

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Visit our website - *Click HERE*

HDV62

Full HD 1080p Frame Grabbers

User's Manual

Manual Rev.	2.00
Revision Date:	January 18, 2010
Part No:	50-11164-1000

Recycled Paper

Advance Technologies; Automate the World.

Copyright 2010 ADLINK TECHNOLOGY INC.

All Rights Reserved.

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.

This document contains proprietary information protected by copyright. All rights are reserved. No part of this manual may be reproduced by any mechanical, electronic, or other means in any form without prior written permission of the manufacturer.

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

Contact us should you require any service or assistance.

ADLINK Technology Inc.
Address: 9F, No.166 Jian Yi Road, Chungho City,
Taipei County 235, Taiwan
台北縣中和市建一路 166 號 9 樓
Tel: +886-2-8226-5877
Fax: +886-2-8226-5717
Email: service@adlinktech.com

Ampro ADLINK Technology Inc.
Address: 5215 Hellyer Avenue, #110, San Jose, CA 95138, USA
Tel: +1-408-360-0200
Toll Free: +1-800-966-5200 (USA only)
Fax: +1-408-360-0222
Email: info@adlinktech.com

ADLINK Technology Beijing
Address: 北京市海淀区上地东路 1 号盈创动力大厦 E 座 801 室
(100085)
Rm. 801, Power Creative E, No. 1, B/D
Shang Di East Rd., Beijing 100085, China
Tel: +86-10-5885-8666
Fax: +86-10-5885-8625
Email: market@adlinktech.com

ADLINK Technology Shanghai
Address: 上海市漕河泾高科技开发区钦江路 333 号 39 幢 4 层
(200233)
Tel: +86-21-6495-5210
Fax: +86-21-5450-0414
Email: market@adlinktech.com

ADLINK Technology Shenzhen
Address: 深圳市南山区科技园南区高新南七道 数字技术园
A1 栋 2 楼 C 区 (518057)
2F, C Block, Bld. A1, Cyber-Tech Zone,
Gao Xin Ave. Sec 7, High-Tech Industrial Park S.,
Shenzhen, 518054 China
Tel: +86-755-2643-4858
Fax: +86-755-2664-6353
Email: market@adlinktech.com

ADLINK Technology Inc. (German Liaison Office)
Address: Nord Carree 3, 40477 Duesseldorf, Germany
Tel: +49-211-495-5552
Fax: +49-211-495-5557
Email: emea@adlinktech.com

ADLINK (French Liaison Office)
Address: 15 rue Emile Baudot, 91300 MASSY Cedex, France
Tel: +33 (0) 1 60 12 35 66
Fax: +33 (0) 1 60 12 35 66
Email: france@adlinktech.com

ADLINK Technology Japan Corporation
Address: 151-0072 東京都渋谷区幡ヶ谷
1-1-2 朝日生命幡ヶ谷ビル 8F
Asahiseimei Hatagaya Bldg. 8F
1-1-2 Hatagaya, Shibuya-ku, Tokyo 151-0072, Japan
Tel: +81-3-4455-3722
Fax: +81-3-5333-6040
Email: japan@adlinktech.com

ADLINK Technology Inc. (Korean Liaison Office)
Address: 서울시 서초구 서초동 1506-25 한도 B/D 2 층
2F, Hando B/D, 1506-25, Seocho-Dong,
Seocho-Gu,Seoul, 137-070, Korea
Tel: +82-2-2057-0565
Fax: +82-2-2057-0563
Email: korea@adlinktech.com

ADLINK Technology Singapore Pte Ltd.
Address: 84 Genting Lane #07-02A, Cityneon Design Centre,
Singapore 349584
Tel: +65-6844-2261
Fax: +65-6844-2263
Email: singapore@adlinktech.com

ADLINK Technology Singapore Pte Ltd. (Indian Liaison Office)
Address: No. 1357, "Anupama", Sri Aurobindo Marg, 9th Cross,
JP Nagar Phase I, Bangalore - 560078, India
Tel: +91-80-65605817
Fax: +91-80-22443548
Email: india@adlinktech.com

Table of Contents

List of Tables ..... **iv**

List of Figures ..... **v**

1 Introduction **1**
1.1 Features 1
1.2 Applications 2
1.3 System Requirements 2

2 Hardware Reference **3**
2.1 HDV62 3
Hardware Specifications 3
HDV62 Outline and Mechanical Dimensions 5
Trigger I/O Timing Diagram 6
HDV62 Connectors and Pin Definitions 7

3 Installation Guide **13**
3.1 Windows Driver Installation 13

4 Function Library **21**
4.1 Function List 22
4.2 Setting up the Build Environment 26
4.3 Device Control Functions 27
Device Count 27
Device Open 28
Device Close 29
Device Vendor Name 30
Device Model Name 31
Device Version 32
Device Firmware Version 33
Driver Version 34
Library Version 35
Device ID 36
Device Reset 37
4.4 Image Format Functions 38
Channel 38
Sensor Format 39
Sensor Width 42

Sensor Height 43
Width 44
Height 45
X Offset 46
Y Offset 47
Output Format 48
HDelay 52
Contrast 53
Hue 54
Saturation 55
4.5 Event and Callback Functions 56
Event Selector 56
Event Handle 57
CallbackSelector 59
Callback 60
4.6 Acquisition Control Functions 61
Acquisition Frame Count 61
Acquisition Start 63
Acquisition Stop 64
One Shot 65
Image Stream 66
Acquisition Status 67
Acquisition Statistics 68
Sensor Status 69
Save Image 70
4.7 Trigger Control Functions 71
Trigger In Source 71
Trigger In Polarity 72
Software Trigger 73
One Pulse Out 74
Trigger Out Polarity 75
Trigger Out Pulse Width 76
4.8 Digital I/O Functions..... 77
4.8.1 Digital IO Selector 77
DI 78
DI Event 79
DO 80
4.9 Other Functions 81
Error Text 81
4.10 EDID Functions..... 82

Ready Status 82
Access Permission 84
Write Protection 86
Rom Selector 87
Rom Read/Write 88

5 Programming Guide 89
5.1 Introduction 89
5.2 Descriptions of Filters 90
Source Filter 90
Example Graphs 92
5.3 Controlling Driver 99
Use Property Pages 99
Use COM interfaces 101
Color Space 103
5.4 Proprietary Interfaces 106
IVideoFormat 106
IAdvance 111
ICardInfo 124
INotify 127
5.5 Build Environment Settings 128

6 ViewCreatorPro Utility 131
6.1 Overview 131
6.2 Component Description 132
Devices panel 133
Adjustment Panel 134
Tool panel 135
Status Panel 139
Display panel 139
Main Menu 141

List of Tables

Table 2-1: Hardware Specifications 3
Table 2-2: DVI-I Connector Pin Definition (CN1) 7
Table 2-3: D-sub Connector Pin Definition (CN2) 8
Table 2-4: Box Header Pin Definition (CN3) 9
Table 2-5: Trigger Output Mode Select (SW1) 10
Table 2-6: Digital Output Mode Select (SW2) 10
Table 2-7: Card ID DIP Switch (SW3) 11
Table 2-8: FPGA Golden Flash Selection (SW4) 12
Table 4-1: ADLINK HDV62 API Function List 25

List of Figures

Figure 2-1: HDV62 Outline and Mechanical Dimensions..... 5
Figure 2-2: Trigger I/O Timing Diagram 6
Figure 4-1: Signals of Frame Image 42
Figure 4-2: EDID ROM Architecture 83

1 Introduction

ADLINK's HDV62 is a PCI Express® x4 lane frame grabber that acquires image and video streams from both HD (High Definition) and SD (Standard Definition) video up to a 170 MHz DVI input, in addition to RGB and YPbPr component analog inputs.

ADLINK's HDV62 delivers the superior quality of high definition video to medical imaging, scientific imaging, military, and high-end video surveillance applications. Equipped with an FPGA (Field Programmable Gate Array) and 512 MB memory buffer, the HDV62 provides the ability to stream image of a specified area to the host PC, and real-time hardware color space conversion to Offload repetitive tasks from the host CPU.

The HDV62 provides the ViewCreator Pro® utility to setup, configure, test, and debug the system without any software programming. And, ADLINK's WDM driver is compatible with Microsoft® Directshow to reduce engineering effort and accelerate the time to market.

1.1 Features

- Resolutions up to 1920 x 1080p, 60 fps
- PCI Express® x4 interface
- Supports standard definition and high-definition video input
- Hardware color space conversion, supports RGB, YUV, and monochrome pixel output formats
- 512 MB of DDR2 SDRAM frame buffer
- Supports graphical overlays and stream imaging of a specified area
- Configurable EDID
- 4 TTL digital inputs/outputs and 1 trigger output Video Input

1.2 Applications

- Medical Imaging
- Scientific Imaging
- Broadcast
- Military & Defense
- Video Surveillance

1.3 System Requirements

The minimum system requirements for 1-CH real-time Full HD image acquisition are:

- Platform: Pentium 4, 2.4 GHz CPU, 512 MB RAM or above.
- Display settings: 1920 x 1080 resolution or above, 16-bit color or above.

2 Hardware Reference

2.1 HDV62

2.1.1 Hardware Specifications

Item	Specifications	
Acquisition Interface	Digital	Standard TMDS , DVI 1.0, HDMI 1.3 (non-HDCP and non-audio)
	Analog	▶ R,G,B ▶ Y,Pb,Pr
Video Input Resolutions	SD (YPbPr@D-Sub 9 Connector)	▶ 720x480i@30fps ▶ 720x480p@60fps ▶ 720x576i@25fps ▶ 720x576p@50fps
	HD (Digital@DVI-I Connector)	▶ 720p@50fps ▶ 720p@60fps ▶ 1080i@25fps ▶ 1080i@30fps ▶ 1080p@25fps ▶ 1080p@30fps ▶ 1080p@50fps ▶ 1080p@60fps
	HD (YPbPr@D-Sub 9 Connector)	▶ 720p@50fps ▶ 720p@60fps ▶ 1080i@25fps ▶ 1080i@30fps ▶ 1080p@50fps ▶ 1080p@60fps
	Computer (Digital@DVI-I Connector)	▶ VGA (640x480@60) ▶ SVGA (800x600@60) ▶ XGA (1024x768@60) ▶ SXGA (1280x1024@60) ▶ UXGA (1600x1200@60) **UXGA mode 30bit 4:4:4 YCrCb/RGB max 30fps
	Computer (Analog@DVI-I Connector)	▶ VGA(640x480@60) ▶ SVGA(800x600@60) ▶ XGA(1024x768@60) ▶ SXGA(1280x1024@60)
ADC Resolution	▶ 10-bit digitization	

Table 2-1: Hardware Specifications

Item	Specifications		
Pixel Output Formats	▶ RGB: 32/30/24 bit ▶ YCrCb: 444/422 ▶ Monochrome:10/8 bit		
ADC Sampling Rate	▶ 170 MHz		
Frame Buffer	▶ DDR2 / 512MB		
Video IO Connector	▶ D-SUB9 female : YPbPr ▶ DVI-I: DVI-D , RGB with sync input		
Digital In/Out	▶ 4 TTL input, 4 TTL output , input can generate interrupt (change of state), Manual enable/disable this function		
Trigger In/Out	▶ 1 TTL Trigger input and 1 TTL trigger output for external hardware trigger capture mode. ▶ Programmable trigger output plus wide and trigger output delay time(See Trigger I/O timing diagram)		
DIO/Trigger IO Level Threshold	Input	▶ Maximum of input voltage:5V ▶ Minimum threshold of high level:2.4V ▶ Maximum threshold of Low level:1V	
	Output (Select by SW1 and SW2)	Open collector	▶ Maximum of IC:200mA ▶ Maximum of VCEO:40V
		TTL output	▶ Maximum of source current:20mA ▶ Output voltage of high level:5V ▶ Output voltage of low level:0V
Configurable EDID	▶ EDID 1.3		
FPGA Golden Boot Flash ROM	▶ Dip switch selection (see FPGA Golden flash selection) Use golden flash ROM to initiate card in failure mode.		
+12 V Power Output Current	▶ Maximum 1 A		
Power Consumption	▶ Maximum 0.5 mA@ +12 V ▶ Maximum 2 A@ +3.3 V		
Operating Temperature	▶ 0 - 55°C		

Table 2-1: Hardware Specifications

2.1.2 HDV62 Outline and Mechanical Dimensions

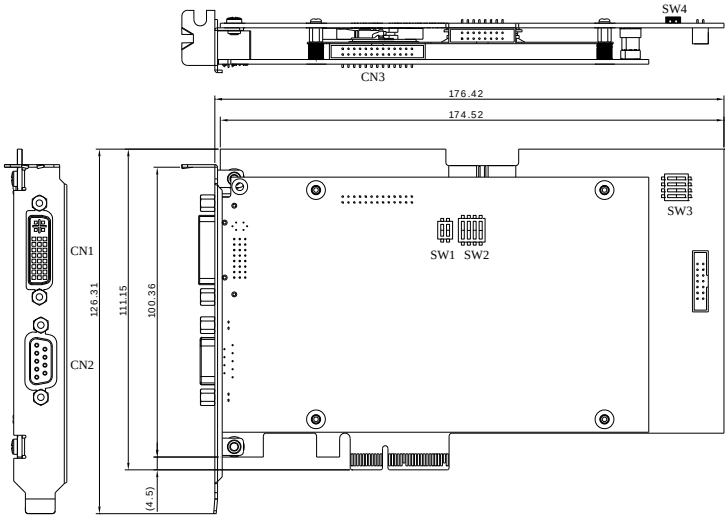
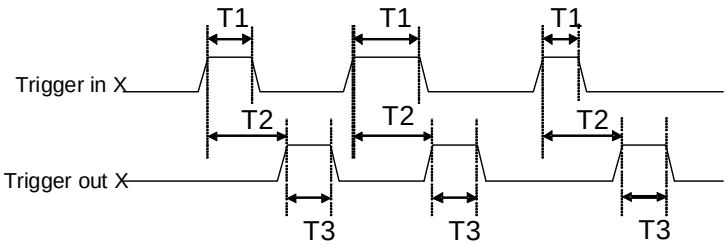


Figure 2-1: HDV62 Outline and Mechanical Dimensions

2.1.3 Trigger I/O Timing Diagram

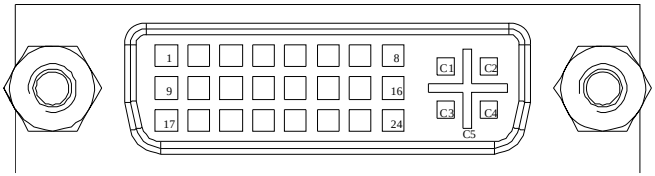


Symbol	Characteristic	Specification
T1	Trigger input pulse width	Minimumwidth is 0.1msec
T2	Trigger delay	0- 1000msec selectable(1 msec/step)
T3	Output trigger pulse width	0.1- 50msec selectable(0.1 msec/step)

Figure 2-2: Trigger I/O Timing Diagram

2.1.4 HDV62 Connectors and Pin Definitions

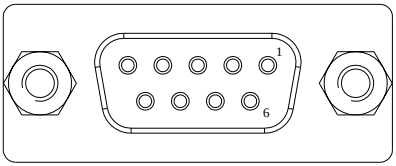
DVI-I Connector Pin Definition (CN1)



Pin	Signal	Pin	Signal
1	T.M.D.S data2-	16	Hot plug detect
2	T.M.D.S data2+	17	T.M.D.S data0-
3	T.M.D.S data2 shield	18	T.M.D.S data0+
4	NC	19	T.M.D.S data0 shield
5	NC	20	NC
6	DDC clock	21	NC
7	DDC data	22	T.M.D.S clock shield
8	Analog Vertical sync	23	T.M.D.S clock+
9	T.M.D.S data1-	24	T.M.D.S clock-
10	T.M.D.S data1+		
11	T.M.D.S data1 shield	C1	Analog Red
12	NC	C2	Analog Green
13	NC	C3	Analog Blue
14	+5V Power	C4	Analog Horizontal Sync
15	Ground (return for +5V,Hsync and Vsync)	C5	Analog Ground (analog R,G and B return)

Table 2-2: DVI-I Connector Pin Definition (CN1)

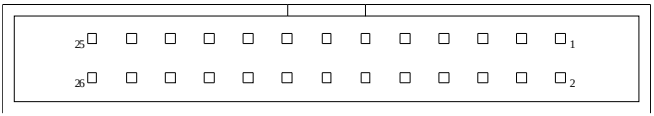
D-sub Connector Pin Definition (CN2)



Pin	Signal	Pin	Signal
1	Analog Ground	6	Analog Ground
2	YPbPr-Pr	7	Analog Ground
3	YPbPr-Pb	8	Analog Ground
4	YPbPr-Y	9	Analog Ground
5	Analog Ground		

Table 2-3: D-sub Connector Pin Definition (CN2)

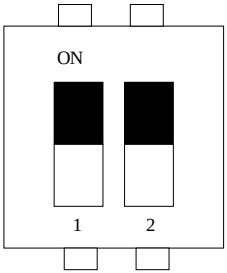
Box Header Pin Definition (CN3)



Pin	Signal	Pin	Signal
1	+12 Power out	14	Digital Output 2
2	+12 Power out	15	Digital Ground
3	+12 Power out	16	Digital Ground
4	+12 Power out	17	Digital Input3
5	Digital Ground	18	Digital Output 3
6	Digital Ground	19	Digital Input4
7	Digital Ground	20	Digital Output 4
8	Digital Ground	21	Digital Ground
9	Digital Ground	22	Digital Ground
10	Digital Ground	23	Trigger Input1
11	Digital Input1	24	Trigger output1
12	Digital Output 1	25	Digital Ground
13	Digital Input2	26	Digital Ground

Table 2-4: Box Header Pin Definition (CN3)

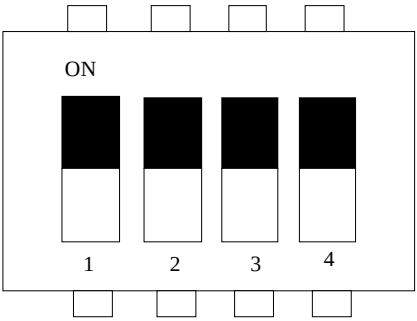
Trigger Output Mode Select (SW1)



Trigger output channel	Switch is On.	Switch is Off.
1	TTL output	Open collector
2	TTL output	Open collector

Table 2-5: Trigger Output Mode Select (SW1)

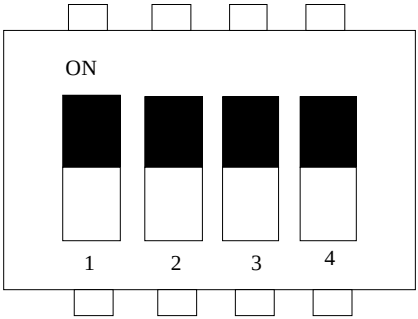
Digital Output Mode Select (SW2)



