AISYS ALTAIR Series Framegrabber

Version of Document and OVK Framework : 20060227

Table of Contents

PART I

AISYS ALTAIR Series Framegrabber Hardware Installation

Chapter 1 Hardware Installation

1-1 Procedures

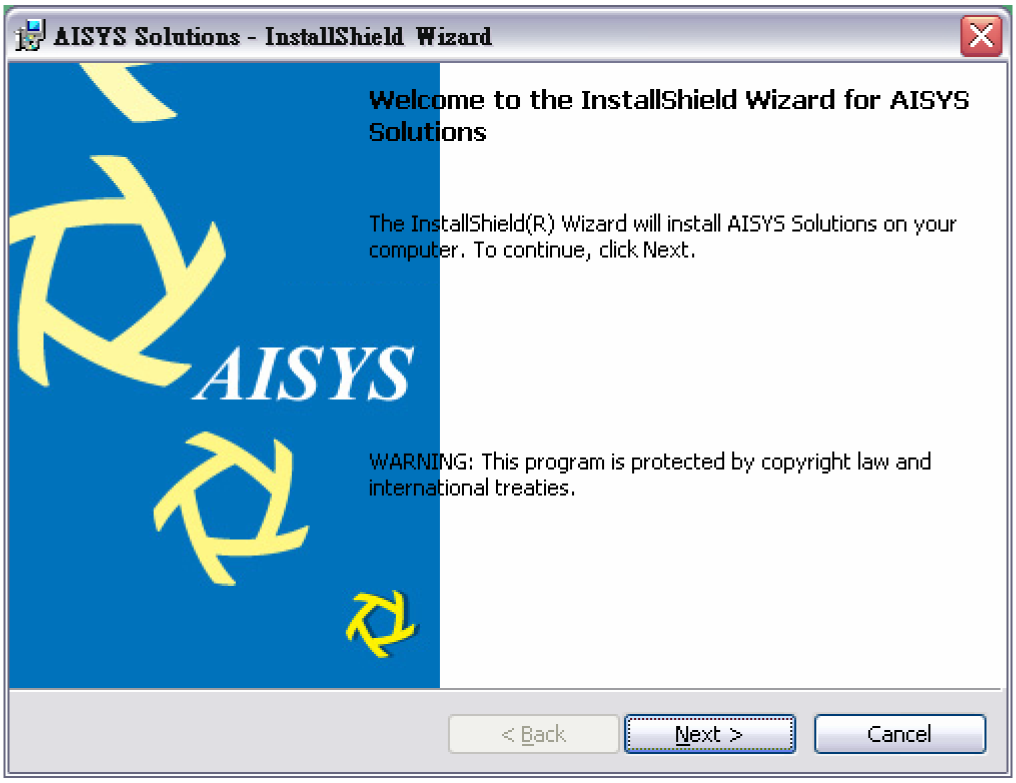
1. Turn on the power of computer, if autorun of CD in not enabled, execute the **AISYSInstallShell.exe** in the CD, choose AISYS OVK Framework Lite, follow the instructions the setup suite prompts and finish the installation procedures.
2. Switch-off computer and all peripheral devices connected to it (monitor, printer …etc).
3. Discharge any electricity that could be accumulated on your body. You can achieve this by touching with bare hand and unpainted metal part of the enclosure of your computer. Make sure that the computer is linked to the AC mains outlet with proper earth connection.
4. Disconnect all cables from your computer, including AC power.
5. Open the computer enclosure to gain access to the PCI slots according to the manufacturer instructions.
6. Locate an available PCI slot and remove the blank bracket associated to its location.
7. Unwrap the AISYS ALTAIR, take the board and hold it carefully. Avoid any contact of the board with unnecessary items including your clothes.
8. Gently insert the board in the targeted PCI slot, taking care of pushing it fully down. If you experience some resistance, remove the board and repeat the operation keeping caution for a perfect mechanical alignment of board relative to slot. Ensure the lower part of the bracket is inserting into the corresponding enclosure fastening.
9. Secure the board with the screw.
10. Close the computer enclosure according to the manufacturer’s instructions

PART II
AISYS ALTAIR Series Framegrabber Software and Device Driver Installation

Chapter 1 Software Installation

1-1 Procedures

1. Insert **"AISYS Solution CD"** and follow the instructions the auto-run application prompts or run the setup utility **"setup.exe"** directly in root directory in the CD to start the installation procedures as shown in Figure 1.



AISYS Solutions - InstallShield Wizard

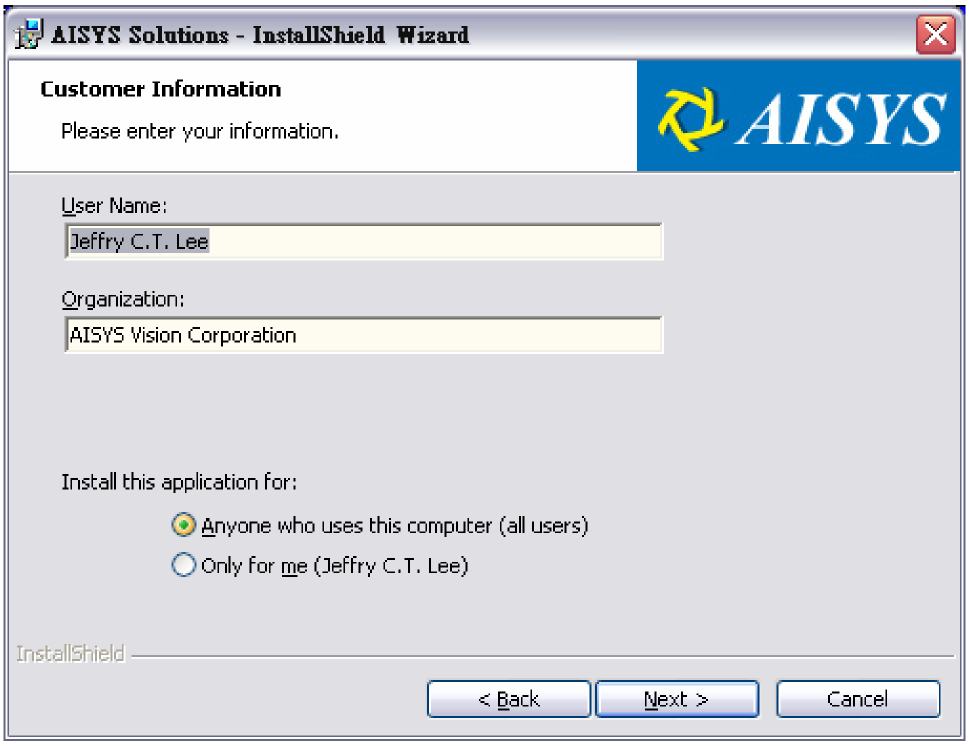
Welcome to the InstallShield Wizard for AISYS Solutions

The InstallShield(R) Wizard will install AISYS Solutions on your computer. To continue, click Next.

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

Figure 1.

2. Enter informations of user and organization and press the button “Next” to continue the procedures as shown in Figure 2.



AISYS Solutions - InstallShield Wizard

Customer Information

Please enter your information.

User Name: Jeffry C.T. Lee

Organization: AISYS Vision Corporation

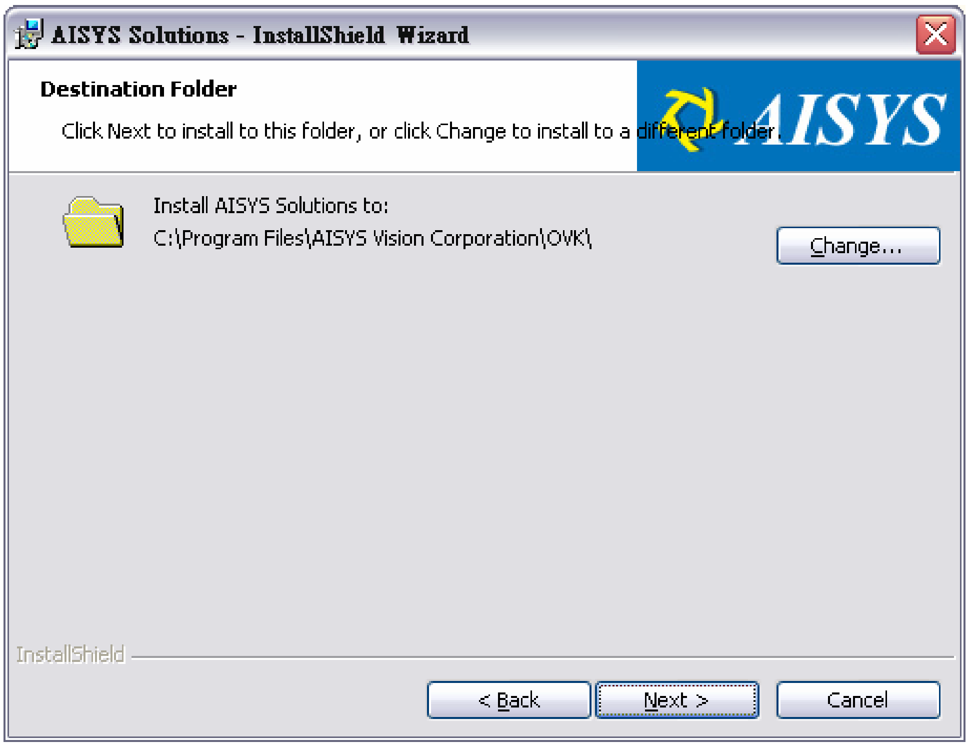
Install this application for:

[✓ Anyone who uses this computer (all users)]

Only for me (Jeffry C.T. Lee)

Figure 2.

3. Change the install destination of necessary and press the button “Next” to continue the procedures as shown in Figure 3.



AISYS Solutions - InstallShield Wizard

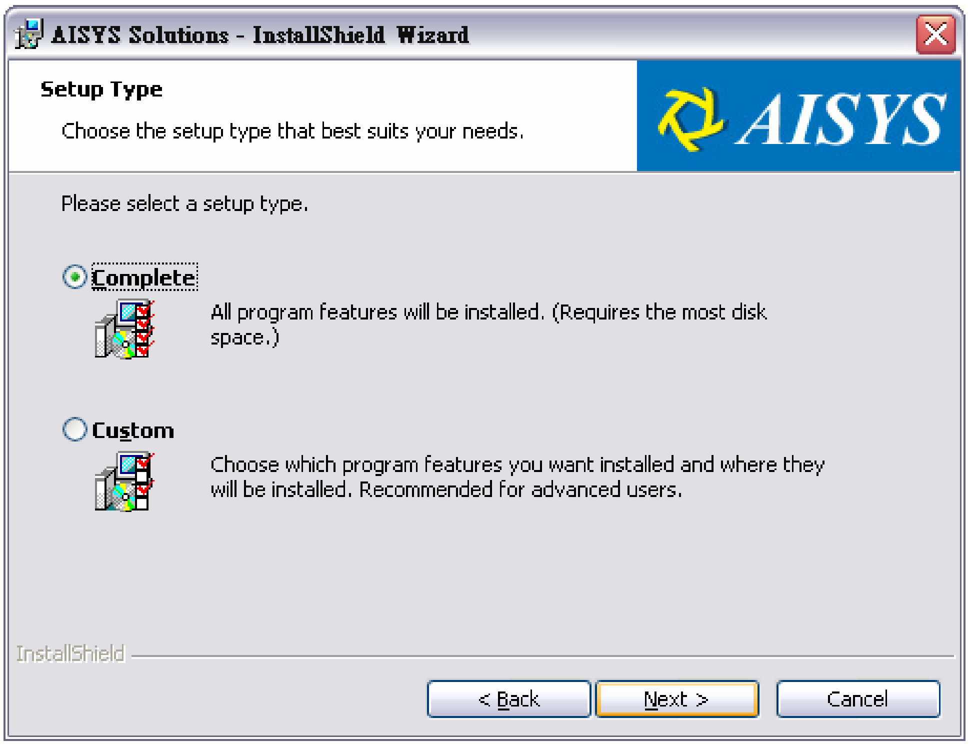
Destination Folder

Click Next to install to this folder, or click Change to install to a different folder.

Install AISYS Solutions to:
C:\Program Files\AISYS Vision Corporation\OVK\

Figure 3.

4. Choose the setuo type and press the button “Next” to continue the procedures as shown in Figure 4.


AISYS Solutions - InstallShield Wizard

Setup Type

Choose the setup type that best suits your needs.

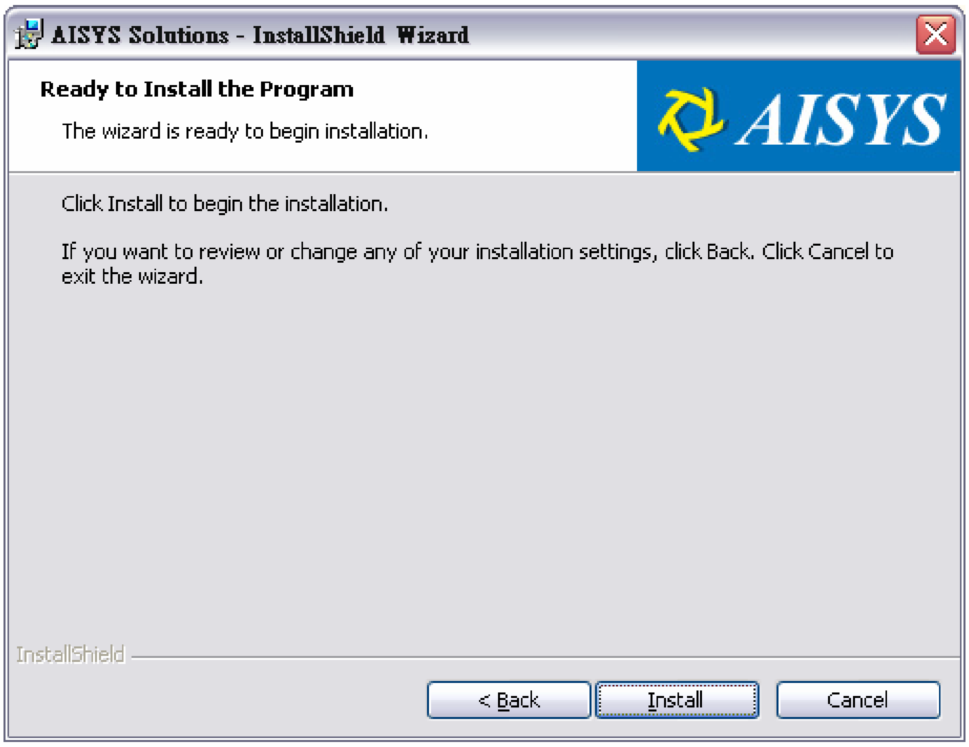
Please select a setup type.

(●) Complete
All program features will be installed. (Requires the most disk space.)

( ) Custom
Choose which program features you want installed and where they will be installed. Recommended for advanced users.

Figure 4.

5. Press the button “Install” to start the installation as shown in Figure 5.


AISYS Solutions - InstallShield Wizard

Ready to Install the Program

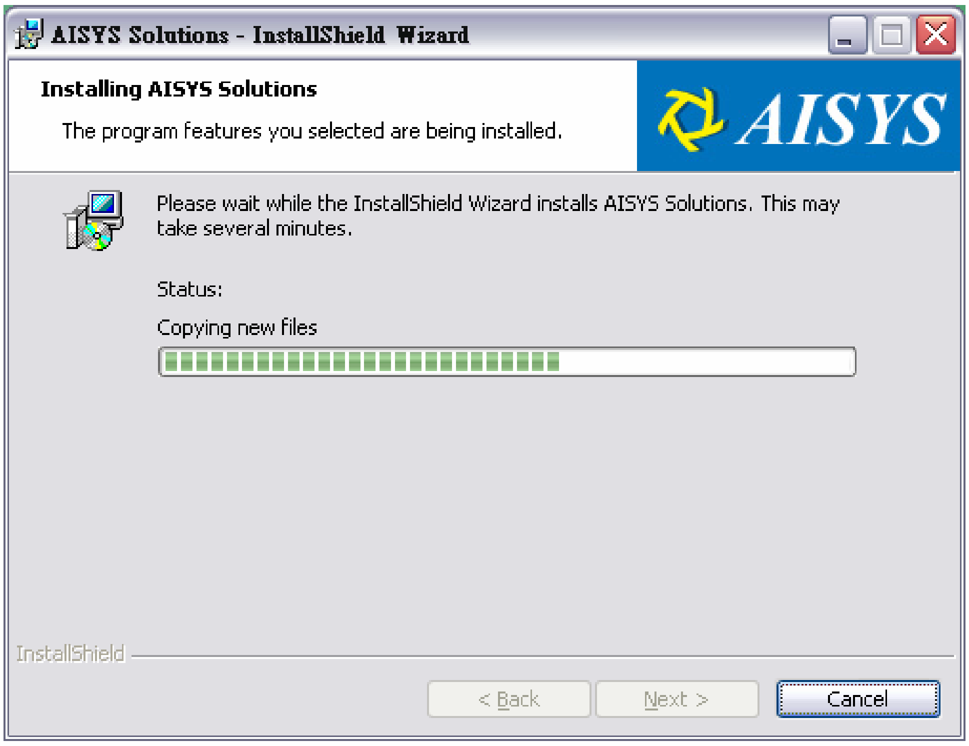
The wizard is ready to begin installation.

Click Install to begin the installation.

If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

Figure 5.

6. Wait for the installation process finished as shown in Figure 6.



AISYS Solutions - InstallShield Wizard

Installing AISYS Solutions

The program features you selected are being installed.

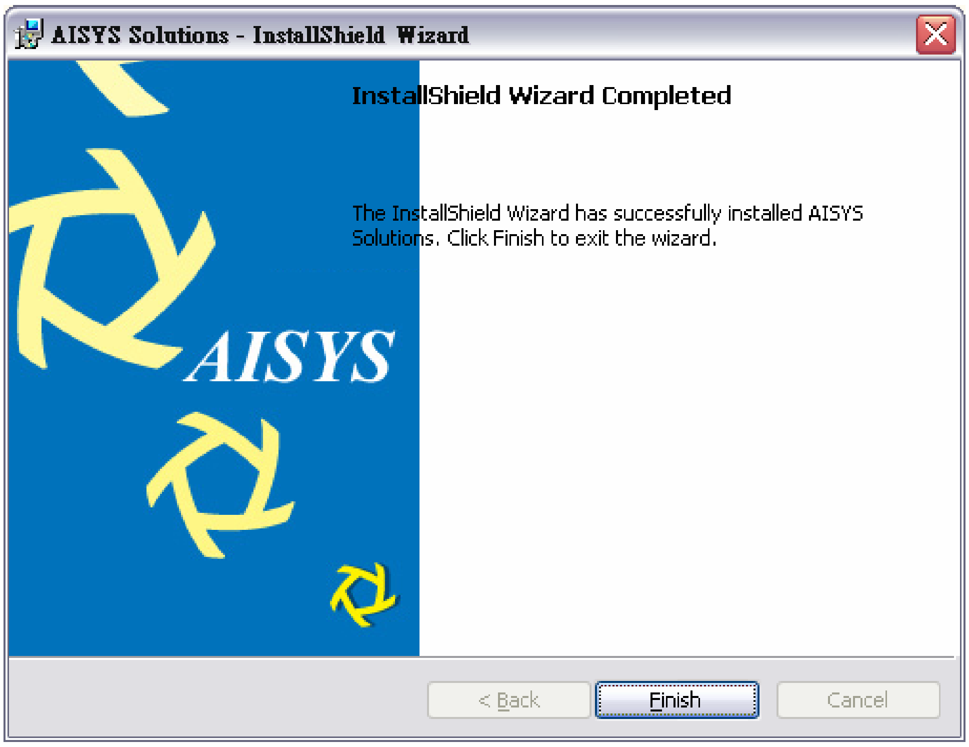
Please wait while the InstallShield Wizard installs AISYS Solutions. This may take several minutes.

Status:

Copying new files

Figure 6.

7. Press the button "Finish" to finish the installation as shown in Figure 7.



AISYS Solutions - InstallShield Wizard

InstallShield Wizard Completed

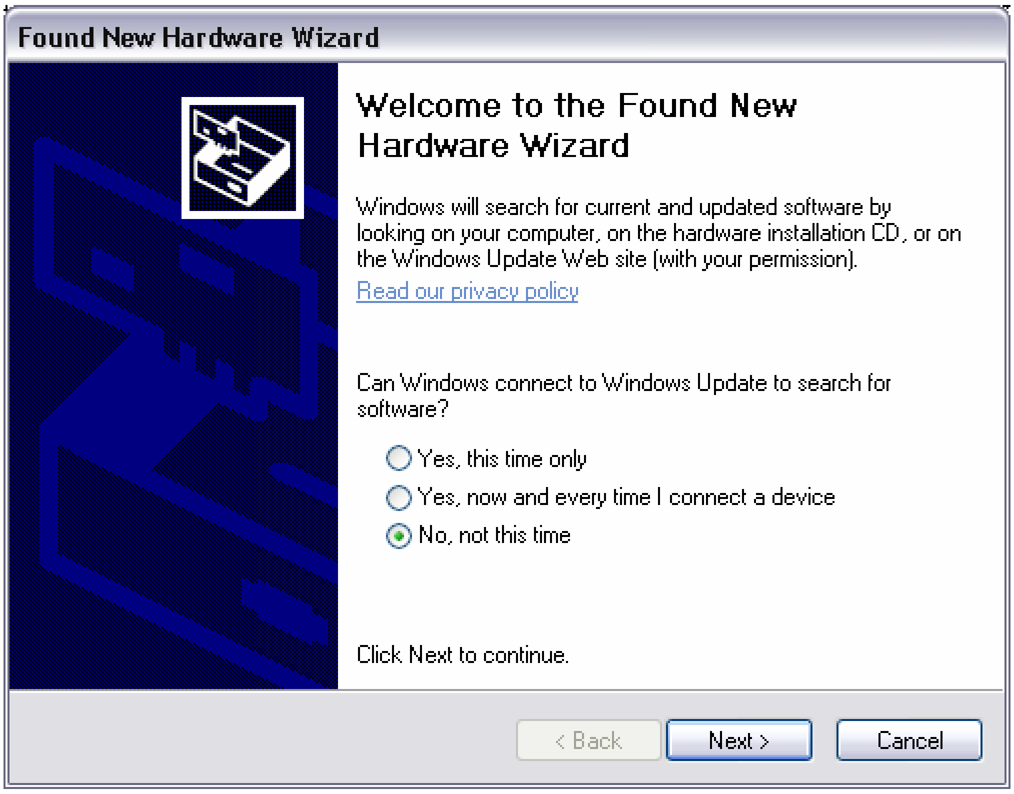
The InstallShield Wizard has successfully installed AISYS Solutions. Click Finish to exit the wizard.

