions** palette.

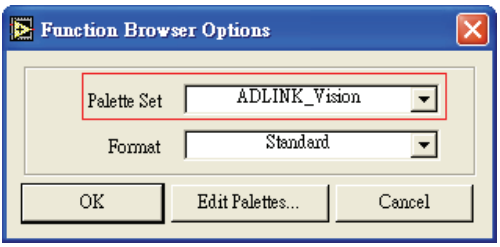
Click **ADLINK Vision** icon to display the **ADLINK_Vision** palette view. Then click the **AngeloRTV** icon, you can find RTV-LVIEW VIs.



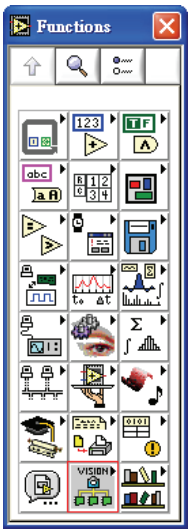
In LabVIEW 6, click the **Options** button on the **Functions** palette toolbar to display the **Function Browser Options** dialog box.



Select **ADLINK_Vision** from the **Palette Set** pull-down menu and click OK button.



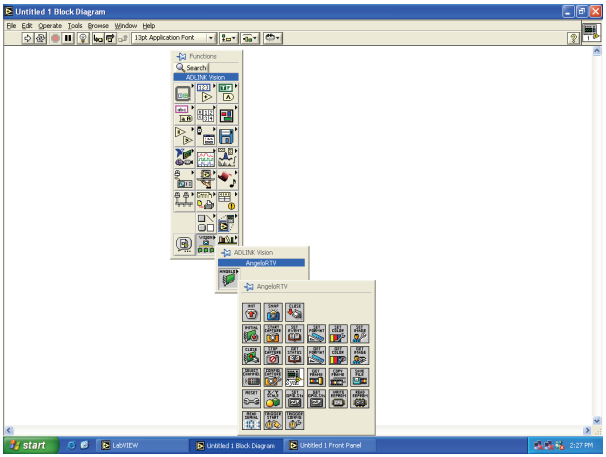
Then you can see the **ADLINK_Vision** Functions Palette as below.

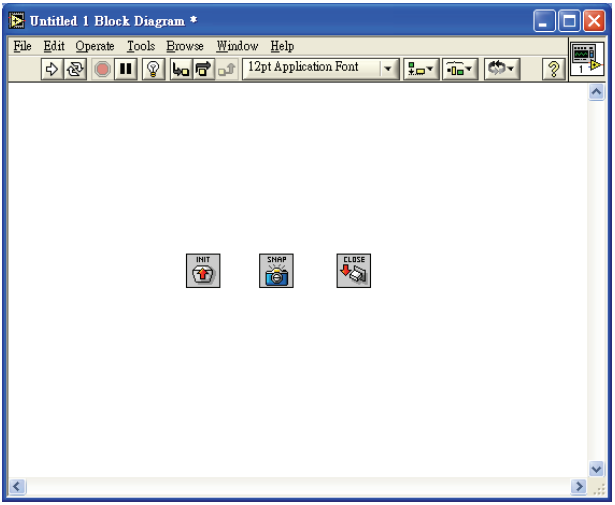


How-to Program with RTV-LVIEW

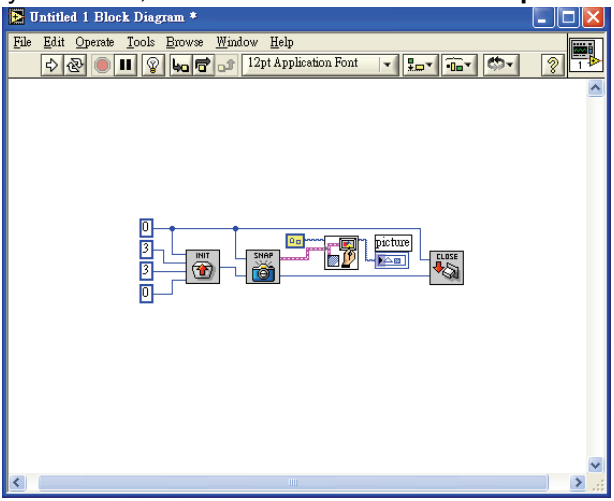
Here we provide a simplest sample showing how to capture a frame with RTV-LVIEW VI. For more complicated samples with RTV-LVIEW, you can reference those located in the **C:\Program Files\ADLINK\RTV-LVIEW\Samples** folder.