Digital output channel	Switch is On.	Switch is Off.
1	TTL output	Open collector
2	TTL output	Open collector
3	TTL output	Open collector
4	TTL output	Open collector

Table 2-6: Digital Output Mode Select (SW2)

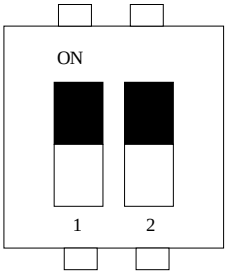
Card ID DIP Switch (SW3)



Card Identification	Setting(4321) On=>0/OFF=>1	Card Identification	Setting (4321)
0	0000	8	1000
1	0001	9	1001
2	0010	10	1010
3	0011	11	1011
4	0100	12	1100
5	0101	13	1101
6	0110	14	1110
7	0111	15	1111

Table 2-7: Card ID DIP Switch (SW3)

FPGA Golden Flash Selection (SW4)



Boot flash selection of Host FPGA	Setting of Switch(21)
Working flash	OFF,OFF
Golden flash	ON,ON

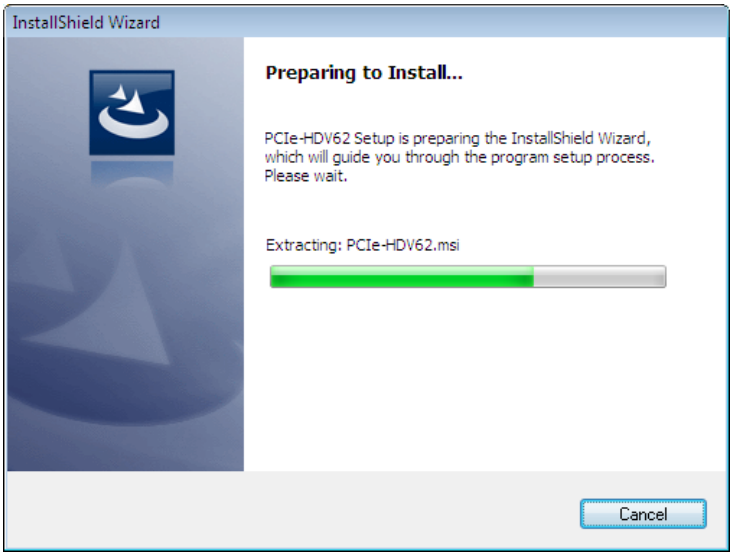
Table 2-8: FPGA Golden Flash Selection (SW4)

3 Installation Guide

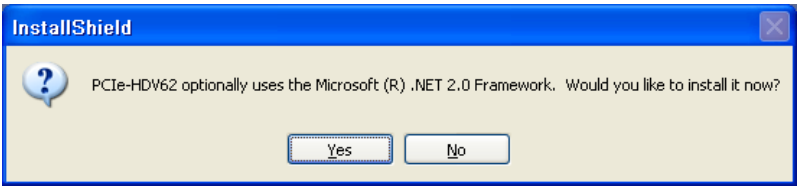
3.1 Windows Driver Installation

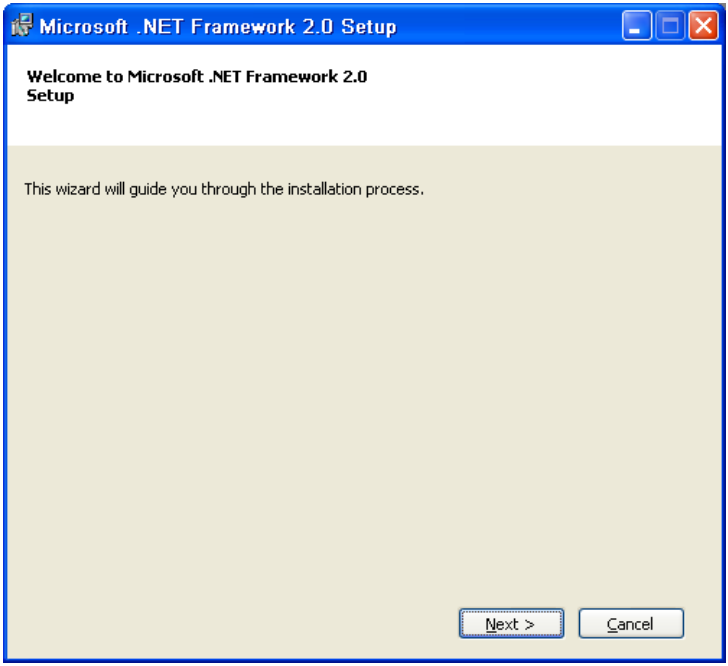
The driver installation procedure for a Windows Vista operation system is described below. Installation on other Windows systems will be similar to these steps.

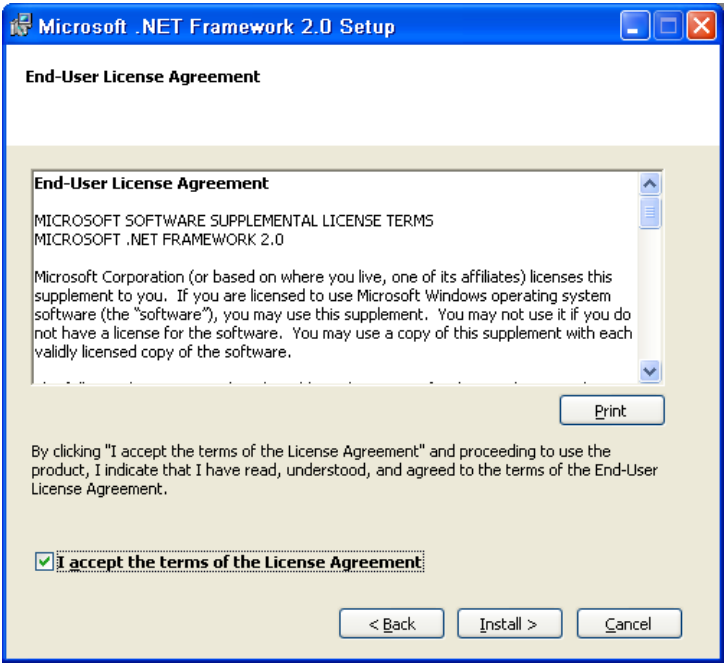
1. Run setup program.
2. The installation will begin



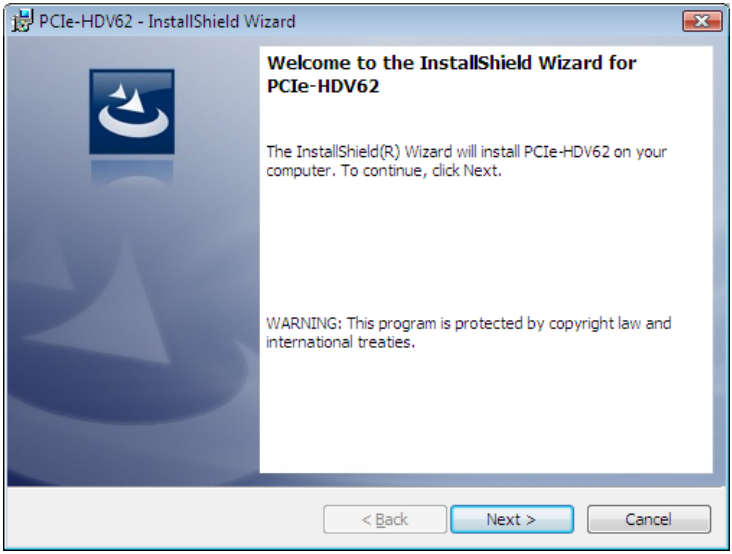
3. If .Net Framework 2.0 is not currently installed on the system, the following window will be displayed. Click Yes to install .Net Framework.

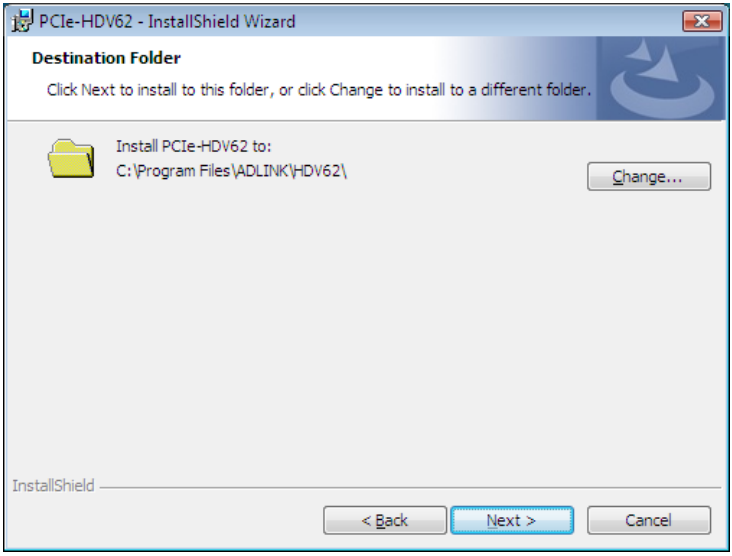


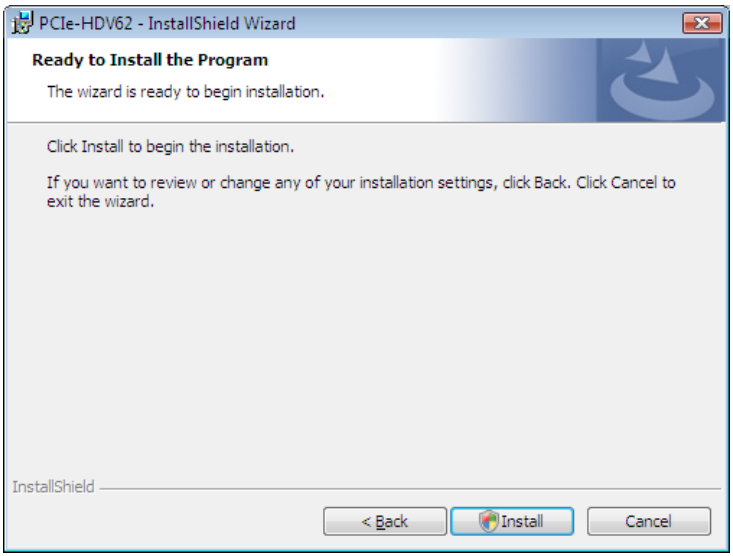

Microsoft .NET Framework 2.0 Setup
Welcome to Microsoft .NET Framework 2.0 Setup
This wizard will guide you through the installation process.
Next >
Cancel


Microsoft .NET Framework 2.0 Setup
End-User License Agreement
End-User License Agreement
MICROSOFT SOFTWARE SUPPLEMENTAL LICENSE TERMS
MICROSOFT .NET FRAMEWORK 2.0
Microsoft Corporation (or based on where you live, one of its affiliates) licenses this supplement to you. If you are licensed to use Microsoft Windows operating system software (the "software"), you may use this supplement. You may not use it if you do not have a license for the software. You may use a copy of this supplement with each validly licensed copy of the software.
Print
By clicking "I accept the terms of the License Agreement" and proceeding to use the product, I indicate that I have read, understood, and agreed to the terms of the End-User License Agreement.
I accept the terms of the License Agreement
< Back
Install >
Cancel

4. Click next until driver installation is complete

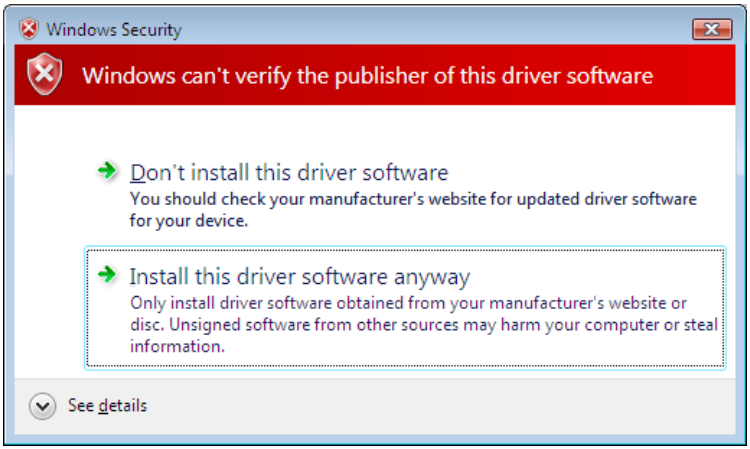




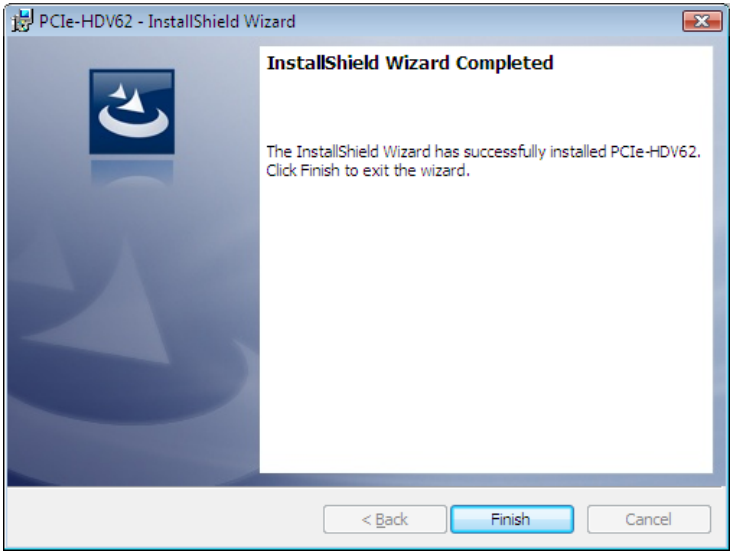


5. When the following window is displayed, please press “Install this driver anyway” to install the device drivers.

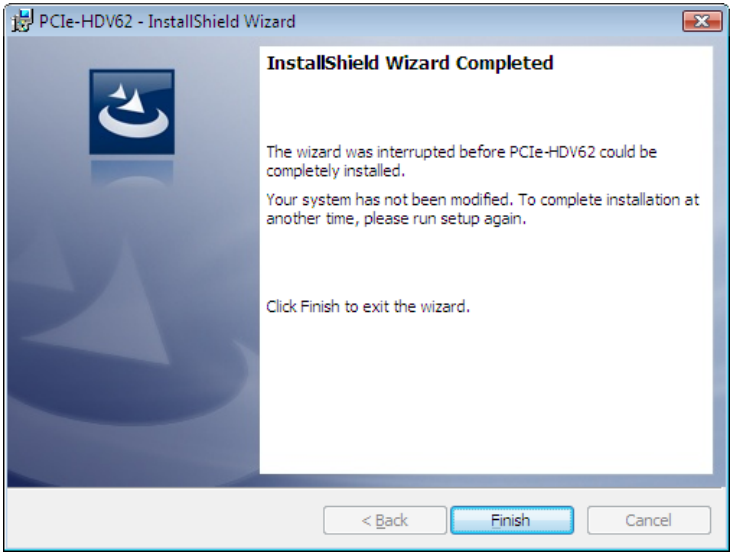
Note: If a “Found New Hardware Wizard” window appears, simply ignore it. After the installation completes, the “Found New Hardware Wizard” window will automatically close.



6. Finally the installation completes. Click Finish.



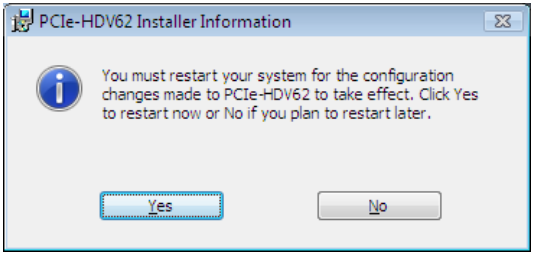
If an error iccurred, the installation will be rolled back.



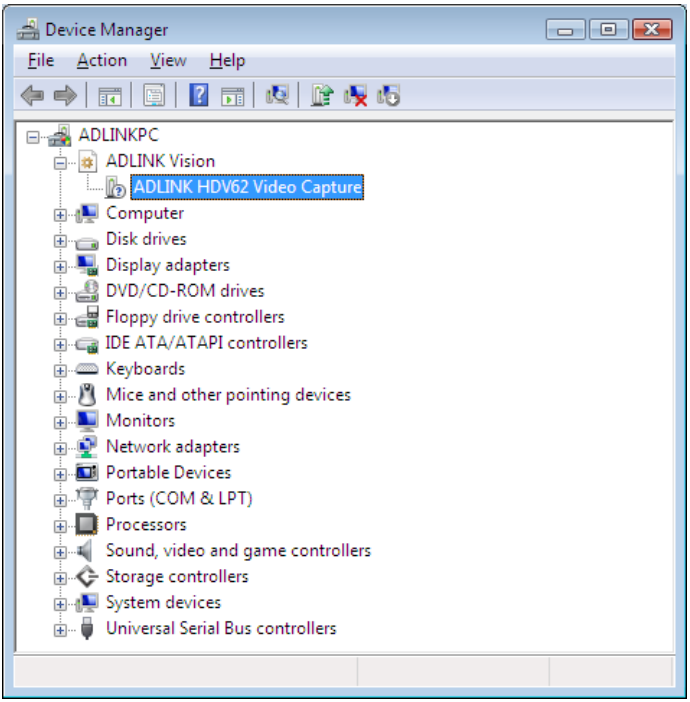
If you see this dialog, please email the file setupapi.log in the Windows folder to ADLINK's technical support.

Note: The log files on Vista system are moved to %windir%\inf and renamed to setupapi.app.log and setupapi.dev.log, where windir is Windows folder.

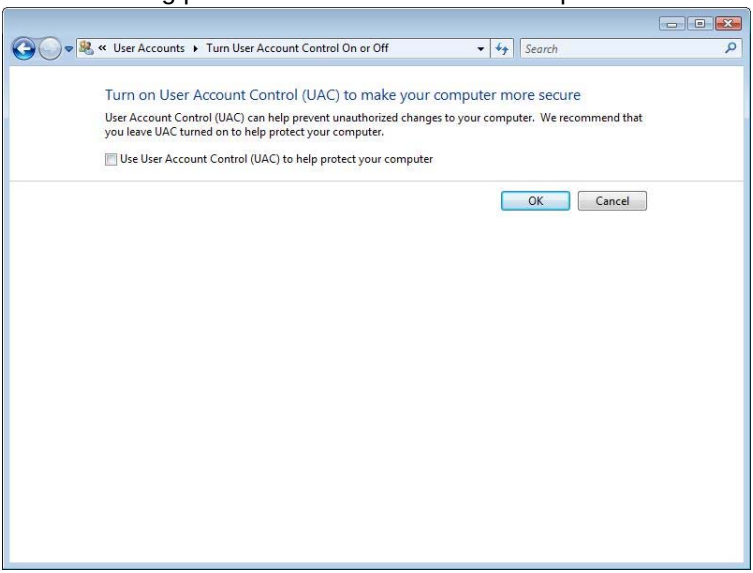
7. Click Yes and restart system.