Figure 7.

Chapter 2 Device Driver Installation

2-1 Procedures

1. Turn on the power of computer again, and you will see the dialog from PNP manager of Windows as shown in Figure 1.



Found New Hardware Wizard

Welcome to the Found New Hardware Wizard

Windows will search for current and updated software by looking on your computer, on the hardware installation CD, or on the Windows Update Web site (with your permission).

Read our privacy policy

Can Windows connect to Windows Update to search for software?

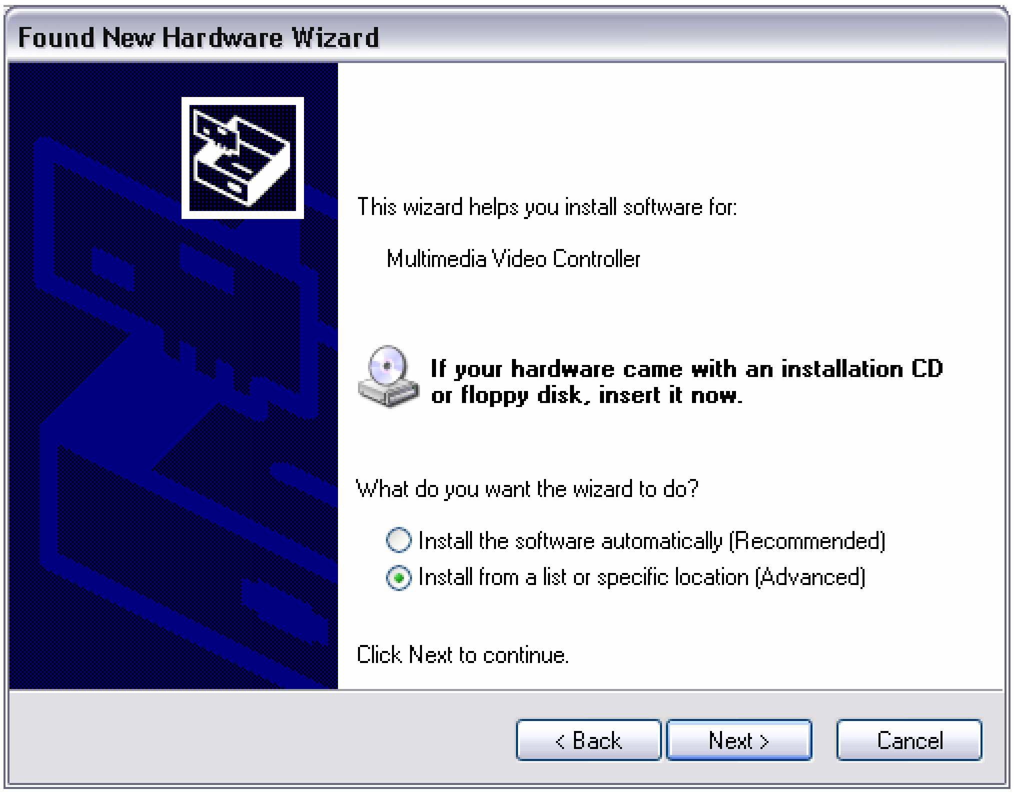
- Yes, this time only
- Yes, now and every time I connect a device
- [✓ No, not this time]

Click Next to continue.

Figure 1.

2. Check the option and press “Next” button as shown in Figure 1.

3. Check the option and press “Next” button to install driver for “AISYS ALTAIR” as shown in Figure 2.


Found New Hardware Wizard

This wizard helps you install software for:

Multimedia Video Controller

If your hardware came with an installation CD or floppy disk, insert it now.

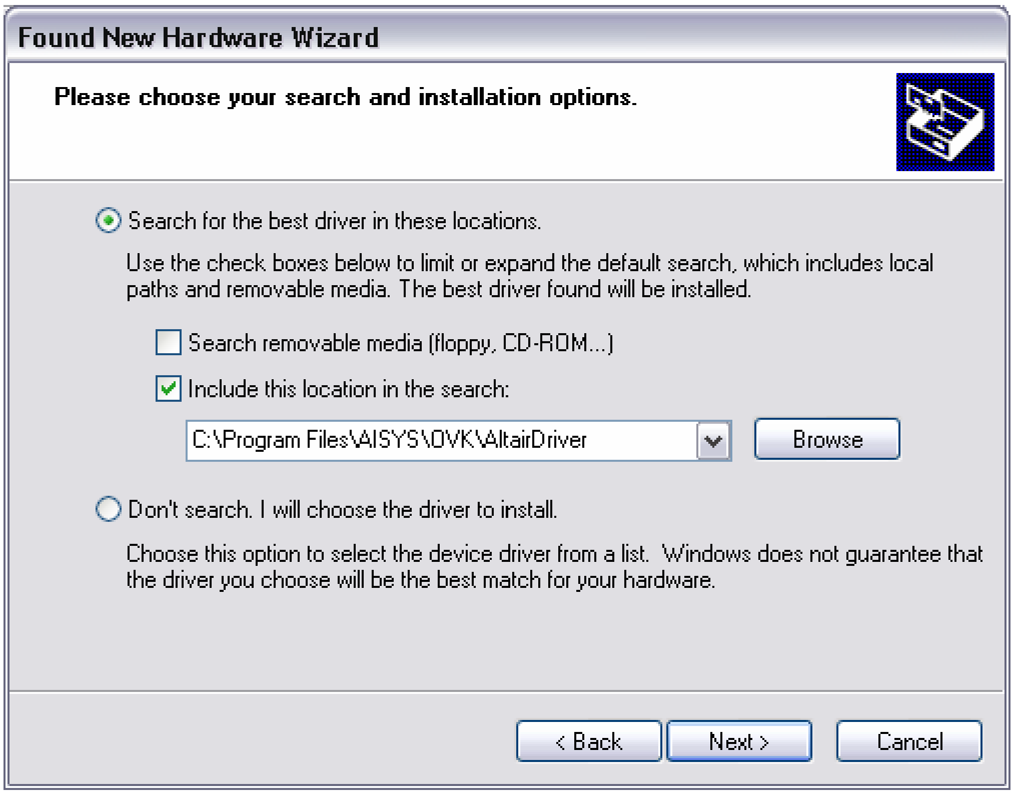
What do you want the wizard to do?

- ○ Install the software automatically (Recommended)
- [✓ Install from a list or specific location (Advanced)]

Click Next to continue.

Figure 2.

4. Check the option and press "Browse" button to introduce Windows the location of ALTAIR driver files as shown in Figure 3 and Figure 4.


Found New Hardware Wizard

Please choose your search and installation options.

[✓ Search for the best driver in these locations.]

Use the check boxes below to limit or expand the default search, which includes local paths and removable media. The best driver found will be installed.

- ☐ Search removable media (floppy, CD-ROM...)
- ☑ Include this location in the search:

C:\Program Files\AISYS\OVK\AltairDriver

○ Don't search. I will choose the driver to install.

Choose this option to select the device driver from a list. Windows does not guarantee that the driver you choose will be the best match for your hardware.

Figure 3.

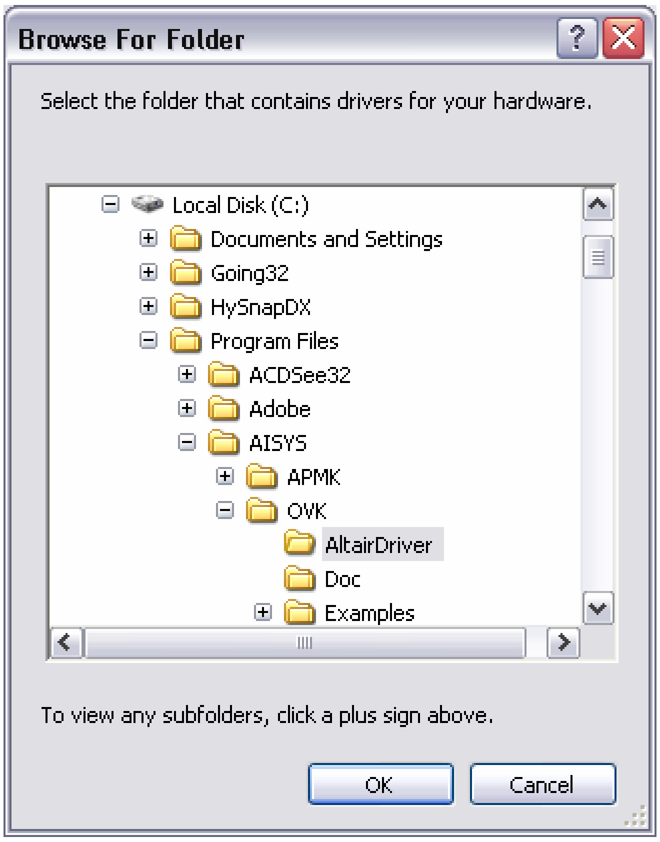
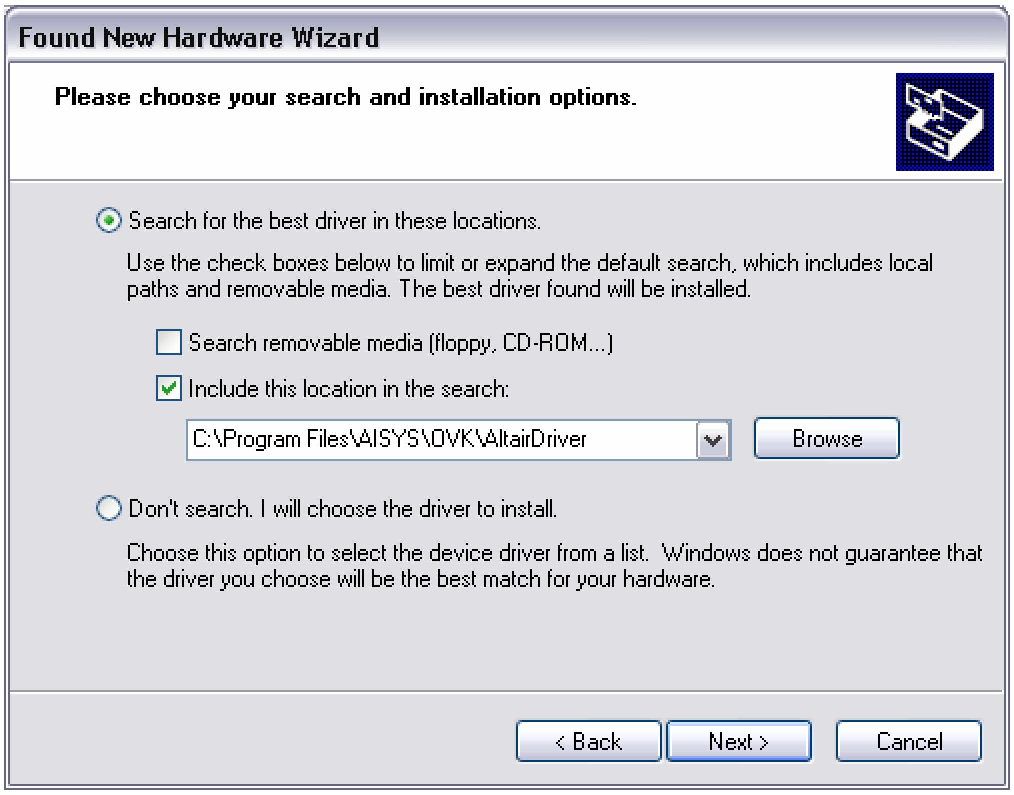


Figure 4.

5. Press “Next” button to install ALTAIR driver files as shown in Figure 5.
6. Press “Continue Anyway” button to install ALTAIR driver files as shown in Figure 6. Here, AISYS ALTAIR is installed complete.
7. Because another dummy driver for “AISYS Component” need to be installed, a new dialog from PNP manager of Windows will show again as shown in Figure 1.
8. Follow the instructions mensioned above (from procedure 13 ~ procedure 16) to install the dummy driver of “AISYS Component” to complete the in whole procedures of driver installation. Now you can see two device records “AISYS ALTAIR” and “AISYS Component” in device manager as shown in Figure 7. Here, AISYS ALTAIR is installed successfully.


Found New Hardware Wizard

Please choose your search and installation options.

[✓ Search for the best driver in these locations.]

Use the check boxes below to limit or expand the default search, which includes local paths and removable media. The best driver found will be installed.

☐ Search removable media (floppy, CD-ROM...)

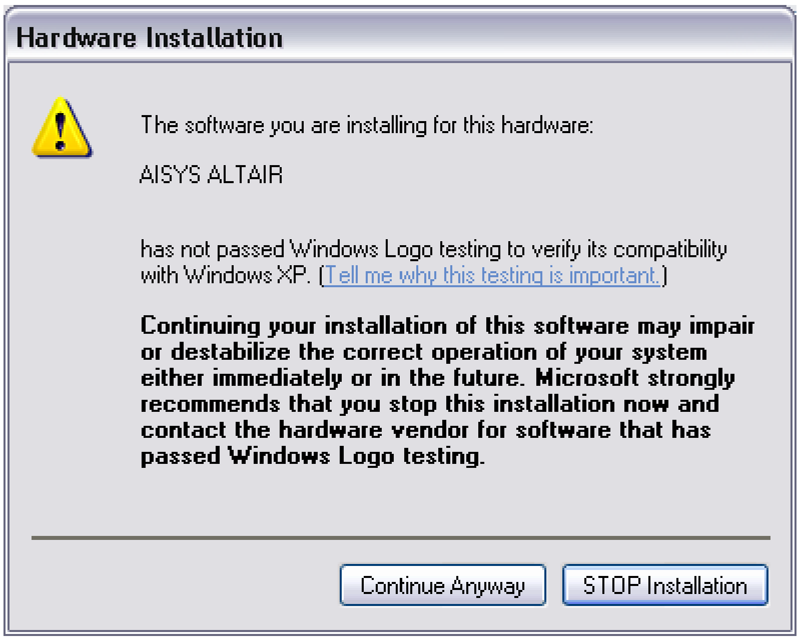
☑ Include this location in the search:

C:\Program Files\AISYS\OVK\AltairDriver

○ Don't search. I will choose the driver to install.

Choose this option to select the device driver from a list. Windows does not guarantee that the driver you choose will be the best match for your hardware.

Figure 5.


Hardware Installation

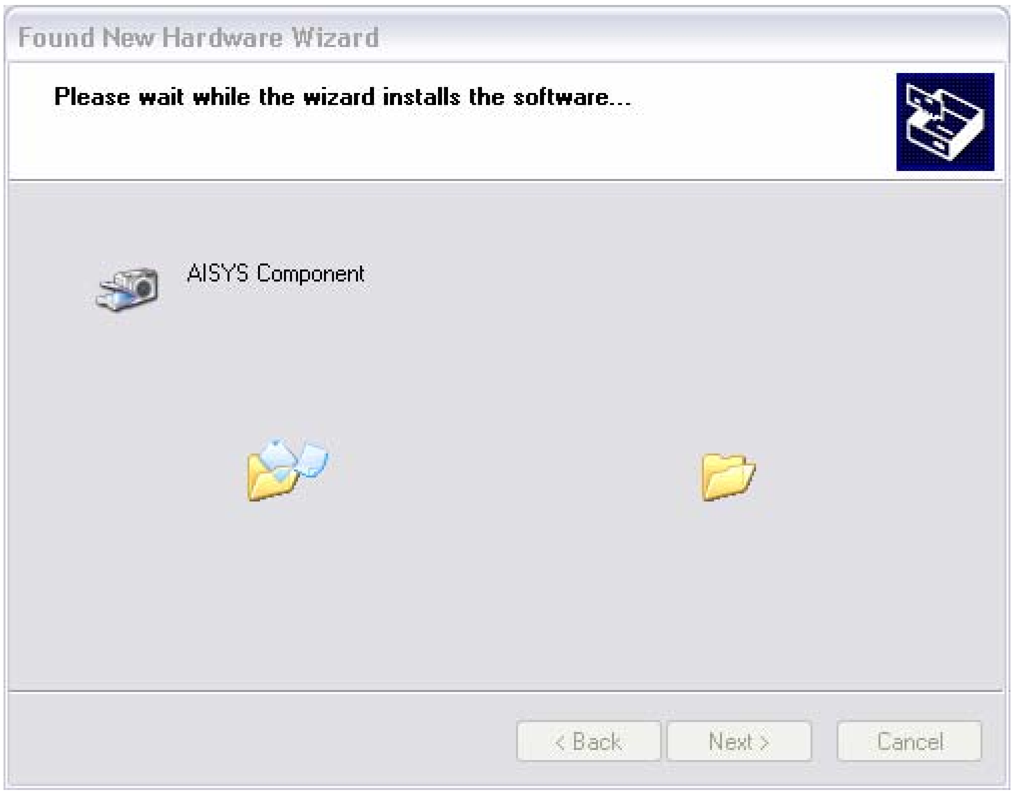
The software you are installing for this hardware:

AISYS ALTAIR

has not passed Windows Logo testing to verify its compatibility with Windows XP. (Tell me why this testing is important.)

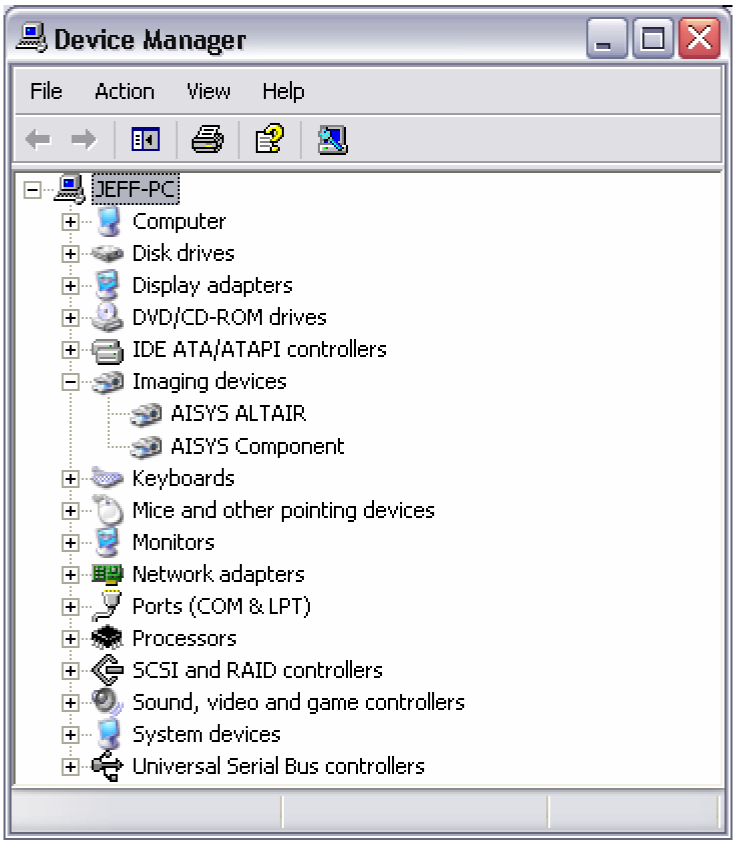
Continuing your installation of this software may impair or destabilize the correct operation of your system either immediately or in the future. Microsoft strongly recommends that you stop this installation now and contact the hardware vendor for software that has passed Windows Logo testing.

Figure 6.


Found New Hardware Wizard

Please wait while the wizard installs the software...

AISYS Component


Device Manager

```
JEFF-PC
  Computer
  Disk drives
  Display adapters
  DVD/CD-ROM drives
  IDE ATA/ATAPI controllers
  Imaging devices
    AISYS ALTAIR
    AISYS Component
  Keyboards
  Mice and other pointing devices
  Monitors
  Network adapters
  Ports (COM & LPT)
  Processors
  SCSI and RAID controllers
  Sound, video and game controllers
  System devices
  Universal Serial Bus controllers
```

Figure 7.

PART III

AISYS ALTAIR Series Framegrabber Programming Manual

Chapter 1 Introduction

1-1 *AISYS* Vision Framework：The Rapid Prototyping and Developing SDK

AISYS provides an easiest way called **“Vision Framework”** to access all the hardware products designed by ***AISYS***. A vision framework, which is a **patented** software framework, helps users to develop vision system easily and rapidly. Thanks to ActiveX architechture, vision framework dramatically reduces the most parts of efforts that users should contribute that adopt traditional SDK. Even with no programming codes at all, users can view the live image **in 1 minute** with the help of the vision framework. By the way, vision framework is free of charge. See the later sections for the details of software development.

1-2 Minimum System Requirement

To make ALTAIR series framegrabber operates smoothly, minimum system capability is required. AISYS suggests users to have the following minimum system requirement to connect ALTAIR series gramegrabbers :

CPU	P4 Celeron 1.6G or above.
DRAM	DDR266 256MB or above.
HD Space	30MB for installing OVK Framework Lite.
VGA Interface	Independent VGA interface card with 8MB RAM or above is strongly suggested.
Operating System	Microsoft Windows 2000 SP4 / Microsoft Windows XP SP1 or later.
VGA Resolution	800x600@16bpp or above ; 1024x768@24bpp is suggested for the best UI appearance.

Table 1.

Chapter 2 Integration with Vision Framework

Vision framework provides the most convenient and versatile way to integrate AISYS hardware products with users' vision system. Briefly speaking, vision framework provides a solution to integrate vision system in a manner **"Pick、Place and Play"**. Let us introduce a **"Quick Start"** to demonstrate how to get a breakthrough to a vision system powered by vision framework. Before we start the introduction, users should follow the instructions mensioned in part I、II and be sure that an ALTAIR series framegrabber is connected to your computer properly.

2-1 Quick Start

2-1-1 Pick

1. Start Microsoft Visual Basic version 6.0 or Visual C++ 6.0 compiler as shown in Figure 1. (Upgrading Visual Basic or Visual C++ to service pack 5 or later is suggested). Here, we use Microsoft Visual Basic 6.0 as the reference platform.