1. Open a blank VI and switch to the block diagram. Drag and drop **AngeloRTV_Init.vi**, **AngeloRTV_Snap.vi**, and **AngeloRTV_Close.vi** on the block diagram.

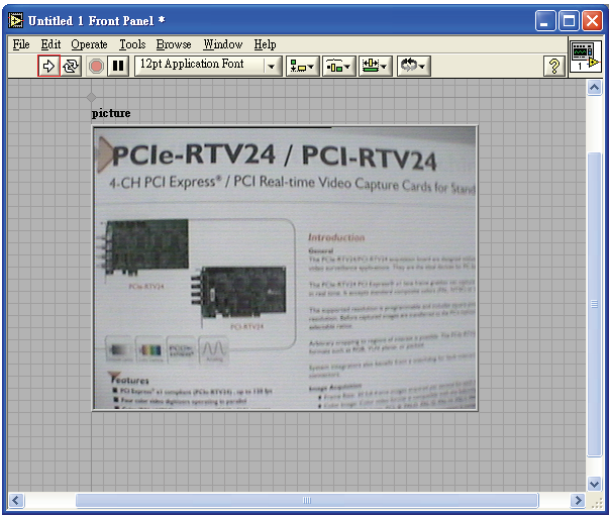




2. Create **Constant** or **Control** to each input and connect these VIs. In order to show the captured frame on the front panel, we also drag and drop another VI provided by LabVIEW, named **Draw Flattened Pixmap.vi**.

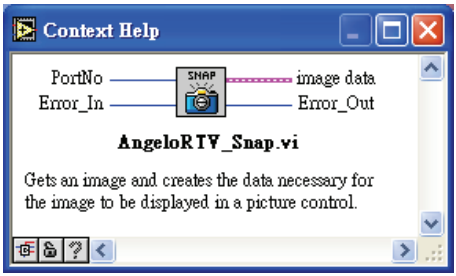


3. Push the upper left **Run** button and you can see a captured frame on the front panel.



Get Help of RTV-LVIEW

You can display the **Context Help** window by selecting **Help>>Show Context Help**. LabVIEW will show the information of the RTV-LVIEW VI when you move the cursor over it.



6.3 Linux Programming Guide

Introduction

Video4 Linux or V4L is intended to provide a standard video capture application programming interface on Linux. V4L is in its second version. V4L2 driver include a compatibility mode for V4L1 application that is V4L application can mix the two modes of V4L1 and V4L2.

A complete documentation on V4L application programming can be found at:

http:// www.linuxtv.org/downloads/video4linux/API/V4L2_API/

The document gives a very detailed description of all APIs. Familiar with it will great help you in writing your video capturing application.

A simple sample

In this chapter, we provide a simple sample as how to program RTV cards.

Open device

The first step is to open a RTV device with open (). The first parameter in it is device name which can be listed under directory /dev with a prefix name ‘video’ and a number appending to it. There will be same number of such files as how many devices your system has.

```
static char dev_name[] = "/dev/video0";// Open
   the first device
int open_device (void)
{
      int fd;
      fd = open (dev_name, O_RDWR | O_NONBLOCK,
         0);
      if (-1 == fd) {
            fprintf (stderr, "Cannot open '%s':
               %d, %s\n", dev_name, errno,
               strerror (errno));
            return -1;// Failed
         }
      return fd;// Success
}
```

Close device

Close the device with close () if you no longer use this device.

```
Close ( fd);
```