8. Open Device Manager from System of Control Panel.



Note: For Windows Vista users, there is a way to avoid security error while operating the ViewCreatorPro utility. Turn off the User Account Control (UAC) by opening [Start] -> [Settings] -> [Control Panel] -> [User Accounts] -> [Turn User Account Control on or off]. Uncheck the UAC as illustrated in the following picture and then restart the computer.



4 Function Library

This chapter describes the ADLINK HDV62 API for the HDV62 frame grabber. Users can use these functions to develop your application programs under Visual C++, C#, Visual Basic.Net, Delphi, and Borland C++ Builder.

The ADLINK HDV62 API is a kind of simple C like function. It is based on DirectShow technologies, but we eliminated the complexity of DirectShow programming and transformed it to simple API functions that users don't need to familiar with DirectShow programming.

Besides the ADLINK API, users also have another choice that is using DirectShow technologies to program your application. DirectShow is a standard interface of audio/video streaming what is provided by Microsoft. It provides many powerful COM objects which are totally different than the ADLINK HDV62 API. Next chapter we will introduce the DirectShow programming guide.

Do not mix the ADLINK HDV62 API and DirectShow programming to access same HDV62 card at the same time. We can't make sure that way can work correctly.

The particular functions associated with each function are presented in following table.

4.1 Function List

The table below lists all of ADLINK HDV62 API functions. For the detail, please reference to the following sections.

Category	Function Name	Section
Device Control	Hdv62_GetDeviceCount	4.3
	Hdv62_DeviceOpen	
	Hdv62_DeviceClose	
	Hdv62_GetDeviceVendorName	
	Hdv62_GetDeviceModelName	
	Hdv62_GetDeviceVersion	
	Hdv62_GetDeviceFirmwareVersion	
	Hdv62_GetDriverVersion	
	Hdv62_GetLibraryVersion	
	Hdv62_GetDeviceID	
	Hdv62_DeviceReset	

Category	Function Name	Section
Image Format Control	Hdv62_SetChannel	4.4
	Hdv62_GetChannel	
	Hdv62_SetSensorFormat	
	Hdv62_GetSensorFormat	
	Hdv62_GetSensorWidth	
	Hdv62_GetSensorHeight	
	Hdv62_SetWidth	
	Hdv62_GetWidth	
	Hdv62_SetHeight	
	Hdv62_GetHeight	
	Hdv62_SetXOffset	
	Hdv62_GetXOffset	
	Hdv62_SetYOffset	
	Hdv62_GetYOffset	
	Hdv62_SetOutputFormat	
	Hdv62_GetOutputFormat	
	Hdv62_SetHDelay	
	Hdv62_GetHDelay	
	Hdv62_SetContrast	
	Hdv62_GetContrast	
	Hdv62_SetHue	
	Hdv62_GetHue	
	Hdv62_SetSaturation	
	Hdv62_GetSaturation	
Event & Callback	Hdv62_SetEventSelector	4.5
	Hdv62_GetEventSelector	
	Hdv62_SetEventHandle	
	Hdv62_GetEventHandle	
	Hdv62_SetCallbackSelector	
	Hdv62_GetCallbackSelector	
	Hdv62_SetCallback	

Category	Function Name	Section
Acquisition Control	Hdv62_SetAcquisitionFrameCount	4.6
	Hdv62_GetAcquisitionFrameCount	
	Hdv62_AcquisitionStart	
	Hdv62_AcquisitionStop	
	Hdv62_OneShot	
	Hdv62_GetImageStream	
	Hdv62_GetAcquisitionStatus	
	Hdv62_GetAcquisitionStatistics	
	Hdv62_GetSensorStatus	
	Hdv62_SaveImage	
Trigger Control	Hdv62_SetTriggerInSource	4.7
	Hdv62_GetTriggerInSource	
	Hdv62_SetTriggerInPolarity	
	Hdv62_GetTriggerInPolarity	
	Hdv62_SoftwareTrigger	
	Hdv62_SendOnePulseOut	
	Hdv62_SetTriggerOutPolarity	
	Hdv62_GetTriggerOutPolarity	
	Hdv62_SetTriggerOutPulseWidth	
	Hdv62_GetTriggerOutPulseWidth	
Digital I/O	Hdv62_SetDigitalIOSelector	4.8
	Hdv62_GetDigitalIOSelector	
	Hdv62_SetDI	
	Hdv62_SetDIEvent	
	Hdv62_GetDIEvent	
	Hdv62_SetDO	
	Hdv62_GetDO	
Others	Hdv62_GetErrorText	4.9

Category	Function Name	Section
EDID	Hdv62_SetEdidReadyStatus	4.10
	Hdv62_GetEdidReadyStatus	
	Hdv62_SetEdidAccessPermission	
	Hdv62_GetEdidAccessPermission	
	Hdv62_SetEdidWriteProtection	
	Hdv62_GetEdidWriteProtection	
	Hdv62_SetEdidRomSelector	
	Hdv62_GetEdidRomSelector	
	Hdv62_SetEdidRom	
	Hdv62_GetEdidRom	

Table 4-1: ADLINK HDV62 API Function List

4.2 Setting up the Build Environment

Include Files

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

Include File	Description
Hdv62.h	The header file required for all C/C++ applications.
Hdv62.vb	The function definitions required for all VB.Net applications.
Hdv62.cs	The function definitions required for all C# applications.

Library File

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

Library File	Description
Hdv62.lib	Exports API function definitions. Required for all Visual C/C++ applications.
Hdv62_bcb.lib	Exports API function definitions. Required for all Borland C++ Builder applications.

DLL Files

All applications using the APIs need the DLL file shown in the following table.

Library File	Description
Hdv62.dll	Dynamic link library. Required for all applications.

All files are located on the directory [Installed directory]\ADLINK\hdv62\Include. ‘Installed directory’ is the destination directory where you specified in setup program.

4.3 Device Control Functions

4.3.1 Device Count

Purpose

This function returns the total number of HDV62 devices in your system. The library can only detect maximum 16 devices.

Prototype

- C/C++
  ```
  int Hdv62_GetDeviceCount(UINT &Count)
  ```
- C#
  ```
  int GetDeviceCount(out uint Count)
  ```
- VB.Net
  ```
  GetDeviceCount (ByRef Count as UInteger) As Integer
  ```

Parameters

- Count
  the total number of devices installed on computer.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.3.2 Device Open

Purpose

This function initializes the device referred to by number. You should call this function before you call other functions except some functions which have no Number parameter.

Prototype

- C/C++
  ```
  int Hdv62_DeviceOpen (UINT Number)
  ```
- C#
  ```
  int DeviceOpen (uint Number)
  ```
- VB.Net
  ```
  DeviceOpen (ByVal Number As UInteger) As Integer
  ```

Parameters

- Number
  The number of which device you want to open. The allowed value is from 0 to 15.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.3.3 Device Close

Purpose

This function closes the device and release all allocated resources. Application should call this function before you terminate your application.

Prototype

- C/C++
  ```
  int Hdv62_DeviceClose (UINT Number)
  ```
- C#
  ```
  int DeviceClose (uint Number)
  ```
- VB.Net
  ```
  DeviceClose (ByVal Number As UInteger) As Integer
  ```

Parameters

- Number
  The number of which device you want to close. The allowed value is from 0 to 15.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.3.4 Device Vendor Name

Purpose

This function gets or returns the vendor name.

Prototype

- C/C++
  ```
  int Hdv62_GetDeviceVendorName (char *Name)
  ```
- C#
  ```
  string GetDeviceVendorName ()
  ```
- VB.Net
  ```
  GetDeviceVendorName () As String
  ```

Parameters

- Name

  Pointer to a user-allocated buffer into which the function copies the vendor name string, for example, “ADLINK”. The name is NULL-terminated.

Return Value

- C/C++
  No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.
- C#
  Return vendor name.
- VB.Net
  Return vendor name.

4.3.5 Device Model Name

Purpose

This function gets or returns the model name.

Prototype

- C/C++
```
int Hdv62_GetDeviceModelName (UINT Number, char
   *Name)
```
- C#
```
string GetDeviceModelName (uint Number)
```
- VB.Net
```
GetDeviceModelName (ByVal Number as UInteger) As
   String
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Name
  Pointer to a user-allocated buffer into which the function copies the model name string, for example, “HDV62”. The name is NULL-terminated.

Return Value

- C/C++
  No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.
- C#
  Return model name.
- VB.Net
  Return model name.

4.3.6 Device Version

Purpose

This function gets or returns the version of hardware device.

Prototype

- C/C++
```
int Hdv62_GetDeviceVersion (UINT Number, char
   *Version)
```
- C#
```
string GetDeviceVersion (uint Number)
```
- VB.Net
```
GetDeviceVersion (ByVal Number as UInteger) As
   String
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Version
  Pointer to a user-allocated buffer into which the function copies the version string. The version is NULL-terminated.

  There will be two types:

  One is “A2/A1” for carrier board plus daughter board type. The former is carrier board version; the later is daughter board version.

  Another one is “A2” for single board type.

Return Value

- C/C++
  No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.
- C#
  Return version string.
- VB.Net
  Return version string.

4.3.7 Device Firmware Version

Purpose

This function gets or returns the version of firmware.

Prototype

- C/C++
```
int Hdv62_GetDeviceFirmwareVersion (UINT Number,
   char *Version)
```
- C#
```
string GetDeviceFirmwareVersion (uint Number)
```
- VB.Net
```
GetDeviceFirmwareVersion (ByVal Number as UInteger)
   As String
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Version
  Pointer to a user-allocated buffer into which the function copies the version string. Its format is “Year/Month/Day Hour:Minute”, for example, “2009/11/19 14:22”. The version is NULL-terminated.

Return Value

- C/C++
  No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.
- C#
  Return version string.
- VB.Net
  Return version string.

4.3.8 Driver Version

Purpose

This function gets or returns the version of driver.

Prototype

- C/C++
```
int Hdv62_GetDriverVersion (UINT Number, char
   *Version)
```
- C#
```
string GetDriverVersion (uint Number)
```
- VB.Net
```
GetDriverVersion (ByVal Number as UInteger) As
   String
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Version
  Pointer to a user-allocated buffer into which the function copies the version string, for example, “1.0.0.0”. The version is NULL-terminated.

Return Value

- C/C++
  No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.
- C#
  Return version string.
- VB.Net
  Return version string.

4.3.9 Library Version

Purpose

This function gets or returns the version of library.

Prototype

- C/C++
```
int Hdv62_GetLibraryVersion (UINT Number, char
   *Version)
```
- C#
```
string GetLibraryVersion (uint Number)
```
- VB.Net
```
GetLibraryVersion (ByVal Number as UInteger) As
   String
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Version
  Pointer to a user-allocated buffer into which the function copies the version string, for example, “1.0.0.0”. The version is NULL-terminated.

Return Value

- C/C++
  No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.
- C#
  Return version string.
- VB.Net
  Return version string.

4.3.10 Device ID

Purpose

This function gets card ID of device.

Prototype

- C/C++
  ```
  int Hdv62_GetDeviceID (UINT Number, UINT& ID)
  ```
- C#
  ```
  int GetDeviceID (uint Number, out uint ID)
  ```
- VB.Net
  ```
  GetDeviceID (ByVal Number as UInteger, ByRef ID as
     UInteger) As Integer
  ```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- ID
  Card ID can be set by DIP Switch on card. Its possible value is from 0 to 15. Card ID can distinguish cards when multi-cards were installed on one system. Set them to different number according to chapter 2 Hardware Reference. Leave card ID as default, all are same is no problem, if you don't care about it.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.3.11 Device Reset

Purpose

This function restores HDV62 device to initial state of booting up. After resetting, you need to reconfigure all settings and then the device can work correctly. Call this function only when the device behaves abnormal and can't restore it to proper state. The effect of this function is same as rebooting computer, but save lot of time of rebooting.

Prototype

- C/C++
  ```
  int Hdv62_DeviceReset (UINT Number)
  ```
- C#
  ```
  int DeviceReset (uint Number)
  ```
- VB.Net
  ```
  DeviceReset (ByVal Number as UInteger) As Integer
  ```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.4 Image Format Functions

4.4.1 Channel

Purpose

These functions set or get channel of device. HDV62 supports multi-source inputs that only one of them is available at same time. Select the channel you prefer. Refer to the Hardware Reference chapter to know the connections of source input.

Prototype

- C/C++

```
int Hdv62_SetChannel (UINT Number, UINT Channel)
int Hdv62_GetChannel (UINT Number, UINT &Channel)
```

- C#

```
int SetChannel (uint Number, uint Channel)
int GetChannel (uint Number, out uint Channel)
```

- VB.Net

```
SetChannel (ByVal Number as UInteger, ByVal Channel
   as UInteger) As Integer
GetChannel (ByVal Number as UInteger, ByRef Channel
   as UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Channel
  The source of input. The allowed value is from 0 to 5, defined as follows:
  - 0: Analog RGB signal coming from DVI-I connector
  - 1: YPbPr signal coming from D-sub connector
  - 2: HDMI signal coming from DVI-I connector

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.4.2 Sensor Format

Purpose

These functions set or get image format of source input. The format is different according to the channel of input.

Prototype

- C/C++

```
int Hdv62_SetSensorFormat (UINT Number, UINT Format)
int Hdv62_GetSensorFormat (UINT Number, UINT
   &Format)
```

- C#

```
int SetSensorFormat (uint Number, uint Format)
int GetSensorFormat (uint Number, out uint Format)
```

- VB.Net

```
SetSensorFormat (ByVal Number as UInteger, ByVal
   Format as UInteger) As Integer
GetSensorFormat (ByVal Number as UInteger, ByRef
   Format as UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Format
  The format of source input. Following are possible values fort different channel:
- Channel 0
  Analog RGB signal from DVI-I connector
  - 0: VGA 60 fps (640 x 480),
  - 1: SVGA 60 fps (800 x 600),
  - 2: XVGA 60 fps (1024 x 768),
  - 3: SXVGA 60 fps (1280 x 1024),

- Channel 1
  YPbPr signal from D-SUB connector
  - 0: 525i 30 fps (720 x 480 interlace, in frames per second),
  - 1: 625i 25 fps (720 x 576 interlace, in frames per second),
  - 2: 525p 60 fps (720 x 480 progressive),
  - 3: 625p 50 fps (720 x 576 progressive),
  - 4: 720p 30 fps (1280 x 720 progressive),
  - 5: 720p 50 fps (1280 x 720 progressive),
  - 6: 720p 60 fps (1280 x 720 progressive),
  - 7: 1080i 25 fps (1920 x 1080 interlace, in frames per second),
  - 8: 1080i 30 fps (1920 x 1080 interlace, in frames per second),
  - 9: 1080p 50 fps (1920 x 1080 progressive)
  - 10: 1080p 60 fps (1920 x 1080 progressive)

- Channel 2
  HDMI signal from DVI-I connector
  - 0: 720p 50 fps YCrCb In (1280 x 720 progressive),
  - 1: 720p 60 fps YCrCb In (1280 x 720 progressive),
  - 2: 1080i 25 fps YCrCb In (1920 x 1080 interlace, in frames per second),
  - 3: 1080i 30 fps YCrCb In (1920 x 1080 interlace, in frames per second),
  - 4: 1080p 25 fps YCrCb In (1920 x 1080 progressive)
  - 5: 1080p 30 fps YCrCb In (1920 x 1080 progressive)
  - 6: 1080p 50 fps YCrCb In (1920 x 1080 progressive)
  - 7: 1080p 60 fps YCrCb In (1920 x 1080 progressive)
  - 8: VGA 60 fps (640 x 480),
  - 9: SVGA 60 fps (800 x 600),
  - 10: XGA 60 fps (1024 x 768),
  - 11: SXGA 60 fps (1280 x 1024),
  - 12: UXGA 60 fps (1600 x 1200)
  - 13: 720p 50 fps RGB In (1280 x 720 progressive),
  - 14: 720p 60 fps RGB In (1280 x 720 progressive),
  - 15: 1080i 25 fps RGB In (1920 x 1080 interlace, in frames per second),
  - 16: 1080i 30 fps RGB In (1920 x 1080 interlace, in frames per second),
  - 17: 1080p 25 fps RGB In (1920 x 1080 progressive)
  - 18: 1080p 30 fps RGB In (1920 x 1080 progressive)
  - 19: 1080p 50 fps RGB In (1920 x 1080 progressive)
  - 20: 1080p 60 fps RGB In (1920 x 1080 progressive)

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.4.3 Sensor Width

Purpose

This function gets image width of source input. The width is same as the one of settings of SensorFormat.

Prototype

- C/C++
  ```
  int Hdv62_GetSensorWidth (UINT Number, UINT &Width)
  ```
- C#
  ```
  int GetSensorWidth (uint Number, out uint Width)
  ```
- VB.Net
  ```
  GetSensorWidth (ByVal Number as UInteger, ByRef
     Width as UInteger) As Integer
  ```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Width
  The image width of source input as illustrated in Figure 4-1.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

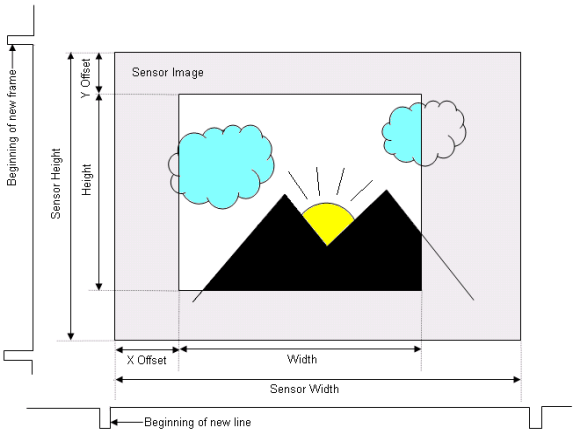


Figure 4-1: Signals of Frame Image

4.4.4 Sensor Height

Purpose

This function gets image height of source input. The height is same as the one of settings of SensorFormat.

Prototype

- C/C++
```
int Hdv62_GetSensorHeight (UINT Number, UINT &
   Height)
```
- C#
```
int GetSensorHeight (uint Number, out uint Height)
```
- VB.Net
```
GetSensorHeight (ByVal Number as UInteger, ByRef
   Height as UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Height
  The image height of source input as illustrated in Figure 4-1.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.4.5 Width

Purpose

These functions set or get active width of image. They are used to reduce horizontal pixels per line.