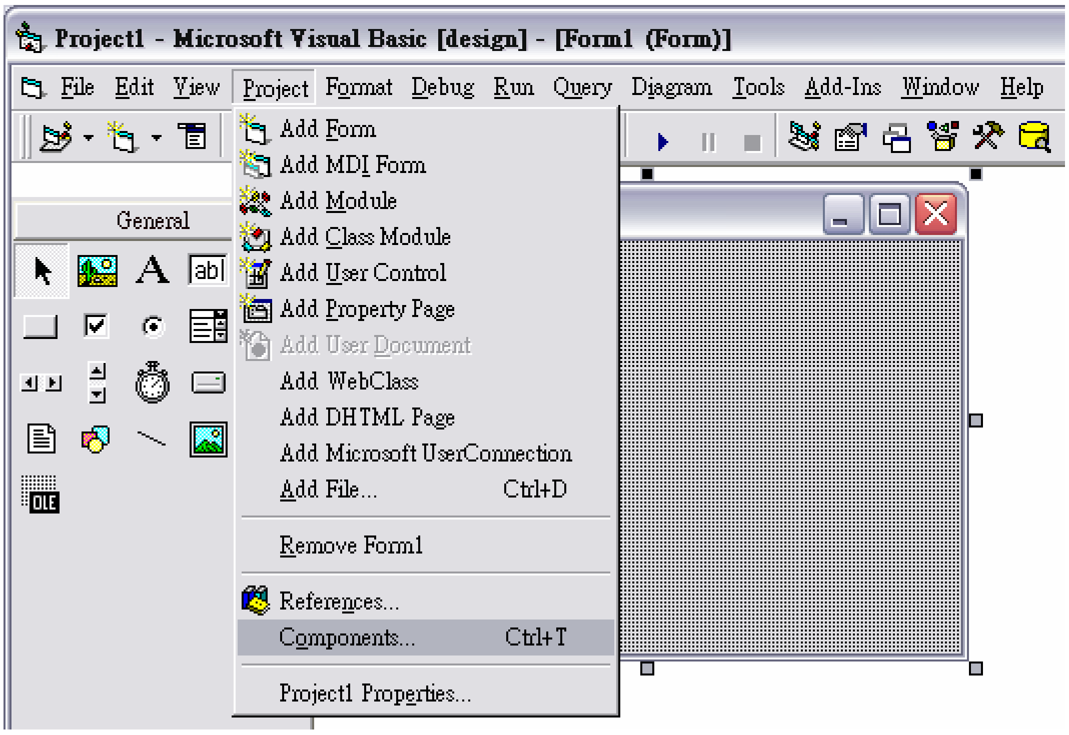
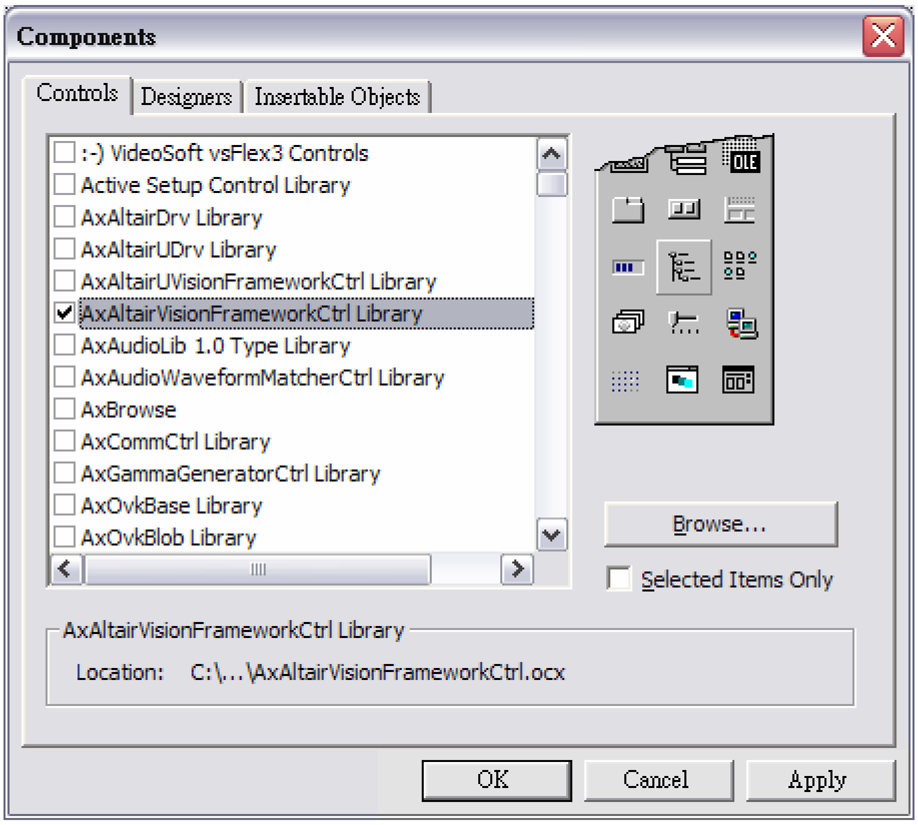


Figure 1

2. Choose menu **Project** → **Components** → **Figure 1** → **Check AxAltairVisionFrameworkCtrl Library** → **OK Button** as shown in Figure 2.



Components

Controls | Designers | Insertable Objects

- [ ] :-) VideoSoft vsFlex3 Controls
- [ ] Active Setup Control Library
- [ ] AxAltairDrv Library
- [ ] AxAltairUDrv Library
- [ ] AxAltairUVisionFrameworkCtrl Library
- [x] AxAltairVisionFrameworkCtrl Library
- [ ] AxAudioLib 1.0 Type Library
- [ ] AxAudioWaveformMatcherCtrl Library
- [ ] AxBrowse
- [ ] AxCommCtrl Library
- [ ] AxGammaGeneratorCtrl Library
- [ ] AxOvkBase Library
- [ ] AxOvkBlob Library

Browse...

- [ ] Selected Items Only

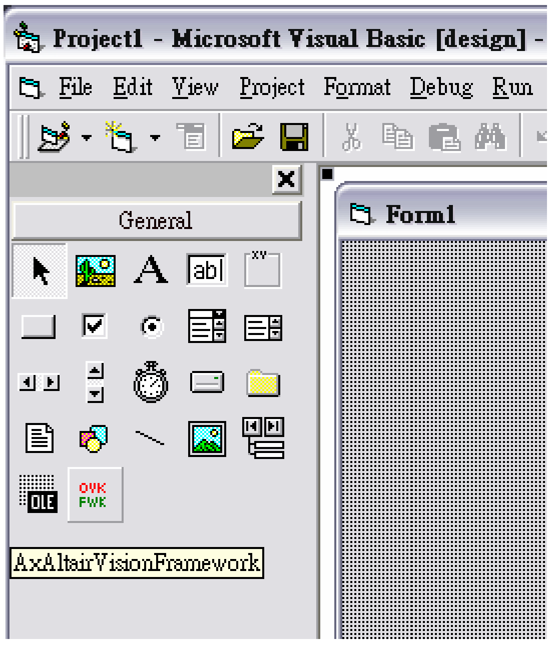
AxAltairVisionFrameworkCtrl Library

Location: C:\...\AxAltairVisionFrameworkCtrl.ocx

OK | Cancel | Apply

Figure 2

3. You can see the components in toolbox after finishing above procedures that names "**AxAltairVisionFramework**", as shown in Figure 3.



Project1 - Microsoft Visual Basic [design] -

File Edit View Project Format Debug Run

General

AxAltairVisionFramework

Form1

Figure 3

4. Pick the component from the palette as shown in Figure 4.

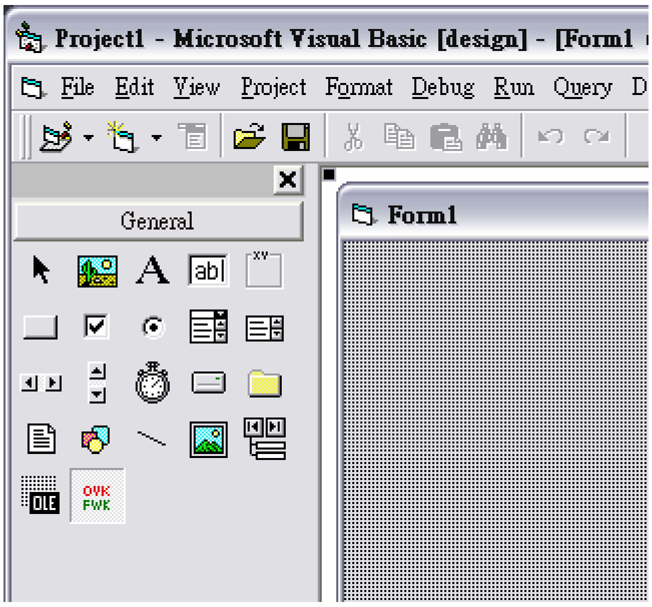

Project1 - Microsoft Visual Basic [design] - [Form1
File Edit View Project Format Debug Run Query D
General
Form1
OVK FWK

Figure 4.

2-1-2 Place

Place the component onto the main form and drag it to adjust the dimension as shown in Figure 5.

2-1-3 Play

1. Here, an integration between ALTAIR framegrabber and vision system is finished.
2. Press the button “Start” of compiler to start the vision system and then press the button “Acquisition Hardware Setup” to show the control panel of hardware as shown Figure 5 and Figure 6.

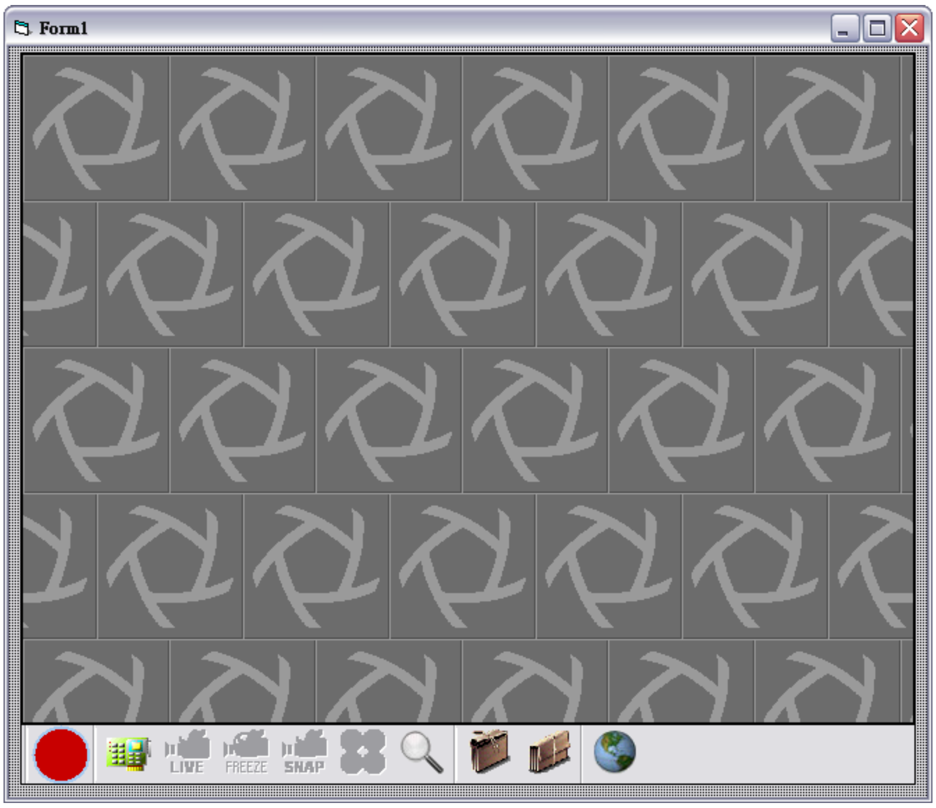
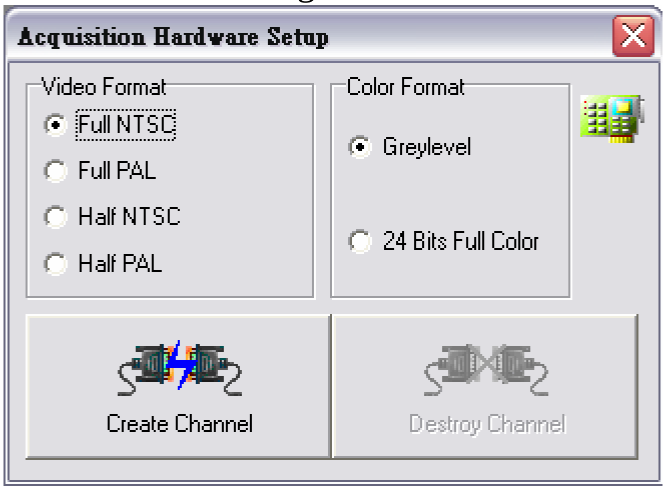


Figure 5.

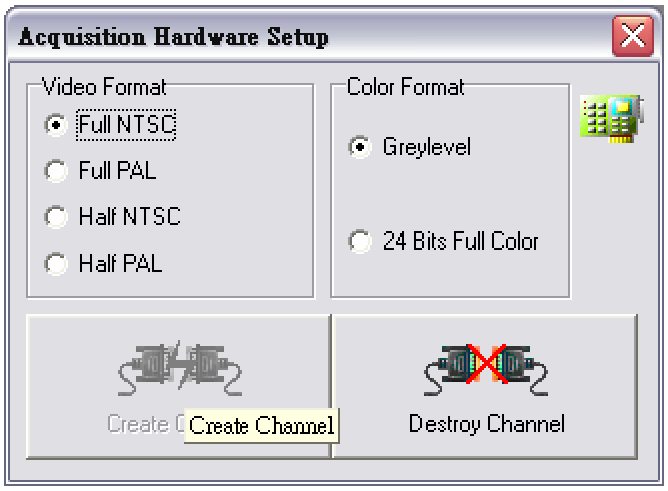


Video Format: [✓ Full NTSC] Full PAL Half NTSC Half PAL

Color Format: [✓ Greylevel] 24 Bits Full Color

Figure 6.

3. Select video and color format by clicking the appropriate radio buttons and then press the button “Create Channel” to initialize the framegrabber as shown in Figure 7.



Acquisition Hardware Setup

Video Format: [✓ Full NTSC] Full PAL Half NTSC Half PAL

Color Format: [✓ Greylevel] 24 Bits Full Color

Figure 7.

4. Press the button “LIVE” to make the framegrabber camera in “Live Image” mode as shown in Figure 8.

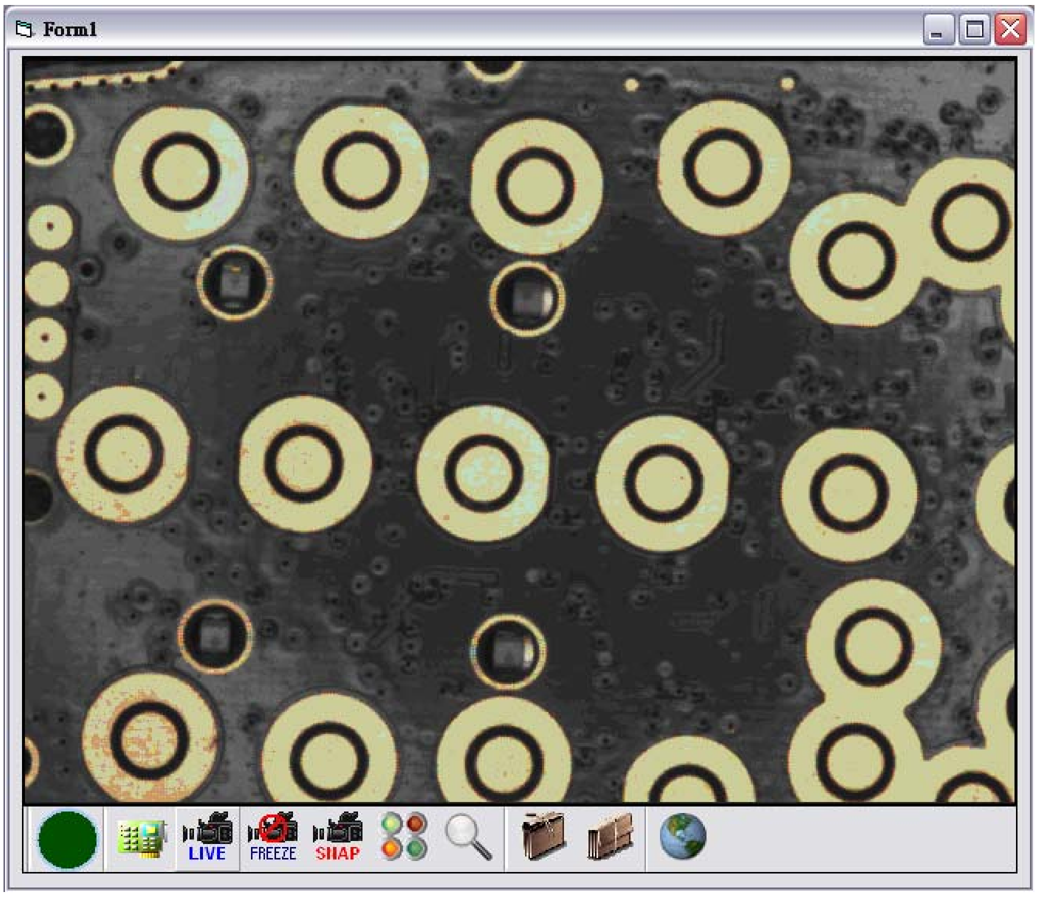


Figure 8.

2-2 Hide The Toolbar

It is the simplest case that users integrate ALTAIR framegrabber with their vision system even with no programming codes. In fact, users need to customize vision framework by configuring associated properties / methods / events. See the chapter 2~5 for more details about customizing vision framework. For example, by configuring the property “ShowToolBar” of vision framework as “FALSE”, vision framework hide the toolbar and shows only the canvas region as shown in Figure 9. Of course, the functions provided by toolbar can also be accessed by properties / methods / events.

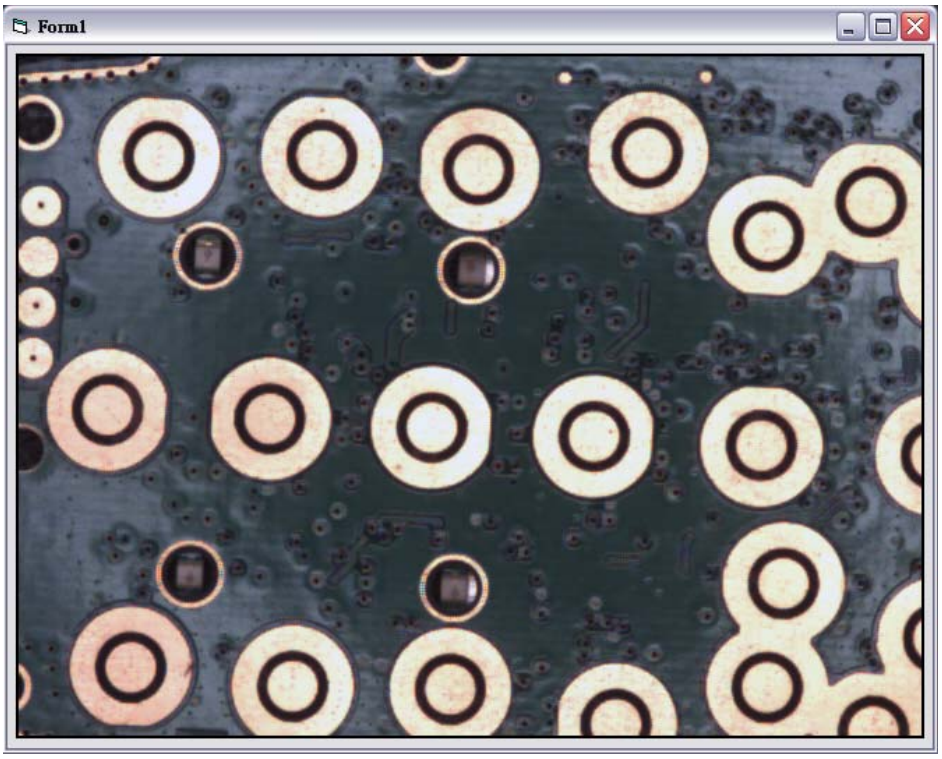


Figure 9.

Chapter 3 User Interface of Vision Framework

Pick and place **AxAltairVisionFramework** component onto form, the functions of the user interface are shown in Figure 10.

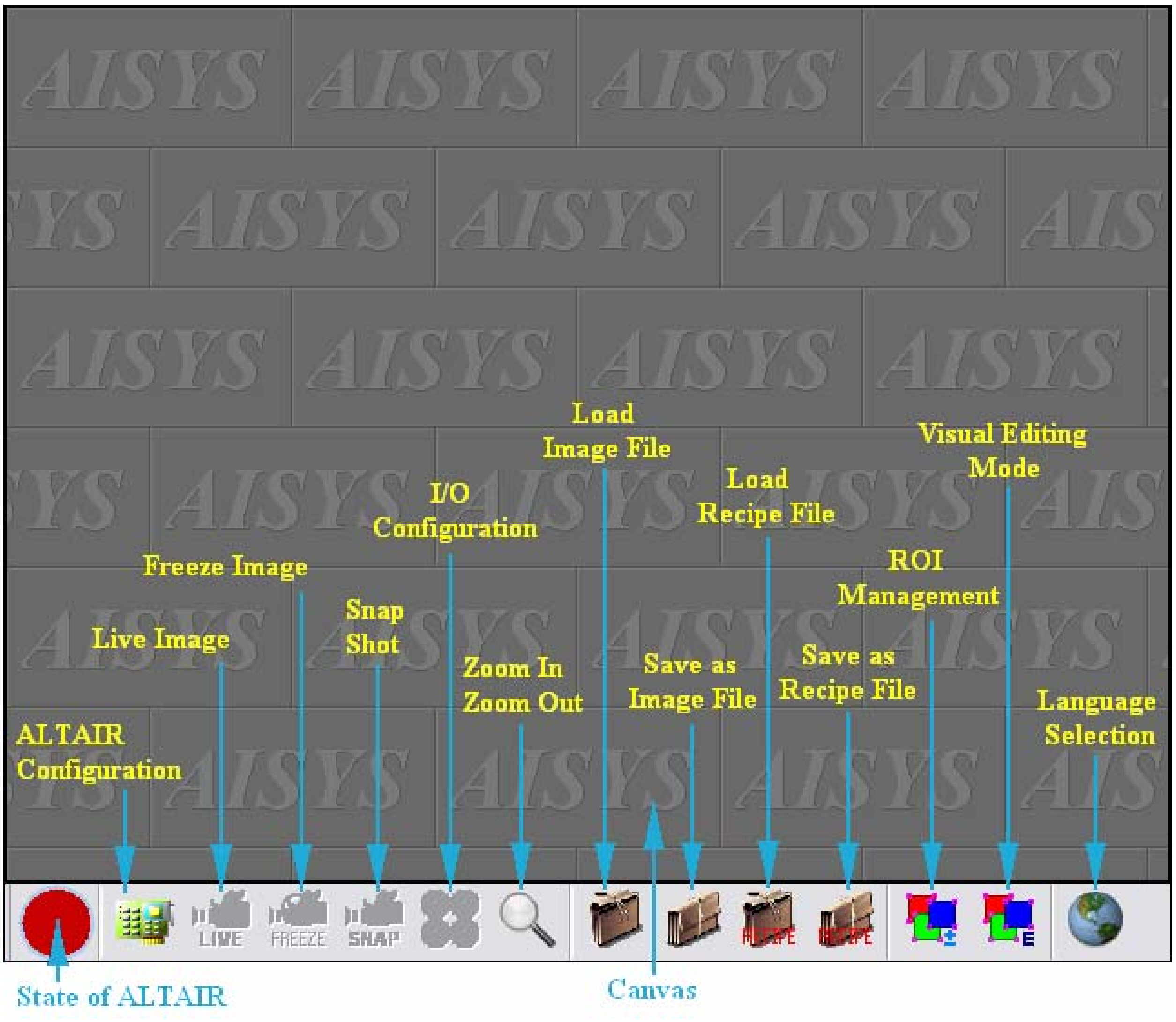


Figure 10

3-1 State of ALTAIR

Three states are identified by the LED-typed mark :

A. Red means **“Channel is not created yet”**;

B. Dark green means **“Channel is created but idle”**;

C. Light green means **“Channel is created and active (Acquiring image)”**.

3-2 ALTAIR Configuration

Press this button to present “Acquisition Hardware Setup” form which helps user to configure ALTAIR easily as shown in Figure 7. Choose appropriate video format / color format and press **“Create Channel”** button to create chanel. Of course, users can also create channel by programming method.