IO control

IO control is a technology communication with driver. V4L sets up many standard IO controls which control video parameters to or get information from driver. Here we give you an example of simple settings.

```
void init_device (void)
{
      struct v4l2_capability cap;
      struct v4l2_cropcap cropcap;
      struct v4l2_crop crop;
      struct v4l2_format fmt
      v4l2_std_id  std  = V4L2_STD_NTSC_M;

      if (-1 == ioctl (fd, VIDIOC_QUERYCAP, &cap))
      {
            if (EINVAL == errno) {
                  fprintf (stderr, "%s is no V4L2
                     device\n", dev_name);
                  exit (EXIT_FAILURE);
            } else {
                  exit (EXIT_FAILURE);
            }
      }
      if (!(cap.capabilities &
         V4L2_CAP_VIDEO_CAPTURE)) {
            fprintf (stderr, "%s is no video
               capture device\n", dev_name);
            exit (EXIT_FAILURE);
      }
      if (!(cap.capabilities &
         V4L2_CAP_STREAMING)) {
            fprintf (stderr, "%s does not support
               streaming i/o\n", dev_name);
            exit (EXIT_FAILURE);
      }
      /* Select video input, video standard and
      tune here. */
      if (-1 == ioctl (fd, VIDIOC_S_STD, &std)) {
```

```
            exit (EXIT_FAILURE);
        }
        /* Change to the default channel */
        int channel = 0;
        if (-1 == ioctl (fd, VIDIOC_S_INPUT,
           &channel)) {
            exit (EXIT_FAILURE);
        }
        memset (&cropcap, 0, sizeof (cropcap));
        cropcap.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;

        if (0 == ioctl (fd, VIDIOC_CROPCAP,
        &cropcap)) {
            crop.type =
           V4L2_BUF_TYPE_VIDEO_CAPTURE;
            crop.c = cropcap.defrect; /* reset to
               default */
            if (-1 == ioctl (fd, VIDIOC_S_CROP,
               &crop)) {
                  switch (errno) {
                  case EINVAL:
                        /* Cropping not supported */
                        break;
                  default:
                        /* Errors ignored. */
                        break;
                  }
            }
        } else {
            /* Errors ignored. */
        }

        memset (&fmt, 0, sizeof (fmt));
        fmt.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
        fmt.fmt.pix.width = 640;
        fmt.fmt.pix.height = 240;
        fmt.fmt.pix.pixelformat =
           V4L2_PIX_FMT_BGR24;
        fmt.fmt.pix.field = V4L2_FIELD_ALTERNATE;//
           per field (odd and even)
        if (-1 == ioctl (fd, VIDIOC_S_FMT, &fmt))
            exit (EXIT_FAILURE);
}
```

Memory map

Memory map system call, mmap (), allows the mapping of device memory directly into a user processor's address space. From device viewpoint, Direct Memory Access (DMA) operations provide peripherals with direct access to system memory without CPU processing. This can save large of time and loading that application or driver doesn't need to move data from devices to system memory. Here we give and example showing how to set 4 buffer queues which store video data in turn.

```
/* global variables */
struct buffer {
      void *start;
      size_t length;
};
struct buffer *buffers = NULL;
static unsigned int n_buffers = 0;

void init_mmap(void)
{
      struct v4l2_requestbuffers req;
      memset (&req, 0, sizeof (reg));
      req.count = 4;
      req.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
      req.memory = V4L2_MEMORY_MMAP;
      if (-1 == ioctl (fd, VIDIOC_REQBUFS, &req))
      {
            if (EINVAL == errno) {
                  fprintf (stderr, "%s does not
                     support " "memory mapping\n",
                     dev_name);
                  exit (EXIT_FAILURE);
            } else {
                  exit (EXIT_FAILURE);
            }
      }
      if (req.count < 2) {
            fprintf (stderr, "Insufficient buffer
              memory on %s\n", dev_name);
                 exit (EXIT_FAILURE);
      }
      buffers = calloc (req.count, sizeof
         (*buffers));
```

```
        if (!buffers) {
              fprintf (stderr, "Out of memory\n");
              exit (EXIT_FAILURE);
        }

        for (n_buffers = 0; n_buffers < req.count;
           ++n_buffers) {
              struct v4l2_buffer buf;
              memset (&buf, o, sizeof (buf));
              buf.type =
                 V4L2_BUF_TYPE_VIDEO_CAPTURE;
              buf.memory = V4L2_MEMORY_MMAP;
              buf.index = n_buffers;
              if (-1 == ioctl (fd, VIDIOC_QUERYBUF,
                 &buf))
                    exit (EXIT_FAILURE);
              buffers[n_buffers].length =
                 buf.length;
              buffers[n_buffers].start =
                    mmap (NULL /* start anywhere */,
                          buf.length,
                          PROT_READ | PROT_WRITE /*
                             required */,
                          MAP_SHARED /* recommended */
                             ,
                          fd, buf.m.offset);
              if (MAP_FAILED ==
                 buffers[n_buffers].start)
                    exit (EXIT_FAILURE);
        }
  }
```