Prototype

- C/C++

```
int Hdv62_SetWidth (UINT Number, UINT Width)
int Hdv62_GetWidth (UINT Number, UINT & Width)
```

- C#

```
int SetWidth (uint Number, uint Width)
int GetWidth (uint Number, out uint Width)
```

- VB.Net

```
SetWidth (ByVal Number as UInteger, ByVal Width as
   UInteger) As Integer
GetWidth (ByVal Number as UInteger, ByRef Width as
   UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Width
  The active width of image as illustrated in Figure 4-1. The width coupled with XOffset is the cropping parameters. The device crops sensor image if the width is less than Sensor-Width which is obtained from section 4.4.3. The following criteria must be met:
  - Width <= SensorWidth
  - Width + XOffset <= SensorWidth
  - Width needs to be multiple of 8 pixels.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.4.6 Height

Purpose

These functions set or get active height of image. They can be used to reduce vertical scan lines.

Prototype

- C/C++

```
int Hdv62_SetHeight (UINT Number, UINT Height)
int Hdv62_GetHeight (UINT Number, UINT& Height)
```

- C#

```
int SetHeight (uint Number, uint Height)
int GetHeight (uint Number, out uint Height)
```

- VB.Net

```
SetHeight (ByVal Number as UInteger, ByVal Height as
   UInteger) As Integer
GetHeight (ByVal Number as UInteger, ByRef Height as
   UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Height
  The active height of image as illustrated in Figure 4-1. The height coupled with YOffset is the cropping parameters. The device crops sensor image if the height is less than Sensor-Height which is obtaineed from section 4.4.4. The following criteria must be met:
  - Height <= SensorHeight
  - Height + YOffset <= SensorHeight
  - Height is an even number.
  - YOffset is an even number if sensor source is interlaced.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.4.7 X Offset

Purpose

These functions set or get start pixels of image cropping per line.

Prototype

- C/C++

```
int Hdv62_SetXOffset (UINT Number, UINT XOffset)
int Hdv62_GetXOffset (UINT Number, UINT & XOffset)
```

- C#

```
int SetXOffset (uint Number, uint XOffset)
int GetXOffset (uint Number, out uint XOffset)
```

- VB.Net

```
SetXOffset (ByVal Number as UInteger, ByVal XOffset
   as UInteger) As Integer
GetXOffset (ByVal Number as UInteger, ByRef XOffset
   as UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- XOffset
  The start pixels of image cropping per line as illustrated in Figure 4-1. The XOffset coupled with Width is the cropping parameters. The device crops sensor image if the XOffset is larger than 0. Following criteria must be met:
  - Width <= SensorWidth
  - Width + XOffset <= SensorWidth
  - Width needs to be multiple of 8 pixels.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.4.8 Y Offset

Purpose

These functions set or get start lines of image cropping.

Prototype

- C/C++

```
int Hdv62_SetYOffset (UINT Number, UINT YOffset)
int Hdv62_GetYOffset (UINT Number, UINT & YOffset)
```

- C#

```
int SetYOffset (uint Number, uint YOffset)
int GetYOffset (uint Number, out uint YOffset)
```

- VB.Net

```
SetYOffset (ByVal Number as UInteger, ByVal YOffset
   as UInteger) As Integer
GetYOffset (ByVal Number as UInteger, ByRef YOffset
   as UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- YOffset
  The start lines of image cropping as illustrated in Figure 4-1. The YOffset coupled with Height is the cropping parameters. The device crops sensor image if the YOffset is larger than 0. Following criteria must be met:
  - Height <= SensorHeight
  - Height + YOffset <= SensorHeight
  - Height is an even number.
  - YOffset is an even number if sensor source is interlaced.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.4.9 Output Format

Purpose

These functions set or get output format of pixel.

Prototype

- C/C++

```
int Hdv62_SetOutputFormat (UINT Number, UINT Format)
int Hdv62_GetOutputFormat (UINT Number, UINT &
   Format)
```

- C#

```
int SetOutputFormat (uint Number, uint Format)
int GetOutputFormat (uint Number, out uint Format)
```

- VB.Net

```
SetOutputFormat (ByVal Number as UInteger, ByVal
   Format as UInteger) As Integer
GetOutputFormat (ByVal Number as UInteger, ByRef
   Format as UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Format
  The output format of pixel. Could be one of the following values:

  0: 24bit RGB (RGB24) – 8bit R + 8bit G + 8bit B

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	B1	R0	G0	B0
dw1	G2	B2	R1	G1
dw2	R3	G3	B3	R2

1: 30bit RGB – 10bit R + 10bit G + 10bit B

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	xx+B[9:4]	B[3:0]+G[9:6]	G[5:0]+R[9:8]	R[7:0]

'x' means 'don't care bit'.

2: 32bit RGB (RGB32) – 8bit R + 8bit G + 8bit B + 8bit Alpha

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	Alpha	R	G	B

3: 8bit gray scale (Y8) – 8bit Y

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	Y3	Y2	Y1	Y0

4: 24bit YCbCr 4:4:4 – 8bit Y + 8bit Cb + 8bit Cr

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	Y1	Cr0	Cb0	Y0
dw1	Cb2	Y2	Cr1	Cb1
dw2	Cr3	Cb3	Y3	Cr2

5: 16bit YCbCr 4:2:2 (YUY2) – 8bit Y + 8bit Cb/Cr

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	Cr	Y	Cb	Y

6: 20bit YCbCr 4:2:2 – 10bit Y + 10bit Cb/Cr

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	Cr0[1:0]+Y1[9:4]	Y1[3:0]+Cb0[9:6]	Cb0[5:0]+Y0[9:8]	Y0[7:0]
dw1	xxxx+Cb2[9:6]	Cb2[5:0]+Y2[9:8]	Y2[7:0]	Cr0[9:2]
dw2	Cb4[1:0]+Y4[9:4]	Y4[3:0]+Cr2[9:6]	Cr2[5:0]+Y3[9:8]	Y3[7:0]
dw3	xxxx+Cr4[9:6]	Cr4[5:0]+Y5[9:8]	Y5[7:0]	Cb4[9:2]

'x' means 'don't care bit'.

Note:

1. Actual width of image setting in WriteCropping routine needs to be multiple of 8 and actual height needs to be an even number.
2. Total bytes of one scan line need to be aligned to multiple of 16. If the requirement is not met, HDV62 appends dummy bytes to the end of each line. For example, if 20bit YCbCr 4:2:2 is selected and

   width = 800 pixels, each line = 2144 bytes (11 dummy bytes appended);

   width = 640 pixles, each line = 1712 bytes (5 dummy bytes appended);

   width = 720 pixles, each line = 1920 bytes (0 dummy bytes appended).

   The formula is: total bytes of each line = ((width * 16 / 6) + 15) & ~15 with all integer calculation.
3. The formula of YCbCr to RGB is as following:

   R = Y + 1.371(Cr-128)

   G = Y – 0.698(Cr-128) – 0.336(Cb-128)

   B = Y + 1.732(Cb-128)

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.4.10 HDelay

Purpose

These functions set or get the horizontal delay of frame images. Horizontal delay is like an X offset. It can move images left or right to remove a black vertical line.

Prototype

- C/C++

```
int Hdv62_SetHDelay (UINT Number, int Delay)
int Hdv62_GetHDelay (UINT Number, int & Delay)
```

- C#

```
int SetHDelay (uint Number, int Delay)
int GetHDelay (uint Number, out int Delay)
```

- VB.Net

```
SetHDelay (ByVal Number as UInteger, ByVal Delay as
   Integer) As Integer
GetHDelay (ByVal Number as UInteger, ByRef Delay as
   Integer) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Delay
  The horizontal delay of frame images. The allowed value is from -3 to 3. Each resolution has its default value that users call this routine after channel and sensor format have been set.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section4.9 for error information about return codes.

4.4.11 Contrast

Purpose

These functions set or get the contrast of frame images from the YPbPr input.

Prototype

- C/C++

```
int Hdv62_SetContrast (UINT Number, int Value)
int Hdv62_GetContrast (UINT Number, int & Value)
```

- C#

```
int SetContrast (uint Number, int Value)
int GetContrast (uint Number, out int Value)
```

- VB.Net

```
SetContrast (ByVal Number as UInteger, ByVal Value
   as Integer) As Integer
GetContrast (ByVal Number as UInteger, ByRef Value
   as Integer) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Value
  The contrast of frame images. The allowed value is from 0 to 255 and default value is 128.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.4.12 Hue

Purpose

These functions set or get the hue of frame images from the YPbPr input.

Prototype

- C/C++i

```
nt Hdv62_SetHue (UINT Number, int Value)
int Hdv62_GetHue (UINT Number, int & Value)
```

- C#

```
int SetHue (uint Number, int Value)
int GetHue (uint Number, out int Value)
```

- VB.Net

```
SetHue (ByVal Number as UInteger, ByVal Value as
   Integer) As Integer
GetHue (ByVal Number as UInteger, ByRef Value as
   Integer) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Value
  The hue of frame images. The allowed value is from -128 to 127 and default value is 0.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.4.13 Saturation

Purpose

These functions set or get the saturation of frame images from the YPbPr input.

Prototype

- C/C++

```
int Hdv62_SetSaturation (UINT Number, int Value)
int Hdv62_GetSaturation (UINT Number, int & Value)
```

- C#

```
int SetSaturation (uint Number, int Value)
int GetSaturation (uint Number, out int Value)
```

- VB.Net

```
SetSaturation (ByVal Number as UInteger, ByVal Value
   as Integer) As Integer
GetSaturation (ByVal Number as UInteger, ByRef Value
   as Integer) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Value
  The saturation of frame images. The allowed value is from 0 to 255 and default value is 128.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.5 Event and Callback Functions

4.5.1 Event Selector

Purpose

These functions set or get the type of event.

Prototype

- C/C++

```
int Hdv62_SetEventSelector (UINT Number, UINT Mode)
int Hdv62_GetEventSelector (UINT Number, UINT &
   Mode)
```

- C#

```
int SetEventSelector (uint Number, uint Mode)
int GetEventSelector (uint Number, out uint Mode)
```

- VB.Net

```
SetEventSelector (ByVal Number as UInteger, ByVal
   Mode as UInteger) As Integer
GetEventSelector (ByVal Number as UInteger, ByRef
   Mode as UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Mode
  The type of event. HDV62 has 2 kind of event; one is frame event and another is DI event. The frame event is the library issue an event when a frame is ready. The DI event is the library issue an event when the state of any DI has changed. Mode could be one of the following values:

  0: Frame event

  1: DI event

  Use the SetEventHandle routinue to set an event handle.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.5.2 Event Handle

Purpose

These functions set or get event handle. Event and callback are the methods to know when to get frame or to check DI's state. Usually applications use only one of them, but both of them were set is allowed.

Prototype

- C/C++

```
int Hdv62_SetEventHandle (UINT Number, HANDLE
   Handle)
int Hdv62_GetEventSelector (UINT Number, HANDLE
   &Handle)
```

- C#

```
int SetEventHandle (uint Number, IntPtr Handle)
int SetEventHandle (uint Number, SafeWaitHandle
   Handle)
int GetEventHandle (uint Number, out IntPtr Handle)
int GetEventHandle (uint Number, out SafeWaitHandle
   Handle)
```

- VB.Net

```
SetEventHandle (ByVal Number as UInteger, ByVal
   Handle as IntPtr) As Integer
SetEventHandle (ByVal Number as UInteger, ByVal
   Handle As SafeWaitHandle) As Integer
GetEventHandle (ByVal Number as UInteger, ByRef
   Handle as IntPtr) As Integer
GetEventHandle (ByVal Number as UInteger, ByRef
   Handle As SafeWaitHandle) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Handle
  The handle of event which is created by your application. After user's application waits for an event, users can call GetImageStream routine to get pointer of frame buffer or call GetDI routine to get the state of DI.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.5.3 CallbackSelector

Purpose

These functions set or get the type of callback function.

Prototype

- C/C++

```
int Hdv62_SetCallbackSelector (UINT Number, UINT
   Mode)
int Hdv62_GetCallbackSelector (UINT Number, UINT
   Mode)
```

- C#

```
int SetCallbackSelector (uint Number, uint Mode)
int GetCallbackSelector (uint Number, out uint Mode)
```

- VB.Net

```
SetCallbackSelector (ByVal Number as UInteger, ByVal
   Mode as UInteger) As Integer
GetCallbackSelector (ByVal Number as UInteger, ByRef
   Mode as UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Mode
  The type of callback. HDV62 has 2 kind of callback; one is frame callback and another is DI callback. The frame callback is the library calls the callback routine when a frame is ready. The DI event is the library calls the callback routine when the state of any DI has changed. Mode could be one of the following values:

  0: Frame callback

  1: DI callback

  Use SetCallback routinue to set a callback function.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.5.4 Callback

Purpose

These functions set or get event handle. Event and callback are the methods to know when to get frame or to check DI's state. Usually applications use only one of them, but both of them were set is allowed.

Prototype

- C/C++

```
int Hdv62_SetCallback (UINT Number, HDV62CALLBACK
   Fun)
int Hdv62_GetCallback (UINT Number, HDV62CALLBACK
   Fun)
```

- C#

```
int SetCallback (uint Number, HDV62CALLBACK Fun)
int GetCallback (uint Number, out HDV62CALLBACK Fun)
```

- VB.Net

```
SetCallback (ByVal Number as UInteger, ByVal Fun as
   HDV62CALLBACK) As Integer
GetCallback (ByVal Number as UInteger, ByRef Fun as
   HDV62CALLBACK) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Fun
  The pointer of callback routine. Users need to declare a callback function and set it as the parameter. The prototype of Fun please refer to the include files. In the callback function, users can call GetImageStream routine to get pointer of frame buffer or call GetDI routine to get the state of DI.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.6 Acquisition Control Functions

4.6.1 Acquisition Frame Count

Purpose

These functions set or get the count of frames you want to capture at once. Call AcquisitionStart to start capturing, check the acquisition state by calling GetAcquisitionStatus routine, and call AcquisitionStop to stop capturing.

Prototype

- C/C++

```
int Hdv62_SetAcquisitionFrameCount (UINT Number,
   UINT Count)
int Hdv62_GetAcquisitionFrameCount (UINT Number,
   UINT & Count)
```

- C#

```
int SetAcquisitionFrameCount (uint Number, uint
   Count)
int GetAcquisitionFrameCount (uint Number, out uint
   Count)
```

- VB.Net

```
SetAcquisitionFrameCount (ByVal Number as UInteger,
   ByVal Count as UInteger) As Integer
GetAcquisitionFrameCount (ByVal Number as UInteger,
   ByRef Count as UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Count
  The frame count desired to capture. Could be the following value:

  0: Capture until AcquisitionStop is called.

  > 0: Acquire the amount of Count of frames. When the Count is reached, the acquisition status will be changed to 0 (stopped). Users must call AcquisitionStop to stop the acquisition.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.6.2 Acquisition Start

Purpose

This function starts to capture frames.

Prototype

- C/C++
  ```
  int Hdv62_AcquisitionStart (UINT Number)
  ```
- C#
  ```
  int AcquisitionStart (uint Number)
  ```
- VB.Net
  ```
  AcquisitionStart (ByVal Number as UInteger) As
     Integer
  ```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.6.3 Acquisition Stop

Purpose

This function stops capturing frames.

Prototype

- C/C++
  ```
  int Hdv62_AcquisitionStop (UINT Number)
  ```
- C#
  ```
  int AcquisitionStop (uint Number)
  ```
- VB.Net
  ```
  AcquisitionStop (ByVal Number as UInteger) As
     Integer
  ```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.6.4 One Shot

Purpose

This function gets one frame image within a specific time. One-Shot is an independent function which can not be used with AcquisitionStart, Callback, and Event. When this function finished without error, call GetImageStream to get the pointer of frame image.

Prototype

- C/C++
  ```
  int Hdv62_OneShot (UINT Number, UINT Timeout)
  ```
- C#
  ```
  int OneShot (uint Number, uint Timeout)
  ```
- VB.Net
  ```
  OneShot (ByVal Number as UInteger, ByVal Timeout as
     UInteger) As Integer
  ```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Timeout
  The maximum waiting time of getting a frame in milliseconds.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.6.5 Image Stream

Purpose

This function gets the pointer of image buffer. Usually this function is called in callback routine, after waiting frame event, or after call OneShot routine.

Prototype

- C/C++
```
int Hdv62_GetImageStream (UINT Number, void
   **Buffer)
```
- C#
```
int GetImageStream (uint Number, out IntPtr Buffer)
```
- VB.Net
```
GetImageStream (ByVal Number as UInteger, ByRef
   Buffer as IntPtr) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Buffer
  The pointer of image buffer.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.6.6 Acquisition Status

Purpose

This function gets the status of acquisition.

Prototype

- C/C++
```
int Hdv62_GetAcquisitionStatus (UINT Number, UINT
   &Status)
```
- C#
```
int GetAcquisitionStatus (uint Number, out uint
   Status)
```
- VB.Net
```
GetAcquisitionStatus (ByVal Number as UInteger,
   ByRef Status as UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Status
  The status of acquisition. Could be one of the following values:

  0: Stopped

  1: Running

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.6.7 Acquisition Statistics

Purpose

This function gets the amount of captured frames since Acquisition Start.