3-3 Live Image / Freeze Image / Snap Shot

These buttons controls image acquisition of ALTAIR framegrabber. Press **“Live Image”** button to acquire intifite image sequence and after acquiring each image vision framework (**AxAltairVisionFramework** component) invoke an **“OnSurfaceFilled”** and an **“OnCustomDraw”** events sequencially to notify users to do something about image processing and image drawing.

3-4 I/O Configuration

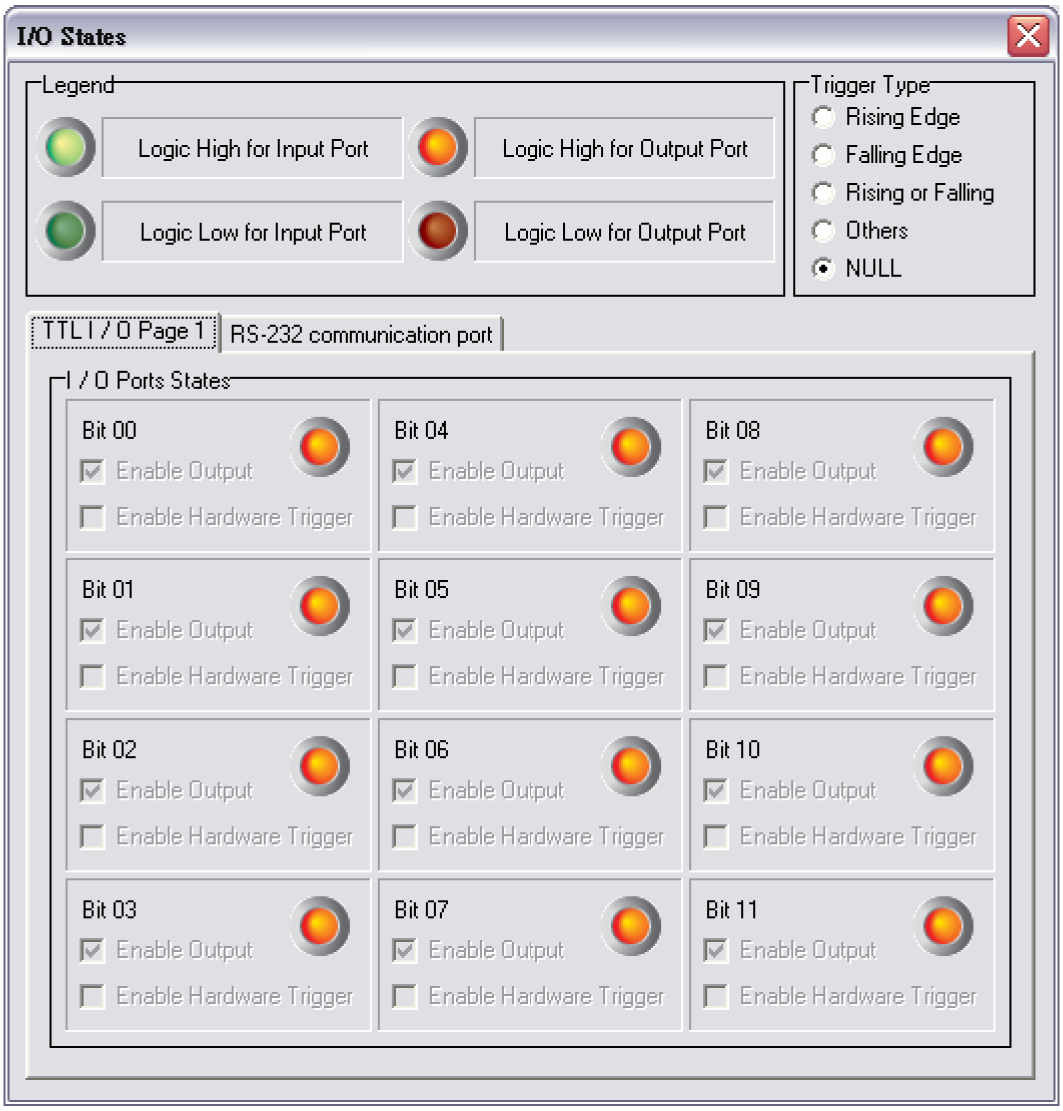
Press this button to present **“I/O States”** form which helps user to configure ALTAIR easily as shown in Figure 11 and Figure 12. Figure 11 shows 4 Input / 4 Output lines states and users can configure external trigger type. ALTAIR framegrabber accept trigger from digital input lines and RS232 communication port. Flexible trigger configurations make users realize I/O application without any other hardware and software programming. Figure 12 shows RS232 configuration form and users can configure and connect RS232 ports by **“Select-and-Click”**.

3-5 Zoom In / Zoom Out

Press this button to present **“Zoom In / Zoom Out”** form which helps user to zoom-in or zoom-out the image show in canvas as shown in Figure 13. Zoom factors can be configured separately in X direction and Y direction which give users the best viewpoint to render the image.

3-6 Load Image File / Save as Image File

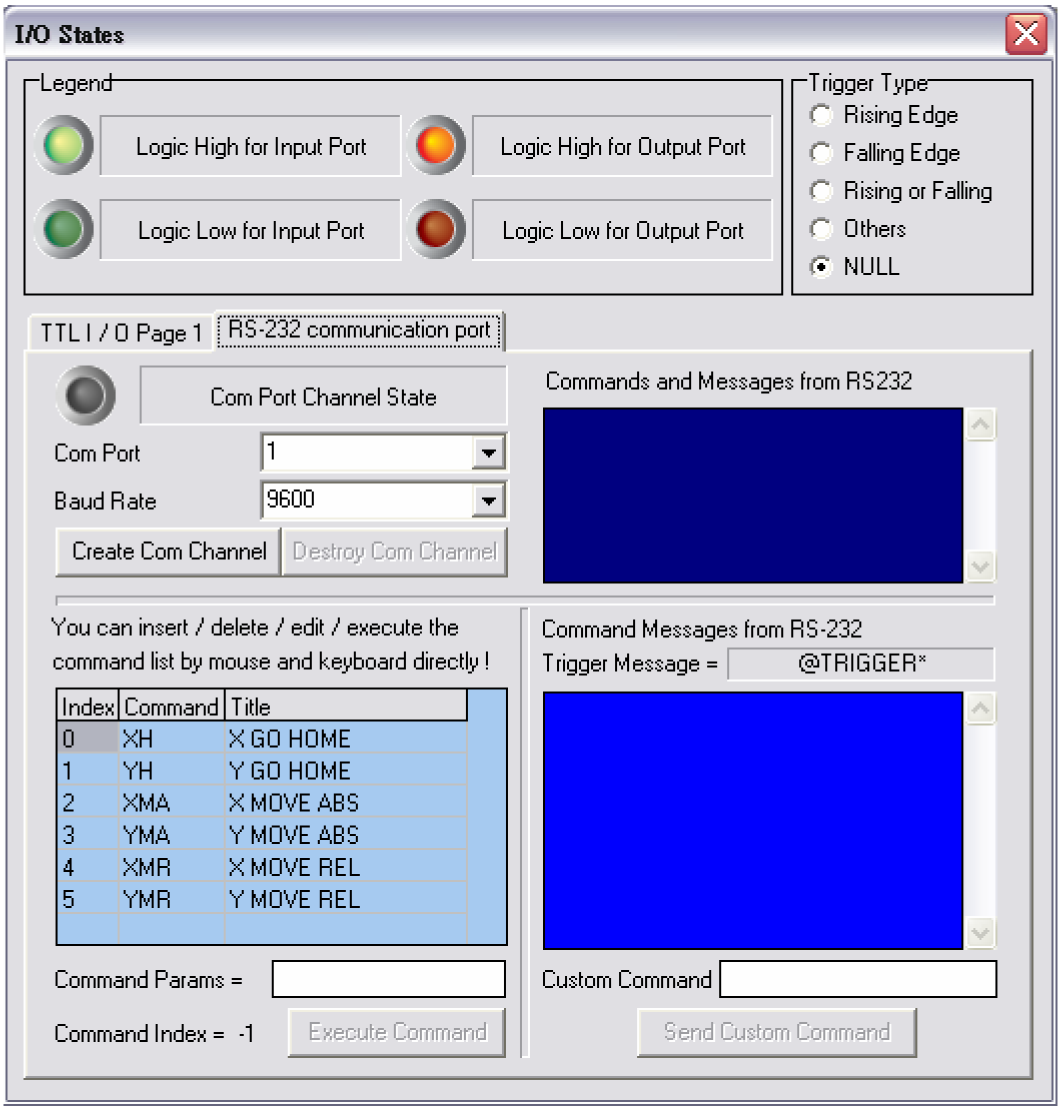
Press this button to present **“Open Image from File”** form or **“Save Image to File”** form which help user to load image from image file or save current image buffer to image file. The supported image formats include BMP / JPEG / TIFF.



Trigger Type: Rising Edge, Falling Edge, Rising or Falling, Others, [✓ NULL]

Bit 00 ● [✓ Enable Output] Enable Hardware Trigger	Bit 04 ● [✓ Enable Output] Enable Hardware Trigger	Bit 08 ● [✓ Enable Output] Enable Hardware Trigger
Bit 01 ● [✓ Enable Output] Enable Hardware Trigger	Bit 05 ● [✓ Enable Output] Enable Hardware Trigger	Bit 09 ● [✓ Enable Output] Enable Hardware Trigger
Bit 02 ● [✓ Enable Output] Enable Hardware Trigger	Bit 06 ● [✓ Enable Output] Enable Hardware Trigger	Bit 10 ● [✓ Enable Output] Enable Hardware Trigger
Bit 03 ● [✓ Enable Output] Enable Hardware Trigger	Bit 07 ● [✓ Enable Output] Enable Hardware Trigger	Bit 11 ● [✓ Enable Output] Enable Hardware Trigger

Figure 11



I/O States

Legend

●	Logic High for Input Port	●	Logic High for Output Port
●	Logic Low for Input Port	●	Logic Low for Output Port

Trigger Type: Rising Edge / Falling Edge / Rising or Falling / Others / [✓ NULL]

TTL I / O Page 1 | RS-232 communication port

Com Port Channel State

Com Port: 1

Baud Rate: 9600

Create Com Channel | Destroy Com Channel

Commands and Messages from RS232

You can insert / delete / edit / execute the command list by mouse and keyboard directly !

Index	Command	Title
0	XH	X GO HOME
1	YH	Y GO HOME
2	XMA	X MOVE ABS
3	YMA	Y MOVE ABS
4	XMR	X MOVE REL
5	YMR	Y MOVE REL

Command Messages from RS-232

Trigger Message = @TRIGGER*

Command Params =

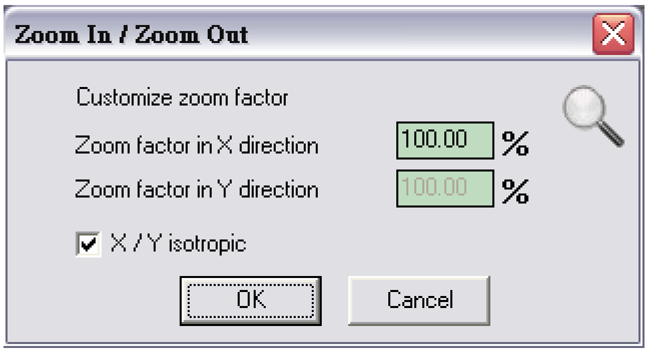
Command Index = -1

Execute Command

Custom Command

Send Custom Command

Figure 12



Zoom In / Zoom Out

Customize zoom factor

Zoom factor in X direction: 100.00 %

Zoom factor in Y direction: 100.00 %

[✓] X / Y isotropic

OK | Cancel

Figure 13

Chapter 4 Programming Model of Vision Framework

4-1 Introduction

The vision framework component **(AxAltairVisionFramework)** is a rapid vision development framework dedicately designed to all the vision hardwares from ***AISYS* Vision Corporation**. The vision framework component is not only the easiest way to integrate ***AISYS*** vision hardware with user's system but an exquisite kernel of ***AISYS* OVK Framework** which is a powerful vision software library. Through the help of the vision framework, the complexity of developing vision system dramatically drops off. In addition, vision framework can also reduce the complexity and effort of software maintain due to its patented structure. No matter users adopt only hardware or whole solution from ***AISYS***, vision framework always meets users' needs seamlessly and we hope technology and products from ***AISYS*** brings users the all new experience in vision development.

4-2 Programming Model

Thanks to **Microsoft ActiveX™ component model** , all the software products from ***AISYS*** adopts P-M-E (Property-Method-Event) programming model. The acquisition flow is fully controlled by vision framework and vision framework will assert two successive events **"OnSurfaceFilled" 、"OnCustomDraw"** to notify user to do something like **"Process the underlying image data" 、"Draw the underlying image or somewhat"** whenever a frame buffer (or so-called surface) is filled with new image data just acquired by framegrabber. The **"OnSurfaceFilled" 、"OnCustomDraw" two-event-operation-mode** is the key role in vision framework.

Prototype : void OnSurfaceFilled (long SurfaceHandle) ;

The event **"OnSurfaceFilled"** bring in a handle of surface which is a handle of complex data structure that contains the image data just acquired. If user adopt OVK Framework as the image processing library, one can directly pass the handle to the vision library to complete the analysis tasks. If user adopt their own image processing routines, one can invoke the method of vision framework **"long GetImagePtr (long SurfaceHandle , int X , int Y)"** by passing the handle of surface as the first parameter and the starting position in the image as the second and third parameters in the event **"OnSurfaceFilled"** to get the physical memory address the image stored in. For example,

```
void AxAltairVisionFramework::OnSurfaceFilled ( long SurfaceHandle )
{
        // Fetch the physical address of image ( Top-left corner )
        pPhyImageAddr=(BYTE*)VisionFramework->GetImagePtr ( SurfaceHandle , 0 , 0 ) ;
        // Processing this image
        OemProcessingRoutine ( pPhyImageAddr ) ;
}
```

Prototype : void OnCustomDraw (long hDC) ;

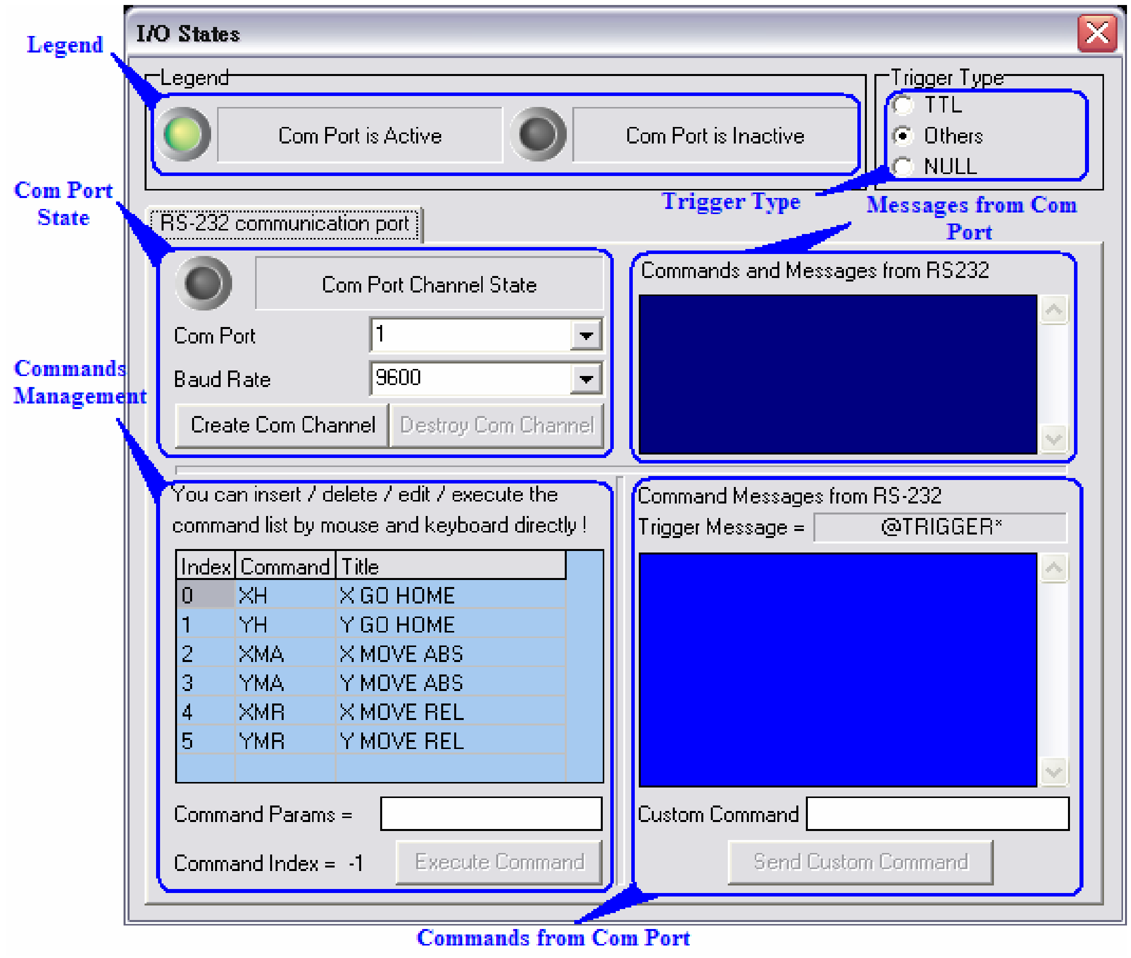
The event **“OnCustomDraw”** bring five parameters in. The first parameter is a Microsoft Windows HDC of canvas and one can draw any texts 、lines 、circles 、points even motifs by GDI APIs. Because vision framework draw the underlying image onto canvas automatically, users do not need to draw the underlying image in the event **“OnCustomDraw”**. Set the property **“bool AutoDrawSurface “** as FALSE to disable this automatic drawing facility. For example,

```
void AxAltairVisionFramework::OnCustomDraw ( long hDC )
{
        // Users do not need to draw the underlying image here because vision framework draw it onto canvas
        // automatically if the property “bool AutoDrawSurface “ is set as TRUE ( which is a default value )
        // Select white pen and draw a white straight line from ( 0 , 0 ) to ( 100 , 100 )
        SelectObject ( (HDC) hDC , GetStockObject (WHITE_PEN) ) ;
        MoveToEx ( (HDC) hDC , 0*VisionFramework->ZoomX+ VisionFramework->PanX , 0*
                        VisionFramework->ZoomY+ VisionFramework->PanY ) ;
        LineTo ( (HDC) hDC , 100* VisionFramework->ZoomX+ VisionFramework->PanX , 100*
                    VisionFramework->ZoomY+ VisionFramework->PanY ) ;
}
```

Typically, users can arrange image processing code fragment in the event **“OnSurfaceFilled”** while arranging image drawing code fragment in the event **“OnCustomDraw”**. The **“OnSurfaceFilled”** 、**”OnCustomDraw” two-event-operation-mode** is just the key reason that vision framework can dramatically reduce the effort of developing and maintaining software.

Chapter 5 Introduction to RS232 configuration

Press the button “**I / O States”** to present “I/O States” form as shown in Figure 14.



I/O States

Legend

● Com Port is Active ● Com Port is Inactive

Trigger Type

TTL
[✓ Others]
NULL

RS-232 communication port

● Com Port Channel State

Com Port: 1

Baud Rate: 9600

Create Com Channel | Destroy Com Channel

Commands and Messages from RS232

You can insert / delete / edit / execute the command list by mouse and keyboard directly !

Index	Command	Title
0	XH	X GO HOME
1	YH	Y GO HOME
2	XMA	X MOVE ABS
3	YMA	Y MOVE ABS
4	XMR	X MOVE REL
5	YMR	Y MOVE REL

Command Params =

Command Index = -1

Execute Command

Command Messages from RS-232

Trigger Message = @TRIGGER*

Custom Command

Send Custom Command

Figure 14

5-1 Introduction

For being the best bare development system of vision, vision framework support versatile RS232 functionality. Through the help of vision framework, RS232 communication becomes very simple. The kernel of vision framework send or receive any data to or from com port automatically; buffering, decoding and filtering the incoming messages according to an user-defined string format, then assert the events to notify users if any user-defined string pattern was received. This is a nice feature and very useful for interfacing vision framework with devices such as PLC. Also, the kernel of vision framework send strings users committed automatically. Users can customize command string table themselves and send specific command string simply by assigning the index. Vision framework makes RS232 hardware transparent to developers and dramatically reduce the effort developers should contribute in developing a vision system.

5-2 Legend

“Legend” shows the legend of com port state. Light green means com port is connected while grey means com port is disconnected.

5-3 Com Port State

The LED-liked region shows the com port connectivity according to the logic described in section 4-2. Light green means com port is connected while grey means com port is disconnected. The combo box of **“Com Port”** shows the index of com port users would like to open. The combo box of **“Baud Rate”** shows the baud rate of com port users would like to open. Two buttons **“Create Com Channel”、 “Destroy Com Channel”** can help users to connect to com port or disconnect from com port. Of course, before connecting to com port, users should configure parameters of communication such as **“Com Port”、”Baud Rate”**…etc. (other parameters will be stated in chapter 5) , and then press the button **“Create Com Port”** to connect the com port.