Start capturing

```
void start_capturing (void)
{
      unsigned int i;
      enum v4l2_buf_type type;
      for (i = 0; i < n_buffers; ++i) {
            struct v4l2_buffer buf;
            memset (&buf, 0, sizeof (buf));
            buf.type  =
               V4L2_BUF_TYPE_VIDEO_CAPTURE;
            buf.memory = V4L2_MEMORY_MMAP;
            buf.index = i;
            if (-1 == ioctl (fd, VIDIOC_QBUF,
               &buf))
                  exit (EXIT_FAILURE);
      }
      type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
      if (-1 == ioctl (fd, VIDIOC_STREAMON,
         &type))
            exit (EXIT_FAILURE);
}
```

Stop capturing

```
Void stop_capturing (void)
{
      enum v4l2_buf_type type;
      type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
      if (-1 == ioctl (fd, VIDIOC_STREAMOFF,
         &type));
}
```

Read frame

Read frame image when an image was ready and prepare next frame.

```
Int read_frame (void)
{
      struct v4l2_buffer buf;
      memset (&buf, 0, sizeof (buf));
      buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
      buf.memory = V4L2_MEMORY_MMAP;

      /* read frame */
      if (-1 == ioctl (fd, VIDIOC_DQBUF, &buf)) {
            switch (errno) {
            case EAGAIN:
                  return 0;
            case EIO:
                  /* Could ignore EIO, see spec. */
                  /* fall through */
            default:
                  exit (EXIT_FAILURE);
            }
      }
      /* prepare next frame */
      if (-1 == ioctl (fd, VIDIOC_QBUF, &buf))
            exit (EXIT_FAILURE);
      return 0;
}
```

Proprietary properties

Except standard APIs, we also provide a proprietary IO control which can read and write external general purpose IO pin.

```
/* configure the direction (in or out) of each
     gpio bit prior to reading or writing gpio.
     */
int config_gpio (void)
{
     unsigned int value = 0xC3FEFF;
     if (-1 == ioctl (fd,
        BT878_S_GPIO_OUT_ENABLE, &value))
           return -1;
     return 0;
}
int read_gpio (void)
{
     unsigned int value;
     if (-1 == ioctl (fd, BT878_G_GPIO_VALUE,
        &value))
           return -1;
     value &= 0x100;// bit 8 is used to store the
        input value
     if(value)
           return 1;
     else
           return 0;
}
int write_gpio (int value)
{
     unsigned int gpio;
     if (-1 == ioctl (fd, BT878_G_GPIO_VALUE,
        &gpio))
           return -1;
     gpio |= 0x40;// bit 6 is used to set the
        output
     if(value)
           gpio -= 0x40;
     if (-1 == ioctl (fd, BT878_S_GPIO_VALUE,
        &gpio))
           return -1;
     return 0;
}
```

7 Appendix

7.1 Glossary

Brightness:

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

CCIR:

An acronym to designate a scanning system used in Europe. The CCIR system is made of two interlaced fields of 312.5 lines, for a total of 625 lines. In each field, only 287.5 lines are visible, for a total of 575 visible lines. A line lasts 64 ms, of which approximately 52 ms are conveying visible pixels.

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:

An acronym to designate a scanning system used in America and Japan. The EIA system is made of two interlaced fields of 262.5 lines, for a total of 525 lines. In each field, only 242.5 lines are visible, for a total of 485 visible lines (typical value). A line lasts 63.56 ms, of which approximately 52 ms are conveying visible pixels.

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 very 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:

Acronym to designate a color television broadcast standard used in America and Japan. The (M) NTSC system uses 525 lines per frame (2 interlaced fields), a 29.97 frame per second update rate, and a YIQ or RGB color space. In each field, only 242.5 lines are visible, for a total of 485 visible lines (typical value). A line lasts 63.56 ms, of which approximately 52 ms are conveying visible pixels.

PAL:

Acronym to designate a color television broadcast standard used in Europe. The (B, G, H, I) PAL (or Phase Alternation Line) uses 625 lines per frame (2 interlaced fields), a 25 frame per second update rate, and the RGB color space. In each field, only 287.5 lines are visible, for a total of 575 visible lines. A line lasts 64 ms, of which approximately 52 ms are conveying visible pixels.

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.

7.2 Standards 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.