Prototype

- C/C++
```
int Hdv62_GetAcquisitionStatistics (UINT Number,
   UINT &Count)
```
- C#
```
int GetAcquisitionStatistics (uint Number, out uint
   Count)
```
- VB.Net
```
GetAcquisitionStatistics (ByVal Number as UInteger,
   ByRef Count as UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Count
  The amount of captured frames.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.6.8 Sensor Status

Purpose

This function obtains the status of whether the device is connecting a proper sensor or not.

Prototype

- C/C++

```
int Hdv62_GetSensorStatus (UINT Number, UINT&
   Locked)
```

- C#

```
int GetSensorStatus (uint Number, out uint Locked)
```

- VB.Net

```
GetSensorStatus (ByVal Number as UInteger, ByRef
   Locked as UInterger) As Integer
```

Parameters

- Number

  The number of device. The allowed value is from 0 to 15.

- Locked

  Whether the input signal is locked. It can be used as whether a proper sensor is connected. It can be the following values:

  0: No proper signal is detected

  1: A proper signal is detected

  For Analog RGB and YPbPr inputs, Locked = 1 is only if the sensor format is the one you select. For HDMI the input, Locked = 1 if the sensor is a valid HDMI source.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 to get error information about return codes.

4.6.9 Save Image

Purpose

This function saves last image buffer as an image file or raw data file depending on the extension of file name.

Prototype

- C/C++
  ```
  int Hdv62_SaveImage (UINT Number, LPTSTR FileNamet)
  ```
- C#
  ```
  int SaveImage (uint Number, string FileNamet)
  ```
- VB.Net
  ```
  SaveImage (ByVal Number as UInteger, ByVal FileNamet
     as String) As Integer
  ```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- FileName
  The name of image file. The library supports following type of images:

  BMP: if FileName is *.bmp

  JPEG: if FileNAME is *.jpg or *.jpeg

  JIFF: if FileName is *.tif

  PNG: if FileName is *.png

  Raw data: other file type

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.7 Trigger Control Functions

4.7.1 Trigger In Source

Purpose

These functions set which trigger source you are using.

Prototype

- C/C++

```
int Hdv62_SetTriggerInSource ( UINT Number, UINT
   Source)
Int Hdv62_GetTriggerInSource (UINT Number, UINT&
   Source)
```

- C#

```
int SetTriggerInSource (uint Number, uint Source)
int GetTriggerInSource (uint Number, out uint
   Source)
```

- VB.Net

```
SetTriggerInSource (ByVal Number As UInteger, ByVal
   Source As UInteger) As Integer
GetTriggerInSource (ByVal Number As UInteger, ByRef
   Source As UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Source
  The trigger source. There are two types you can choose:

  0: Free run – the source input is CCD camera

  1: External trigger or software trigger – holds frame acquisition until trigger input is active or SoftwareTrigger is called.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.7.2 Trigger In Polarity

Purpose

These functions set or get trigger polarity of trigger input.

Prototype

- C/C++

```
int Hdv62_SetTriggerInPolarity (UINT Number, UINT
   Polarity)
Int Hdv62_GetTriggerInPolarity (UINT Number, UINT&
   Polarity)
```

- C#

```
int SetTriggerInPolarity (uint Number, uint
   Polarity)
int GetSensorWidth (uint Number, out uint Polarity)
```

- VB.Net

```
SetTriggerInPolarity (ByVal Number As UInteger,
   ByVal Polarity As UInteger) As Integer
GetTriggerInPolarity (ByVal Number As UInteger,
   ByRef Polarity As UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Polarity
  The trigger polarity of trigger input. Could be one of the following values:

  0: rising edge

  1: falling edge

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.7.3 Software Trigger

Purpose

This function issues a capturing trigger if the source parameter of GetTriggerInsource is 1.

Prototype

- C/C++

```
int Hdv62_SoftwareTrigger (UINT Number)
```

- C#

```
int SoftwareTrigger (uint Number)
```

- VB.Net

```
      SoftwareTrigger (ByVal Number As UInteger) As
   Integer
```

Parameters

- Number

  The number of device. The allowed value is from 0 to 15.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.7.4 One Pulse Out

Purpose

This function sends one pulse out from the trigger out pin.

Prototype

- C/C++
  ```
  int Hdv62_SendOnePulseOut
  ```
- C#
  ```
  int SendOnePulseOut (uint Number)
  ```
- VB.Net
  ```
  SendOnePulseOut (ByVal Number As UInteger) As
     Integer
  ```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.7.5 Trigger Out Polarity

Purpose

These functions set or get active polarity of trigger output.

Prototype

- C/C++

```
int Hdv62_SetTriggerOutPolarity (UINT Number, UINT
   Polarity)
Int Hdv62_GetTriggerOutPolarity (UINT Number, UINT&
   Polarity)
```

- C#

```
int SetTriggerOutPolarity (uint Number, uint
   Polarity)
int GetTriggerOutPolarity (uint Number, out uint
   Polarity)
```

- VB.Net

```
SetTriggerOutPolarity (ByVal Number As UInteger,
   ByVal Polarity As UInteger) As Integer
GetTriggerOutPolarity (ByVal Number As UInteger,
   ByRef Polarity As UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Polarity
  The active polarity of trigger output. Could be one of following values:

  0: normal low, active high

  1: normal high, active low

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.7.6 Trigger Out Pulse Width

Purpose

These functions set or get pulse width of trigger output.

Prototype

- C/C++

```
int Hdv62_SetTriggerOutPulseWidth (UINT Number,
   UINT Width)
Int Hdv62_GetTriggerOutPulseWidth (UINT Number,
   UINT& Width)
```

- C#

```
int SetTriggerOutPulseWidth (uint Number, uint
   Width)
int GetTriggerOutPulseWidth (uint Number, out uint
   Width)
```

- VB.Net

```
SetTriggerOutPulseWidth (ByVal Number As UInteger,
   ByVal Width As UInteger) As Integer
GetTriggerOutPulseWidth (ByVal Number As UInteger,
   ByRef Width As UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Width
  The pulse width of trigger output, in us.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.8 Digital I/O Functions

4.8.1 4.8.1 Digital IO Selector

Purpose

These functions set or get which channel of DI and DO you are setting when call GetDI, SetDO, and GetDO routines.

Prototype

- C/C++

```
int Hdv62_SetDigitalIOSelector (UINT Number, UINT
   Channel)
Int Hdv62_GetDigitalIOSelector (UINT Number, UINT&
   Channel)
```

- C#

```
int SetDigitalIOSelector (uint Number, uint Channel)
int GetDigitalIOSelector (uint Number, out uint
   Channel)
```

- VB.Net

```
SetDigitalIOSelector (ByVal Number As UInteger,
   ByVal Channel As UInteger) As Integer
GetDigitalIOSelector (ByVal Number As UInteger,
   ByRef Channel As UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Channel
  The channel of DI and DO. Allowed channel is from 0 to 3.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.8.2 DI

Purpose

This function gets state of DI channel.

Prototype

- C/C++
  ```
  int Hdv62_GetDI (UINT Number, UINT& Value)
  ```
- C#
  ```
  int GetDI (uint Number, out uint Value)
  ```
- VB.Net
  ```
  GetDI (ByVal Number As UInteger, ByRef Value As
    UInteger) As Integer
  ```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Value
  The state of DI channel. Could be one of the following values:

  0: Low

  1: High

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.8.3 DI Event

Purpose

These functions enable or disable DI event. If enable, the library issues an event when change of state (COS) of any DI channel. Users can call SetEvent or SetCallback routines to wait for this event.

Prototype

- C/C++

```
int Hdv62_SetDIEvent (UINT Number, UINT Enable)
int Hdv62_GetDIEvent (UINT Number, UINT& Enable)
```

- C#

```
int SetDIEvent (uint Number, uint Enable)
int GetDIEvent (uint Number, out uint Enable)
```

- VB.Net

```
SetDIEvent (ByVal Number As UInteger, ByVal Enable
   As UInteger) As Integer
GetDIEvent (ByVal Number As UInteger, ByRef Enable
   As UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Enable
  Whether enable DI event or not. Could be one of the following values:

  0: disable

  1: enable

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.8.4 DO

Purpose

These functions set or get state of DO channel.

Prototype

- C/C++

```
int Hdv62_SetDO (UINT Number, UINT Value)
int Hdv62_GetDO (UINT Number, UINT& Value)
```

- C#

```
int SetDO (uint Number, uint Value)
int GetDO (uint Number, out uint Value)
```

- VB.Net

```
SetDO (ByVal Number As UInteger, ByVal Value As
   UInteger) As Integer
GetDO (ByVal Number As UInteger, ByRef Value As
   UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Value
  The state of DO channel. Could be one of the following values:

  0: Low

  1: High

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 for error information about return codes.

4.9 Other Functions

4.9.1 Error Text

Purpose

This function gets error text string.

Prototype

- C/C++
  ```
  int Hdv62_GetErrorText (int code, char *Text)
  ```
- C#
  ```
  string GetErrorText (int code)
  ```
- VB.Net
  ```
  GetErrorText (ByVal code As Integer) As String
  ```

Parameters

- Code
  The error code returned by other functions.
- Text
  A string of error text. Users need to allocate a buffer, maximum 160 bytes, to contain this text.

Return Value

- C/C++
  Always return 0.
- C#
  Return the error text.
- VB.Net
  Return the error text.

4.10 EDID Functions

4.10.1 Ready Status

Purpose

These functions set or get the ready status of the EDID ROM.

Prototype

- C/C++

```
int Hdv62_SetEdidReadyStatus (UINT Number, UINT
   Status)
int Hdv62_GetEdidReadyStatus (UINT Number, UINT&
   Status)
```

- C#

```
int SetEdidReadyStatus (uint Number, uint Status)
int GetEdidReadyStatus (uint Number, out uint
   Status)
```

- VB.Net

```
SetEdidReadyStatus (ByVal Number As UInteger, ByVal
   Status As UInteger) As Integer
GetEdidReadyStatus (ByVal Number As UInteger, ByRef
   Status As UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Status
  Indicates whether or not the EDID ROM is ready. This value can be read by external device through DVI-I connector. Some external devices can auto-adjusting their resolution by reading this EDID ROM. Users can configure the content of the EDID ROM and set it as ready state. Could be one of the following values:
  0: EDID ROM is not ready

  1: EDID ROM is ready

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 to get error information about return codes.

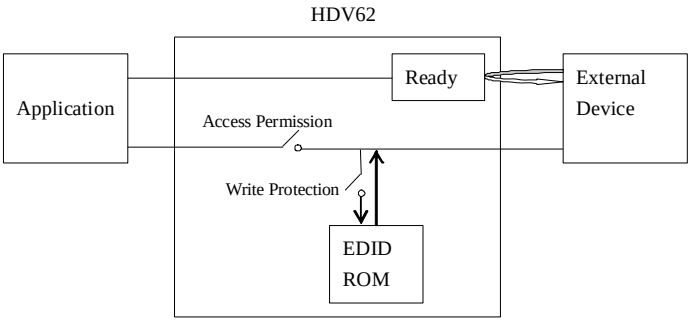


Figure 4-2: EDID ROM Architecture

4.10.2 Access Permission

Purpose

These functions set or get whether or not applications can access the EDID ROM.

Prototype

- C/C++

```
int Hdv62_SetEdidAccessPermission (UINT Number,
   UINT Status)
int Hdv62_GetEdidAccessPermission (UINT Number,
   UINT& Status)
```

- C#

```
int SetEdidAccessPermission (uint Number, uint
   Status)
int GetEdidAccessPermission (uint Number, out uint
   Status)
```

- VB.Net

```
SetEdidAccessPermission (ByVal Number As UInteger,
   ByVal Status As UInteger) As Integer
GetEdidAccessPermission (ByVal Number As UInteger,
   ByRef Status As UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Status
  Indicates whether or not the EDID ROM can be accessed. The EDID ROM can be accessed by either application or external device at the same time. So if you want to open EDID, you need to open access permission, configure EDID ROM, close access permission, and then plug external device into DVI-I connector. Please see figure 4-2 for EDID hardware architecture. Could be one of the following values:

  0: EDID ROM is not accessible

  1: EDID ROM is accessible

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 to get error information about return codes.

4.10.3 Write Protection

Purpose

These functions set or get whether or not the EDID ROM is writable.

Prototype

- C/C++

```
int Hdv62_SetEdidWriteProtection (UINT Number, UINT
   Status)
int Hdv62_GetEdidWriteProtection (UINT Number,
   UINT& Status)
```

- C#

```
int SetEdidWriteProtection (uint Number, uint
   Status)
int GetEdidWriteProtection (uint Number, out uint
   Status)
```

- VB.Net

```
SetEdidWriteProtection (ByVal Number As UInteger,
   ByVal Status As UInteger) As Integer
GetEdidWriteProtection (ByVal Number As UInteger,
   ByRef Status As UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Status
  Indicates whether or not the EDID ROM is writable. Users need to break write protection before write EDID ROM and set it to protect the content of the EDID ROM. Please see figure 4-2 for EDID hardware architecture. Could be one of the following values:
  0: EDID ROM can be read/write

  1: EDID ROM is read only

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 to get error information about return codes.

4.10.4 Rom Selector

Purpose

These functions set or get the offset of the EDID ROM which is currently accessed.

Prototype

- C/C++

```
int Hdv62_SetEdidRomSelector (UINT Number, UINT
   Offset)
int Hdv62_GetEdidRomSelector (UINT Number, UINT&
   Offset)
```

- C#

```
int SetEdidRomSelector (uint Number, uint Offset)
int GetEdidRomSelector (uint Number, out uint
   Offset)
```

- VB.Net

```
SetEdidRomSelector (ByVal Number As UInteger, ByVal
   Offset As UInteger) As Integer
GetEdidRomSelector (ByVal Number As UInteger, ByRef
   Offset As UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Offset
  Indicates the offset of the EDID ROM. Allowed value is between 0 and 255.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 to get error information about return codes.

4.10.5 Rom Read/Write

Purpose

These functions set or get the value of the EDID ROM.

Prototype

- C/C++

```
int Hdv62_SetEdidRom (UINT Number, UINT Value)
int Hdv62_GetEdi Rom (UINT Number, UINT& Value)
```

- C#

```
int SetEdidRom (uint Number, uint Value)
int GetEdidRom (uint Number, out uint Value)
```

- VB.Net

```
SetEdidRom (ByVal Number As UInteger, ByVal Value As
   UInteger) As Integer
GetEdidRom (ByVal Number As UInteger, ByRef Value As
   UInteger) As Integer
```

Parameters

- Number
  The number of device. The allowed value is from 0 to 15.
- Value
  Indicates the value of the EDID ROM. Allowed value is between 0 and 255.

Return Value

No error occurs if return value >= 0; if negative value, please refer to Section 4.9 to get error information about return codes.

5 Programming Guide

5.1 Introduction

Complete documentation on DirectShow application programming can be found at http://msdn.microsoft.com/en-us/library/dd390351(VS.85).aspx. If a DirectX SDK is installed, this documentation is also available from DirectX SDK Help. The last version of DirectShow SDK was moved to Windows SDK.

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 HDV62 Video Capture filter and the ADLINK HDV62 Crossbar 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. Call **IAMCrossbar::Route** to switch source channel. 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 capturing video streams are also illustrated in this chapter and gives examples of two ways of controlling device driver.

5.2 Descriptions of Filters

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

5.2.1 Source Filter

ADLINK HDV62 Video Capture

ADLINK HDV62 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 HDV62 Video Capture
Filter CLSID	Not applicable.
Filter Category Name	WDM Streaming Capture Devices
Filter Category	AM_KSCATEGORY_CAPTURE
Capture Pin Supported Media Types	MEDIATYPE_Video Subtypes: MEDIASUBTYPE_RGB24 MEDIASUBTYPE_BGR30 MEDIASUBTYPE_RGB32 MEDIASUBTYPE_RGB8 MEDIASUBTYPE_YUV8 MEDIASUBTYPE_YUY2 MEDIASUBTYPE_YU10
Exported Interfaces	IAMAnalogVideoDecoder IAMDroppedFrames IAMStreamConfig IAMVideoProcAmp IAdvance IVideoFormat ICardInfo
Merit	MERIT_DO_NOT_USE

In the above, MEDIASUBTYPE_RGB8, MEDIASUBTYPE_BGR30, MEDIASUBTYPE_YUV8, and MEDIASUBTYPE_YU10, please refer to section 5.3.3. For IAdvance, IVideoFormat, and ICardInfo, please refer to section 5.4.

ADLINK HDV62 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 card.

Filter Name	ADLINK HDV62 Crossbar
Filter Category Name	WDM_Streaming Crossbar Devices
Exported Interfaces	IAMCrossbar
Input Pins	Pin Names: Video RGB In Video YRYBY In Video SerialDigital In
Output Pin	Pin Name: Video Decoder Out

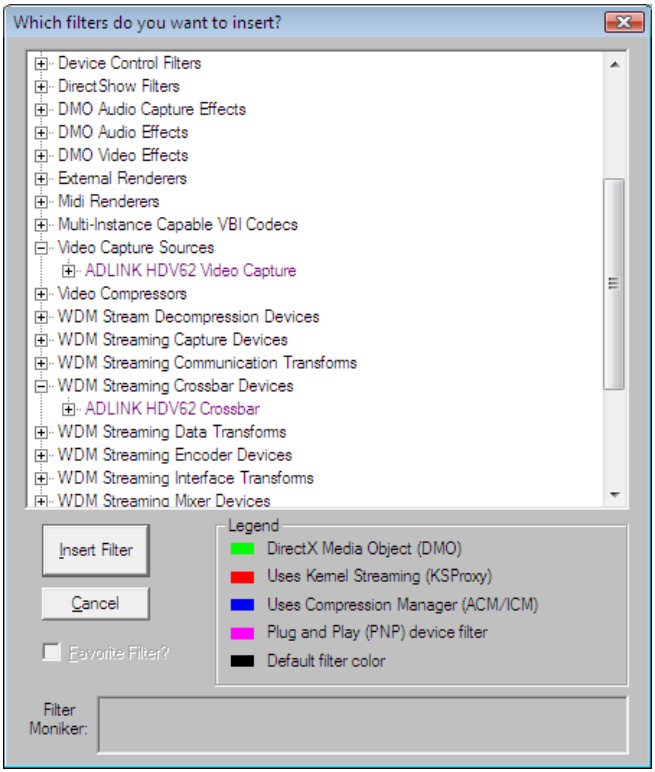
In the above,

- 0. Video RGB In is an analog RGB signal coming from DVI-I connector.
- 1. Video YRYBY In is an YPbPr signal coming from D-SUB connector.
- 2. Video SerialDigital In is a HDMI signal coming from DVI-I connector.

You can call IAMCrossbar::Route routine to switch the input channel.