5-4 Commands Management

The table in the middle of this region shows the default commands list of vision framework. Users can customize their own commands list and send commands simply by assigning the index. Note that the field of **“Title”** can be any description string that help users to understand the meaning of specific command. The edit box **“Command Params=”** determines the extra parameters when users send the commands in the commands list. The **“Command Index”** shows the index users specified (typically, users can specify the command simply by clicking the command in the commands list). Note that **“Command Index”** equals **“-1”** means no command were specified at all. After specifying command and optionally fill the extra parameters, users can press the button **“Execute Command”** to send the command to com port immediately.

5-5 Messages from Com Port

The dark blue region shows all of the messages received by com port. Strings sent from the peer (ex. PLC、PC…etc.) will be shown in this region.

5-6 Commands from Com Port

The dark blue region shows all of the qualified messages (the “qualified” messages is the so-called “commands” sent from the peer and the format of commands can be configured by a leading character and a trailing character in vision framework) received by com port. Qualified messages sent from the peer (ex. PLC、PC…etc.) will be shown in this region. Also, users can define a specific string pattern (i.e. trigger message) to trigger vision framework when the string match the pattern is received. When vision framework is triggered, the event “OnTriggerMsgArrived” will be assert from vision framework to notify user to do something like “acquire image”…etc. Input custom command into the edit box “Custom Command” and press the button “Send Custom Command” to send the custom command to the peer immediatelly.

5-7 Trigger Type

The trigger type region determines the type of trigger that vision framework can accept. There are three options can be configured. Checking the option **“TTL”** configures ALTAIR framegrabber accepting triggers only from digital input (TTL input) and vision framework will assert event **“OnHardwareTrigger”** whenever **“High-to-low”** transition occurs in digital input port. Checking the option **“Others”** configures ALTAIR framegrabber accepting triggers only from communication port (com port) and vision framework will assert event **“OnTriggerMsgArrived”** whenever user-defined trigger command receives in com port. Checking the option **“NULL”** configures ALTAIR framegrabber accepting no triggers at all. Note that configuring trigger type to **“Others”** or **“NULL”** will **not** suppress the event **“OnHardwareTrigger”** when **“High-to-low”** transition occurs in digital input port while configuring trigger type to **“TTL”** will supress the event **“OnTriggerMsgArrived”** when user-defined trigger command receives in com port.

5-8 Releated Properties / Methods / Events

See the chapter 5 for details of programming interface.

Chapter 6 Programming Interface of Vision Framework

6-1 Introduction

The vision framework component **(AxAltairVisionFramework)** is a rapid vision development framework dedicately designed to all the vision hardwares from ***AISYS*** **Vision Corporation**. The vision framework component is not only the easiest way to integrate ***AISYS*** vision hardware with user's system but an exquisite kernel of ***AISYS*** **OVK Framework** which is a powerful vision software library. Through the help of the vision framework, the complexity of developing vision system dramatically drops off. In addition, vision framework can also reduce the complexity and effort of software maintain due to its patented structure. No matter users adopt only hardware or whole hardware and software solution from ***AISYS***, vision framework always meets users' needs seamlessly and we hope technology and products from ***AISYS*** brings users the all new experience in vision development.

6-2 Programming Interface

6-2-1 Properties

bool CanvasAutoSize (R/W, Default=True); Set as TRUE means vision framework will adjust geometry of canvas according to image size and zoom factor automatically ; set as FALSE means vision framework will not adjust geometry of canvas when image size or zoom changing.

int CanvasWidth (R/W, Default=640); This property determine the width of canvas in unit of pixel.

int CanvasHeight (R/W, Default=480); This property determine the height of canvas in unit of pixel.

bool AutoDrawSurface (R/W, Default=TRUE) ; Set as **TRUE** means vision framework will automatically draw the underlying image onto canvas whenever vision framework finishs acquiring image. Set as **FALSE** to disable this automatic drawing facility. With this property set as **FALSE**, ones can draw the image by the method **"DrawSurface"** within the event **"OnCustomDraw"** if users would like to draw the image themselves.

float ZoomX (R/W, Defautl=1.0) ; This property determine the zooming factor in the direction of X. 1.0 means 100% and 2.0 means 200% …etc.

float ZoomY (R/W, Defautl=1.0) ; This property determine the zooming factor in the direction of Y. 1.0 means 100% and 2.0 means 200% …etc.

int PanX (R/W, Defautl=1.0) ; This property determine the panning factor in the direction of X. Positive value means right offset, negative values means left offset. The unit is pixel.

int PanY (R/W, Defautl=1.0) ; This property determine the panning factor in the direction of Y. Positive value means up offset, negative values means down offset. The unit is pixel.

TxAxColor PenColor (R/W, Default=AX_COLOR_WHITE, 20 in decimal); This property determines the pen color of the drawing area in AxCanvas component. **Note that users should configure pen color by this property instead of using windows API to avoid GDI confliction.** The available options are as the following :

AX_COLOR_AQUA	**0 in decimal**
AX_COLOR_BLACK	**1 in decimal**
AX_COLOR_BLUE	**2 in decimal**
AX_COLOR_CREAM	**3 in decimal**
AX_COLOR_DKGRAY	**4 in decimal**
AX_COLOR_FUCHSIA	**5 in decimal**
AX_COLOR_GRAY	**6 in decimal**
AX_COLOR_GREEN	**7 in decimal**
AX_COLOR_LIME	**8 in decimal**
AX_COLOR_LTGRAY	**9 in decimal**
AX_COLOR_MAROON	**10 in decimal**
AX_COLOR_MEDGRAY	**11 in decimal**
AX_COLOR_MONEYGREEN	**12 in decimal**
AX_COLOR_NAVY	**13 in decimal**
AX_COLOR_OLIVE	**14 in decimal**
AX_COLOR_PURPLE	**15 in decimal**
AX_COLOR_RED	**16 in decimal**
AX_COLOR_SILVER	**17 in decimal**
AX_COLOR_SKYBLUE	**18 in decimal**
AX_COLOR_TEAL	**19 in decimal**
AX_COLOR_WHITE	**20 in decimal**
AX_COLOR_YELLOW	**21 in decimal**
AX_COLOR_OTHER	**22 in decimal**

Note that if **PenColor =AX_COLOR_OTHER**, users can customize pen color by assigning color constant to the property **"CustomPenColor"**. See the next property for more details.

long CustomPenColor (R/W, Default= 16777215); This property determines the customized pen color when **PenColor =AX_COLOR_OTHER**. The color value is encoded by the following method : 0x00BBGGRR in hexdecimal. Each color component have 8 bits (256 levels) to quantize. For example, pure red can be expressed as 0x000000FF = 255 ; pure green can be expressed as 0x0000FF00 = 65280 ; pure blue can be expressed as 0x00FF0000 = 16711680.

int PenWidth (R/W, Default= 1); This property determines the width of the pen of drawing area in AxCanvas component. **Note that users should configure pen width by this property instead of using windows API to avoid confliction.**

TxAxPenStyle PenStyle (R/W, Default= AX_PENSTYLE_SOLID, 0 in decimal); This property determines the pen style of the drawing area in AxCanvas component. **Note that users should configure pen style by this property instead of using windows API to avoid GDI confliction.** The available options are as the following :

AX_PENSTYLE_SOLID	**0 in decimal**
AX_PENSTYLE_DASH	**1 in decimal**
AX_PENSTYLE_DOT	**2 in decimal**
AX_PENSTYLE_DASHDOT	**3 in decimal**
AX_PENSTYLE_DASHDOTDOT	**4 in decimal**
AX_PENSTYLE_CLEAR	**5 in decimal**
AX_PENSTYLE_INSIDEFRAME	**6 in decimal**

TxAxColor FontColor (R/W, Default=AX_COLOR_BLACK, 1 in decimal); This property determines the font color of the drawing area in AxCanvas component. **Note that users should configure font color by this property instead of using windows API to avoid GDI confliction.** The available options are as the following :

AX_COLOR_AQUA	**0 in decimal**
AX_COLOR_BLACK	**1 in decimal**
AX_COLOR_BLUE	**2 in decimal**
AX_COLOR_CREAM	**3 in decimal**
AX_COLOR_DKGRAY	**4 in decimal**
AX_COLOR_FUCHSIA	**5 in decimal**
AX_COLOR_GRAY	**6 in decimal**
AX_COLOR_GREEN	**7 in decimal**
AX_COLOR_LIME	**8 in decimal**

AX_COLOR_LTGRAY	**9 in decimal**
AX_COLOR_MAROON	**10 in decimal**
AX_COLOR_MEDGRAY	**11 in decimal**
AX_COLOR_MONEYGREEN	**12 in decimal**
AX_COLOR_NAVY	**13 in decimal**
AX_COLOR_OLIVE	**14 in decimal**
AX_COLOR_PURPLE	**15 in decimal**
AX_COLOR_RED	**16 in decimal**
AX_COLOR_SILVER	**17 in decimal**
AX_COLOR_SKYBLUE	**18 in decimal**
AX_COLOR_TEAL	**19 in decimal**
AX_COLOR_WHITE	**20 in decimal**
AX_COLOR_YELLOW	**21 in decimal**
AX_COLOR_OTHER	**22 in decimal**

Note that if **FontColor =AX_COLOR_OTHER**, users can customize font color by assigning color constant to the property **"CustomFontColor"**. See the next property for more details.

long CustomFontColor (R/W, Default= 16777215); This property determines the customized font color when **FontColor =AX_COLOR_OTHER**. The color value is encoded by the following method : 0x00BBGGRR in hexdecimal. Each color component have 8 bits (256 levels) to quantize. For example, pure red can be expressed as 0x000000FF = 255 ; pure green can be expressed as 0x0000FF00 = 65280 ; pure blue can be expressed as 0x00FF0000 = 16711680.

TxAxColor BrushColor (R/W, Default=AX_COLOR_WHITE, 20 in decimal); This property determines the brush color of the drawing area in AxCanvas component. **Note that users should configure brush color by this property instead of using windows API to avoid GDI confliction.** The available options are as the following :

AX_COLOR_AQUA	**0 in decimal**
AX_COLOR_BLACK	**1 in decimal**
AX_COLOR_BLUE	**2 in decimal**
AX_COLOR_CREAM	**3 in decimal**
AX_COLOR_DKGRAY	**4 in decimal**
AX_COLOR_FUCHSIA	**5 in decimal**
AX_COLOR_GRAY	**6 in decimal**

AX_COLOR_GREEN	**7 in decimal**
AX_COLOR_LIME	**8 in decimal**
AX_COLOR_LTGRAY	**9 in decimal**
AX_COLOR_MAROON	**10 in decimal**
AX_COLOR_MEDGRAY	**11 in decimal**
AX_COLOR_MONEYGREEN	**12 in decimal**
AX_COLOR_NAVY	**13 in decimal**
AX_COLOR_OLIVE	**14 in decimal**
AX_COLOR_PURPLE	**15 in decimal**
AX_COLOR_RED	**16 in decimal**
AX_COLOR_SILVER	**17 in decimal**
AX_COLOR_SKYBLUE	**18 in decimal**
AX_COLOR_TEAL	**19 in decimal**
AX_COLOR_WHITE	**20 in decimal**
AX_COLOR_YELLOW	**21 in decimal**
AX_COLOR_OTHER	**22 in decimal**

Note that if **BrushColor =AX_COLOR_OTHER**, users can customize font color by assigning color constant to the property **"CustomBrushColor"**. See the next property for more details.

long CustomBrushColor (R/W, Default= 16777215); This property determines the customized brush color when **BrushColor =AX_COLOR_OTHER**. The color value is encoded by the following method : 0x00BBGGRR in hexdecimal. Each color component have 8 bits (256 levels) to quantize. For example, pure red can be expressed as 0x000000FF = 255 ; pure green can be expressed as 0x0000FF00 = 65280 ; pure blue can be expressed as 0x00FF0000 = 16711680.

TxAxBrushStyle BrushStyle (R/W, Default= AX_BRUSHSTYLE_SOLID, 0 in decimal); This property determines the brush style of the drawing area in AxCanvas component. **Note that users should configure brush style by this property instead of using windows API to avoid GDI confliction.** The available options are as the following :

AX_BRUSHSTYLE_SOLID	**0 in decimal**
AX_BRUSHSTYLE_CROSS	**1 in decimal**
AX_BRUSHSTYLE_CLEAR	**2 in decimal**
AX_BRUSHSTYLE_DIAGCROSS	**3 in decimal**
AX_BRUSHSTYLE_BDIAGONAL	**4 in decimal**

AX_BRUSHSTYLE_FDIAGONAL	**5 in decimal**
AX_BRUSHSTYLE_HORIZONTAL	**6 in decimal**
AX_BRUSHSTYLE_VERTICAL	**7 in decimal**

Long hDC (Read Only) ; This property is the handle of device context (HDC) of canvas. All the GDI APIs and drawing functions in ***AISYS* OVK Framework** need this property to draw something in canvas. Because the canvas of vision framework is double buffered, users need to refresh canvas by themselves through invoking method **"RefreshCanvas"**.

bool ShowToolBar (R/W, Default=TRUE); This property determines the visibility of tool bar which is located at the bottom of vision framework. Set as FALSE to hide the tool bar if users would like to make the vision framework more compact as shown in Figure 15.

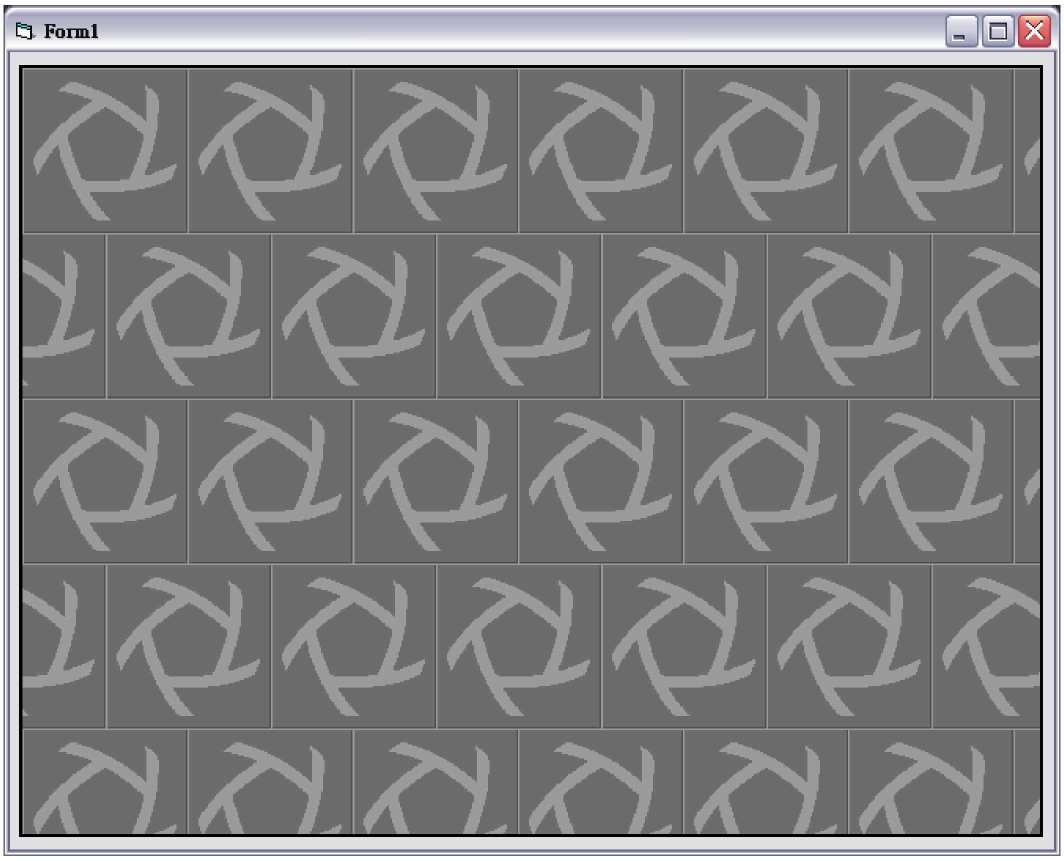


Figure 15.

TxAxaPortID PortID (R/W, Default=AXA_PORT_0); This property determine the index of framegrabber users would like to connect. Typically, after assigning **PortID**, invoking the method **“CreateChannel”** to connect the framegrabber specified by **PortID**. The available options are as the following:

AXA_PORT_0	**0 in decimal**
AXA_PORT_1	**1 in decimal**
AXA_PORT_2	**2 in decimal**
AXA_PORT_3	**3 in decimal**

TxAxaAcqColFmt AcqColFmt (R/W, Default=AXA_ACQCOLFMT_GREYLEVEL, 0 in decimal); This property determine the color format of connected camera. The available options are as the following:

AXA_ACQCOLFMT_GREYLEVEL	**0 in decimal**
AXA_ACQCOLFMT_RGB24	**1 in decimal**

TxAxaVideoFmt VideoFmt (R/W, Default=AXA_VIDEOFMT_FULL_NTSC, 0 in decimal); This property determine the video sync format of connected camera. The available options are as the following:

AXA_VIDEOFMT_FULL_NTSC	**0 in decimal**
AXA_VIDEOFMT_FULL_PAL	**1 in decimal**
AXA_VIDEOFMT_CIF_NTSC	**2 in decimal**
AXA_VIDEOFMT_CIF_PAL	**3 in decimal**

int GrabCount (R/W, Default=0); This property determine the number of frames need to acquire. Vision framework control ALTAIR series framegrabbers by two properties **“GrabCount”、”ChannelState”** mainly. The property **”ChannelState”** controls the operation mode of framegrabber. There are two operation modes of framegrabber available **“ACTIVE”、”IDLE”**. See the property **“ChannelState”** for details. **“GrabCount”** will be decreased by 1 when each acquisition is finished. When **“GrabCount”** becomes zero, framegrabber stop acquiring although the **”ChannelState”** is still in **“ACTIVE”** mode. It mean acquires image infinitely if **“GrabCount”** is -1 and **”ChannelState”** is in **“ACTIVE”** mode which is so-called **“Live Image”**.

int FrameRate (R/W, Default=30); This property determine the data output rate (so-called frame rate) of the framegrabber.

bool IsChannelCreated (Read Only); Read this property to determine whether any camera is connected by vision framework or not.

TxAxaChannelState ChannelState (R/W, Default=AXA_CHANNELSTATE_IDLE, 0 in decimal); This property determine the operation mode of the channel. This property should be configured combined with the property **“GrabCount”**. See the property **“GrabCount”** for the detail. The available options are the followings:

AXA_CHANNELSTATE_IDLE	**0 in decimal**
AXA_CHANNELSTATE_ACTIVE	**1 in decimal**

int Brightness (R/W, Default=128); This property determine the brightness of video amplifier in ALTAIR framegrabber. It ranges from 0 to 255 and the neutral value is 128.

int Contrast (R/W, Default=108); This property determine the contrast of video amplifier in ALTAIR framegrabber. It ranges from 0 to 255 and the neutral value is 108.

int Hue (R/W, Default=0); This property determine the hue of ALTAIR framegrabber. It ranges 0 to 255.

int ChromaU (R/W, Default=127); This property determine the U gain of chroma in ALTAIR framegrabber. It ranges from 0 to 255 and the neutral value is 127.

int ChromaV (R/W, Default=127); This property determine the V gain of chroma in ALTAIR framegrabber. It ranges from 0 to 255 and the neutral value is 127.

int ImageWidth (Read Only); Read this property to determine the width of image in pixels that the framegrabber output. This property will be changed according to the property **“VideoFmt”**.

int ImageHeight (Read Only); Read this property to determine the height of image in pixels that the framegrabber output. This property will be changed according to the property **“VideoFmt”**.

bool ShowControlPanel (R/W, Default=False); A versatile control panel for ALTAIR framegrabber was embedded in vision framework. With the help of control panel, users can integrate framegrabber to their system very easily and rapidly. Important functions of the framegrabber can be configured through the control panel without any programming codes. Set this property as **True** to show the embedded control panel while **False** to hide the control panel.

long ActiveSurfaceHandle (Read Only); Read this property to fetch the handle of the latest acquired image. With the handle at hand, users can draw it anywhere they would like to by invoking the method **“DrawSurface”** of vision framework or pass it to the image processing library provided by AISYS (such as **VAlign、OVK Framework**…etc.) for further processing and analysis. Note that this handle is completely a object handle specific to software products from AISYS not a physical memory address of image data.

To obtain the physical memory address of image data, pass this handle to the method **"GetImagePtr"** to get the address.

TxAxComPortID ComPortID (R/W, Default= AX_COMPORT_COM1, 0 in decimal); This property determine the index of com port users would like to connect. Typically, after assigning **ComPortID**, invoking the method **"CreateComPortChannel"** to connect com port. The available options are as the following:

AX_COMPORT_COM1	**0 in decimal**
AX_COMPORT_COM2	**1 in decimal**
AX_COMPORT_COM3	**2 in decimal**
AX_COMPORT_COM4	**3 in decimal**

TxAxComPortBaudRate ComPortBaudRate (R/W, Default=AX_COMPORT_BAUDRATE_9600_BPS, 6 in decimal); This property determine the speed of com port users would like to connect. The available options are as the following:

AX_COMPORT_BAUDRATE_110_BPS	**0 in decimal**
AX_COMPORT_BAUDRATE_300_BPS	**1 in decimal**
AX_COMPORT_BAUDRATE_600_BPS	**2 in decimal**
AX_COMPORT_BAUDRATE_1200_BPS	**3 in decimal**
AX_COMPORT_BAUDRATE_2400_BPS	**4 in decimal**
AX_COMPORT_BAUDRATE_4800_BPS	**5 in decimal**
AX_COMPORT_BAUDRATE_9600_BPS	**6 in decimal**
AX_COMPORT_BAUDRATE_14400_BPS	**7 in decimal**
AX_COMPORT_BAUDRATE_19200_BPS	**8 in decimal**
AX_COMPORT_BAUDRATE_38400_BPS	**9 in decimal**
AX_COMPORT_BAUDRATE_56000_BPS	**10 in decimal**
AX_COMPORT_BAUDRATE_57600_BPS	**11 in decimal**
AX_COMPORT_BAUDRATE_115200_BPS	**12 in decimal**
AX_COMPORT_BAUDRATE_128000_BPS	**13 in decimal**
AX_COMPORT_BAUDRATE_256000_BPS	**14 in decimal**

TxAxComPortDataBits ComPortDataBits (R/W, Default=AX_COMPORT_DATABITS_8, 4 in decimal); This property determine the data bits of com port users would like to connect. The available options are as the following:

AX_COMPORT_DATABITS_4	**0 in decimal**

AX_COMPORT_DATABITS_5	**1 in decimal**
AX_COMPORT_DATABITS_6	**2 in decimal**
AX_COMPORT_DATABITS_7	**3 in decimal**
AX_COMPORT_DATABITS_8	**4 in decimal**

TxAxComPortFlowControl ComPortFlowControl (R/W, Default= AX_COMPORT_FLOWCONTROL_DEFAULT, 4 in decimal); This property determine the flow control of com port users would like to connect. The available options are as the following:

AX_COMPORT_FLOWCONTROL_NONE	**0 in decimal**
AX_COMPORT_FLOWCONTROL_CTS	**1 in decimal**
AX_COMPORT_FLOWCONTROL_DTR	**2 in decimal**
AX_COMPORT_FLOWCONTROL_SOFTWARE	**3 in decimal**
AX_COMPORT_FLOWCONTROL_DEFAULT	**4 in decimal**

TxAxComPortParity ComPortParity (R/W, Default= AX_COMPORT_PARITY_NONE, 0 in decimal); This property determine the parity of com port users would like to connect. The available options are as the following:

AX_COMPORT_PARITY_NONE	**0 in decimal**
AX_COMPORT_PARITY_ODD	**1 in decimal**
AX_COMPORT_PARITY_EVEN	**2 in decimal**
AX_COMPORT_PARITY_MARK	**3 in decimal**
AX_COMPORT_PARITY_SPACE	**4 in decimal**

TxAxComPortStopBits ComPortStopBits (R/W, Default= AX_COMPORT_STOPBITS_10, 0 in decimal); This property determine the stop bits of com port users would like to connect. The available options are as the following:

AX_COMPORT_STOPBITS_10	**0 in decimal**
AX_COMPORT_STOPBITS_15	**1 in decimal**
AX_COMPORT_STOPBITS_20	**2 in decimal**

bool ComPortConnected (Read Only); Read this property to determine whether the com port is connected by vision framework or not.

String CmdMsgBeginToken (R/W, Default=”@”)、String CmdMsgEndToken (R/W, Default=”*”); These two properties determine the start and end token of a **“Command”**. The **“Command”** is a qualified message that can be configured by a leading character and a trailing character. The property **“CmdMsgBeginToken “** specifies the leading character of a **“Command”** while the property **“CmdMsgEndToken”** specifies the trailing character of a **“Command”**.

String SeparateToken (R/W, Default=" "); This property determines the separate token. The separate token is inserted between stock command (vision framework can store a list of "stock" commands) and parameters when users send the stock commands by user interface.

String CmdMsg (Read Only); Read this property to determine current **"Command"** received by com port.

String TriggerMsg (R/W, Default="TRIGGER"); This property defines a specific string pattern (i.e. trigger message) to trigger vision framework when the string match this pattern is received. When vision framework is triggered, the event **"OnTriggerMsgArrived"** will be assert from vision framework to notify user to do something like "acquire image"…etc. In default case, when a command **"@TRIGGER*"** received, vision framework is triggered and the event **"OnTriggerMsgArrived"** is asserted.

int NumOfCommands (R/W, Default=6); This property determines the numbers of stock commands currently stored in vision framework. Change the this property will force vision framework allocate the necessary memory space for storing the incoming commands. Typically, users do not need to change this property directly, because the command management methods do this for users.

int CommandIndex (R/W, Default=6); This property determines the index of specific command currently stored in vision framework. This property is typically used in conjunction with the property **"CommandTitle"**. Users read the property **"CommandTitle"** to get the description string of a specific command or write the property **"CommandTitle"** to change the description string of a specific command after assigning the property **"CommandIndex"** to indicate the specific command.

6-2-2 Methods

bool CreateChannel() ; Connect to ALTAIR framegrabber. This method should be invoked before any attempts to control ALTAIR framegrabber. Before invoking this method, users should configure **PortID**、**VideoFmt**、**AcqColorFmt** properties to appropriate values. Return value **TRUE** means the vision framework connects framegrabber successfully. Return value **FALSE** means the vision framework connects framegrabber failed and users should check the hardware and device driver installation to shoot where the trouble is.

bool DestroyChannel() ; Disconnect from ALTAIR framegrabber. This method should be invoked if the connected is not used anymore. Return value **TRUE** means the vision framework disconnects from framegrabber successfully. Return value **FALSE** means the vision framework disconnects from framegrabber failed and users should check the hardware and device driver installation to shoot where the trouble is.

void Live() ; Invoke this method makes vision framework enter **"Live**

Image” mode. At this time, the framegrabber acquires image infinitely and assert two successive events **“OnSurfaceFilled”、”OnCustomDraw”** to notify user to complete necessary tasks. Invoking this method is equivalent to the following code fragment:

```
AxAltairVisionFramework1.GrabCount = -1
Ax AltairVisionFramework1.ChannelState =
AXA_CHANNELSTATE_ACTIVE
```

void Freeze() ; Invoke this method makes vision framework enter **“Freeze Image”** mode. At this time, the framegrabber stop acquiring image. Invoking this method is equivalent to the following code fragment:

```
Ax AltairVisionFramework1.ChannelState =
AXA_CHANNELSTATE_IDLE
Ax AltairVisionFramework1. GrabCount = 0
```

void Snap() ; Invoke this method makes vision framework enter “Snap Image” mode. At this time, the framegrabber acquires one image and assert two successive events **“OnSurfaceFilled”、”OnCustomDraw”** to notify user to complete necessary tasks. Invoking this method is equivalent to the following code fragment:

```
Ax AltairVisionFramework1.ChannelState =
AXA_CHANNELSTATE_IDLE
AxAltairVisionFramework1.GrabCount = 1
Ax AltairVisionFramework1.ChannelState =
AXA_CHANNELSTATE_ACTIVE
```

void LoadFile(String FileName) ; Invoke this method makes vision framework loads the image file **FileName** specified into the current frame buffer and assert two successive events **“OnSurfaceFilled”、”OnCustomDraw”** to notify user to complete necessary tasks.

void SaveFile(String FileName , TxAxImageFileFormat FileFormat) ; Invoke this method makes vision framework saves the current frame buffer to image file with **FileName** as name and **FileFormat** as file format. The available options for **FileFormat** is the following:

AX_IMAGE_FILE_TYPE_GREYLEVEL_BMP: Greylevel in BMP format
AX_IMAGE_FILE_TYPE_FULLCOLOR_BMP: True color in BMP format
AX_IMAGE_FILE_TYPE_GREYLEVEL_JPG: Greylevel in JPEG format
AX_IMAGE_FILE_TYPE_FULLCOLOR_JPG: True color in JPEG format

long DrawSurface (Long SurfaceHandle) ; Draw the image specified by **SurfaceHandle** onto the canvas of vision framework. Returned value means the number of scanlines actually drawed. Typically, vision framework draw the underlying image onto canvas automatically, users do not need to draw the underlying image by themselves. When **AutoDrawSurface** is set as FALSE, vision framework will not draw the underlying image automatically and thus users can use this method draw the image in the event **"OnCustomDraw"**. Note that users should not need to invoke **"RefreshCanvas"** method after drawing in the event **"OnCustomDraw"** because vision framework will do this for you. **"RefreshCanvas"** method should be invoked after drawing anything out of the event **"OnCustomDraw"**.

void DrawText (String Text , int X , int Y) ; Draw the string **Text** onto canvas at the position(X,Y). Same as the method **"DrawSurface"**, users should not need to invoke **"RefreshCanvas"** method after drawing in the event **"OnCustomDraw"** because vision framework will do this for you. **"RefreshCanvas"** method should be invoked after drawing anything out of the event **"OnCustomDraw"**.

void DrawLine (int SX , int SY , int DX , int DY) ; Draw the straight line from position (SX,SY) to position (DX,DY) onto canvas. Same as the method **"DrawSurface"**, users should not need to invoke **"RefreshCanvas"** method after drawing in the event **"OnCustomDraw"** because vision framework will do this for you. **"RefreshCanvas"** method should be invoked after drawing anything out of the event **"OnCustomDraw"**.

long GetImagePtr (Long SurfaceHandle , int X , int Y) ; Get the physical memory address of the image data specified by **SurfaceHandle** and starting position **(X , Y)**. Returned value is the physical memory address of the specified image data at the starting position **(X , Y)**. Users can cast the returned value to a byte pointer and thus can be a data source for the coming procession.

void RefreshCanvas () ; When users draw anything onto canvas (through hDC property) out of the event **"OnCustomDraw"**, you always need to invoke the method **"RefreshCanvas"** manually because of double buffered mechanism of canvas in vision framework. Users should not need to invoke **"RefreshCanvas"** method after drawing in the event **"OnCustomDraw"** because vision framework will do this for you.

void ForceReinspect () ; This method automatically simulate the flow of actions that vision framework plays when acquiring and filling a new surface. When invoking the method **"ForceReinspect"**, vision framework assert the event **"OnSurfaceFilled"** with the **"active"** surface handle bringing in. The so-called **"active"** surface handle can be the most recently acquired surface if no new image is acquired now or just the newly acquired image. This method is useful especially when users want to **"inspect manually"** or test the whole inspection process for their software.

void ForceRedraw() ; This method automatically asserts event **"OnCustomDraw"** to let users have a chance to **"Draw the underlying image or somewhat"**. This method is useful especially when users want to **"draw something manually"** or test the whole inspection process for their software. Typically, this method is used in conjunction with the method

"ForceReinspect" and is usually invoked after invoking the method **"ForceReinspect"**.

TxAxaIoState GetIoState(TxAxaIoPortType PortType, TxAxaIoPort Port); ALTAIR framegrabber support 4 In / 4 Out digital IO. Invoke this method to read the current state of input or output port. **PortType** determine the type of port users would like to read. The available options are the followings:

AXA_IO_PORT_TYPE_INPUT	**0 in decimal**
AXA_IO_PORT_TYPE_OUTPUT	**1 in decimal**

Port determine the index of port users would like to read. The available options are the followings:

AXA_IO_PORT_00	**0 in decimal**
AXA_IO_PORT_01	**1 in decimal (Reserved for future use)**
AXA_IO_PORT_02	**2 in decimal (Reserved for future use)**
AXA_IO_PORT_03	**3 in decimal (Reserved for future use)**
AXA_IO_PORT_04	**4 in decimal (Reserved for future use)**
AXA_IO_PORT_05	**5 in decimal (Reserved for future use)**
AXA_IO_PORT_06	**6 in decimal (Reserved for future use)**
AXA_IO_PORT_07	**7 in decimal (Reserved for future use)**
AXA_IO_PORT_11	**11 in decimal (Reserved for future use)**
AXA_IO_PORT_12	**12 in decimal (Reserved for future use)**
AXA_IO_PORT_13	**13 in decimal (Reserved for future use)**
AXA_IO_PORT_14	**14 in decimal (Reserved for future use)**
AXA_IO_PORT_15	**15 in decimal (Reserved for future use)**

The returned value determines the state of the specified port. The state are the followings:

AXA_IO_STATE_LOW	**0 in decimal**
AXA_IO_STATE_HIGH	**1 in decimal**
AXA_IO_STATE_NULL	**2 in decimal**

"AXA_IO_STATE_NULL" indicates the failure of execution of the method.

void SetIoState(TxAxaIoPort Port, TxAxaIoState IoState); Invoke this method to write the state of the specified output port. **Port** determines the port index users specified. **IoState** determines the state of the port to change. The available options are listed as above.

float GetFocusRatio (Long SurfaceHandle) ; Estimate the focus ratio of the image specified by **SurfaceHandle**. This is a very nice feature that help users to design an auto-focus system without any other software or hardware. When the focus is reached, this focus ratio will reach a local maximum thus users can refer to this ratio to control the motion device and search the best focus easily. The algorithm of this focus ratio is very fast and robust so that the estimation is almost real-time and will not low down the frame rate at all.

bool CreateComPortChannel(); Connect to com port. This method should be invoked before any attempts to communicate with com port. Before invoking this method, users should configure **ComPortID**、**ComPortBaudRate**、**ComPortDataBits**、**ComPortFlowControl**、**ComPortParity**、**ComPortStopBits** to suit the peer's need. Return value **TRUE** means the vision framework connects com port successfully. Return value **FALSE** means the vision framework connects com port failed.

bool DestroyComPortChannel(); Disconnect from com port. This method should be invoked if com port is not used anymore. Return value **TRUE** means the vision framework disconnects com port successfully. Return value **FALSE** means the vision framework disconnects com port failed.

bool InsertCmd(String Cmd, String CmdTitle); Insert a new command to vision framework. **Cmd** determines the command string while **CmdTitle** determines the description string for this command. Return value **TRUE** means the vision framework accept this command as stock command successfully. Return value **FALSE** means the vision framework accept this command as stock command failed.

bool InsertCmdByIndex(int CmdIndex, String Cmd, String CmdTitle); Insert a new command at a specific index to vision framework. **CmdIndex** determines the index. **Cmd** determines the command string while **CmdTitle** determines the description string for this command. Return value **TRUE** means the vision framework accept this command as stock command successfully. Return value **FALSE** means the vision framework accept this command as stock command failed.

bool DeleteCmdByIndex(int CmdIndex); Delete a command at a specific index. **CmdIndex** determines the index. Return value **TRUE** means the vision framework delete this command from stock command successfully. Return value **FALSE** means the vision framework delete this command from stock command failed.

bool DeleteCmdByTitle(AnsiString CmdTitle); Delete a command that the description string matches **CmdTitle**. Return value **TRUE** means the vision framework delete this command from stock command successfully. Return value **FALSE** means the vision framework delete this command from stock command failed.

bool ModifyCmdByIndex(int CmdIndex,AnsiString NewCmd); Modify command at a specific index. **CmdIndex** determines the index. **NewCmd** determines the new command string. Return value **TRUE** means the vision framework accept this modification successfully. Return value **FALSE** means the vision framework accept this modification failed.

bool ModifyCmdByTitle(AnsiString CmdTitle,AnsiString NewCmd); Modify command that the description string matches **CmdTitle**. **CmdTitle** determines the description string. **NewCmd** determines the new command string. Return value **TRUE** means the vision framework accept this modification successfully. Return value **FALSE** means the vision framework accept this modification failed.

bool ModifyCmdTitleByIndex(int CmdIndex,AnsiString NewCmdTitle); Modify command description string at a specific index. **CmdIndex** determines the index. **NewCmdTitle** determines the new command description string. Return value **TRUE** means the vision framework accept this modification successfully. Return value **FALSE** means the vision framework accept this modification failed.

bool ModifyCmdTitleByTitle(AnsiString CmdTitle,AnsiString NewCmdTitle); Modify command description string that the description string matches **CmdTitle**. **CmdTitle** determines the description string. **NewCmdTitle** determines the new command description string. Return value **TRUE** means the vision framework accept this modification successfully. Return value **FALSE** means the vision framework accept this modification failed.

bool ExecuteCmdByIndex(int CmdIndex, String Params); Send the command at a specific index to com port. **CmdIndex** determines the index. **Params** determines the companion parameters for the command. Return value **TRUE** means the vision framework sends this command successfully. Return value **FALSE** means the vision framework sends this command failed.

bool ExecuteCmdByTitle(String CmdTitle, String Params); Send the command whose description string matches **CmdTitle** to com port. **CmdTitle** determines the description string. **Params** determines the companion parameters for the command. Return value **TRUE** means the vision framework sends this command successfully. Return value **FALSE** means the vision framework sends this command failed.

void DirectSendCmd(String Cmd); Send the command **Cmd** to com port directly.

6-2-3 Events

void OnCanvasMouseDown (int X , int Y) ; Vision framework assert this event when users click within the region of canvas. The X and Y parameters indicates the position users clicked in the unit of pixel.

void OnCanvasMouseMove (int X , int Y) ; Vision framework assert this event when users move cursor within the region of canvas. The X and Y parameters indicates the position users moving in the unit of pixel.

void OnCanvasMouseUp (int X , int Y) ; Vision framework assert this event when users release the mouse button within the region of canvas. The X and Y parameters indicates the position users released in the unit of pixel.

void OnSurfaceFilled (Long SurfaceHandle) ; The acquisition flow is fully controlled by vision framework and vision framework will assert two successive events **“OnSurfaceFilled”、”OnCustomDraw”** to notify user to do something like **“Process the underlying image data”、“Draw the underlying image or somewhat”** whenever a frame buffer (or so-called surface) is filled with new image data just acquired by framegrabber. The **“OnSurfaceFilled”、”OnCustomDraw” two-event-operation-mode** is the key role in vision framework.

void OnCustomDraw (Long hDC) ; The acquisition flow is fully controlled by vision framework and vision framework will assert two successive events whenever a frame buffer (or so-called surface) is filled with new image data just acquired by framegrabber. The **“OnSurfaceFilled”、”OnCustomDraw” two-event-operation-mode** is the key role in vision framework.

void OnHardwareTrigger (Long IoStates) ; The vision framework assert this event just **after** the connected camera received a valid hardware trigger from digital input line. IoStates means the current state of I/O lines.

void OnCmdMsgArrived () ; Vision framework assert this event when com port receives a **“Command”**. A **“Command”** is a qualified message (the “qualified” message is the so-called “commands” sent from the peer and the format of commands can be configured by a leading character and a trailing character in vision framework) received by com port.

void OnTriggerMsgArrived () ; Vision framework assert this event when com port receives **“Trigger Command”**. A **“Trigger Command”** is a command that interior string (leading and trailing characters are exclusive) matches a specific string pattern (can be defined by the property **“TriggerMsg”**).

Chapter 7 Another way of Integration : Stand Alone Component

In addition to integrate vision system with vision framework, AISYS provides another way **"Stand Alone Component"** method to let user integrate with more flexilibity. The difference between vision framework and stand alone component method is : vision framework will do almost all for users (e.g. image rendering、zooming control、RS232 interfacing…etc) but stand alone component doesn't. Some users may prefer to integrate ALTAIR with stand alone component rather than vision framework because of needing more flexibility. Briefly speaking, stand alone component also provides a solution to integrate vision system in a manner of **"Pick、Place and Play"** just like vision framework except rendering image automatically. Let us introduce a **"Quick Start"** to demonstrate how to get a breakthrough to a vision system powered by stand alone component. Before we start the introduction, users should follow the instructions mensioned in part I、II and be sure that an ALTAIR framegrabber is installed to your computer properly.

7-1 Quick Start

7-1-1 Pick

1. Start Microsoft Visual Basic version 6.0 or Visual C++ 6.0 compiler as shown in Figure 16. (Upgrading Visual Basic or Visual C++ to service pack 5 or later is suggested). Here, we use Microsoft Visual Basic 6.0 as the reference platform.

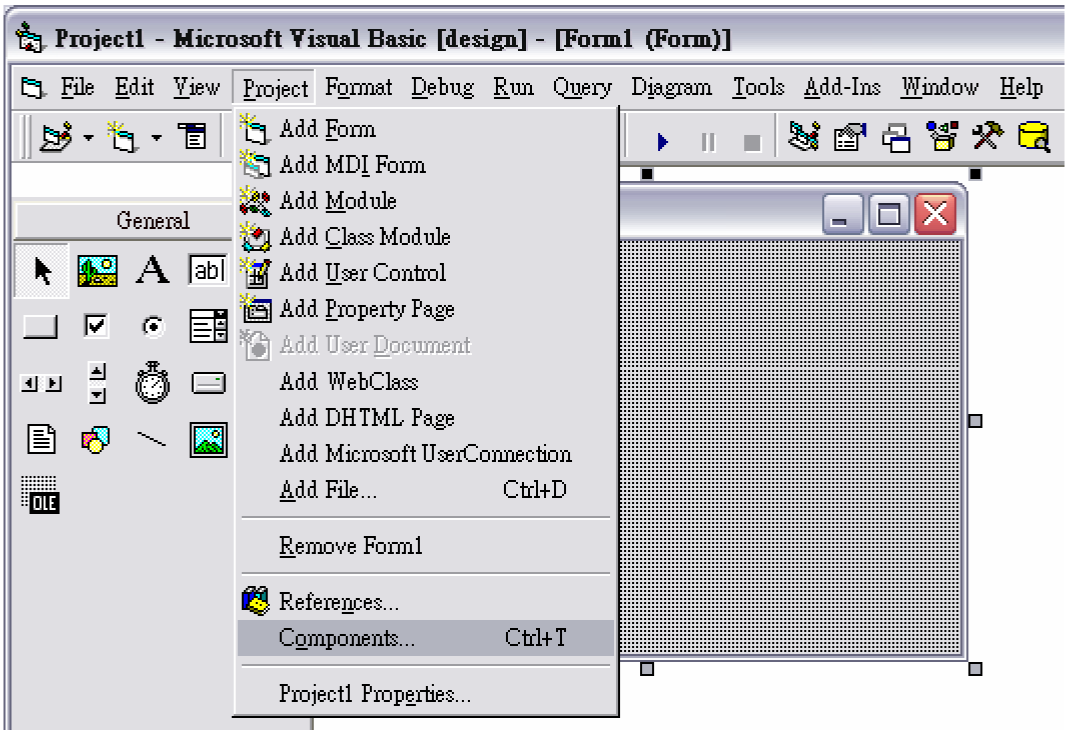
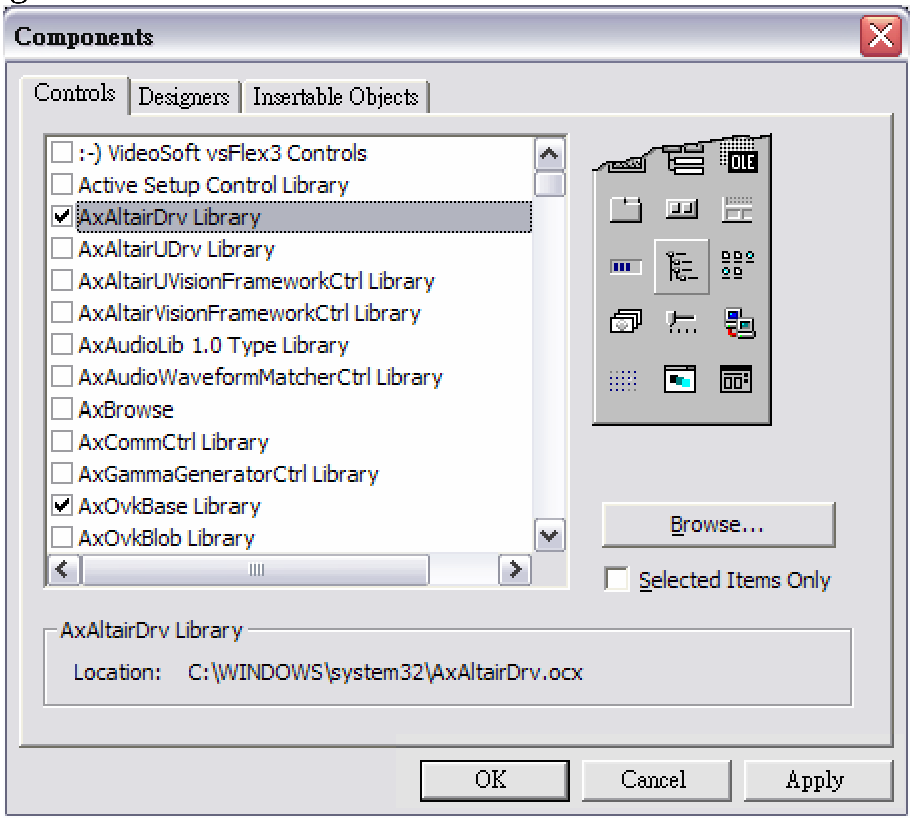


Figure 16.