5.2.2 Example Graphs

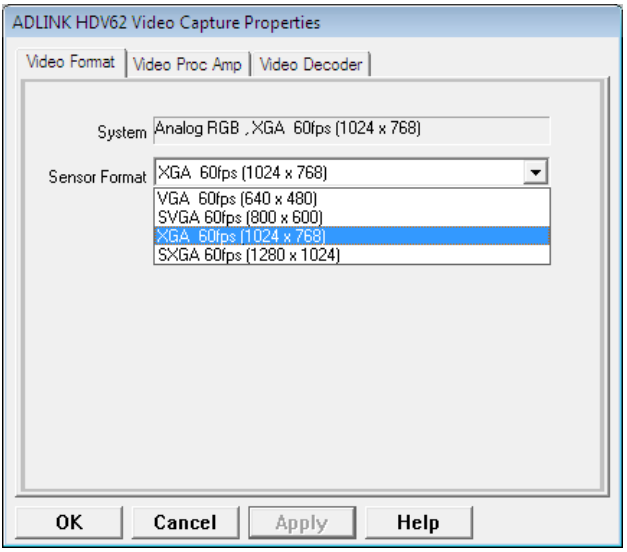
The Microsoft DirectX SDK provides a very useful debugging utility called GraphEdit, which can be used to simulate graph building. From the **Graph** menu of the GraphEdit application, click **Insert Filters…** and choose the desired filters. Filters are organized by categories. Click I**nsert Filter** 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 HDV62 Video Capture filter and ADLINK HDV62 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:

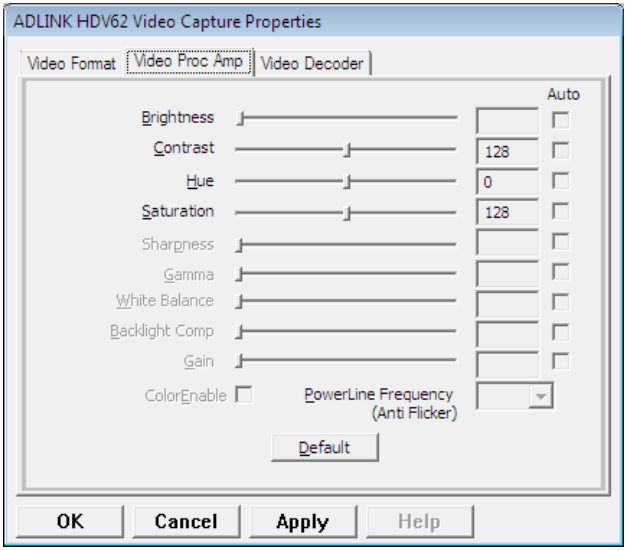
ADLINK HDV62 Video Capture filter:

Video Format:

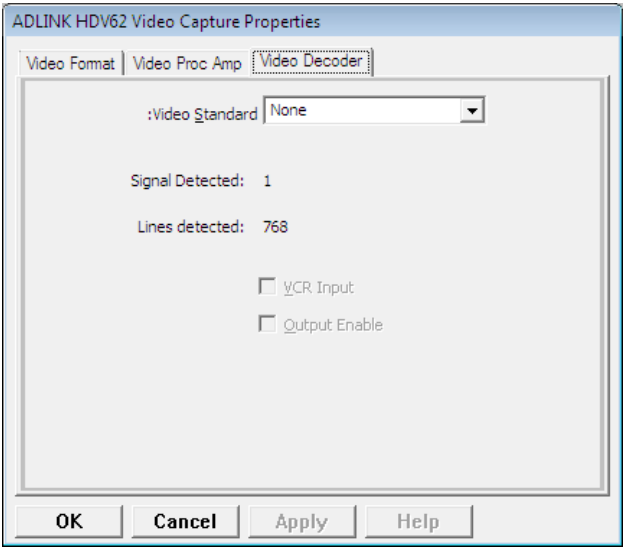


In the above picture, System is current settings. Content of Sensor Format is changed with different crossbar input. Supported sensor format please refer to section 5.4.1.1

Video Proc Amp:

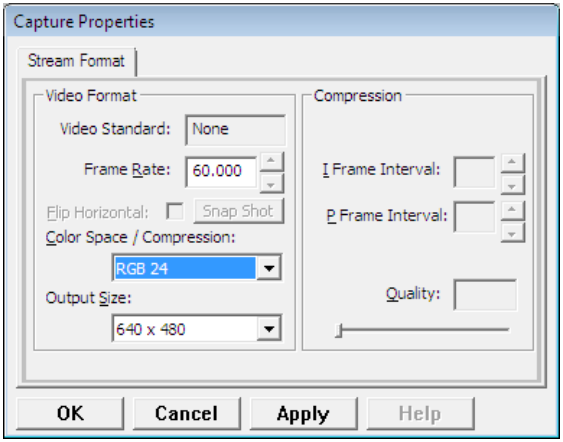


Video Decoder:



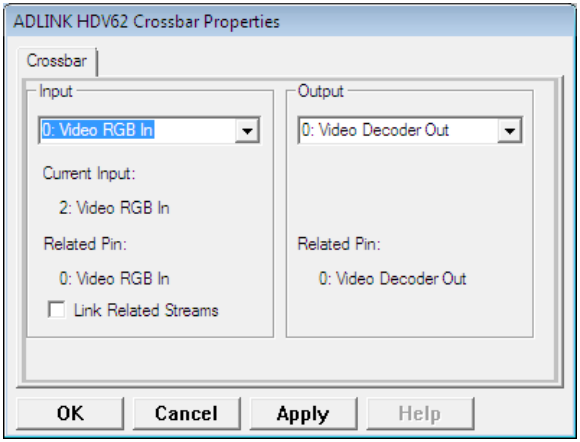
In the above graphic, Signal Detected represents whether the input source is valid, and Lines detected represents the valid lines of the input source.

Capture Pin Properties:



In the above, Color Space/Compression is BGRA means BGR30. Supported Color Space please refers to section 5.3.3.

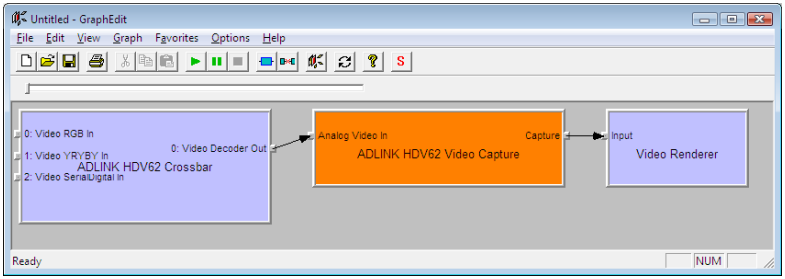
ADLINK HDV62 Crossbar filter:



Select input before the Capture pin of ADLINK HDV62 Video Capture filter is connected.

Example Graph

1. Open GraphEdit.exe.
2. Click ‘Insert a filter into the graph’ on the toolbar and insert ‘ADLINK HDV62 Video Capture’, ‘ADLINK HDV62 Crossbar’, and ‘Video Renderer’ filters where ‘ADLINK HDV62 Video Capture’ is in ‘Video Capture Sources’ group, ‘ADLINK HDV62 Crossbar’ in ‘WDM Streaming Crossbar Devices’ group, and ‘Video Renderer’ in ‘DirectShow Filters’ group.
3. Right click on ‘ADLINK HDV62 Crossbar’ filter and select ‘Filter Properties…’. In ‘ADLINK Crossbar Properties’ dialog, select the Input channel. Click ‘OK’ to close it.
4. Right click on ‘ADLINK HDV62 Video Capture’ filter and select ‘Filter Properties…’. In ‘ADLINK HDV62 Video Capture Properties’ dialog, select the Sensor Format. Click ‘OK’ to close it.
5. Right click on Capture pin of ‘ADLINK HDV62 Video Capture’ filter and select ‘Pin Properties…’. In ‘Capture Properties’ dialog, select the Color Space/Compression. Click ‘OK’ to close it.
6. Drag cursor from ‘Video Decoder Out’ pin to ‘Analog Video In’ pin and from ‘Capture’ pin to ‘Input’ pin as illustrated in the following graphic.
7. Click ‘Play the graph’ on the toolbar to start a preview.



Note: If you choice VMR to substitute for default Video Renderer, the preview video will be a vertical mirror video. Please insert a Color Space Converter filter before it to correct this symptom.

5.3 Controlling Driver

The ADLINK HDV62 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.

5.3.1 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 an ADLINK HDV62 Video Capture
filter
// or an ADLINK HDV62 Crossbar 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);
}
```

5.3.2 Use COM interfaces

Use the methods of IAMVideoProcAmp interface of standard DirectShow Interface to get or set the qualities of an incoming video signal, such as contrast, saturation… etc. Other interfaces please refer to DirectX SDK help and next section.

```
// pFilter points to an ADLINK HDV62 Video Capture
filter

IAMVideoProcAmp *pAmp;
HRESULT hr;
long contrast, flags;
hr = pFilter->QueryInterface(IID_IAMVideoProcAmp,
(void **)&pAmp);
if (SUCCEEDED(hr))
{
        pAmp->Get(VideoProcAmp_Contrast, &contrast,
&flags);

        pAmp->Release();
}
```

ADLINK HDV62 Crossbar

The ADLINK HDV62 Crossbar filter implements an IAMCrossbar interface. It routes signals from an analog or digital source to a video capture filter.

```
// pFilter points to an ADLINK HDV62 Crossbar filter

IAMCrossbar *pXbar;
HRESULT hr;
hr = pFilter->QueryInterface(IID_IAMCrossbar, (void
**)&pXbar);
if (SUCCEEDED(hr))
{
        // Route from input 1 to output 0
        pXbar->Route(0, 1);
        pXBar->Release();
}
```

5.3.3 Color Space

The ADLINK HDV62 Video Capture supports 7 kind of following color spaces:

MEDIASUBTYPE_RGB24 – 8bit R + 8bit G + 8bit B

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	B1	R0	G0	B0
dw1	G2	B2	R1	G1
dw2	R3	G3	B3	R2

MEDIASUBTYPE_BGR30 – 10bit R + 10bit G + 10bit B

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	xx+B[9:4]	B[3:0]+G[9:6]	G[5:0]+R[9:8]	R[7:0]

'x' means 'don't care bit'.

Its compression (FOURCC code) is 'BGRA' and GUID is 41524742-0000-0010-8000-00AA00389B71.

MEDIASUBTYPE_RGB32 – 8bit R + 8bit G + 8bit B + 8bit Alpha

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	Alpha	R	G	B

MEDIASUBTYPE_RGB8 – 8bit Y (Gray scale)

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	Y3	Y2	Y1	Y0

MEDIASUBTYPE_YUV8 – 8bit Y + 8bit Cb + 8bit Cr – YCbCr 4:4:4

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	Y1	Cr0	Cb0	Y0
dw1	Cb2	Y2	Cr1	Cb1
dw2	Cr3	Cb3	Y3	Cr2

Its compression (FOURCC code) is 'YUV8' and GUID is 38565559-0000-0010-8000-00AA00389B71.

MEDIASUBTYPE_YUY2 – 8bit Y + 8bit Cb/Cr – YCbCr 4:2:2

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	Cr	Y	Cb	Y

MEDIASUBTYPE_YU10 – 10bit Y + 10bit Cb/Cr – 20bit YCbCr 4:2:2

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	Cr0[1:0]+Y1[9:4]	Y1[3:0]+Cb0[9:6]	Cb0[5:0]+Y0[9:8]	Y0[7:0]
dw1	xxxx+Cb2[9:6]	Cb2[5:0]+Y2[9:8]	Y2[7:0]	Cr0[9:2]
dw2	Cb4[1:0]+Y4[9:4]	Y4[3:0]+Cr2[9:6]	Cr2[5:0]+Y3[9:8]	Y3[7:0]
dw3	xxxx+Cr4[9:6]	Cr4[5:0]+Y5[9:8]	Y5[7:0]	Cb4[9:2]

‘ x’ means ‘don’t care bit’.

Its compression (FOURCC code) is ‘YU10’ and GUID is 30315559-0000-0010-8000-00AA00389B71.

Notes for the MEDIASUBTYPE_YU10 format:

Total bytes of one scan line need to be aligned to multiple of 16. If the requirement is not met, HDV62 appends dummy bytes to the end of each line. For example,

if width = 800 pixels, each line = 2144 bytes (11 dummy bytes appended);

if width = 640 pixles, each line = 1712 bytes (5 dummy bytes appended);

if width = 720 pixles, each line = 1920 bytes (0 dummy bytes appended).

The formula is: total bytes of each line = ((width * 16 / 6) + 15) & ~15 with all integer calculation.

The formula to convert YCbCr to RGB is as following:

R = Y + 1.371(Cr-128)

G = Y – 0.698(Cr-128) – 0.336(Cb-128)

B = Y + 1.732(Cb-128)

5.4 Proprietary Interfaces

The proprietary interfaces are dedicated to ADLINK HDV62 Video Capture filter that they are not the standard interfaces of DirectShow. They are kind of COM interfaces and can be got from ADLINK HDV62 Video Capture filter by call IBaseFilter::QueryInterface routine.

5.4.1 IVideoFormat

IVideoFormat provides methods to select sensor format, select output format, horizontal delay, and set video cropping.

5.4.1.1 Sensor Format

Purpose

These functions read or write source format of CCD sensor including video standard, resolution, and frame rate.

Prototype

- C/C++

```
HRESULT WriteSensorFormat (UINT Format)
HRESULT ReadSensorFormat (UINT *Format)
```

- C#

```
int WriteSensorFormat (uint Format)
int ReadSensorFormat (out uint Format)
```

- VB.Net

```
WriteSensorFormat (ByVal Format As UInteger) As
   Integer
ReadSensorFormat (ByRef Format As UInteger) As
   Integer
```

Parameters

- Format
  The format of source sensor. Following are possible values fort different crossbar input:

Channel 0: Analog RGB from DVI-I connector

- 0: VGA 60 fps (640 x 480),
- 1: SVGA 60 fps (800 x 600),
- 2: XGA 60 fps (1024 x 768),
- 3: SXGA 60 fps (1280 x 1024),

Channel 1: YPbPr from D-SUB connector

- 0: 525i 30 fps (720 x 480 interlace, in frame per second),
- 1: 625i 25 fps (720 x 576 interlace, in frame per second),
- 2: 525p 60 fps (720 x 480 progressive),
- 3: 625p 50 fps (720 x 576 progressive),
- 4: 720p 30 fps (1280 x 720 progressive),
- 5: 720p 50 fps (1280 x 720 progressive),
- 6: 720p 60 fps (1280 x 720 progressive),
- 7: 1080i 25 fps (1920 x 1080 interlace, in frame per second),
- 8: 1080i 30 fps (1920 x 1080 interlace, in frame per second),
- 9: 1080p 50 fps (1920 x 1080 progressive)
- 10: 1080p 60 fps (1920 x 1080 progressive)

Channel 2: HDMI from DVI-I connector

- 0: 720p 50 fps YCrCb In (1280 x 720 progressive),
- 1: 720p 60 fps YCrCb In (1280 x 720 progressive),
- 2: 1080i 25 fps YCrCb In (1920 x 1080 interlace, in frame per second),
- 3: 1080i 30 fps YCrCb In (1920 x 1080 interlace, in frame per second),
- 4: 1080p 25 fps YCrCb In (1920 x 1080 progressive)
- 5: 1080p 30 fps YCrCb In (1920 x 1080 progressive)
- 6: 1080p 50 fps YCrCb In (1920 x 1080 progressive)
- 7: 1080p 60 fps YCrCb In (1920 x 1080 progressive)
- 8: VGA 60 fps (640 x 480),
- 9: SVGA 60 fps (800 x 600),
- 10: XGA 60 fps (1024 x 768),
- 11: SXGA 60 fps (1280 x 1024),
- 12: UXGA 60 fps (1600 x 1200)
- 13: 720p 50 fps RGB In (1280 x 720 progressive),
- 14: 720p 60 fps RGB In (1280 x 720 progressive),
- 15: 1080i 25 fps RGB In (1920 x 1080 interlace, in frame per second),
- 16: 1080i 30 fps RGB In (1920 x 1080 interlace, in frame per second),
- 17: 1080p 25 fps RGB In (1920 x 1080 progressive)
- 18: 1080p 30 fps RGB In (1920 x 1080 progressive)
- 19: 1080p 50 fps RGB In (1920 x 1080 progressive)
- 20: 1080p 60 fps RGB In (1920 x 1080 progressive)

Return Value

No error occurs if return value >= 0; if negative value, call AMGet-ErrorText function to get error information about return codes.

5.4.1.2 Cropping

Purpose

These functions read or write actual output resolution.

Prototype

- C/C++

```
HRESULT WriteCropping (RECT Rt)
HRESULT ReadCropping (RECT * Rt)
```

- C#

```
int WriteCropping (Rectangle RECT Rt)
int ReadCropping (out Rectangle RECT Rt)
```

- VB.Net

```
WriteCropping (ByVar Rt as RECT) As Integer
ReadCropping (ByRef Rt as RECT) As Integer
```

Parameters

- Rt
  A rectangle setting to crop the sensor image as illustrated in Figure 4-1. The rectangle must be located inside the sensor image. Actual image starts from (Rt.left, Rt.top) and its width is Rt.right – Rt.left and height is Rt.bottom – Rt.top. Actual image width needs to be multiple of 16 pixels and actual height is even. If the sensor source is interlace, the top value in Rt need to be even

Return Value

No error occurs if return value >= 0; if negative value, call AMGet-ErrorText function to get error information about return codes.

5.4.1.3 Horizontal Delay

Purpose

These functions read or write the horizontal delay of frame images. Horizontal delay is like X offset. It can move images left or right to remove black vertical line.

Prototype

- C/C++
```
HRESULT WriteHDelay (INT Delay)
HRESULT ReadHDelay (INT * Delay)
```
- C#
```
int WriteHDelay (int Delay)
int ReadHDelay (out int Delay)
```
- VB.Net
```
WriteHDelay (ByVal Delay As Integer) As Integer
ReadHDelay (ByRef Delay As Integer) As Integer
```

Parameters

- Delay
  The horizontal delay of frame images. The allowed value is from -3 to 3. Each resolution has its default value that users call this routine after channel and sensor format have been set.

Return Value

No error occurs if return value >= 0; if negative value, call AMGet-ErrorText function to get error information about return codes.

5.4.2 IAdvance

IAdvace provides methods to access peripheral IOs including DIs, DOs, Trigger In, and Trigger Out.

5.4.2.1 DIO

Purpose

These functions read and write the state of digital inputs and digital outputs. HDV62 has 4 DI pins and 4 DO pins.

Prototype

- C/C++

```
HRESULT DIO_ReadPins (UINT StartPin, UINT StopPin,
   UINT *Values)
HRESULT DIO_WritePins (UINT StartPin, UINT StopPin,
   UINT Values)
HRESULT DIO_ReadDOs (UINT StartPin, UINT StopPin,
   UINT *Values)
```

- C#

```
int DIO_ReadPins (uint StartPin, uint StopPin, out
   uint Values)
int DIO_WritePins (uint StartPin, uint StopPin, uint
   Values)
int DIO_ReadDOs (uint StartPin, uint StopPin, out
   uint Values)
```

- VB.Net