2. Choose menu **Project** → **Components** → **Figure 1** → **Check AxAltairDrv Library and AxOvkBase**→ **OK Button** as shown in Figure 25、Figure 17.



Components

Controls | Designers | Insertable Objects

- [ ] :-) VideoSoft vsFlex3 Controls
- [ ] Active Setup Control Library
- [✓] AxAltairDrv Library
- [ ] AxAltairUDrv Library
- [ ] AxAltairUVisionFrameworkCtrl Library
- [ ] AxAltairVisionFrameworkCtrl Library
- [ ] AxAudioLib 1.0 Type Library
- [ ] AxAudioWaveformMatcherCtrl Library
- [ ] AxBrowse
- [ ] AxCommCtrl Library
- [ ] AxGammaGeneratorCtrl Library
- [✓] AxOvkBase Library
- [ ] AxOvkBlob Library

Browse...

[ ] Selected Items Only

AxAltairDrv Library

Location: C:\WINDOWS\system32\AxAltairDrv.ocx

OK | Cancel | Apply

Figure 17

3. You can see the components in toolbox after finishing above procedures that name "**AxAltair**" and "**AxCanvas**", as shown in Figure 18.

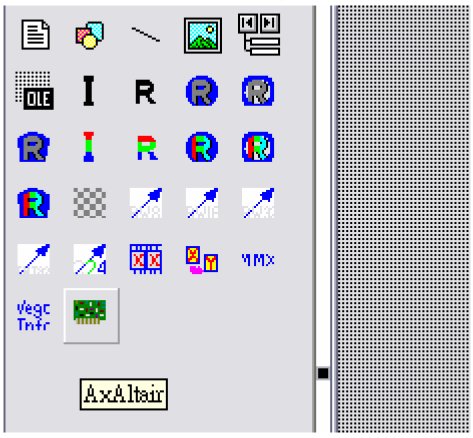


Figure 18

4. Pick the component from the palette as shown in Figure 19、Figure 20.

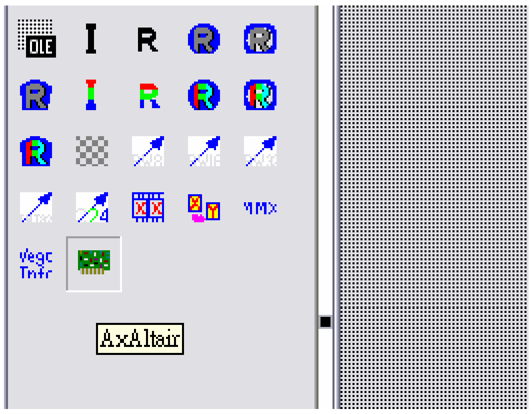


Figure 19.

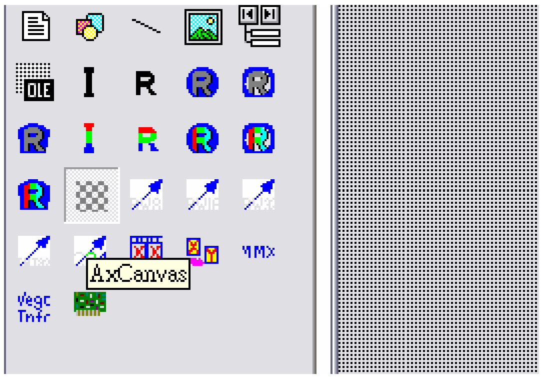


Figure 20.

7-1-2 Place

Place the components onto the main form and drag it to adjust the dimension as shown in Figure 21. Change the canvas geometry of AxCanvas component to 640x480 by assigning the properties of **AxAliatr** : **CanvasWidth=640**, **CanvasHeight=480** as shown in Figure 22.

7-1-3 Play

As mensioned above, stand alone component behaves just like vision framework except rendering image automatically. Now, let us construct the codes for rendering image. **AxAltair** supports two important events **"OnSurfaceFilled"** and **"OnSurfaceDraw"** to provide the chances of processing and rendering image for users. The event **"OnCustomDraw"** in **AxAltair** is similar as the event **"OnCustomDraw"** in **AxAltairVisionFramework** except that **AxAltairVisionFramework** draw the image automatically before the event **"OnCustomDraw"** is invoked but **AxAltair** doesn't. So that users should draw the image by themselves within the event **"OnSurfaceDraw"** to render image in stand alone component method.

First, override the event **"OnSurfaceFilled"** to write some codes to process the underlying image if necessary as shown in Figure 23.

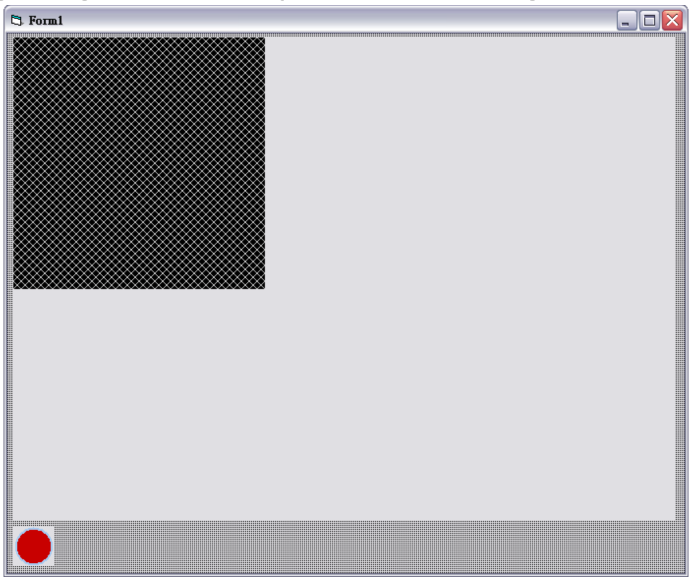


Figure 21.

Second, override the event **“OnSurfaceDraw”** to write some codes to render the underlying image as shown in Figure 24.

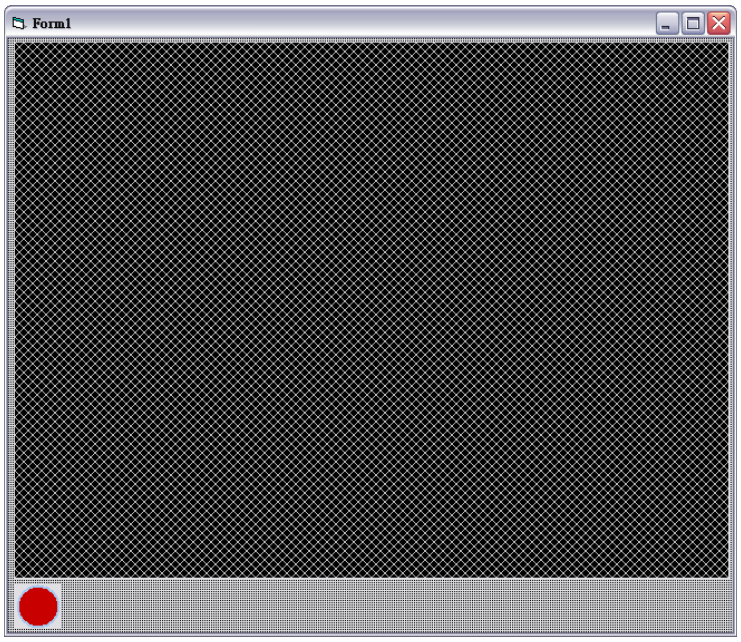
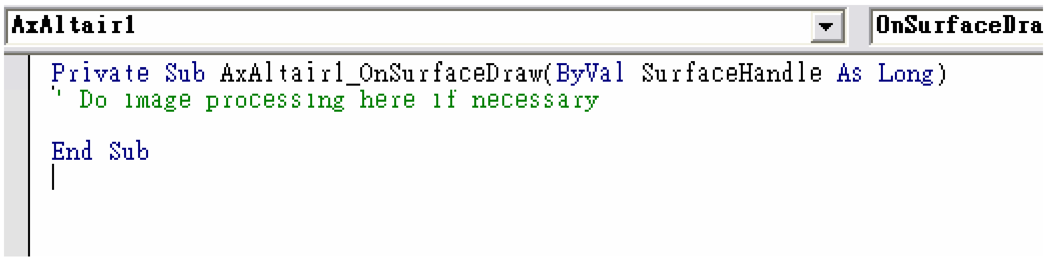


Figure 22.

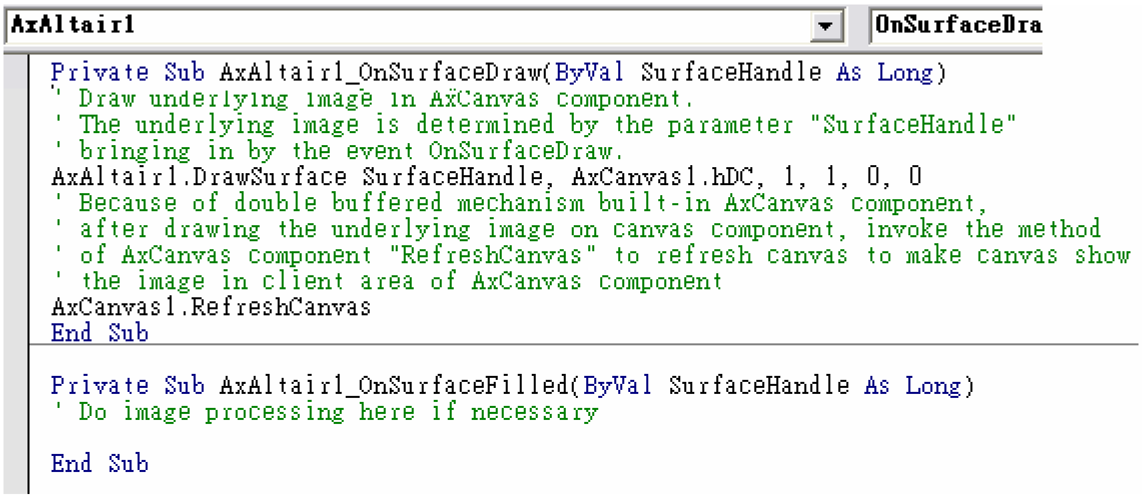
Here, add four buttons with captions “Control Panel”, “Live”, “Freeze”, “Snap” and add some codes to control AxAltairU component to enter associated modes accordingly as shown in Figure 25 and Figure 26.



```
AxAltair1                                          OnSurfaceDra
Private Sub AxAltair1_OnSurfaceDraw(ByVal SurfaceHandle As Long)
' Do image processing here if necessary

End Sub
```

Figure 23.



```
AxAltair1                                          OnSurfaceDra
Private Sub AxAltair1_OnSurfaceDraw(ByVal SurfaceHandle As Long)
' Draw underlying image in AxCanvas component.
' The underlying image is determined by the parameter "SurfaceHandle"
' bringing in by the event OnSurfaceDraw.
AxAltair1.DrawSurface SurfaceHandle, AxCanvas1.hDC, 1, 1, 0, 0
' Because of double buffered mechanism built-in AxCanvas component,
' after drawing the underlying image on canvas component, invoke the method
' of AxCanvas component "RefreshCanvas" to refresh canvas to make canvas show
' the image in client area of AxCanvas component
AxCanvas1.RefreshCanvas
End Sub

Private Sub AxAltair1_OnSurfaceFilled(ByVal SurfaceHandle As Long)
' Do image processing here if necessary

End Sub
```

Figure 24

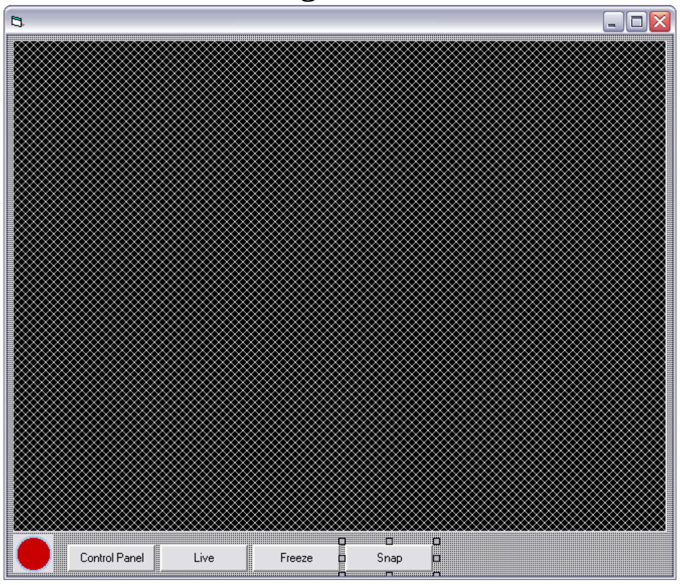
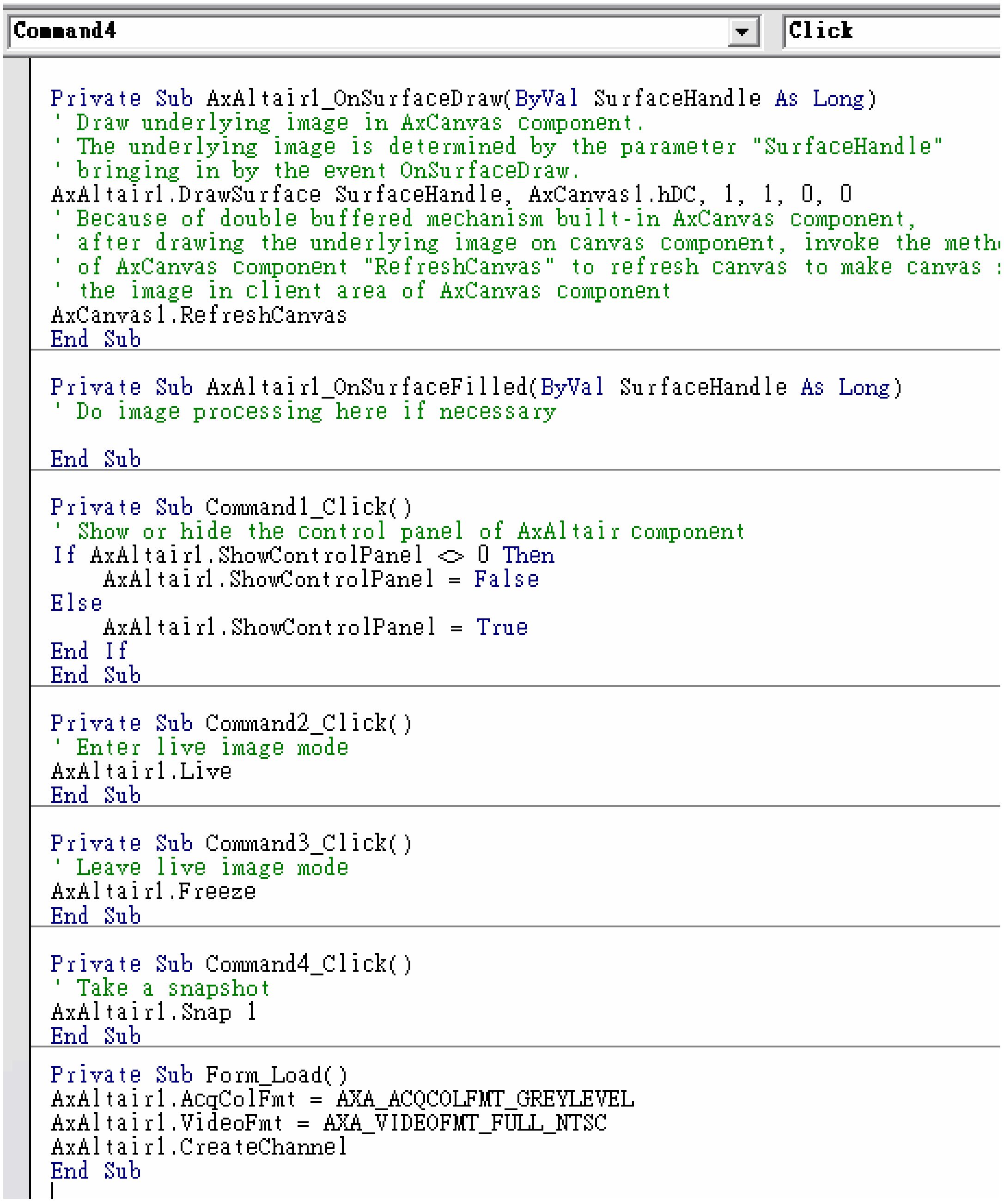


Figure 25



```
Private Sub AxAltair1_OnSurfaceDraw(ByVal SurfaceHandle As Long)
' Draw underlying image in AxCanvas component.
' The underlying image is determined by the parameter "SurfaceHandle"
' bringing in by the event OnSurfaceDraw.
AxAltair1.DrawSurface SurfaceHandle, AxCanvas1.hDC, 1, 1, 0, 0
' Because of double buffered mechanism built-in AxCanvas component,
' after drawing the underlying image on canvas component, invoke the meth
' of AxCanvas component "RefreshCanvas" to refresh canvas to make canvas :
' the image in client area of AxCanvas component
AxCanvas1.RefreshCanvas
End Sub

Private Sub AxAltair1_OnSurfaceFilled(ByVal SurfaceHandle As Long)
' Do image processing here if necessary

End Sub

Private Sub Command1_Click()
' Show or hide the control panel of AxAltair component
If AxAltair1.ShowControlPanel <> 0 Then
    AxAltair1.ShowControlPanel = False
Else
    AxAltair1.ShowControlPanel = True
End If
End Sub

Private Sub Command2_Click()
' Enter live image mode
AxAltair1.Live
End Sub

Private Sub Command3_Click()
' Leave live image mode
AxAltair1.Freeze
End Sub

Private Sub Command4_Click()
' Take a snapshot
AxAltair1.Snap 1
End Sub

Private Sub Form_Load()
AxAltair1.AcqColFmt = AXA_ACQCOLFMT_GREYLEVEL
AxAltair1.VideoFmt = AXA_VIDEOFMT_FULL_NTSC
AxAltair1.CreateChannel
End Sub
```

Figure 26

Now, a simple example for stand alone component method is completed. Press the button **"Start"** of compiler to start the vision system and then press the button **"Live"** to enter **"Live Image"** mode as shown in Figure 27.

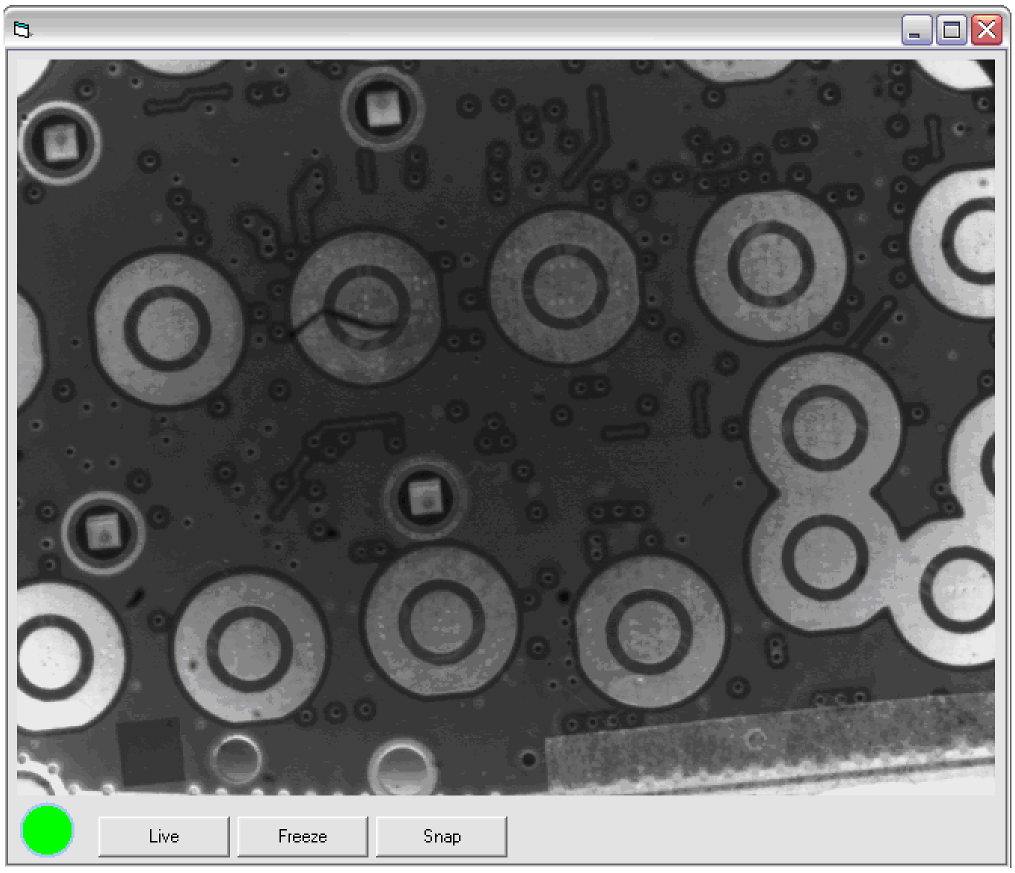


Figure 27

7-2 Programming Model

Thanks to **Microsoft ActiveX™ component model** , all the software products from ***AISYS*** adopts P-M-E (Property-Method-Event) programming model. The acquisition flow is fully controlled by AxAltair component and the component will assert two successive events **“OnSurfaceFilled”** and **”OnSurfaceDraw”** to notify user to do something like **“Process the underlying image data”**、**“Draw the underlying image or somewhat”** whenever a frame buffer (or so-called surface) is filled with new image data just acquired by framegrabber. The **“OnSurfaceFilled”**、**”OnSurfaceDraw” two-event-operation-mode** is the key role in AxAltair component.