```
DIO_ReadPins (ByVal StartPin As UInteger, ByVal
   StopPin As UInteger, ByRef Values As UInteger) As
   Integer
DIO_WritePins (ByVal StartPin As UInteger, ByVal
   StopPin As UInteger, ByVal Values As UInteger) As
   Integer
DIO_ReadPins (ByVal StartPin As UInteger, ByVal
   StopPin As UInteger, ByRef Values As UInteger) As
   Integer
```

Parameters

- StartPin
  The beginning DIO pin you want to read from or write to. The StartPin is less than 4.
- StopPin
  The end DIO pin you want to read from or write to. The StopPin is larger than or equal to StartPin and less than 4.
- Values
  The states of DIO you read from or write to.

  If StartPin = StopPin, Values is the state of single pin. Values = 0 means low level and Values = 1 means high level.

  If StartPin < StopPin, Values is the states of multi-pins. Each respective bit of Values is the states of each pin counting from StartPin. For example, StartPin = 0, StopPin = 3, and Values = 12 = 2b1010, that is Pin0 = low level, Pin1 = high level, Pin2 = low level, and Pin3 = high level.

Return Value

No error occurs if return value >= 0; if negative value, call AMGetErrorText function to get error information about return codes.

5.4.2.2 Interrupt

Purpose

These functions enable or disable interrupt of DIs. The HDV62 will issue a hardware interrupt when change of state (COS) of any DI. Please refer to INotify section in this chapter to know how to capture the interrupt signal.

Prototype

- C/C++
```
HRESULT DIO_EnableInterrupt ( )
HRESULT DIO_DisableInterrupt ( )
```
- C#
```
int DIO_EnableInterrupt ( )
int DIO_DisableInterrupt ( )
```
- VB.Net
```
DIO_EnableInterrupt ( ) As Integer
DIO_DisableInterrupt ( ) As Integer
```

Parameters

Return Value

No error occurs if return value >= 0; if negative value, call AMGetErrorText function to get error information about return codes.

5.4.2.3 Trigger Input Mode

Purpose

These functions read or write the mode of external trigger signal. Trigger Input is an external pin which can accept external signal. This signal can control when to capture a frame.

Prototype

- C/C++

```
HRESULT TriggerIn_WriteMode (UINT Value)
HRESULT TriggerIn_ReadMode (UINT *Value)
```

- C#

```
int TriggerIn_WriteMode (uint Value)
int TriggerIn_ReadMode (out uint Value)
```

- VB.Net

```
TriggerIn_WriteMode (ByVal Value As UInteger) As
   Integer
TriggerIn_ReadMode (ByRef Value As UInteger) As
   Integer
```

Parameters

- Value
  Enable or disable trigger input. Could be one of the following values:

  0: Free run – the source input is according to CCD camera

  1: External trigger or software trigger – holds frame acquisition until trigger input is active or SoftwareTrigger is called.

Return Value

No error occurs if return value >= 0; if negative value, call AMGetErrorText function to get error information about return codes.

5.4.2.4 Trigger Input Polarity

Purpose

These functions read or write the polarity of external trigger signal. HDV62 captures a frame when Trigger Input is enabled and the state of polarity is reached.

Prototype

- C/C++

```
HRESULT TriggerIn_WritePolarity (UINT Value)
HRESULT TriggerIn_ReadPolarity (UINT *Value)
```

- C#

```
int TriggerIn_WritePolarity (uint Value)
int TriggerIn_ReadPolarity (out uint Value)
```

- VB.Net

```
TriggerIn_WritePolarity (ByVal Value As UInteger) As
   Integer
TriggerIn_ReadPolarity (ByRef Value As UInteger) As
   Integer
```

Parameters

- Value

  The polarity of trigger input. Could be one of the following values:

  0: rising edge

  1: falling edge

Return Value

No error occurs if return value >= 0; if negative value, call AMGet-ErrorText function to get error information about return codes.

5.4.2.5 Software Trigger

Purpose

This function controls when to get a frame when Trigger Input is enabled.

Prototype

- C/C++
  ```
  HRESULT TriggerIn_SoftwareTrigger ( )
  ```
- C#
  ```
  int TriggerIn_SoftwareTrigger ( )
  ```
- VB.Net
  ```
  TriggerIn_SoftwareTrigger ( ) As Integer
  Parameters
  ```

Return Value

No error occurs if return value >= 0; if negative value, call AMGet-ErrorText function to get error information about return codes.

5.4.2.6 One Pulse Out

Purpose

These functions read or write the mode of trigger output. Trigger Output is an external pin which can output continuous pulse signal.

Prototype

- C/C++
  ```
  HRESULT TriggerOut_OnePulseOut ()
  ```
- C#
  ```
  int TriggerOut_OnePulseOut ()
  ```
- VB.Net
  ```
  TriggerOut_OnePulseOut () As Integer
  ```

Return Value

No error occurs if return value >= 0; if negative value, call AMGetErrorText function to get error information about return codes.

5.4.2.7 Trigger Output Polarity

Purpose

These functions read or write the polarity of trigger output.

Prototype

- C/C++

```
HRESULT TriggerOut_WritePolarity (UINT Value)
HRESULT TriggerOut_ReadPolarity (UINT *Value)
```

- C#

```
int TriggerOut_WritePolarity (uint Value)
int TriggerOut_ReadPolarity (out uint Value)
```

- VB.Net

```
TriggerOut_WritePolarity (ByVal Value As UInteger)
   As Integer
TriggerOut_ReadPolarity (ByRef Value As UInteger) As
   Integer
```

Parameters

- Value
  The active polarity of trigger output. Could be one of the following values:

  0: normal low, active high

  1: normal high, active low

Return Value

No error occurs if return value >= 0; if negative value, call AMGetErrorText function to get error information about return codes.

5.4.2.8 Trigger Output Pulse Width

Purpose

These functions read or write the pulse width of trigger output.

Prototype

- C/C++

```
HRESULT TriggerIn_WritePulseWidth (UINT Value)
HRESULT TriggerIn_ReadPulseWidth (UINT *Value)
```

- C#

```
int TriggerIn_WritePulseWidth (uint Value)
int TriggerIn_ReadPulseWidth (out uint Value)
```

- VB.Net

```
TriggerIn_WritePulseWidth (ByVal Value As UInteger)
   As Integer
TriggerIn_ReadPulseWidth (ByRef Value As UInteger)
   As Integer
```

Parameters

- Value
  The pulse width of trigger output, in us.

Return Value

No error occurs if return value >= 0; if negative value, call AMGet-ErrorText function to get error information about return codes.

5.4.2.9 EDID ROM Ready Status

Purpose

These functions read or write the ready status of the EDID ROM.

Prototype

- C/C++

```
HRESULT EDID_WriteReadyStatus (UINT Value)
HRESULT EDID_ReadReadyStatus (UINT *Value)
```

- C#

```
int EDID_WriteReadyStatus (uint Value)
int EDID_ReadReadyStatus (out uint Value)
```

- VB.Net

```
EDID_WriteReadyStatus (ByVal Value As UInteger) As
   Integer
EDID_ReadReadyStatus (ByRef Value As UInteger) As
   Integer
```

Parameters

- Value
  Indicates whether or not the EDID ROM is ready. This value can be read by external device through DVI-I connector. Some external devices can auto-adjusting their resolution by reading this EDID ROM. Users can configure the content of the EDID ROM and set it as ready state. Please see figure 4-2 for EDID hardware architecture. Could be one of the following values:
  0: EDID ROM is not ready

  1: EDID ROM is ready

Return Value

No error occurs if return value >= 0; if negative value, call AMGetErrorText function to get error information about return codes.

5.4.2.10EDID ROM Access Permission

Purpose

These functions read or write whether or not applications can access the EDID ROM.

Prototype

- C/C++

```
HRESULT EDID_WriteAccessPermission (UINT Value)
HRESULT EDID_ReadAccessPermission (UINT *Value)
```

- C#

```
int EDID_WriteAccessPermission (uint Value)
int EDID_ReadAccessPermission (out uint Value)
```

- VB.Net

```
EDID_WriteAccessPermission (ByVal Value As
   UInteger) As Integer
EDID_ReadAccessPermission (ByRef Value As UInteger)
   As Integer
```

Parameters

- Status
  Indicates whether or not the EDID ROM can be accessed. The EDID ROM can be accessed by either application or external device at the same time. So if you want to open EDID, you need to open access permission, configure EDID ROM, close access permission, and then plug external device into DVI-I connector. Please see figure 4-2 for EDID hardware architecture. Could be one of the following values:
  0: EDID ROM is not accessible

  1: EDID ROM is accessible

Return Value

No error occurs if return value >= 0; if negative value, call AMGet-ErrorText function to get error information about return codes.

5.4.2.11EDID ROM Write Protection

Purpose

These functions read or write whether or not the EDID ROM is writable.

Prototype

- C/C++

```
HRESULT EDID_WriteWriteProtection (UINT Value)
HRESULT EDID_ReadWriteProtection (UINT *Value)
```

- C#

```
int EDID_WriteWriteProtection (uint Value)
int EDID_ReadWriteProtection (out uint Value)
```

- VB.Net

```
EDID_WriteWriteProtection (ByVal Value As UInteger)
   As Integer
EDID_ReadWriteProtection (ByRef Value As UInteger)
   As Integer
```

Parameters

- Status
  Indicates whether or not the EDID ROM is writable. Users need to break write protection before write EDID ROM and set it to protect the content of the EDID ROM. Please see figure 4-2 for EDID hardware architecture. Could be one of the following values:
  0: EDID ROM can be read/write

  1: EDID ROM is read only

Return Value

No error occurs if return value >= 0; if negative value, call AMGet-ErrorText function to get error information about return codes.

5.4.2.12EDID ROM Read/Write

Purpose

These functions read or write the value of EDID ROM.

Prototype

- C/C++

```
HRESULT EDID_WriteRom (UINT Offset, UINT Value)
HRESULT EDID_ReadRom (UINT Offset, UINT *Value)
```

- C#

```
int EDID_WriteRom (uint Offset, uint Value)
int EDID_ReadRom (uint Offset, out uint Value)
```

- VB.Net

```
EDID_WriteRom (ByVal Offset As UInteger , ByVal
   Value As UInteger) As Integer
EDID_ReadRom (ByVal Offset As UInteger , ByRef Value
   As UInteger) As Integer
```

Parameters

- Offset
  Indicates the offset of the EDID ROM. Allowed value is between 0 and 255.
- Value
  Indicates the value of the EDID ROM. Allowed value is between 0 and 255.

Return Value

No error occurs if return value >= 0; if negative value, call AMGet-ErrorText function to get error information about return codes.

5.4.3 ICardInfo

5.4.3.1 Version

Purpose

These functions get miscellaneous versions

Prototype

- C/C++

```
HRESULT GetHardwareVersion (UINT *Version)
HRESULT GetFirmwareVersion (UINT *Version)
HRESULT GetDriverVersion (UINT *Version)
```

- C#

```
int GetHardwareVersion (out uint Version)
int GetFirmwareVersion (out uint Version)
int GetDriverVersion (out uint Version)
```

- VB.Net

```
GetHardwareVersion (ByRef Version as UInteger) As
   Integer
GetFirmwareVersion (ByRef Version as UInteger) As
   Integer
GetDriverVersion (ByRef Version as UInteger) As
   Integer
```

Parameters

- Version
  A hexadecimal number that each byte represents a version digital.

  For driver version, Version = (Major << 24 + Minor << 16 + Revision << 8 + Release).

  For example, if Version = 0x1000001, it represents “1.0.0.1”.

  For firmware version, Version = (Year-2000) << 28 + Month << 24 + (Day / 16) << 20 + (Day % 16) << 16 + (Hour / 16) << 12 + (Hour % 16) << 8 + (Minute /16) << 4 + (Minute % 16).

  To simplify, it can be expressed as the output of printf in C language as following:

```
int Year = (Version >> 28) + 2000;
int Month = (Version >> 24) & 0x0F;
int Day = (Version >> 16) & 0xFF;
int Hour = (Version >> 8) & 0xFF;
int Minute = Version & 0xFF;
printf("%d/%d/%x %x:%x", Year, Month, Day, Hour,
   Minute);
```

For example, if Version = 0x9b191407, it represents "2009/11/19 14:07".

For hardware version, the least 16 bits are carrier board version and the most 16 bits are daughter board version. If most 16 bits are all zero, the hardware is a single board. Version of carrier board or daughter board = (Major << 8 + Minor).

For example, if Version = 0xA1A2, it represents carrier board version is "A2" and daughter board version is "A1".

Return Value

No error occurs if return value >= 0; if negative value, call AMGetErrorText function to get error information about return codes.

5.4.3.2 Card ID

Purpose

This function gets card ID which is set by DIP switch on HDV62 card.

Prototype

- C/C++
  ```
  HRESULT GetCardID (UINT* ID)
  ```
- C#
  ```
  int GetCardID (out uint ID)
  ```
- VB.Net
  ```
  GetCardID (ByRef ID As UInteger) As Integer
  ```

Parameters

- ID
  Card ID can be set by DIP switch on card. Its possible value is from 0 to 15. Card ID can distinguish cards when multi-cards were installed on one system. Set them to different number according to chapter 2 Hardware Reference. Leave card ID as default, all are same is no problem, if you don't care about it.

Return Value

No error occurs if return value >= 0; if negative value, call AMGet-ErrorText function to get error information about return codes.

5.4.4 INotify

5.4.4.1 Enable and Disable Event

Purpose

These functions enable or disable event of DI interrupt.

Prototype

- C/C++

```
HRESULT EnableEvent (ULONG EventID, BYTE
   *EventHandle, BYTE **Cookie)
HRESULT DisableEvent (ULONG EventID, BYTE *Cookie)
```

- C#

```
int EnableEvent (uint EventID, IntPtr EventHandle,
   out IntPtr Cookie)
int DisableEvent (uint EventID, IntPtr Cookie)
```

- VB.Net