Prototype : void OnSurfaceFilled (long SurfaceHandle) ;

The event **“OnSurfaceFilled”** bring in a handle of surface which is a handle of complex data structure that contains the image data just acquired. If user adopt OVK Framework as the image processing library, one can directly pass the handle to the vision library to complete the analysis tasks. If user adopt their own image processing routines, one can invoke the method of AxAltair component **“long GetImagePtr (long SurfaceHandle , int X , int Y)”** by passing the handle of surface as the first parameter and the starting position in the image as the second、third parameters in the event **“OnSurfaceFilled”** to get the physical memory address the image stored in. For example,

```
void AxAltair::OnSurfaceFilled ( long SurfaceHandle )
{
        // Fetch the physical address of image ( Top-left corner )
        pPhyImageAddr=(BYTE*) AxAltair1->GetImagePtr ( SurfaceHandle , 0 , 0 ) ;
        // Processing this image
        OemProcessingRoutine ( pPhyImageAddr ) ;
}
```

Prototype : void OnSurfaceDraw (long SurfaceHandle) ;

The event **"OnSurfaceDraw"** bring in a handle of surface which is a handle of complex data structure that contains the image data just acquired. For example,

```
void AxAltair::OnSurfaceDraw ( long SurfaceHandle )
{
        Float ZoomX=1.0 , ZoomY=1.0;
        Int PanX=0 , PanY=0;
        // Users should draw the underlying image here because AxAltair component doesn't draw it
        // by themselves
        AxAltair1->DrawSurface ( SurfaceHandle , AxCanvas1->hDC , ZoomX , ZoomY , PanX , PanY ) ;
        // Select white pen and draw a white straight line from ( 0 , 0 ) to ( 100 , 100 )
        SelectObject ( (HDC) AxCanvas1->hDC , GetStockObject (WHITE_PEN) ) ;
        MoveToEx ( (HDC) AxCanvas1->hDC ,
                        0* ZoomX+ PanX ,
                        0* ZoomY+ PanY ) ;
        LineTo ( (HDC) AxCanvas1->hDC ,
                     100* ZoomX+ PanX ,
                     100* ZoomY+ PanY ) ;
        // Remember to refresh the AxCanvas object you draw
        AxCanvas1->RefreshCanvas();
}
```

Typically, users can arrange image processing code fragment in the event **"OnSurfaceFilled"** while arranging image drawing code fragment in the event **"OnSurfaceDraw"**. The **"OnSurfaceFilled" 、" OnSurfaceDraw"** **two-event-operation-mode** is just the key reason that **AxAltair** component can dramatically reduce the effort of developing and maintaining software.

7-3 Programming Interface of AxAltair Component

7-3-1 Properties

TxAxaPortID PortID (R/W, Default=AXA_PORT_0); This property determine the index of framegrabber users would like to connect. Typically, after assigning **PortID**, invoking the method **"CreateChannel"** to connect the framegrabber specified by **PortID**. The available options are as the following:

AXA_PORT_0	**0 in decimal**
AXA_PORT_1	**1 in decimal**
AXA_PORT_2	**2 in decimal**
AXA_PORT_3	**3 in decimal**

TxAxaAcqColFmt AcqColFmt (R/W, Default=AXA_ACQCOLFMT_GREYLEVEL, 0 in decimal); This property determine the color format of connected camera. The available options are as the following:

AXA_ACQCOLFMT_GREYLEVEL	**0 in decimal**
AXA_ACQCOLFMT_RGB24	**1 in decimal**

TxAxaVideoFmt VideoFmt (R/W, Default=AXA_VIDEOFMT_FULL_NTSC, 0 in decimal); This property determine the video sync format of connected camera. The available options are as the following:

AXA_VIDEOFMT_FULL_NTSC	**0 in decimal**
AXA_VIDEOFMT_FULL_PAL	**1 in decimal**
AXA_VIDEOFMT_CIF_NTSC	**2 in decimal**
AXA_VIDEOFMT_CIF_PAL	**3 in decimal**

int GrabCount (R/W, Default=0); This property determine the number of frames need to acquire. Vision framework control ALTAIR series framegrabbers by two properties **"GrabCount"、"ChannelState"** mainly. The property **"ChannelState"** controls the operation mode of framegrabber. There are two operation modes of framegrabber available **"ACTIVE"、"IDLE"**. See the property **"ChannelState"** for details. **"GrabCount"** will be decreased by 1 when each acquisition is finished. When **"GrabCount"** becomes zero, framegrabber stop acquiring although the **"ChannelState"** is still in **"ACTIVE"** mode. It mean acquires image infinitely if **"GrabCount"** is -1 and **"ChannelState"** is in **"ACTIVE"** mode which is so-called **"Live Image"**.

int FrameRate (R/W, Default=30); This property determine the data output rate (so-called frame rate) of the framegrabber.

bool IsPortCreated (Read Only); Read this property to determine whether the framegrabber is connected by **AxAltair** component or not.

TxAxaChannelState ChannelState (R/W, Default=AXA_CHANNELSTATE_IDLE, 0 in decimal); This property determine the operation mode of the channel. This property should be configured combined with the property **"GrabCount"**. See the property **"GrabCount"** for the detail. The available options are the followings:

AXA_CHANNELSTATE_IDLE	**0 in decimal**
AXA_CHANNELSTATE_ACTIVE	**1 in decimal**

int Brightness (R/W, Default=128); This property determine the brightness of video amplifier in ALTAIR framegrabber. It ranges from 0 to 255 and the neutral value is 128.

int Contrast (R/W, Default=108); This property determine the contrast of video amplifier in ALTAIR framegrabber. It ranges from 0 to 255 and the neutral value is 108.

int Hue (R/W, Default=0); This property determine the hue of ALTAIR framegrabber. It ranges 0 to 255.

int ChromaU (R/W, Default=127); This property determine the U gain of chroma in ALTAIR framegrabber. It ranges from 0 to 255 and the neutral value is 127.

int ChromaV (R/W, Default=127); This property determine the V gain of chroma in ALTAIR framegrabber. It ranges from 0 to 255 and the neutral value is 127.

int ImageWidth (Read Only); Read this property to determine the width of image in pixels that the framegrabber output. This property will be changed according to the property **"VideoFmt"**.

int ImageHeight (Read Only); Read this property to determine the height of image in pixels that the framegrabber output. This property will be changed according to the property **"VideoFmt"**.

7-3-2 Methods

bool CreateChannel() ; Connect to ALTAIR framegrabber. This method should be invoked before any attempts to control ALTAIR framegrabber. Before invoking this method, users should configure **PortID**、**VideoFmt**、**AcqColorFmt** properties to appropriate values. Return value **TRUE** means the vision framework connects framegrabber successfully. Return value **FALSE** means the vision framework connects framegrabber failed and users should check the hardware and device driver installation to shoot where the trouble is.

bool DestroyChannel() ; Disconnect from ALTAIR framegrabber. This method should be invoked if the connected is not used anymore. Return value **TRUE** means the vision framework disconnects from framegrabber successfully. Return value **FALSE** means the vision framework disconnects from framegrabber failed and users should check the hardware and device driver installation to shoot where the trouble is.

void Live() ; Invoke this method makes vision framework enter **"Live Image"** mode. At this time, the framegrabber acquires image infinitely and assert two successive events **"OnSurfaceFilled"、"OnCustomDraw"** to notify user to complete necessary tasks. Invoking this method is equivalent to the following code fragment:

```
AxAltair1.GrabCount = -1
Ax Altair1.ChannelState = AXA_CHANNELSTATE_ACTIVE
```

void Freeze() ; Invoke this method makes vision framework enter **"Freeze Image"** mode. At this time, the framegrabber stop acquiring image. Invoking this method is equivalent to the following code fragment:

```
Ax Altair1.ChannelState = AXA_CHANNELSTATE_IDLE
Ax Altair1. GrabCount = 0
```

void Snap(int nFrames) ; Invoke this method makes vision framework enter "Snap Image" mode. At this time, the framegrabber acquires **nFrames** images and assert two successive events **"OnSurfaceFilled"、"OnCustomDraw"** to notify user to complete necessary tasks for each acquired image. Invoking this method is equivalent to the following code fragment:

```
Ax Altair1.ChannelState = AXA_CHANNELSTATE_IDLE
AxAltair1.GrabCount = 1
Ax Altair1.ChannelState = AXA_CHANNELSTATE_ACTIVE
```

void SaveFile(String FileName , TxAxImageFileFormat FileFormat) ; Invoke this method makes **AxAltair** component saves the current frame buffer to image file with **FileName** as name and **FileFormat** as file format. The available options for **FileFormat** is the following:

AX_IMAGE_FILE_TYPE_GREYLEVEL_BMP: Greylevel in BMP format
AX_IMAGE_FILE_TYPE_FULLCOLOR_BMP: True color in BMP format
AX_IMAGE_FILE_TYPE_GREYLEVEL_JPG: Greylevel in JPEG format
AX_IMAGE_FILE_TYPE_FULLCOLOR_JPG: True color in JPEG format

long DrawSurface (Long SurfaceHandle , Long hDC , float ZoomX , float ZoomY , int PanX , int PanY) ; Draw the image specified by **SurfaceHandle** onto the specified canvas. Returned value means the number of scanlines actually drawed.

long GetImagePtr (Long SurfaceHandle , int X , int Y) ; Get the physical memory address of the image data specified by **SurfaceHandle** and starting position **(X , Y)**. Returned value is the physical memory address of the specified image data at the starting position **(X , Y)**. Users can cast the returned value to a byte pointer and thus can be a data source for the coming procession.

TxAxaIoState GetIoState(int Port); ALTAIR framegrabber support 4 In / 4 Out digital IO. Invoke this method to read the current state of input or output port. **Port** determine the index of port users would like to read which ranges from 0 to 7.The returned value determines the state of the specified port. The state are the followings:

AXA_IO_STATE_LOW	**0 in decimal**
AXA_IO_STATE_HIGH	**1 in decimal**
AXA_IO_STATE_NULL	**2 in decimal**

"AXA_IO_STATE_NULL" indicates the failure of execution of the method.

void SetIoState(int Port, TxAxaIoState IoState); Invoke this method to write the state of the specified output port. **Port** determines the port index users specified. **IoState** determines the state of the port to change. The available options are listed as above.

float GetFocusRatio (Long SurfaceHandle) ; Estimate the focus ratio of the image specified by **SurfaceHandle**. This is a very nice feature that help users to design an auto-focus system without any other software or hardware. When the focus is reached, this focus ratio will reach a local maximum thus users can refer to this ratio to control the motion device and search the best focus easily. The algorithm of this focus ratio is very fast and robust so that the estimation is almost real-time and will not low down the frame rate at all.

7-3-3 Events

void OnSurfaceFilled (Long SurfaceHandle) ; The acquisition flow is fully controlled by **AxAltair** component and **AxAltair** component will assert two successive events **"OnSurfaceFilled"**、**"OnCustomDraw"** to notify user to do something like **"Process the underlying image data"** 、**"Draw the underlying image or somewhat"** whenever a frame buffer (or so-called surface) is filled with new image data just acquired by framegrabber. Here, users can write some codes for image processing or somewhat.

void OnSurfaceDraw (Long SurfaceHandle) ; Same as above. Here, users can write some codes for drawing image or somewhat..

void OnHardwareTrigger (Long IoStates) ; The **AxAltair** component assert this event just **after** the framegrabber received a valid hardware trigger from digital input line. IoStates means the current state of I/O lines.

7-4 Programming Interface of AxCanvas Component

7-4-1 Properties

int CanvasWidth (R/W, Default=256); This property determines the width of drawing area in AxCanvas compoment which in unit of pixels. If the width or height of drawing area is larger than the width or height of AxCanvas component, horizontal or vertical scroll bar will appear automatically to help users to view the whole drawing area.

int CanvasHeight (R/W, Default=256); This property determines the height of drawing area in AxCanvas compoment which in unit of pixels. If the width or height of drawing area is larger than the width or height of AxCanvas component, horizontal or vertical scroll bar will appear automatically to help users to view the whole drawing area.

long hDC (Read Only); This property determines the handle of device context of the drawing area in AxCanvas compoment. If users need to draw some primitives, the hDC is needed for drawing. Because of the double buffered mechanism in AxCanvas, users need to "refresh" AxCanvas component manually by invoking the method **"RefreshCanvas"** to make the drawing area keep updated.

TxAxColor PenColor (R/W, Default=AX_COLOR_WHITE, 20 in decimal); This property determines the pen color of the drawing area in AxCanvas component. **Note that users should configure pen color by this property instead of using windows API to avoid GDI confliction.** The available options are as the following :

AX_COLOR_AQUA	**0 in decimal**
AX_COLOR_BLACK	**1 in decimal**
AX_COLOR_BLUE	**2 in decimal**
AX_COLOR_CREAM	**3 in decimal**
AX_COLOR_DKGRAY	**4 in decimal**
AX_COLOR_FUCHSIA	**5 in decimal**
AX_COLOR_GRAY	**6 in decimal**
AX_COLOR_GREEN	**7 in decimal**
AX_COLOR_LIME	**8 in decimal**
AX_COLOR_LTGRAY	**9 in decimal**
AX_COLOR_MAROON	**10 in decimal**

AX_COLOR_MEDGRAY	**11 in decimal**
AX_COLOR_MONEYGREEN	**12 in decimal**
AX_COLOR_NAVY	**13 in decimal**
AX_COLOR_OLIVE	**14 in decimal**
AX_COLOR_PURPLE	**15 in decimal**
AX_COLOR_RED	**16 in decimal**
AX_COLOR_SILVER	**17 in decimal**
AX_COLOR_SKYBLUE	**18 in decimal**
AX_COLOR_TEAL	**19 in decimal**
AX_COLOR_WHITE	**20 in decimal**
AX_COLOR_YELLOW	**21 in decimal**
AX_COLOR_OTHER	**22 in decimal**

Note that if **PenColor =AX_COLOR_OTHER**, users can customize pen color by assigning color constant to the property **"CustomPenColor"**. See the next property for more details.

long CustomPenColor (R/W, Default= 16777215); This property determines the customized pen color when **PenColor =AX_COLOR_OTHER**. The color value is encoded by the following method : 0x00BBGGRR in hexdecimal. Each color component have 8 bits (256 levels) to quantize. For example, pure red can be expressed as 0x000000FF = 255 ; pure green can be expressed as 0x0000FF00 = 65280 ; pure blue can be expressed as 0x00FF0000 = 16711680.

int PenWidth (R/W, Default= 1); This property determines the width of the pen of drawing area in AxCanvas component. **Note that users should configure pen width by this property instead of using windows API to avoid confliction.**

TxAxPenStyle PenStyle (R/W, Default= AX_PENSTYLE_SOLID, 0 in decimal); This property determines the pen style of the drawing area in AxCanvas component. **Note that users should configure pen style by this property instead of using windows API to avoid GDI confliction.** The available options are as the following :

AX_PENSTYLE_SOLID	**0 in decimal**
AX_PENSTYLE_DASH	**1 in decimal**
AX_PENSTYLE_DOT	**2 in decimal**
AX_PENSTYLE_DASHDOT	**3 in decimal**

AX_PENSTYLE_DASHDOTDOT	**4 in decimal**
AX_PENSTYLE_CLEAR	**6 in decimal**
AX_PENSTYLE_INSIDEFRAME	**7 in decimal**

TxAxColor FontColor (R/W, Default=AX_COLOR_BLACK, 1 in decimal); This property determines the font color of the drawing area in AxCanvas component. **Note that users should configure font color by this property instead of using windows API to avoid GDI confliction.** The available options are as the following :

AX_COLOR_AQUA	**0 in decimal**
AX_COLOR_BLACK	**1 in decimal**
AX_COLOR_BLUE	**2 in decimal**
AX_COLOR_CREAM	**3 in decimal**
AX_COLOR_DKGRAY	**4 in decimal**
AX_COLOR_FUCHSIA	**5 in decimal**
AX_COLOR_GRAY	**6 in decimal**
AX_COLOR_GREEN	**7 in decimal**
AX_COLOR_LIME	**8 in decimal**
AX_COLOR_LTGRAY	**9 in decimal**
AX_COLOR_MAROON	**10 in decimal**
AX_COLOR_MEDGRAY	**11 in decimal**
AX_COLOR_MONEYGREEN	**12 in decimal**
AX_COLOR_NAVY	**13 in decimal**
AX_COLOR_OLIVE	**14 in decimal**
AX_COLOR_PURPLE	**15 in decimal**
AX_COLOR_RED	**16 in decimal**
AX_COLOR_SILVER	**17 in decimal**
AX_COLOR_SKYBLUE	**18 in decimal**
AX_COLOR_TEAL	**19 in decimal**
AX_COLOR_WHITE	**20 in decimal**
AX_COLOR_YELLOW	**21 in decimal**
AX_COLOR_OTHER	**22 in decimal**

Note that if **FontColor =AX_COLOR_OTHER**, users can customize font color by assigning color constant to the property **"CustomFontColor"**. See the next property for more details.

long CustomFontColor (R/W, Default= 16777215); This property determines the customized font color when **FontColor**

=AX_COLOR_OTHER. The color value is encoded by the following method : 0x00BBGGRR in hexdecimal. Each color component have 8 bits (256 levels) to quantize. For example, pure red can be expressed as 0x000000FF = 255 ; pure green can be expressed as 0x0000FF00 = 65280 ; pure blue can be expressed as 0x00FF0000 = 16711680.

TxAxColor BrushColor (R/W, Default=AX_COLOR_WHITE, 20 in decimal); This property determines the brush color of the drawing area in AxCanvas component. **Note that users should configure brush color by this property instead of using windows API to avoid GDI confliction.** The available options are as the following :

AX_COLOR_AQUA	**0 in decimal**
AX_COLOR_BLACK	**1 in decimal**
AX_COLOR_BLUE	**2 in decimal**
AX_COLOR_CREAM	**3 in decimal**
AX_COLOR_DKGRAY	**4 in decimal**
AX_COLOR_FUCHSIA	**5 in decimal**
AX_COLOR_GRAY	**6 in decimal**
AX_COLOR_GREEN	**7 in decimal**
AX_COLOR_LIME	**8 in decimal**
AX_COLOR_LTGRAY	**9 in decimal**
AX_COLOR_MAROON	**10 in decimal**
AX_COLOR_MEDGRAY	**11 in decimal**
AX_COLOR_MONEYGREEN	**12 in decimal**
AX_COLOR_NAVY	**13 in decimal**
AX_COLOR_OLIVE	**14 in decimal**
AX_COLOR_PURPLE	**15 in decimal**
AX_COLOR_RED	**16 in decimal**
AX_COLOR_SILVER	**17 in decimal**
AX_COLOR_SKYBLUE	**18 in decimal**
AX_COLOR_TEAL	**19 in decimal**
AX_COLOR_WHITE	**20 in decimal**
AX_COLOR_YELLOW	**21 in decimal**
AX_COLOR_OTHER	**22 in decimal**

Note that if **BrushColor =AX_COLOR_OTHER**, users can customize font color by assigning color constant to the property **“CustomBrushColor”**. See the next property for more details.

long CustomBrushColor (R/W, Default= 16777215); This property determines the customized brush color when **BrushColor =AX_COLOR_OTHER**. The color value is encoded by the following method : 0x00BBGGRR in hexdecimal. Each color component have 8 bits (256 levels) to quantize. For example, pure red can be expressed as 0x000000FF = 255 ; pure green can be expressed as 0x0000FF00 = 65280 ; pure blue can be expressed as 0x00FF0000 = 16711680.

TxAxBrushStyle BrushStyle (R/W, Default= AX_BRUSHSTYLE_SOLID, 0 in decimal); This property determines the brush style of the drawing area in AxCanvas component. **Note that users should configure brush style by this property instead of using windows API to avoid GDI confliction.** The available options are as the following :

AX_BRUSHSTYLE_SOLID	**0 in decimal**
AX_BRUSHSTYLE_CROSS	**1 in decimal**
AX_BRUSHSTYLE_CLEAR	**2 in decimal**
AX_BRUSHSTYLE_DIAGCROSS	**3 in decimal**
AX_BRUSHSTYLE_BDIAGONAL	**4 in decimal**
AX_BRUSHSTYLE_FDIAGONAL	**5 in decimal**
AX_BRUSHSTYLE_HORIZONTAL	**6 in decimal**
AX_BRUSHSTYLE_VERTICAL	**7 in decimal**

7-4-2 Methods

void DrawText(String Text , int X , int Y , float ZoomX , float ZoomY , int PanX , PanY) ; Draw the specified text in drawing area at (X,Y) location with specified zooming and panning conditions. Because of the double buffered mechanism in AxCanvas, users need to "refresh" AxCanvas component manually by invoking the method **"RefreshCanvas"** to make the drawing area keep updated.

void DrawLine(int SX , int SY , int DX , int DY , float ZoomX , float ZoomY , int PanX , PanY) ; Draw the specified line with starting point (SX,SY) and ending point (DX,DY) in drawing area with specified zooming and panning conditions. Because of the double buffered mechanism in AxCanvas, users need to "refresh" AxCanvas component manually by invoking the method **"RefreshCanvas"** to make the drawing area keep updated.

void DrawSurface(Long SurfaceHandle , float ZoomX , float ZoomY , int PanX , PanY) ; Draw the specified image (specified by parameter **SurfaceHandle**) in drawing area with specified zooming and panning conditions. Because of the double buffered mechanism in AxCanvas, users need to “refresh” AxCanvas component manually by invoking the method **“RefreshCanvas”** to make the drawing area keep updated.

void RefreshCanvas() ; Make the drawing area keep updated. Normally, users can invoke this method once after **all** the drawing routines finished to keep the best rendering performance. For example,

```
void RefreshImage ()
{
        // Draw image in AxCanvas
        AxCanvas1->DrawSurface ( SurfaceHandle , 1 , 1 , 0 , 0 ) ;
        // Draw other primitives in AxCanvas
        DrawOtherPrimitivesRoutine (AxCanvas1->hDC) ;
        // Finally, refresh AxCanvas to render the result
        AxCanvas1->RefreshCanvas();
}
```

7-4-3 Events

void OnCanvasMouseDown (int X , int Y) ; AxCanvas component assert this event when users click within the region of drawing area. The X and Y parameters indicates the position users clicked in the unit of pixel.

void OnCanvasMouseMove (int X , int Y) ; AxCanvas component assert this event when users move cursor within the region of drawing area. The X and Y parameters indicates the position users moving in the unit of pixel.

void OnCanvasMouseUp (int X , int Y) ; AxCanvas component assert this event when users release the mouse button within the region of drawing area. The X and Y parameters indicates the position users released in the unit of pixel.