```
EnableEvent (ByVal EventID As UInteger, ByVal
   EventHandle As IntPtr, ByRef Cookie As IntPtr) As
   Integer
DisableEvent (ByVal EventID As UInteger, ByValCookie
   As IntPtr) As Integer
```

Parameters

- EventID
  A user defined positive number
- EventHandle
  A handle created by user's application
- Cookie
  A pointer to retain the address of object created by EnableEvent routine. This variable is used for releasing this created object.

Return Value

No error occurs if return value >= 0; if negative value, call AMGet-ErrorText function to get error information about return codes.

5.5 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.
Hdv62Proxy.h	The header file is required for all C++ applications.
Hdv62Guids.h	The header file is required for all C++ applications
DirectShowLib	Imports this name space for all Microsoft .Net application.
Hdv62ProxyLib	Imports this name space for all Microsoft .Net application.

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 for C++ applications. If you do not call this function, this library is not required.
DirectShowLib-2005.dll	The class library of DirectShow is required for all Microsoft .Net applications.
Hdv62ProxyLib.dll	The class library of the interfaces of HDV62 is required for all Microsoft .Net applications.

Note: In the above, the libraries for Microsoft .Net applications work and test on .Net Framwork 2.0. You should use Microsoft Visual Studio 2005 to build your .Net application.

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.
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
   or,
   %DXSDK%\Samples\C++\DirectShow\BaseClasses\Debug

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

.Net Programming Users

Microsoft DirectShow only provides C++ programming. As for .net users, they need convert DirectShow COM objects to .net classes. 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 HDV62 cards in the installation directory.

6 ViewCreatorPro Utility

ViewCreatorPro provides a simple but powerful way to setup, configure, test, and debug a vision system. This chapter outlines how to verify the vision system by ViewCreatorPro.

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

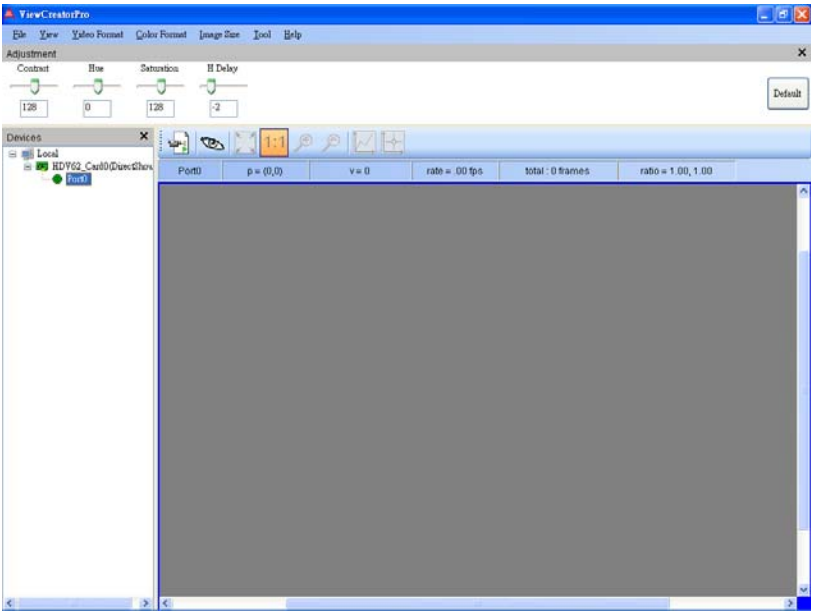
6.1 Overview

ViewCreatorPro offers the following features:

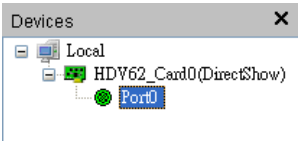
- 32-bit operations under Windows XP/Vista DirectShow driver
- Access and configuration of HDV62 cards
- Video picture adjustments (brightness, contrast, hue, saturation, video cropping)
- Open an image file (BMP or JPG)
- Save an image file (BMP)
- Direct access to general purpose I/O
- Trigger I/O
- EDID R/W

6.2 Component Description

Launch ViewCreatorPro and you can see the following view:



6.2.1 Devices panel



Local
List all cards that ViewCreatorPro supports in the system.

Active Device
All operations will apply to this device.

Inactive Device
Click the device name after this icon can activate this device.

Active port
All operations will apply to this port.

Inactive port
Click the port name after this icon can activate this port.

Close this panel

6.2.2 Adjustment Panel

This panel is used to adjust the parameters of images, including contrast, hue, saturation, and horizontal delay. Users can drag and drop the slider, or directly input a value in the edit box, to change the value of each parameter. Note that contrast, hue, and saturation can be adjusted only when YPbPr is selected.



Default Button
Press the Default button to reset all the parameters on the adjustment panel to default values.

Close this panel
Close the adjustment panel.

6.2.3 Tool panel



Continue Grab
Click it to grab images continuously and click it again to stop grabbing. This is a toggle button.



Stop Grab
Stop grabbing.



Snap Shot
Capture an image.



Hide/Show Image
Hide or show images. This is a toggle button.



Fit Size
Fit the images to the display region.



Original Size
Restore the images to the original size.



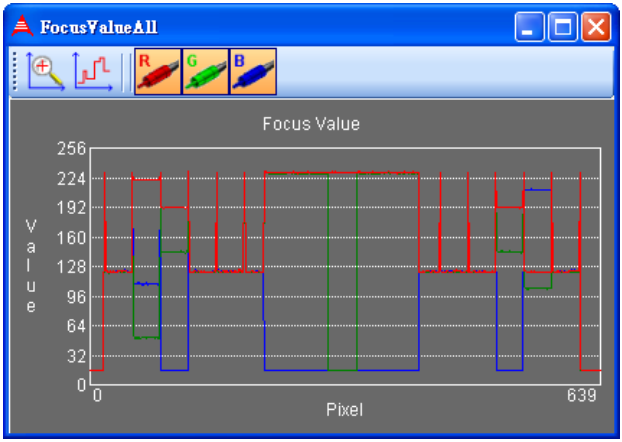
Zoom In
Zoom in the images .



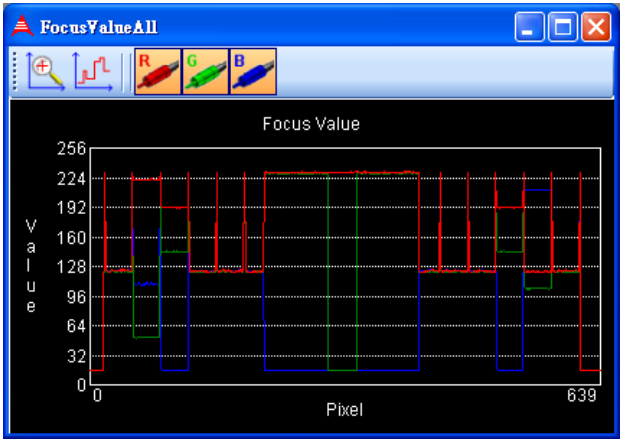
Zoom Out
Zoom out the images.



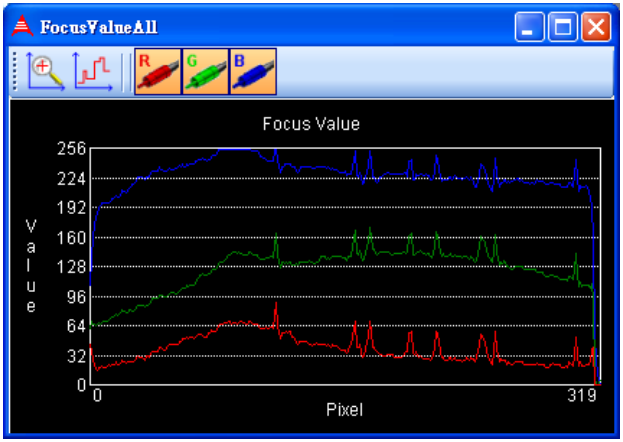
Focus Value
Open a window to show the focus value chart of the selected horizontal line in an image. The selected horizontal line is marked in red, and users can change this line by clicking anywhere on the image. The background color of the window is gray and the focus value chart updates immediately while grabbing.



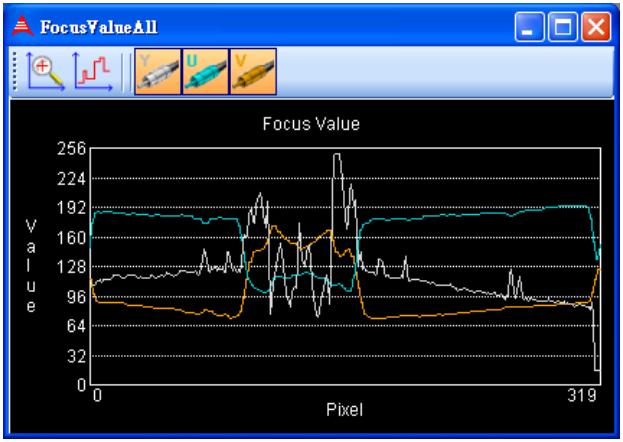
When the capturing is stopped, the background color of the window turns into black.



If the color format of captured images is RGB, there are three curves individually represented red, green, and blue in the chart.



If the color format of captured images is YUV, there are three curves individually represented y, u, and v in the chart.



Zoom In



Open a new window to zoom in the green rectangle region appeared in the focus value window. Users can resize the green rectangle by dragging the vertical green line on the right side or on the left side.

Differential



Open a new window to show the slope chart in the green rectangle region appeared in the focus value window. Users can resize the green rectangle by dragging the vertical green line on the right side or on the left side.



Show/Hide Red Values

Show or hide the red value of pixels.



Show/Hide Green Values

Show or hide the green value of pixels.



Show/Hide Blue Values

Show or hide the blue value of pixels.



Show/Hide Y Values

Show or hide the y value of pixels.



Show/Hide U Values

Show or hide the u value of pixels.



Show/Hide V Values

Show or hide the v value of pixels.

Focus Cross



Display a blue cross on the image, and the pixel value of the cross point will be updated on the Status Panel. Users can move the position of the cross point by clicking anywhere of the image.

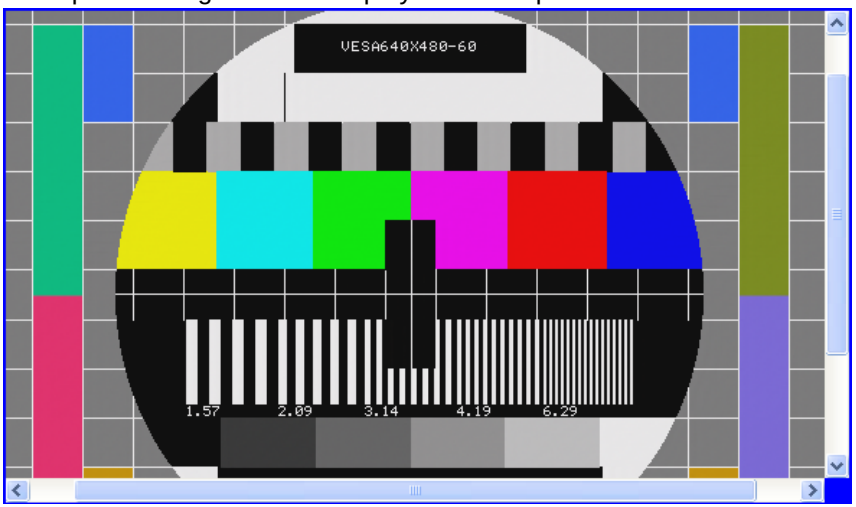
6.2.4 Status Panel



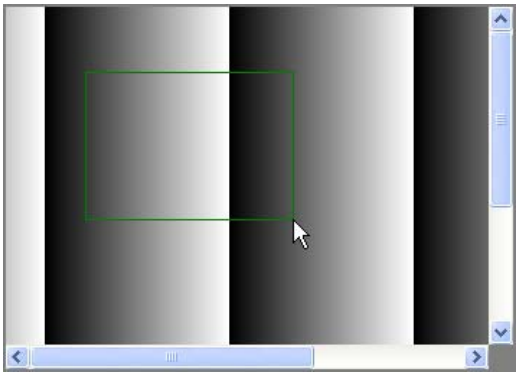
Items from left to right are selected port and channel, cursor position, pixel value, frame rate, total captured frames, and magnification (horizontal ratio, vertical ratio).

6.2.5 Display panel

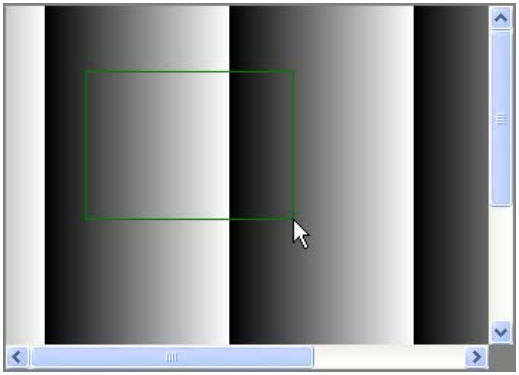
Captured images will be displayed on this panel as below.



Click the left mouse button and then drag the cursor on the image. A green rectangle appears and this region will be magnified to the same size as the display region. If you want to drag the green rectangle in the same proposition of width and height, keep pressing the “Shift” key before dragging.



Click the right mouse button on the image, and the cursor will become a move2D icon. When the size of the image is larger than that of the display panel, users can drag the image when this icon appears.



6.2.6 Main Menu

File menu

Open Image

Open an image file and display it on Display panel.

Save Image

Save current displaying image to a bitmap file.

Exit

Exit ViewCreatorPro.

View menu

Device

Hide or unhide Devices panel.

Adjustment

Hide or unhide Adjustment panel.

Video Format menu

Change the video format here according to the video source. HDV62 support three kinds of video source: RGB (DVI-I), YPbPr (DSUB), and HDMI (DVI-I).

RGB (DVI-I)

VGA 60 fps (640 x 480)

Set the video format of captured images to VGA 60 fps (640 x 480).

SVGA 60 fps (800 x 600)

Set the video format of captured images to SVGA 60 fps (800 x 600).

XGA 60 fps (1024 x 768)

Set the video format of captured images to XGA 60 fps (1024 x 768).

SXGA 60 fps (1280 x 1024)

Set the video format of captured images to SXGA 60 fps (1280 x 1024).

YPbPr (DSUB)

720p 25 fps (1280 x 720 progressive)

Set the video format of captured images to 720p 25 fps (1280 x 720 progressive).

720p 30 fps (1280 x 720 progressive)

Set the video format of captured images to 720p 30 fps (1280 x 720 progressive).

720p 50 fps (1280 x 720 progressive)

Set the video format of captured images to 720p 50 fps (1280 x 720 progressive).

720p 60 fps (1280 x 720 progressive)

Set the video format of captured images to 720p 60 fps (1280 x 720 progressive).

1080i 25 fps (1920 x 1080 interlace)

Set the video format of captured images to 1080i 25 fps (1920 x 1080 interlace).

1080i 30 fps (1920 x 1080 interlace)
Set the video format of captured images to 1080i 30 fps (1920 x 1080 interlace).

1080p 25 fps (1920 x 1080 progressive)
Set the video format of captured images to 1080p 25 fps (1920 x 1080 progressive).

1080p 30 fps (1920 x 1080 progressive)
Set the video format of captured images to 1080p 30 fps (1920 x 1080 progressive).

1080p 50 fps (1920 x 1080 progressive)
Set the video format of captured images to 1080p 50 fps (1920 x 1080 progressive).

1080p 60 fps (1920 x 1080 progressive)
Set the video format of captured images to 1080p 60 fps (1920 x 1080 progressive).

HDMI (DVI-I)

720p 50 fps (1280 x 720 progressive) YCRCB IN
Set the video format of captured images to 720p 50 fps (1280 x 720 progressive).

720p 60 fps (1280 x 720 progressive) YCRCB IN
Set the video format of captured images to 720p 60 fps (1280 x 720 progressive).

1080i 25 fps (1920 x 1080 interlace) YCRCB IN
Set the video format of captured images to 1080i 25 fps (1920 x 1080 interlace).

1080i 30 fps (1920 x 1080 interlace) YCRCB IN
Set the video format of captured images to 1080i 30 fps (1920 x 1080 interlace).

1080p 25 fps (1920 x 1080 progressive) YCRCB IN
Set the video format of captured images to 1080p 25 fps (1920 x 1080 progressive).

1080p 30 fps (1920 x 1080 progressive) YCRCB IN
Set the video format of captured images to 1080p 30 fps (1920 x 1080 progressive).

1080p 50 fps (1920 x 1080 progressive) YCRCB IN
Set the video format of captured images to 1080p 50 fps (1920 x 1080 progressive).

1080p 60 fps (1920 x 1080 progressive) YCRCB IN
Set the video format of captured images to 1080p 60 fps (1920 x 1080 progressive).

VGA 60 fps (640 x 480)
Set the video format of captured images to VGA 60 fps (640 x 480).

SVGA 60 fps (800 x 600)
Set the video format of captured images to SVGA 60 fps (800 x 600).

XGA 60 fps (1024 x 768)
Set the video format of captured images to XGA 60 fps (1024 x 768).

SXGA 60 fps (1280 x 1024)

Set the video format of captured images to SXGA 60 fps (1280 x 1024).

UXGA 60 fps (1600 x 1200)
Set the video format of captured images to UXGA 60 fps (1600 x 1200).

720p 50 fps (1280 x 720 progressive) RGB IN
Set the video format of captured images to 720p 50 fps (1280 x 720 progressive).

720p 60 fps (1280 x 720 progressive) RGB IN
Set the video format of captured images to 720p 60 fps (1280 x 720 progressive).

1080i 25 fps (1920 x 1080 interlace) RGB IN
Set the video format of captured images to 1080i 25 fps (1920 x 1080 interlace).

1080i 30 fps (1920 x 1080 interlace) RGB IN
Set the video format of captured images to 1080i 30 fps (1920 x 1080 interlace).

1080p 25 fps (1920 x 1080 progressive) RGB IN
Set the video format of captured images to 1080p 25 fps (1920 x 1080 progressive).

1080p 30 fps (1920 x 1080 progressive) RGB IN
Set the video format of captured images to 1080p 30 fps (1920 x 1080 progressive).

1080p 50 fps (1920 x 1080 progressive) RGB IN
Set the video format of captured images to 1080p 50 fps (1920 x 1080 progressive).

1080p 60 fps (1920 x 1080 progressive) RGB IN
Set the video format of captured images to 1080p 60 fps (1920 x 1080 progressive).

Color Format menu

Gray

Set the color format of captured images to gray.

RGB32

Set the color format of captured images to RGB32.

RGB24

Set the color format of captured images to RGB24.

BGR30

Set the color format of captured images to BGR30.

YUY2

Set the color format of captured images to YUY2.

YUV8

Set the color format of captured images to YUV8.

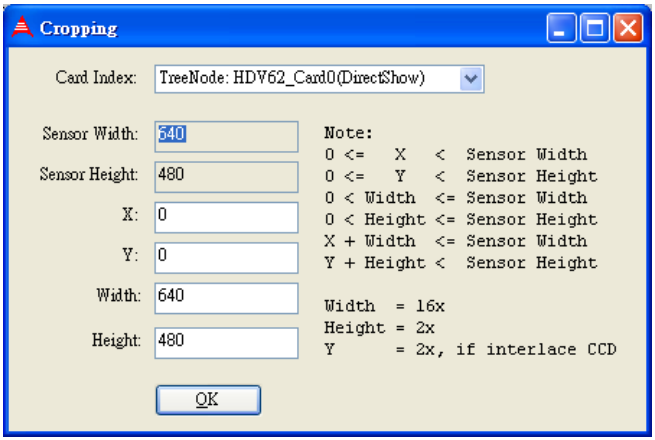
YU10

Set the color format of captured images to YU10.

Image Size menu

Cropping

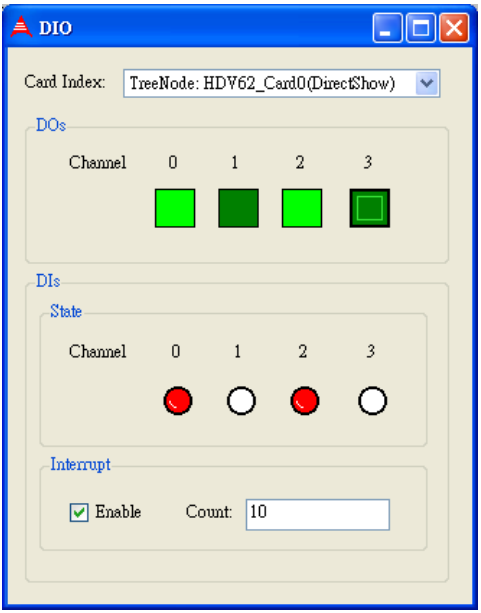
Before using cropping, a correct card index must be selected first. Sensor Width and Sensor Height are the width and height of an original image, and these two values vary with your selected video format automatically. Before cropping an image, four parameters: X, Y, Width, Height must be properly set. X and Y represent the coordinates of the start position (upper left corner)?Width and Height stand for the width and height after cropping. Click the OK button to apply change, and then click the Continue Grab icon on the Tool panel to get cropped images.



Tool menu

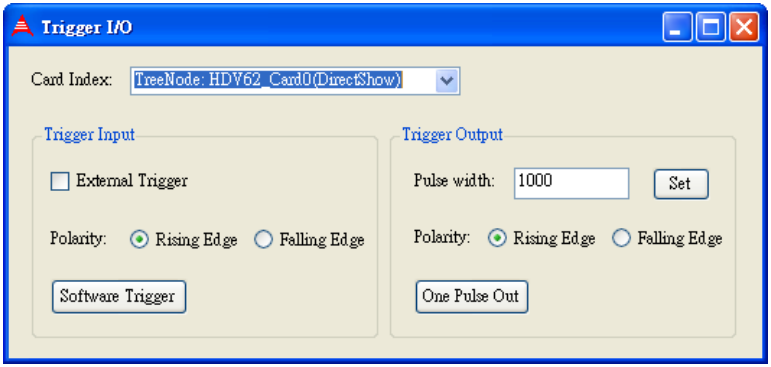
DIO

Before using digital I/O, a correct card index must be selected first. There are four rectangles and four circles respectively representing four channels DO and four channels DI. Click on the rectangle to output a digital signal, and the signal level can be identified by the rectangle color (Lime green: High, Green: Low). The status of DI can be read from the circle color in a similar way as DO (Red: High, White: Low). Besides, if the interrupt is enabled, the interrupt count will be increased while the change of state (COS) of DI occurs.



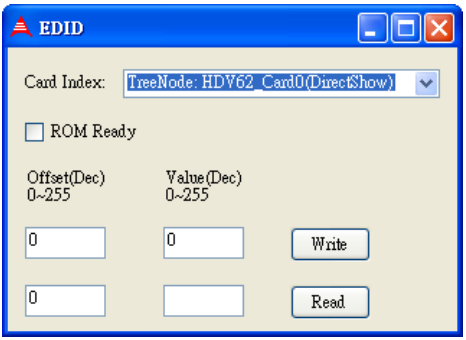
Trigger I/O

Before using trigger I/O, a correct card index must be selected first. When External Trigger is checked, users can set the trigger input polarity, and then click the Software Trigger button to generate a trigger pulse, or just wait for an external trigger input. Besides trigger input, HDV62 can generate an output trigger by clicking the One Pulse Out button. Users can also set the polarity and pulse width for the output trigger.



EDID

Before using EDID, a correct card index must be selected first. Enter the offset and value, and then click the Write or Read button to write to or read from EDID ROM. Valid offset and input value range from 0 to 255, and the values in EDID ROM will not be erased when the system is powered off.



Help menu

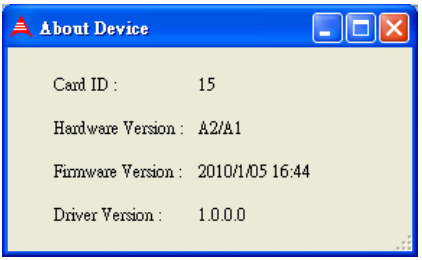
About ViewCreatorPro

List the version of ViewCreatorPro.



About Device

List the card id, hardware version, firmware version, and driver version of the HDV62.



Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP