DAQBench

32-bit ActiveX controls for Measurement and Automation

User's Guide

Recycled Paper

Trademarks

Getting Service from ADLINK

- **Customer Satisfaction is always the most important thing for ADLINK Tech Inc. If you need any help or service, please contact us and get it.**

ADLINK Technology Inc.			
Web Site	http://www.adlink.com.tw http://www.adlinktechnology.tw		
Sales & Service	service@ADLINK.com.tw		
Technical	NuDAQ	nudaq@ADLINK.com.tw	
Support	NuDAM	nudam@ADLINK.com.tw	
	NuIPC	nuipc@ADLINK.com.tw	
	NuPRO	nupro@ADLINK.com.tw	
	Software	sw@ADLINK.com.tw	
	AMB	amb@ADLINK.com.tw	
TEL	+886-2-82265877	FAX	+886-2-82265717
Address	9F, No. 166, Jian Yi Road, Chungho City, Taipei, 235 Taiwan, R.O.C.		

- **Please inform or FAX us of your detailed information for a prompt, satisfactory and constant service.**

Detailed Company Information			
Company/Organization			
Contact Person			
E-mail Address			
Address			
Country			
TEL		FAX	
Web Site			

Questions	
Product Model	
Environment to Use	☐OS: ______ ☐Computer Brand: ______ ☐M/B: ☐CPU: ☐Chipset: ☐Bios: ☐Video Card: ☐Network Interface Card: ☐Other:
Challenge Description	
Suggestions for ADLINK	

Table of Contents

How to Use This Guide

This manual is designed to help you use the DAQBench software package for developing your measurement or automation applications. The manual describes how to install and use the software to meet your requirements and help you program your own software applications. It is organized as follows.

The DAQBench *User' s Guide* is organized as follows:

Chapter 1, "*Introduction to DAQBench*" , contains an overview of DAQBench, lists the DAQBench system requirements, describes how to install the software, and explains the basics of ActiveX controls.

Chapter 2, "*Building DAQBench Applications with Visual Basic*" , describes how you can use DAQBench controls with Visual Basic; insert the controls into the Visual Basic environment, set their properties, and use their methods and events; and perform their operations using ActiveX controls in general. This chapter also outlines Visual Basic features that simplify working with ActiveX controls.

Chapter 3, "*Building DAQBench Applications with Visual C++*" , describes how you can use DAQBench controls with Visual C++, explains how to insert the controls into the Visual C++ environment and create the necessary wrapper classes, shows you how to create an application compatible with the DAQBench controls using the Microsoft Foundation Classes Application Wizard (MFC AppWizard) and how to build your program using the ClassWizard with the controls, and discusses how to perform these operations using ActiveX controls in general.

Chapter 4, "*Building DAQBench Applications with Delphi*" , describes how you can use DAQBench controls with Delphi; insert the controls into the Delphi environment, set their properties, and use their methods and events; and perform these operations using ActiveX controls. This chapter also outlines Delphi features that simplify working with ActiveX controls.

Chapter 5, *"Introducing the ActiveX Controls of DAQBench"*, simply describes all ActiveX controls of DAQBench; explains the individual controls, their object structure and different style of control.

Chapter 6, *"NuDAQ Configuration"*, describes how you can use the NuDAQ Configuraton Utility to register NuDAQ cards on Widnows 98/NT/2000 and define local or remote devices.

1

Introduction to DAQBench

This chapter contains an overview of DAQBench, lists the DAQBench system requirements, describes how to install the software, and explains the basics of ActiveX controls.

1.1 What is DAQBench?

DAQBench is a collection of ActiveX controls for acquiring, analyzing, and presenting data within any compatible ActiveX control container. ActiveX controls are also known as OLE (Object Linking and Embedding) controls, and the two terms can be used interchangeably in this context. Use the online reference for specific information about the properties, methods, and events of the individual ActiveX controls.

With DAQBench, you can easily develop complex custom user interfaces to display your data, control your ADLINK Data Acquisition (DAQ) boards, and analyze data you acquired or received from some other sources. The DAQBench package contains the following components:

User Interface Controls -- 32-bit ActiveX controls for presenting your data in a technical format. These controls include a graph/chart control, sliders, thermometers, tanks, knobs, seven segment, meters, LEDs, and switches.

NuDAQ PCI Controls -- 32-bit ActiveX controls for analog I/O, digital I/O, and counter/timer I/O operations using ADLINK NuDAQ PCI-bus cards.

NuDAM Controls -32-bit ActiveX controls for controlling and retrieving data from NuDAM modules connected to a serial port in your computer.

Analysis Library Control -- Functions for basic statistics, vector and matrix algebra , array manipulations and FFT operation. These functions are packaged in one 32-bit ActiveX control.

Equipment Controls -- 32-bit ActiveX controls for displaying some popular equipment patterns in industry automation. These patterns is convenient to develop a MMI or SCADA system.

Thermocouple Control -- 32-bit ActiveX controls for Thermocouple operation. User can just use the voltage value as the input parameter, then the Thermocouple control converts the voltage value to the temperature data. The control now support J-type, K-type and T-type Thermocouple.

Information Integration Controls -- 32-bit ActiveX controls for integrating information system. These controls include Excel linker, Database access for ODBC, Web snapshot for browser and OPC client for OPC server.

The DAQBench ActiveX controls are designed for use in Visual Basic, a premier ActiveX control container application. However, you can use ActiveX controls in any application that supports them, including Visual C++, Access, and Delphi.

1.2 Installing DAQBench

The DAQBench setup program installs DAQBench through a process that lasts approximately five minutes.

1.2.1 System Requirements

To use the DAQBench ActiveX controls, you must have the following:

- ➢ Microsoft Windows 95/98/NT/2000 operating system. NuDAQ PCI OCX' s can only be used in Windows NT/98/2000 environment.
- ➢ Personal computer using 66 MHz 80486 or higher microprocessor
- ➢ VGA resolution (or higher) video adapter
- ➢ ActiveX control container such as Visual Basic (32-bit version), Visual C++, or Delphi (32-bit version)
- ➢ Minimum of 32 MB of memory
- ➢ Minimum of 10 MB of free hard disk space
- ➢ Microsoft-compatible mouse

1.2.2 Installation Instructions

This section provides instructions for installing different pieces of your DAQBench software. You can start most of these installers directly from the startup screen that appears when you load the "ADLINK DAQBench" CD.

Installing the DAQBench ActiveX Controls

Complete the following steps to install DAQBench.

Note: To install DAQBench on a Windows NT/2000 system, you must be logged in with Administrator privileges to complete the installation.

Make sure that your computer and monitor are turned on and that you have installed Windows 95/98/NT/2000.

Insert the "ADLINK DAQBench" CD in the CD-ROM drive of your computer. From the CD startup screen, click on *Install DAQBench*. If the CD startup screen does not appear, use the Windows Explorer or File Manager to run the *x*:\SETUP.EXE (*x* identifies the drive that contains the CD).

If you install DAQBench in Windows 98/NT/2000 environment, when the software component installation process is completed, Setup will launch the NuDAQ Configuration utility "NuDAQCfg" for you to make the NuDAQ PCI card's driver registries, board configuration and device definition. For a full description of the utility, see the chapter 6 "NuDAQ Configuration".

1.2.3 Installed Files

The DAQBench setup program installs the following groups of files on your hard disk.

ActiveX Controls

Directory:	\DAQBench\OCX
Files:	Digital.ocx, Multiple.ocx, DBUI.ocx, DBGraph.ocx, NDDigital.ocx, NDAnalog.ocx, NDHost.ocx, NDCounter.ocx, DQAnalysis.ocx, DBEquip.ocx, ExcelLinker.ocx, DBAccess.ocx, WebSnapshot.ocx, OPCClient.ocx, Thermocouple.ocx

NuDAQPCI utilities

Directory:	\DAQBench\PCIDAQ
Files:	DAQDevMgr.exe, DMProxyStub.dll, NuDAQCfg.exe, RDASvr .exe

Example Programs

Directory:	\DAQBench\Samples\VB \DAQBench\Samples\VC

PDF Manual Files

Directory:	\DAQBench\Manual

One-line Help Files

Directory:	Windows system directory (\Windows\system for Win95/98) (\Windows\system32 for WinNT/2000)
Files:	DAQBenchRef.hlp, DAQBenchRef.CNT

Utility Files

Directory:	\DAQBench\Util
Files:	DAQCvt.exe

VC++ Data Type Wraping Library Files

Directory:	\DAQBench\Varpacker
Files:	VarPacker.lib, VarPacker.dll, VarPacker.h

1.3 About the DAQBench Controls

Before learning how to use DAQBench, you should be familiar with using ActiveX controls. This section outlines some background information about ActiveX controls, in particular the DAQBench controls. If you are not already familiar with the concepts outlined in this section, make sure you understand them before continuing. You also might want to refer to your programming environment documentation for more information on using ActiveX (OLE) controls in your particular environment.

1.3.1 Properties, Methods, and Events

ActiveX controls consist of three different parts “ properties, methods, and events “ used to implement and program the controls.

Properties are the attributes of a control. These attributes describe the current state of the control and affect the display and behavior of the control. The values of the properties are stored in variables that are part of the control.

Methods are functions defined as part of the control. Methods are called with respect to a particular control and usually have some effect on the control itself. The operation of most methods also is affected by the current property values of the control.

Events are notifications generated by a control in response to some particular occurrence. The events are passed to the control container application to execute a particular subroutine in the program (event handler).

For example, the DAQBench DGraph control has a wide variety of properties that determine how the graph looks and operates. To customize the graph appearance and behavior, set properties for color, axes, scale, tick marks, and plots.

The DGraph control also has a series of high-level methods, or functions, that you can invoke to set several properties at once and to perform a particular operation. For example, you can use the PlotGraph method to pass an array of data to the DGraph control.

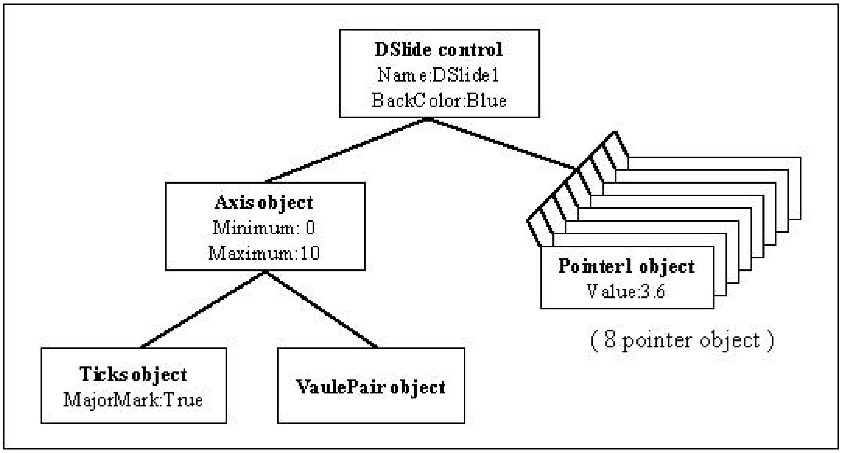
1.3.2 Object Hierarchy

As described in the previous section, each ActiveX control has properties, methods, and events. These three parts are stored in a software object, which is the piece of software that makes up the ActiveX control and contains all its parts. Certain ActiveX controls are very complex, containing many different parts (properties). For this reason, complex ActiveX controls are often subdivided into different software objects, the sum of which make up the ActiveX control. Each individual object in a control contains some specific parts (properties) and functionality (methods and events) of the ActiveX control. The relationships between different objects of a control are maintained in an object hierarchy. At the top of the hierarchy is the actual control itself.

This top-level object contains its own properties, methods, and events. Some of the top-level object properties are actually reference to other objects that define specific parts of the control. Objects below the top-level have their own methods and properties, and their properties can reference to other objects. The number of objects in this hierarchy is not limited.

Another advantage of subdividing controls is the re-use of different objects between different controls. One object might be used at different places in the same object hierarchy or in several different controls/object hierarchies.

The following illustration shows part of the object hierarchy of the DAQBench DSlide control.



The DSlide object contains some of its own properties, such as Name and BackColor. It also contains properties such as Axis and Pointers, which are separate objects from the DSlide object. The Axis object contains all the information about the axis used on the slide and has properties such as Maximum and Minimum. The Axis object contains Ticks object and ValuePair object of its own. Ticks object has properties, such as MajorMark, MajorColor, MinorMark, MinorColor. ValuePair object has properties, such as Count, a array of name, a array of value. If the style of DSlide is Numeric then Axis object use the information of Ticks object. Otherwise, use the information of ValuePair.. The DSlide object contains eight Pointer objects. Each Pointer object has its own properties, such as Value, PointerStyle, FillColor.

1.4 Setting the Properties of an ActiveX Control

You can modify the properties of an ActiveX control from its property pages or directly from the program.

1.4.1 Using Property Pages

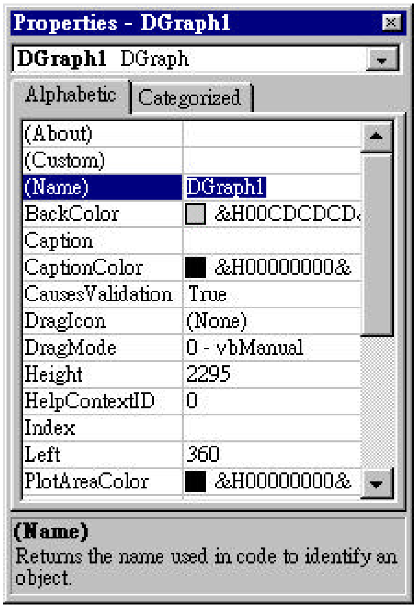
Property pages are common throughout the Windows 95/98/NT/2000 interface. When you want to change the appearance or options of a particular object, right click on the object and select Properties. A property page or tabbed dialog box appears with a variety of properties that you can set for that particular object. You customize ActiveX controls in exactly the same way. Once you place the control on a form in your programming environment, right click on the control and select **Properties...** to customize the appearance and operation of the control.

Use the property pages to set the property values for each ActiveX control while you are creating your application. The property values you select at this point represent the state of the control at the beginning of your application. You can change the property values from within your program, as shown in the next section, " Changing Properties Programmatically" .

In some programming environments (such as Visual Basic and Delphi), you have two different property pages. The property page common to the programming environment is called the default property sheet; it contains the most basic properties of a control.

Your programming environment assigns default values for some of the basic properties, such as the control name and the tab order. You must edit these properties through the default property sheet.

The following illustration shows the Visual Basic default property sheet for the DGraph control.


Properties - DGraph1

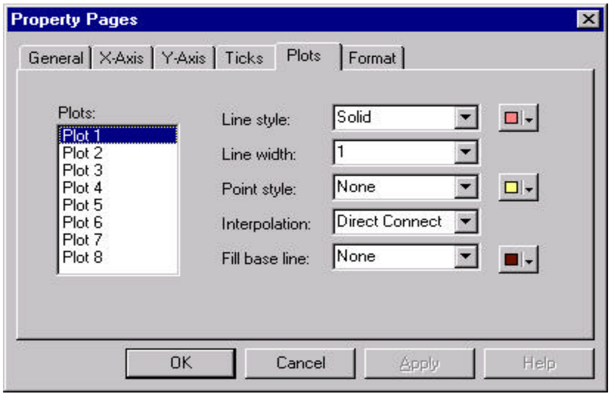
DGraph1 DGraph

(About)	
(Custom)	
(Name)	DGraph1
BackColor	⬜ &H00CDCDCD&
Caption	
CaptionColor	⬛ &H00000000&
CausesValidation	True
DragIcon	(None)
DragMode	0 - vbManual
Height	2295
HelpContextID	0
Index	
Left	360
PlotAreaColor	⬛ &H00000000&

(Name)
Returns the name used in code to identify an object.

Visual Basic Default Property Sheets

The second property sheet is called the custom property page. The layout and functionality of the custom property pages vary for different controls. The following illustration shows the custom property page for the DGraph control.


Property Pages

Plots: Plot 1, Plot 2, Plot 3, Plot 4, Plot 5, Plot 6, Plot 7, Plot 8

Line style: Solid
Line width: 1
Point style: None
Interpolation: Direct Connect
Fill base line: None

DAQBench Custom Property Pages

1.4.2 Changing Properties Programmatically

You can also set or read the properties of your controls programmatically. For example, if you want to change the state of an DBoolean control during program execution, change the Value property from True to False or from False to True. The exact syntax for reading and writing property values depends on your programming language, so consult the appropriate section of the Help system for using your programming environment. In this discussion, properties are set with Visual Basic syntax, which is similar to most implementations.

Each control you create in your program has a name (like a variable name) , it is a reference of the control in your program. You can set the value of a property on a top-level object with the following syntax.

```
ObjectName.property = new_value
```

For example, you can change the Value property of an DBoolean control to off by using the following line of code, where DBoolean1 is the default name of the DBoolean control.

```
    DBoolean1.Value = 0
```

To access properties of sub-objects referenced by the top-level object, use the control name, followed by the name of the sub-object and the property name. For example, consider the following code for the DAQBench graph control.

```
    DGraph1.YAxis.Log = True
```

In the above code, YAxis is a property of the DGraph control and refers to a Axis object.

Log is one of Axis properties. The DGraph control also has a XAxis property that refers to a different Axis object.

You can retrieve the value of control properties from your program in the same way. For example, you can print the value of the DBoolean control.

```
    Print DBoolean1.Value
```

You can display the Maximum used by the DGraph control in a Visual Basic text box with the following code.

```
    Text1.Text = DGraph1.YAxis.Maximum
```

1.5 Working with Control Methods

ActiveX controls and objects have their own methods, or functions, that you can call from your program. Methods can have parameters that are passed to the method and return values that pass information back to your program.

For example, the PlotGraph method for the DAQBench DGraph control has a required parameter --The array of data to be plotted -- that you must include when you call the method. If you want to plot the data returned from an Analog Input control, use the following line of code (the array ScaledData is automatically generated by the Pci9112 control).

```
    DGraph1.PlotGraph ScaledData
```

Depending on your programming environment, the parameters might be enclosed in parentheses. Visual Basic does not use parentheses to pass parameters if the function or method is not assigned a return variable. The ReadDIPort Method in the DAQ Digital Input control has the following form when used with a return variable lresult.

```
    lresult = Pci7250.ReadDIPort(port, value)
```

1.6 Developing Event Handler Routines

After you configure your controls on a form, you can create event handler routines in your program to respond to events generated by the controls. For example, the DAQ Analog Input control has an AiComplete event that fires (occurs) when the acquired data have completed.

You can configure the control to collect 1,000 points of data from a particular channel at a rate of 1,000 points per second. After ONE second, the data buffer is ready and the AiComplete event is fired. In your AiComplete event routine, you can write code to analyze the data buffer, plot it, or store it to disk.

To develop the event routine code, most programming environments generate a skeleton function to handle each event. The Visual Basic , Visual C++ , and Delphi sections outline how to generate these function skeletons to build your event handler routines. For example, the Visual Basic environment generates the following function skeleton into which you insert the functions to call when the AiComplete event occurs.

```
Private Sub Pci9112_AiComplete(ScaledData As Variant,
BinaryCodes As Variant)

End Sub
```

In most cases, the event also returns some data to the event handler, such as the ScaledData and BinaryCodes arrays in the previous example, that can be used in your event handler routine.

1.7 Using the Analysis Library

The DAQBench Analysis Library is packaged as one ActiveX control, named DQAnalysis. You can add analysis functions to your project in the same way you add user interface or data acquisition controls. After adding the Analysis controls to your programming environment, use the analysis functions like any other method on a control. To use any specific function, place the appropriate Analysis control on a form. In your program, call the name of the control followed by the name of the analysis function:

```
    MeanValue = DQAnalysis1.Mean (Data)
```

Consult the online reference for more information on the individual analysis functions and their use.

1.7.1 The Online Reference--Learning the Properties, Methods, and Events

The DAQBench online reference contains detailed information on each control and its associated properties, methods, and events. Refer to DAQBench online reference when you are using a control for the first time. Remember that many of the DAQBench controls share sub-objects and properties, so when you learn how to use one control, you also learn how to use others. You can open the online reference from within most programming environments by clicking on the **Help** button in the custom property pages

Some programming environments have built-in mechanisms for detailing the available properties, methods, and events for a particular control and sometimes include automatic links to the help file. Refer to the Help system on your particular programming environment to learn about additional tools.

2

Building DAQBench Applications with Visual Basic

This chapter describes how you can use the DAQBench controls with Visual Basic; insert the controls into the Visual Basic environment, set their properties, and use their methods and events; and perform these operations using ActiveX controls in general. This section of the online reference also outlines Visual Basic features that simplify working with ActiveX controls.

At this point you should be familiar with the general structure of ActiveX controls described in *Introduction to DAQBench* . The individual DAQBench controls are described in the various function references.

2.1 Developing Visual Basic Applications

The following procedure explains how you can start developing Visual Basic applications with DAQBench.

1. Select the type of application you want to build. Initially select a Standard EXE for your application type.
2. Design the form. A form is a window or area on the screen on which you can place controls and indicators to create the user interface for your program. The toolbox in Visual Basic contains all of the controls available for developing the form.
3. After placing each control on the form, configure the properties of the control using the default and custom property pages.

Each control on the form has associated code (event handler routines) in your Visual Basic program that automatically executes when the user operates that control.

4. To create this code, double click on the control while editing your application and the Visual Basic code editor opens to a default event handler routine.

2.1.1 Loading the DAQBench Controls into the Toolbox

Before building an application using the DAQBench controls and libraries, you must add them to the Visual Basic toolbox. The DAQBench ActiveX controls are divided into different groups including user interface controls (DBUI.OCX, DBGraph.OCX), data acquisition controls (Digital.OCX, Multiple.OCX), NuDAM controls (NDDigital.OCX, NDAnalog.OCX, NDHost.OCX, DNCounter.OCX), equipment controls(DBEquip.OCX), analysis library controls (DQAnalysis .OCX), ExcelLinker control (ExcelLinker.OCX), WebSnapshot control (WebSnapshot.OCX), DBAccess controls (DBAccess.OCX), OPCClient control (OPCClient.OCX) and Thermocouple control(Thermocouple.OCX). The exact list of controls depends on the DAQBench package you use.

Use the following procedure to add DAQBench controls to the project toolbox.

1. In a new Visual Basic project, right click on the toolbox and select **Components....**
2. Scroll down to the control list , then you can find the DAQBench controls which beginning with the “ DAQBench” .
3. Place a checkmark in the box next to the control groups to select the controls you want to use in your project. If the DAQBench controls are not in the list, select the control files from the DAQBench\OCX directory by pressing the **Browse** button.

If you need to use the DAQBench controls in several projects, create a new default project in Visual Basic to include the DAQBench controls and serve as a template.

1. Create a new Standard EXE application in the Visual Basic environment.
2. Add the DAQBench controls to the project toolbox as described in the preceding procedure.
3. Save the form and project in the \Template\Projects directory under your Visual Basic directory.
4. Give the form and project a descriptive name, such as DBForm and DBProject.

After creating this default project, you have a new option, DBProject, that includes the DAQBench controls in the **New Project** dialog by default. You can create additional project templates with different combinationsof controls.

2.1.2 Building the User Interface Using DAQBench

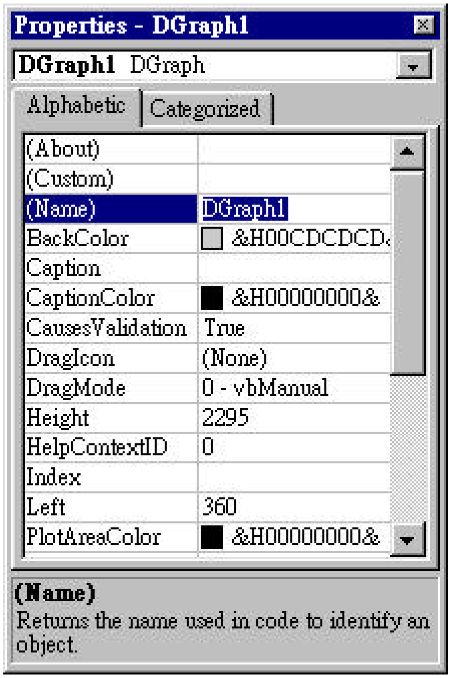
After you add the DAQBench controls to the Visual Basic toolbox, use them to create the front panel of your application. To place the controls on the form, select the corresponding icon in the toolbox and click and drag the mouse on the form. This step creates the corresponding control. After you create controls, move and size them by using the mouse. To move a control, click and hold the mouse on the control and drag the control to the desired location. To resize a control, select the control and place the mouse pointer on one of the hot spots on the border of the control. Drag the border to the desired size. Notice that the icons for all but the user interface controls cannot be resized and will not be visible at run time.

Once ActiveX controls are placed on the form, you can edit their properties using their property sheets. You can also edit the properties from within the Visual Basic program at run time.

2.1.3 Using Property Sheets in Visual Basic

After placing a control on a Visual Basic form, configure the control by setting its properties in the Visual Basic property pages and DAQBench custom control property pages (illustrated below). Visual Basic assigns some default properties, such as the control name and the tab order. When you create the control, you can edit these stock properties in the Visual Basic default property sheet. To access this sheet, select a control and select **Properties Window** from the **View** menu, or press <F4>. To edit a property, highlight the property value on the right side of the property sheet and type in the new value or select it from a pull down menu. The most important property in the default property sheet is Name, which is used to reference the control in the program.

Edit all other properties of an ActiveX control in the custom property sheets. To open the custom property sheets, right click on the control on the form and select **Properties...** or select the controls and press <Shift-F4>.



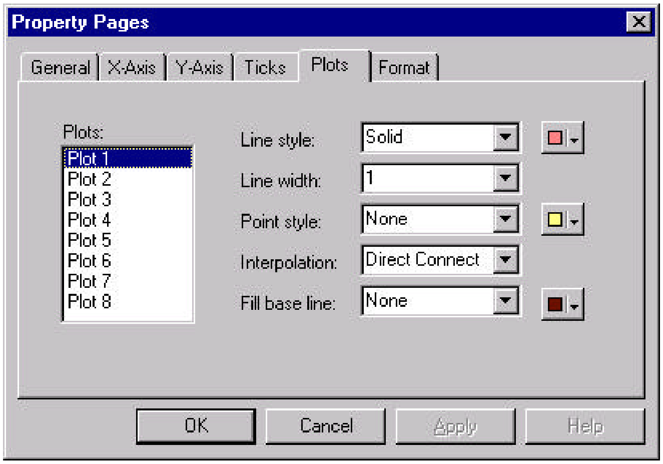
Properties - DGraph1

DGraph1 DGraph

(About)	
(Custom)	
(Name)	DGraph1
BackColor	&H00CDCDCD
Caption	
CaptionColor	&H00000000&
CausesValidation	True
DragIcon	(None)
DragMode	0 - vbManual
Height	2295
HelpContextID	0
Index	
Left	360
PlotAreaColor	&H00000000&

(Name)
Returns the name used in code to identify an object.

Visual Basic Property Pages



Property Pages

Plots:
Plot 1
Plot 2
Plot 3
Plot 4
Plot 5
Plot 6
Plot 7
Plot 8

Line style:	Solid
Line width:	1
Point style:	None
Interpolation:	Direct Connect
Fill base line:	None

DAQBench Custom Property Pages

2.1.4 Using Your Visual Basic Program to Edit Properties

You can set and read the properties of your controls programmatically in Visual Basic. Use the name of the control with the name of the property as you would with any other variable in Visual Basic. The syntax for setting a property in Visual Basic is `name.property = new value`.

For example, if you want to change the state of an DBoolean control during program execution, change the `Value` property from `True` to `False` or `False` to `True`.

```
    DBoolean.Value = 0x3
```

Some properties of a control can be objects that have their own properties. In this case, specify the name of the control, sub-object, and property separated by periods. For example, consider the following code for the analog input of DAQ Pci9112 control.

```
    Pci91121.ScanRate = 10000
```

In the above code, `ScanRate` is a property of the Pci9112 control. You can retrieve the value of control properties from your program in the same way. For example, you can print the value of an DBoolean control.

```
    Print DBoolean.Value
```

In Visual Basic most controls have a default property such as Value for the Knob, DBoolean, and Slide controls. You can access the default property of a control by using its control name (without the property name attached).

```
    DSlide1 = 5.0
```

is programmatically equivalent to

```
    DSlide1.Value = 5.0
```

2.1.5 Working with Control Methods from Visual Basic

Calling the methods of an ActiveX control in Visual Basic is similar to working with the control properties. To call a method, add the name of the method after the name of the control (and sub-object if applicable). For example, you can call the `StartContAI` method on the DAQ analog input control.

```
Pci9112.StartContAI
```

Methods can have parameters that you pass to the method, and return values that pass information back to your program. For example, the PlotGraph method for the DAQBench DGraph control has two required parameter -- The array of scaled data to be plotted and the index of plot -- That you must include when you call the method. If you want to plot the data returned from an Analog Input control, you must process the event of control, see the following line of code.

```
DGraph1.PlotGraph ScaledData, 0
```

In Visual Basic if you call a method without assigning a return variable, any parameters passed to the method are listed after the method name, separated by commas without parentheses.

```
Pci9112.ReadSingleAI 0, Data
```

If you assign the return value of a method to a return variable, enclose the parameters in parentheses.

```
result = Pci9112.ReadSingleAI(0, Data)
```

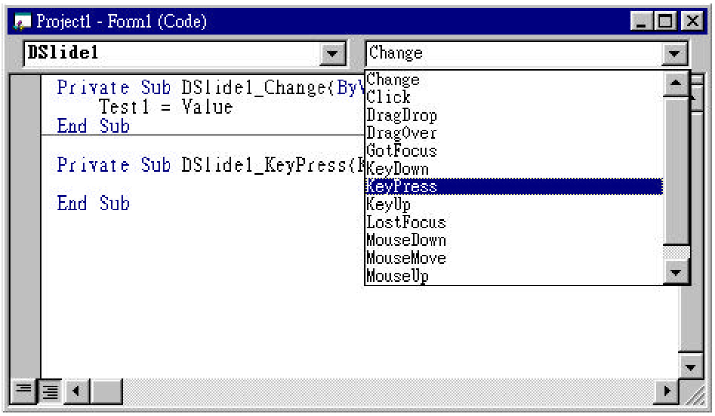
You can use the Visual Basic Object Browser to add method calls to your program.

2.1.6 Developing Control Event Routines

After you configure your controls in the forms editor, write Visual Basic code to respond to events on the controls. The controls generate these events in response to user interactions with the controls or in response to some other occurrence in the control. To develop the event handler routine code for an ActiveX control in Visual Basic, double click on the control to open the code editor, which automatically generates a default event handler routine for the control. The event handler routine skeleton includes the control name, the default event, and any parameters that are passed to the event handler routine. The following code is an example of the event routine generated for the DSlide control. This event routine (`Change`) is called when the value of the slide is changed by the user or by some other part of the program.

```
Private Sub DSlide1_Change(ByVal PointerNo As Integer,
      ByVal Value As Variant)
End Sub
```

To generate an event handler for a different event of the same control, double click the control to generate the default handler, and select the desired event from the right pull-down menu in the code window, as shown in the following illustration.



```
Private Sub DSlide1_Change(ByV
    Test1 = Value
End Sub

Private Sub DSlide1_KeyPress(K

End Sub
```

Selecting Events in the Code Window

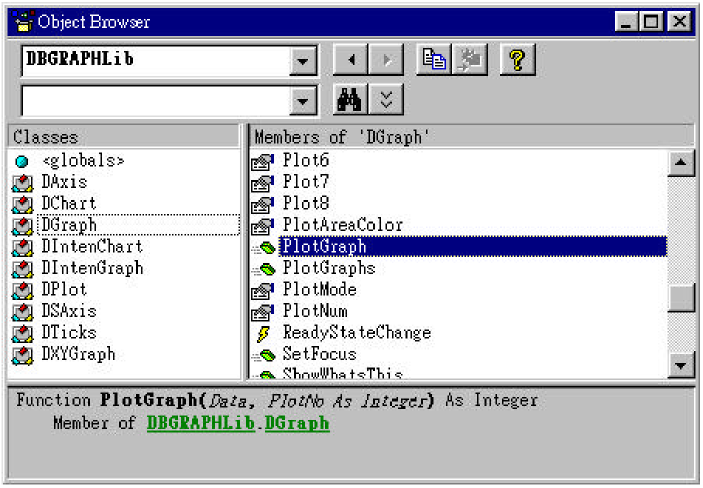
Use the left pull-down menu in the code window to change to another control without going back to the form window.

2.1.7 Using the Object Browser to Build Code in Visual Basic

Visual Basic includes a tool called the Object Browser that you can use to work with ActiveX controls. The Object Browser displays a detailed list of the available properties, methods, and events for a particular control. It presents a three-step hierarchical view of controls or libraries and their properties, methods, functions, and events. To open the Object Browser, select **Object Browser** from the **View** menu, or press <F2>.

In the Object Browser, use the top left pull-down menu to select a particular ActiveX control file, library, or instrument driver. You can select any currently loaded control or driver. The Classes list on the left side of the object browser displays a list of controls, objects, and function classes available in the selected control file or driver.

The following illustration shows the DAQBench User Interface (UI) control file selected in the Object Browser. The Classes list shows all the UI controls and associated object types. Each time you select an item from the Classes list in the Object Browser, the Members list on the right side displays the properties, methods, and events for the selected object or class.

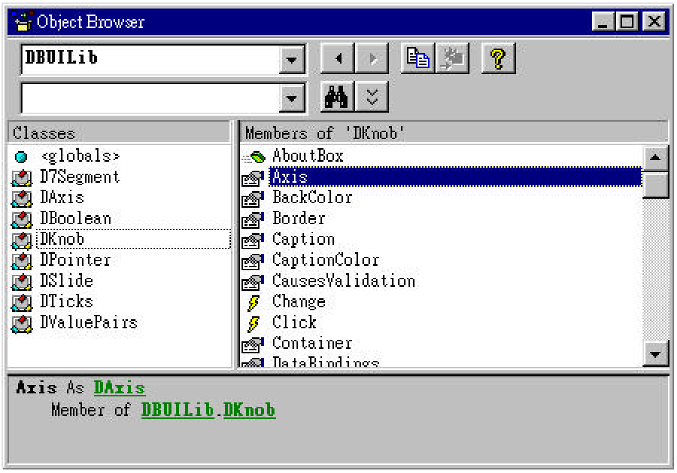


```
Function PlotGraph(Data, PlotNo As Integer) As Integer
    Member of DBGRAPHLib.DGraph
```

Viewing DGraph in the Object Browser

When you select an item in the Members list, the prototype and description of the selected property, method, or event are displayed at the bottom of the Object Browser dialog box. In the illustration above, the DGraph control is selected from the list of available UI objects. For this control, the `PlotGraph` method is selected and the prototype and description of the method appear in the dialog box. The prototype of a method or function lists all parameters, required and optional.

When you select a property of a control or object in the Members list which is an object in itself, the description of the property includes a reference to the object type of the property. For example, the following illustration shows the Knob control (DKnob) selected in the Classes list and its `Axis` property selected in the Members list.



Axis As DAxis
    Member of DBUILib.DKnob

Viewing DKnob in the Object Browser

The Axis on the Knob control is a separate object, so the description at the bottom of the dialog window lists the Axis property as DAxis. DAxis is the type name of the Axis object, and you can select DAxis in the Classes list to see its properties and methods. Move from one level of the object hierarchy to the next level using the Object Browser to explore the structure of different controls.

The question mark (?) button at the top of the Object Browser opens the help file to a description of the currently selected item. To find more information about the DGraph control, select the control in the window and press the ? button.

2.1.8 Pasting Code into Your Program

If you open the Object Browser from the Visual Basic code editor, you can copy the name or prototype of a selected property, method, or function to the clipboard and then paste it into your program. To perform this task, select the desired Member item in the Object Browser. Press the **Copy to Clipboard** button at the top of the Object Browser or highlight the prototype at the bottom and press <Ctrl-C> to copy it to the clipboard. Paste it into your code window by selecting **Paste** from the **Edit** menu or pressing <Ctrl-V>.

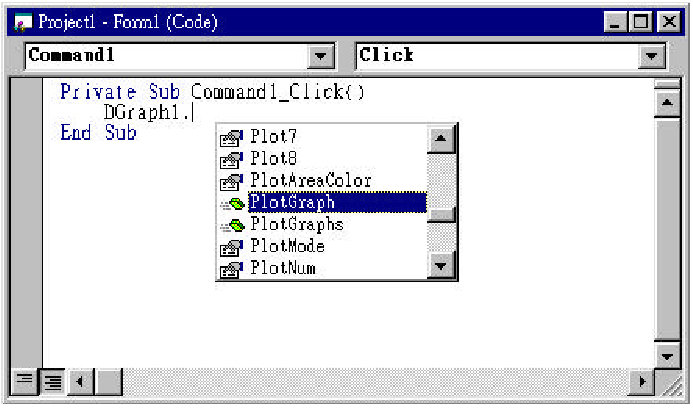
Use this method repeatedly to build a more complex reference to a property of a lower-level object in the object hierarchy. For example, you can create a reference to

```
DKnob1.Axes.ValuePairs.Name(3)
```

by typing in the name of the control (DKnob1) and then using the Object Browser to add each section of the property reference.

2.1.9 Adding Code Using Visual Basic Code Completion

Visual Basic supports automatic code completion in the code editor. As you enter the name of a control, the code editor prompts you with the names of all appropriate properties and methods. Try placing a control on the form and then entering its name in the code editor. After typing the name, add a period as the delimiter to the property or method of the control. As soon as you type the period, Visual Basic drops down a menu of available properties and methods, as shown below.



```
Private Sub Command1_Click()
    DGraph1.
End Sub
```

Visual Basic Code Completion

You can select from the list or properties and events by scrolling through the list and selecting one or by typing in the first few letters of the desired item. Once you have selected the correct item, type the next logical character such as a period, space, equal sign, or carriage return to enter the selected item in your code and continue editing the code.

2.1.10 Learning to Use Specific DAQBench Controls

Each DAQBench control and its use are described in more detail in other sections of this help file. However, these sections do not discuss every property, method, and feature of every control. The DAQBench online reference contains detailed information about each control and all its associated properties, events, and methods. Refer to the online reference to find descriptions of the different features of a particular control. Remember that many of the DAQBench controls share properties. When you learn how to use one control, you are learning how to use others as well.

3

Building DAQBench Applications with Visual C++

This chapter describes how you can use DAQBench controls with Visual C++, explains how to insert the controls into the Visual C++ environment and create the necessary wrapper classes, shows you how to create an application compatible with the DAQBench controls using the Microsoft Foundation Classes Application Wizard (MFC AppWizard) and how to build your program using the ClassWizard with the controls, and discusses how to perform these operations using ActiveX controls in general.

At this point you should be familiar with the general structure of ActiveX controls as well as C++ programming and the Visual C++ environment.

3.1 Developing Visual C++ Applications

The following procedure explains how you can start developing Visual C++ applications with DAQBench.

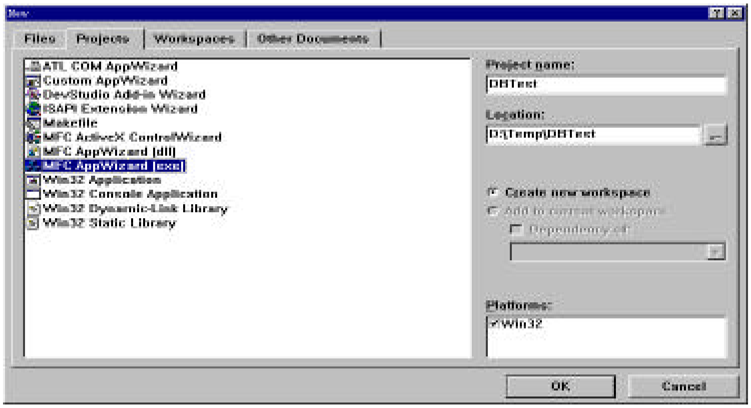
1. Create a new workspace or project in Visual C++.
2. To create a project compatible with the DAQBench ActiveX controls, use the Visual C++ MFC AppWizard to create a skeleton project and program.
3. After building the skeleton project, add the ActiveX controls you need to the controls toolbar. From the toolbar, you can add the controls to the application itself.
4. After adding a control to your application, you can configure its properties by its property pages.
5. While developing your program code, use the control properties and methods and create event handlers to process different events generated by the control.

Create the necessary code for these different operations using the ClassWizard in the Visual C++ environment.

3.1.1 Creating Your Application in Visual C++

When developing new applications, use the MFC AppWizard to create new project workspace so that the project is compatible with ActiveX controls. The MFC AppWizard creates the project skeleton and adds the necessary code that enables you to add ActiveX controls to your program.

1. Create a new project by selecting **New...** from the **File** menu. The **New** dialog box opens.



Project name: DBTest

Location: D:\Temp\DBTest

Platforms: ✓Win32

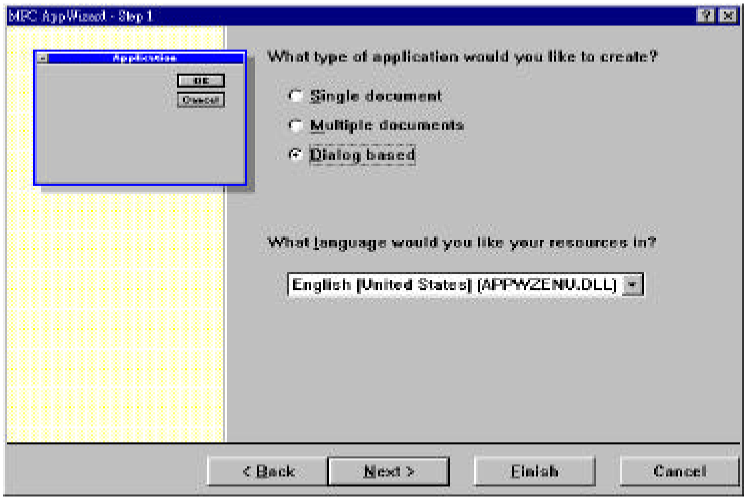
New Dialog Box

2. On the **Projects** tab, select the MFC AppWizard (exe) and enter the project name in the Project name field and the directory in the Location field.

3. Click on **OK** to setup your project.Complete the next series of dialog windows in which the MFC AppWizard prompts you for different project options. If you are a new Visual C++ or the MFC AppWizard user, accept the default options unless otherwise stated in this documentation.

4. In the first step, select the type of application you want to build.

   For this example, select a Dialog based application, as shown in below to make it easier to become familiar with the DAQBench controls.



What type of application would you like to create?

( ) Single document
( ) Multiple documents
[✓ Dialog based]

What language would you like your resources in?

English [United States] (APPWZENU.DLL)

MFC AppWizard -- Step 1

5. Click on the **Next>** button to continue.
6. Enable ActiveX controls support. If you have selected a Dialog based application, step two of the MFC AppWizard enables **ActiveX Controls** support by default.
7. Continue selecting desired options through the remainder of the MFC AppWizard. When you finish the MFC AppWizard, it builds a project and program skeleton according to the options you specified. The skeleton includes several classes, resources, and files, all of which can be accessed from the Visual C++ development environment.
8. Use the Workspace window, which you can select from the **View** menu, to see the different components in your project.

3.1.2 Adding DAQBench Controls to the Visual C++ Controls Toolbar

Before building an application using the DAQBench controls, you must load the controls into the Controls toolbar in Visual C++ from the Component Gallery in the Visual C++ environment. When you load the controls using the Component Gallery, a set of C++ wrapper classes automatically generate in your project. You must have wrapper classes to work with the DAQBench controls.

The Controls toolbar is visible in the Visual C++ environment only when the Visual C++ dialog editor is active. Use the following procedure to open the dialog editor.

1. Open the Workspace window by selecting **Workspace** from the **View** menu.
2. Select the Resource View (second tab along the bottom of the Workspace window).
3. Expand the resource tree and double click on one of the Dialog entries.
4. If necessary, right click on any existing toolbar and enable the Controls option.

By adding controls to your project, you can create the necessary wrapper classes for the control in your project and add the control to the toolbox, then use the following procedures to add the controls to the toolbar.

1. Select **Project**>>**Add To Project...>>Components and Controls** and, in the following dialog, double click on Registered ActiveX Controls.
2. Select and insert registered ActiveX controls into your project and control toolbox.
3. Select the controls you need and click the **Insert** button. All DAQBench controls start with DAQBench.
4. Click on **OK** in the following dialog windows.
5. When you have inserted all controls, click **Close** in the Components and Controls Gallery.

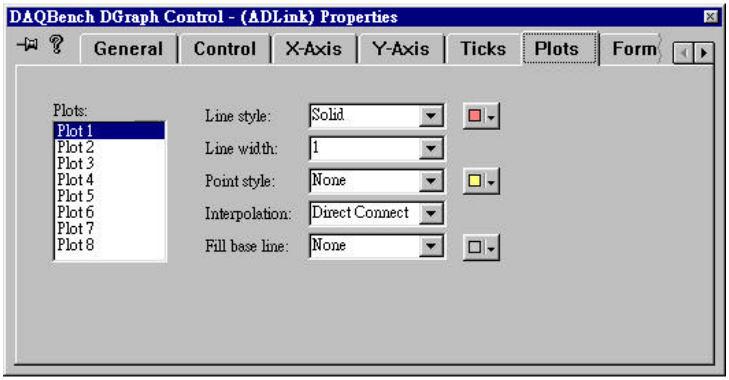
3.1.3 Building the User Interface Using DAQBench Controls

After adding the controls to the Controls toolbar, use the controls in the design of the application user interface. Place the controls on the dialog form using the dialog editor. You can size and move individual controls in the form to customize the interface. Use the custom property sheets to configure control representation on the user interface and control behavior at run time.

To add DAQBench controls to the form, open the dialog editor by selecting the dialog from the Resource View of the Workspace window. If the Controls toolbar is not displayed in the dialog editor, open it by right clicking on any existing toolbar and enabling the Controls option.

To place a DAQBench control on the dialog form, select the desired control in the Controls toolbar and click , then drag the mouse on the form to create the control. After placing the controls, move and resize them on the form as needed.

After you add a DAQBench control to a dialog form, configure the default properties of the control by right clicking on the control and selecting **Properties…** to display its custom property sheets.



Plots: Plot 1, Plot 2, Plot 3, Plot 4, Plot 5, Plot 6, Plot 7, Plot 8

Line style: Solid

Line width: 1

Point style: None

Interpolation: Direct Connect

Fill base line: None

DGraph Control Property Sheets

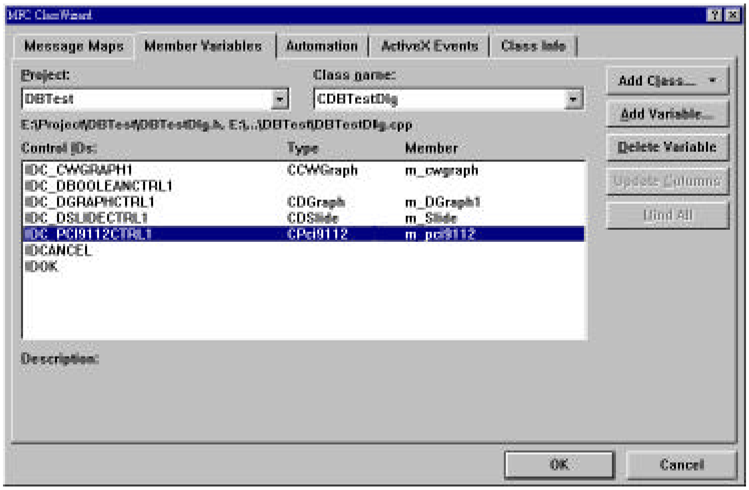
So you can see immediately how different properties affect the control, a separate window displays a sample copy of the control that reflects the property changes as you make them in the property sheets.

3.1.4 Programming with the DAQBench Controls

To program with DAQBench controls, use the properties, methods, and events of the controls as defined by the wrapper classes in Visual C++.

Before you can use the properties or methods of a control in your Visual C++ program, assign a member variable name to the control. This member variable becomes a variable of the application dialog class in your project.

To create a member variable for a control on the dialog form, right click on the control and select **ClassWizard**. In the **MFC Class Wizard** window, activate the **Member Variables** tab.



Project: DBTest

Class name: CDBTestDlg

E:\Project\DBTest\DBTestDlg.h, E:\...\DBTest\DBTestDlg.cpp

Control IDs:	Type	Member
IDC_CWGRAPH1	CCWGraph	m_cwgraph
IDC_DBOOLEANCTRL1		
IDC_DGRAPHCTRL1	CDGraph	m_DGraph1
IDC_DSLIDECTRL1	CDSlide	m_Slide
IDC_PCI9112CTRL1	CPci9112	m_pci9112
IDCANCEL		
IDOK		

MFC ClassWizard -- Member Variable Tab

Select the new control in the Control IDs field and press the **Add Variable...** button. In the dialog window that appears, complete the member variable name and press **OK**. Most member variable names start with m_, and you should adhere to this convention. After you create the member variable, use it to access a control from your source code. The illustration above shows the MFC Class Wizard after member variables have been added for a graph and analog input control.

3.1.5 Using Properties

Unlike Visual Basic, you can not read or set the properties of DAQBench controls directly in Visual C++. Instead, the wrapper class of each control contains functions to read and write the value of each property. These functions are named starting with either `Get` or `Set` followed by the name of the property. For example, to set the `Value` property of a slide, use the `SetValue` function of the wrapper class for the Slide control. In the source code, the function call is preceded by the member variable name of the control to which it applies.

```
    m_Slide.SetValue(COleVariant(5.0));
```

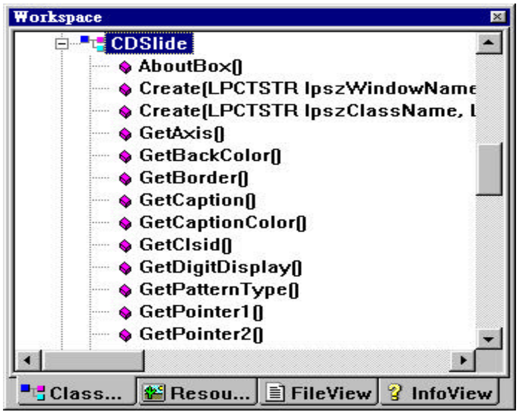
All values passed to properties need to be the variant type. Convert the value passed to the Value property to a variant using COleVariant() or the DAQBench type wrappinh library(Please refer to 3.1.8 section).

Use the `GetValue()` function to read the value of a control or to pass a value of a control to another part of your program. For example, pass the value of a Slide control to a Meter control.

```
    m_Meter.SetValue(m_Slide.Pointerl.GetValue());
```

Because the `GetValue` function returns its value as a variant in the previous line of code, conversion to a variant type is not necessary.

You can view the names of all the property functions (and other functions) for a given control in the ClassView of the Workspace window. In the Workspace window, select **ClassView** and then the control for which you want to view property functions and methods. The following illustration shows the functions for the Slide object as listed in the Workspace. These are created automatically when you add a control to the Controls toolbar in you project.



Workspace
CDSlide
AboutBox()
Create(LPCTSTR lpszWindowName
Create(LPCTSTR lpszClassName, l
GetAxis()
GetBackColor()
GetBorder()
GetCaption()
GetCaptionColor()
GetClsid()
GetDigitDisplay()
GetPatternType()
GetPointer1()
GetPointer2()
Class... | Resou... | FileView | InfoView

Viewing Property Functions and Methods in the Workspace Window

If you need to access a property of a control which is in itself another object, use the appropriate property function to return the sub-object of the control. Make a call to access the property of the sub-object. Include the header file in your program for any new objects. For example, use the following code to configure the Axis object of a Slide control.

```
#include daxis.h
CDAxis   Axis1;
Axis1 = m_Slide.GetAxis();
Axis1.GetTicks().SetMaximum(COleVariant(5.0));
```

You can chain this operation into one function call without having to declare another variable.

```
#include dslide.h
#include daxis.h
#include dticks.h
m_Slide.GetAxis().GetTicks().SetMaximum
(COleVariant(5.0));
```

If you need to access an object in a collection property, use the Item method with the index of the object. Remember to include the header file for the collection object. For example, to set the maximum of the first y-axis on a graph, use the following code.

```
#include dgraph.h
#include daxis.h
#include dticks.h
m_DGraph.GetAxis().GetTicks().SetMaximum(COleVariant(
5.0));
```

3.1.6 Using Methods in Visual C++

Use the control wrapper classes to extract all methods of the control. To call a method, append the method name to the member variable name and pass the appropriate parameters. If the method does not require parameters, use a pair of empty parentheses.

```
m_Pci91121.StartContAI();
```

Most methods take some parameters as variants. You must convert any such parameter to a variant if you have not already done so or use the DAQBench type wrapping library `(Please refer to 3.1.8 section).`. You can convert most scalar values to variants with COleVariant(). For example, the PlotGraph method of the graph control requires one scalar values as variants.

```
m_Graph.PlotGraph(COleVariant(1.0), 0 );
```

If you need to call a method on a sub-object of a control, follow the conventions outlined in Using Properties . For example, To call PlotGraph on one particular plot of your graph, use the following line of code.

```
m_Graph.PlotGraph (*Voltages, 2 );
```

3.1.7 Using Events in Visual C++

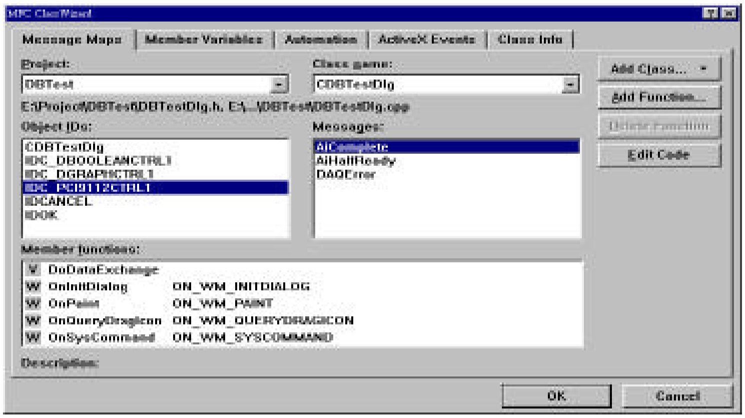
After placing a control on your form, you can start defining event handler functions for the control in your code. Events generate automatically at run time when different controls respond to conditions, such as a user clicking a button on the form or the data acquisition process acquiring a specified number of points.

Use the following procedure to create an event handler.

1. Right click on a control and select **ClassWizard**.
2. Select the **Message Maps** tab and the desired control in the Object IDs field. The Messages field displays the available events for the selected control. See Event Handler for the Change Event of a Knob.
3. Select the event and press the **Add Function..**. button to add the event handler to your code.
4. To switch directly to the source code for the event handler, click on the **Edit Code** button. The cursor appears in the event handler, and you can add the functions to call when the event occurs. You can use the **Edit Code** button at any time by opening the class wizard and selecting the event for the specific control.

The following illustration is an example of an event handler generated for the Change event of a knob. Insert your own code in the event handler:

```
void CTestDlg::OnChangeDKnob1(Short PointerNo, const
VARIANT FAR & Value)
{
// TODO: Add your control notification handler code here
}
```



Project: DBTest

Class name: CDBTestDlg

E:\Project\DBTest\DBTestDlg.h, E:\...\DBTest\DBTestDlg.cpp

Object IDs:
```
CDBTestDlg
IDC_DBOOLEANCTRL1
IDC_DGRAPHCTRL1
IDC_PCI9112CTRL1
IDCANCEL
IDOK
```

Messages:
```
AICompletc
AIHalfReady
DAQError
```

Member functions:
```
V  DoDataExchange
W  OnInitDialog      ON_WM_INITDIALOG
W  OnPaint           ON_WM_PAINT
W  OnQueryDragIcon   ON_WM_QUERYDRAGICON
W  OnSysCommand      ON_WM_SYSCOMMAND
```

Description:

Event Handler for the Change Event of a Knob.

3.1.8 DAQBench enhancement in Visual C++

To make it flexible and ease of use in Visual Basic environment, many properties and methods arguments in DAQBench are with VARIANT type which is not a basic type of C/C++. Actually VARIANT is defined as a structure. Therefore to use VARIANT type in C/C++ is not so straightforward as the basic types. In addition to this, some of the DAQBench controls encapsulate objects in it. For example, DChart control encapsulates Xaxis, Yaxis objects. You can easily access the encapsulated objects in VB. However it is not so straightforward to access them in VC++. In order to let user can access the encapsulated objects in the DAQBench controls, and use VARIANT structure in the VC++ environment in an easier way, DAQBench provides some enhancement functions.

- **The enhancement method of DAQBench control**

Some methods are added in the User-Interface controls to help user with the above difficulties. Take the DChart control object as an example. After adding this control into the project, you will find that some additional functions in the header file “dchart.h”, such as :

```
void SetXAxisViewNumber(long ViewNumber);
void SetYAxisMinMax(double Min, double Max);
```

With these functions, user can set the control properties directly and pass the arguments by the basic data type. Without this kind of functions, if you want to draw the X-Axis grids of the DChart object during the run-time, you need the codes below in C++:

```
CDChart  m_Chart;   //declare a chart object
//set the major grid property as true
m_Chart.GetXAxis().GetTicks().SetMajorGrid(true);
```

Now with these enhancement functions, you can simply show the grids by the following way:

```
CDChart  m_Chart2;   //declare a chart object
//enable major grid and disable minor grid
m_Chart2.SetXAxisGrid(true, false);
```

Please refer to the DAQBench function reference manual for the details of the enhancement methods.

◆ The data type wraping library for DAQBench VARIANT structure

Due to the limitation of parameter passing in COM, some DAQBench control object methods have VARIANT type of parameters. If user wants to convert the VARIANT type data to other basic type data (e.g. integer, real), user can use COleVariant to wrap the VARIANT type data to basic data type. But for some complicated types (such as array), COleVariant can not provide the type casting function. Therefore DAQBench provides a data type conversion library "VarPacker.dll" to help users to wrap other data type into a VARIANT structure.

Before going to next stage, there are some things user has to do:

1. Check if the **VarPacker.dll** is in the Winnt/System32 (for NT/2000) or Windows/System (for 98) directory.
2. Check if the VarPacker.h is in the DAQBench installation\include directory.
3. Open/New the VC++ project workspace.
4. Add the VarPacker.h into your project workspace.
5. Link with the VarPacker.lib library, this library is located in DAQBench installation\OCX directory.

After the above setting, user now can use " VarPacker" library functions. Some usage examples of the library are described below:

Case 1: Suppose user wants to change the Value property (VARIANT type) of a DBoolean control to 16. User can write the code in the following way:

```
DBoolean1.SetValue( LongToVar( (long) 16) );
```

Case 2: Suppose user wants to use the DChart object to draw a sine wave. The PlotCharts method needs an array wrapped in the VARIANT structure as its argument. Here is the solution of this case:

```
double  data[100];  //declare the array to store data
…
// generate the sine wav data and store in data[100]
…
Dchart1.PlotChart(ArrayToVar( data, 100 ));
```

Please refer to the DAQBench function reference manual for the details of the data type wrapping functions in **VarPacker.dll.**

4

Building DAQBench Applications with Delphi

This section of the online reference describes how you can use DAQBench controls with Delphi; insert the controls into the Delphi environment, set their properties, and use their methods and events; and perform these operations using ActiveX controls. This section also outlines Delphi features that simplify working with ActiveX controls.

At this point you should be familiar with the general structure of ActiveX controls.

4.1 Running Delphi Examples

To run the Delphi examples installed with DAQBench, you need to import the appropriate controls into the Delphi environment. See*Loading the DAQBench Controls into the Component Palette* for more information about loading the controls.

4.2 Upgrading from a Previous Version of DAQBench

When you upgrade DAQBench, you must remove the current controls from the Delphi environment and reinsert the controls in the Delphi environment to update the support files.

1. From the Component menu select Install Packages....
2. In the Design packages list, select Delphi User' s Components.
3. Click on Edit... and Yes in the dialog boxes to edit the user component package. The package editor lists all the components currently installed in the user components package, including the DAQBench controls.
4. Select each of the DAQBench entries and click on Remove.
5. Click on Compile to rebuild the package.
6. Close the package editor.

4.3 Developing Delphi Applications

You can start to develop applications in Delphi by using a form. A form is a window or area on the screen on which you can place controls and indicators to create the user interface for your programs. The Component palette in Delphi contains all of the controls available for building applications. After placing each control on the form, configure the properties of the control with the default and custom property pages. Each control you place on a form has associated code (event handler routines) in the Delphi program that automatically executes when the user operates the control or the control generates an event.

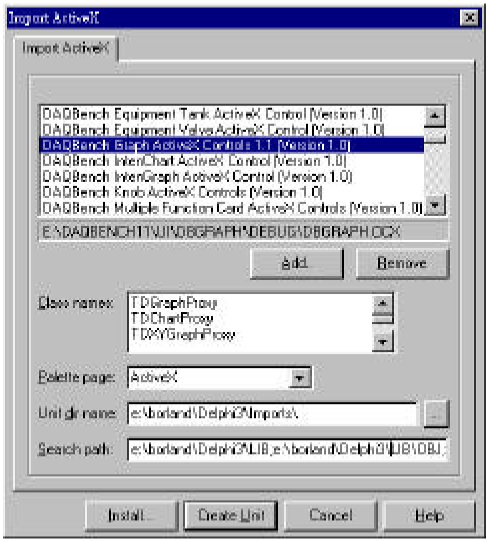
4.3.1 Loading the DAQBench Controls into the Component Palette

Before you can use the DAQBench controls in your Delphi applications, you must add them to the Component palette in the Delphi environment. You need to add the controls to the palette only once because the controls remain in the Component palette until you explicitly remove them. When you add controls to the palette, you create Pascal import units (header files) that declare the properties, methods, and events of a control. When you use a control on a form, a reference to the corresponding import unit is automatically added to the program.

Note: Before adding a new control to the Component palette, make sure to save all your work in Delphi, including files and projects. After loading the controls, Delphi closes any open projects and files to complete the loading process.

Use the following procedure to add ActiveX controls to the Component palette.

1. Select **Import ActiveX Control...** from the Component menu in the Delphi environment. The Import ActiveX Control window displays a list of currently registered controls.



```
DAQBench Equipment Tank ActiveX Control (Version 1.0)
DAQBench Equipment Valve ActiveX Control (Version 1.0)
DAQBench Graph ActiveX Controls 1.1 (Version 1.0)
DAQBench InterChart ActiveX Control (Version 1.0)
DAQBench InterGraph ActiveX Control (Version 1.0)
DAQBench Knob ActiveX Controls (Version 1.0)
DAQBench Multiple Function Card ActiveX Controls (Version 1.0)
E:\DAQBENCH11\WIN\DBGRAPH\DEBUG\DBGRAPH.OCX

Class names:     TDGraphProxy
                 TDChartProxy
                 TDXYGraphProxy
Palette page:    ActiveX
Unit dir name:   e:\borland\Delphi3\Imports\
Search path:     e:\borland\Delphi3\LIB;e:\borland\Delphi3\LIB\OBJ;
```

Delphi Import ActiveX Control Dialog Box

2. Select the control group you want to add to the Component palette. All DAQBench controls start with `DAQBench`.
3. After selecting the control group, click **Install...**.

Delphi generates a Pascal import unit file for the selected .OCX file, which is stored in the Delphi `\Imports` directory. If you have installed the same `.OCX` file previously, Delphi prompts you to overwrite the existing import unit file.

4. In the **Install** dialog box, click on **OK** to add the controls to the Delphi user' s components package.
5. In the following dialog, click on **Yes** to rebuild the user's components package with the added controls. Another dialog box acknowledges the changes you have made to the user's components package, and the package editor displays the components currently installed.

At this point, you can add additional ActiveX controls with the following procedure.

a. Click on the **Add** button.

b. Select the **Import ActiveX** tab.

c. Select the ActiveX control you want to add.

d. Click on **OK**.

e. After adding the ActiveX controls, compile the user' s components package.

If your control does not appear in the list of registered controls, click the **Add...** button. To register a control with the operating system and add it to the list of registered controls, browse to and select the OCX file that contains the control. Most OCX files reside in the `DAQBench\OCX` directory.

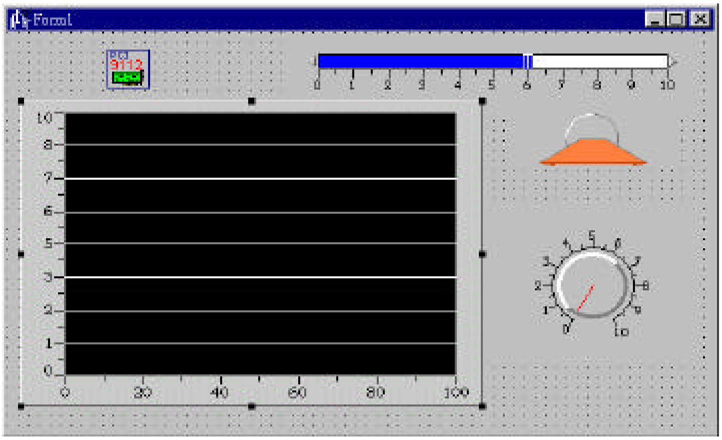
New controls are added to the **ActiveX** tab in the Components palette. You can rearrange the controls or add a new tab to the Components palette by right clicking on the palette and selecting **Properties**....

4.3.2 Building the User Interface

After you add the DAQBench controls to the Component palette, use them to create the user interface. Open a new project, and place different controls on the form. These controls, as part of the program user interface, add specific functionality to the application. After placing the controls on the form, configure their default property values through the stock and custom property sheets.

Placing Controls

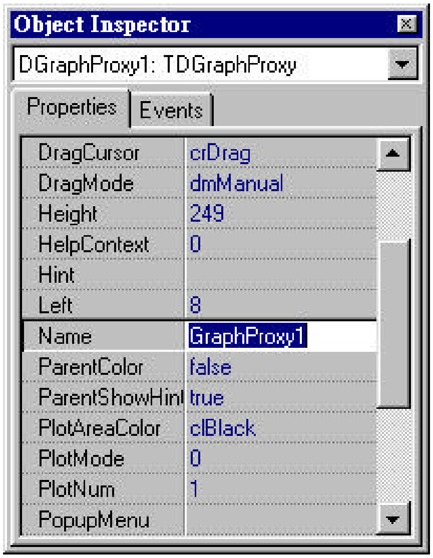
To place a control on the form, select the control from the Component palette and click and drag the mouse on the form. Use the mouse to move and resize controls to customize the interface, as in the following illustration. After you place the controls, you can change their default property values by using the default property sheet (Object Inspector) and custom property sheets.



DAQBench Controls on a Delphi Form

Using Property Sheets

Set property value such as Name in the Object Inspector of Delphi. To open the Object Inspector, select Object Inspector from the View menu or press <F11>. Under the Properties tab of the Object Inspector, you can set different properties of the select control.

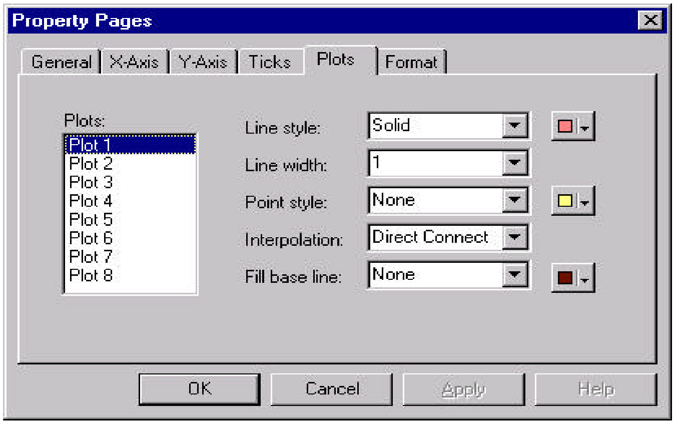


DGraphProxy1: TDGraphProxy

DragCursor	crDrag
DragMode	dmManual
Height	249
HelpContext	0
Hint	
Left	8
Name	GraphProxy1
ParentColor	false
ParentShowHint	true
PlotAreaColor	clBlack
PlotMode	0
PlotNum	1
PopupMenu	

Delphi Object Inspector

To open the custom property pages of a control, double click on the control or right click on the control and select Properties… You can edit most control properties from the custom property pages. The following figure shows the DAQBench Graph control property page.



Plots: Plot 1, Plot 2, Plot 3, Plot 4, Plot 5, Plot 6, Plot 7, Plot 8

Line style: Solid

Line width: 1

Point style: None

Interpolation: Direct Connect

Fill base line: None

DAQBench DGraph Control Property Page

4.3.3 Programming with DAQBench

The code for each form in Delphi is listed in the Associated Unit (code) window. You can toggle between the form and Associated Unit window by pressing <F12>. After placing controls on the form, use their methods in your code and create event handler routines to process events generated by the controls at run time.

Using Your Program to Edit Properties

You can set or read control properties programmatically by referencing the name of the control with the name of the property, as you would any variable name in Delphi. The name of the control is set in the Object Inspector.

If you want to change the state of an DBoolean control during program execution, change the `Value` property from `True` to `False` or from `False` to `True`. The syntax for setting the `Value` property in Delphi is

```
    name.property: = new_value.
    DBoolean1.Value := 1;
```

A property can be an object itself that has its own properties. To set properties in this case, combine the name of the control, sub-object, and property. For example, consider the following code for the DAQ Pci91121 control.

`ScanRate` is a property of the DAQ control..

```
    Pci91121.ScanRate := 10000;
```

You can retrieve the value of acontrol property from your program in the same way. For example, you can assign the scan rate of a Pci9112 control to a text box on the user interface.

```
    Edit1.Text := Pci91121.ScanRate;
```

To use the properties or methods of an object in a collection, use the Item method to extract the object from the collection. Once you extract the object, use its properties and methods as you usually would.

```
    DGraph1.YAxis.Maximum := 5;
```

Consult *the Setting the Properties of an ActiveX Control* section for more information about setting properties programmatically.

Using Methods

Each control has defined methods that you can use in your program. To call a method in your program, use the control name followed by the method name.

```
    Pci91121.StartContAI;
```

Some methods require parameters, as does the following method.

```
    DGraph1.PlotGraph(data, 0);
```

In most cases, parameters passed to a method are of type variant. Simple scalar values can be automatically converted to variants and, therefore, might be passed to methods. Arrays, however, must be explicitly declared as variant arrays.

The following example plots data using the graph `PlotGraph` method. Consult your Delphi documentation for more information about the variant data type.

```
Var
     vData:Variant;
begin
     //Create array in Variant
     vData := VarArrayCreate([0, 99], varDouble);
     for i := 0 to 99 do
     begin
         vData[i] := Random;
     end;
     //Plot Variant Array
     DGraph1.PlotGraph(vData, 0);
end;
```

Using Events

Use event handler routines in your source code to respond to and process events generated by the different DAQBench controls. Events are generated by user interaction with an object such as a DKnob or by other controls (such as the DAQ controls) in response to internal conditions (for example, completed acquisition or an error). You can create a skeleton for an event handler routine using the Object Inspector in the Delphi environment.

To open the Object Inspector, press <F11> or select **Object Inspector** from the **View** menu. In the Object Inspector, select the **Events** tab. This tab lists all the events for the selected control. To create a skeleton function in your code window, double click on the empty field next to the event name. Delphi generates the event handler routine in the code window using the default name for the event handler.

To specify your own event handler name, click in the empty field in the Object Inspector next to the event, and enter the function name. After the event handler function is created, insert the code in the event handler.

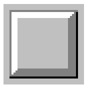
5

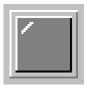
Introducing the ActiveX Controls of DAQBench

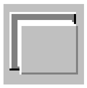
5.1 DBoolean Control

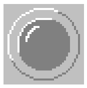
DBoolean ActiveX control is an UI component for operating boolean functions. The maximum bit of the DBoolean is 32. It can be using to indicate the boolean data like the LED signal. It can also be used to control the bit state of data like the switch. So, the DBoolean control is very convenient to be used as the display of digital input and the control of digital output at data acquisition operation.

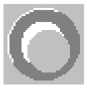
Pattern style

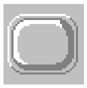

Square Button

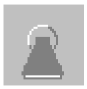

Square Radio Button

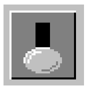

Square Push Button

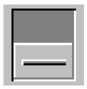

LED Button Round


Push Button


Round Button


Toggle Switch

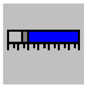

Switch

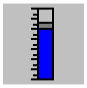

Slide Switch

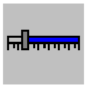
5.2 DSlide Control

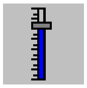
The DSlide control represents different types of linear displays, such as the variant slide, thermometers and tank display. With DSlide control, users can input or output(display) individual or multiple scalar values. A DSlide can have multiple pointers (maximum eight) on the control, Each pointer represents one scalar value.

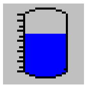
Pattern style

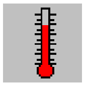

Wide horizon slide


Wide vertical slide


Narrow horizon slide


Narrow vertical slide

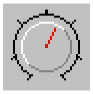

Tank

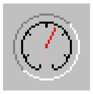

Thermometer

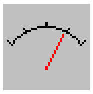
5.3 DKnob Control

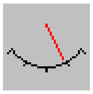
The DKnob control represents different types of circular displays, such as the knob, dial and different type of meters. With DKnob control, users can input or output(display) individual or multiple scalar values. A DKnob can have multiple pointers (maximum eight) on the control, Each pointer represents one scalar value.

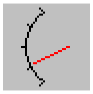
Pattern style

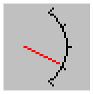

Knob


Dial


Upper meter


Down meter

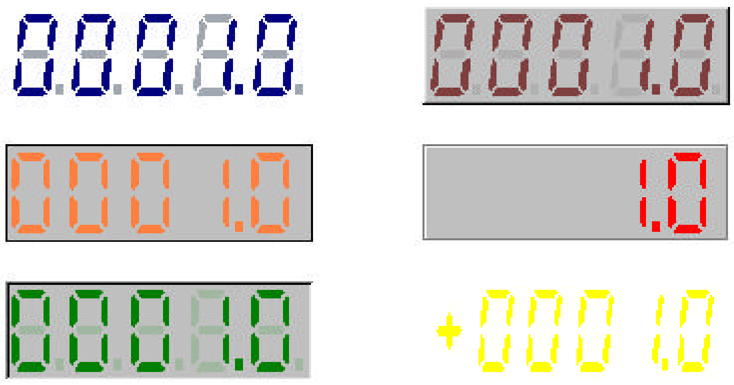

Left meter


Right meter

5.4 D7Segment Control

D7Segnment ActiveX control is an UI component for display number using style of seven segment display. Users can configure the property of control to specify the digit number, declined, Digit number after point, color of segment, transparent and signed, etc.

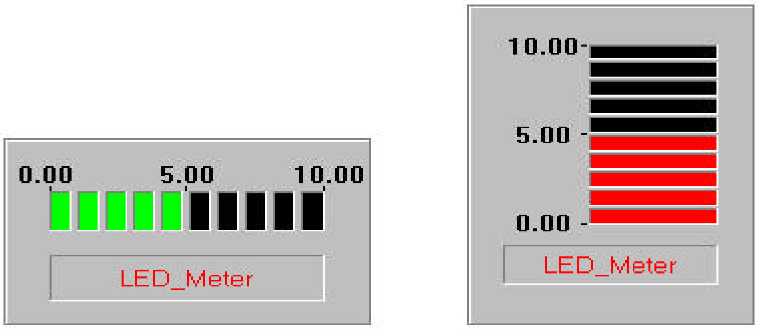
Pattern style



5.5 DLEDMeter Control

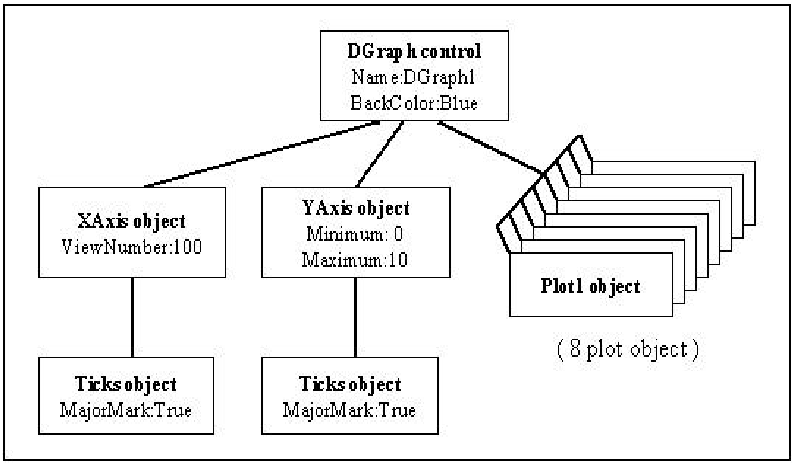
DLEDMeter ActiveX control is an UI component for display number using style of LED Bar display. Users can configure the property of control to specify the bar number, direction, bar color, ticks, max value and min value, etc.

Pattern style



5.6 DGraph Control

The DGraph control is a flexible control used for plotting data. It can display multiple plots(maximum eight plots). Plotting data refers to the process of taking a large number of points and updating one or more plots on the graph with new data. The DGraph control is made up of a hierarchy of objects, as illustrated in the following figure.

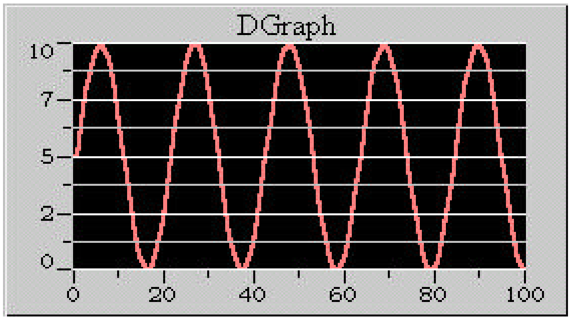


The XAxis object represents the input data points at horizon scale. Users can set the ViewNumber property to specify the DGraph object how many data points will display on plot window. The XAxis object can display the time domain scale when the scale format is "Date" or "Time". The XAxis object include one Ticks object that will process different style of ticks color, ticks mark and ticks label.

The YAxis object represent s the value of data points at the vertical scale. Users can set the maximum and minimum properties to specify the DGraph object has the display range at plot window. The YAxis object has many scale format to display scale label. The YAxis object includes one Ticks object that will process different style of ticks color ,ticks mark and ticks label.

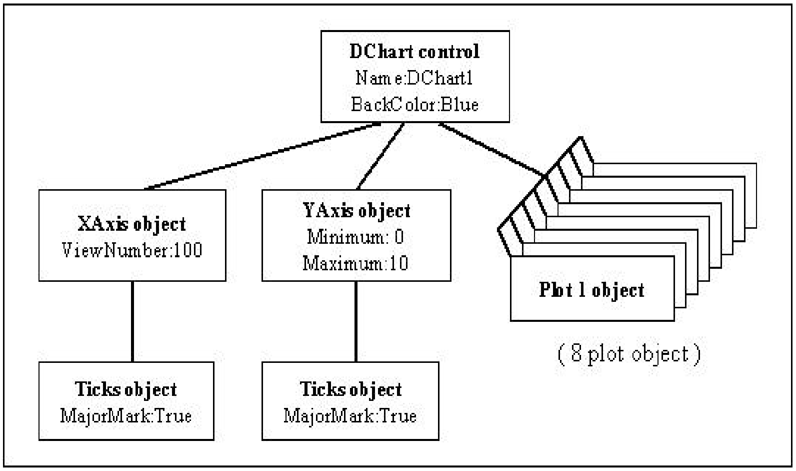
The DGraph object includes eight Plot objects. Users can specify the property of each plot object that include line style, line width, pointer style, fill style, line color, fill color, pointer color, interpolation type.

Example



5.7 DChart Control

The DChart control is a flexible control used for charting data. It can display multiple plots(maximum eight plots). Charting data appends new data points to an existing plot over time. Charting is used with slow processes where only few data points per second are added to the graph. The DChart control is made up of a hierarchy of objects, as illustrated in the following figure.



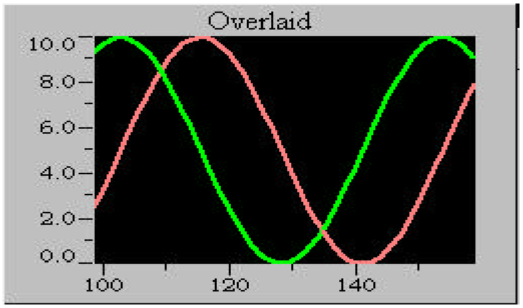
The XAxis object represents the input data points at horizon scale. Users can set the ViewNumber property to specify how many data points will display on plot window. The XAxis object can display the time domain scale when the scale format is " Date" or " Time". The XAxis object includes one Ticks object that will process different style of ticks color ,ticks mark and ticks label.

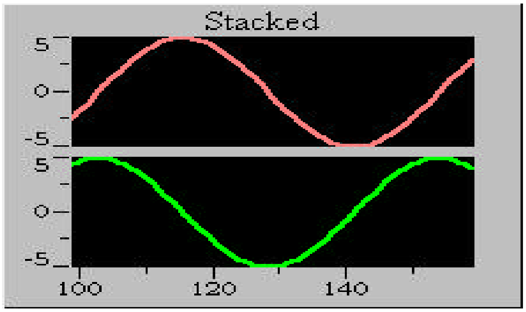
The YAxis object represent the value of data points at vertical scale. Users can set the `Maximum` and `Minimum` property to specify the display range at plot window. The YAxis object has many scale format to display scale label. The YAxis object includes one Ticks object that will process different style of ticks color, ticks mark and ticks label.

The DChart object include eight Plot object. Users can specify the property of each plot object that include line style, line width, pointer style, fill style, line color, fill color, pointer color, interpolation type.

Users can set the `PlotMode` property of DChart to " Overlaid" or " Stacked" to specify different type for multiple plot data. The `UpdateMode` property of DChart can determinate different update method while the charting data would be continuously input and the plot window would be scrolling.

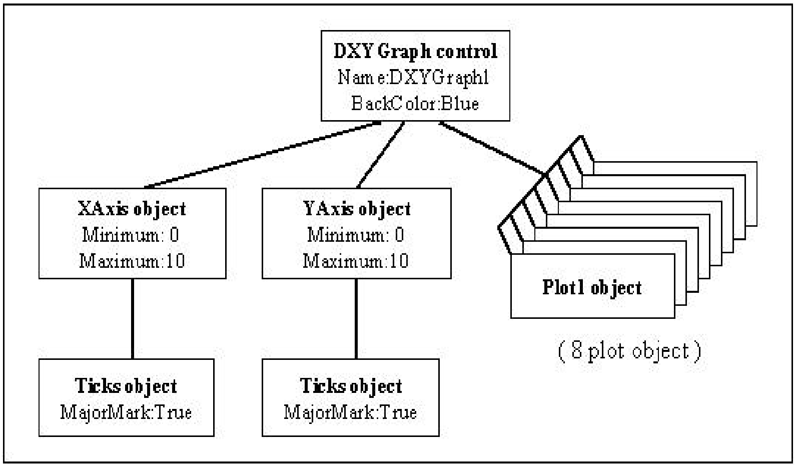
Example





5.8 DXYGraph Control

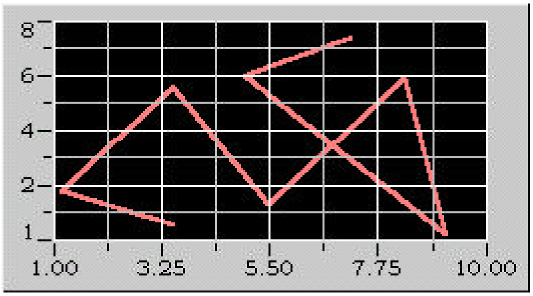
The DXYGraph control is a flexible control used for drawing XY data. It can display multiple plots(maximum eight plots). Plotting XY graph data is drawing the curve of a (x,y) data array. The DXYGraph control is made up of a hierarchy of objects, as illustrated in the following figure.



The XAxis object represents the value of data points at horizon scale. Users can set the `Maximum` and `Minimum` properties to specify the display range at plot window. The XAxis object has many scale format to display scale label. The XAxis object includes one Ticks object that will process different style of ticks color ,ticks mark and ticks label.

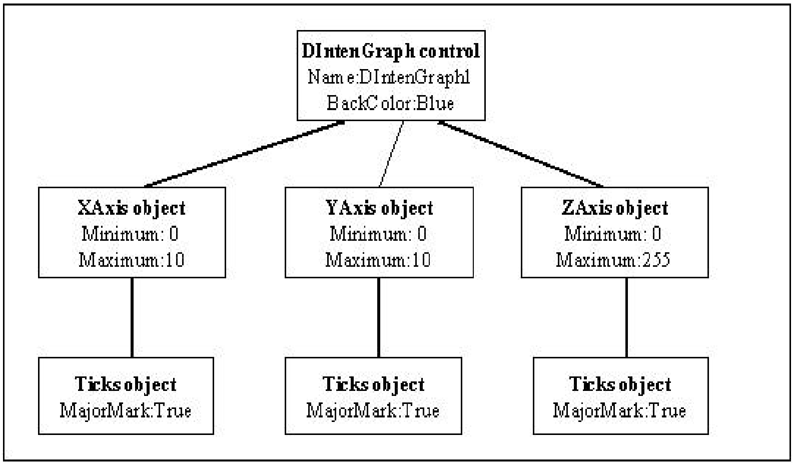
The YAxis object represents the value of data points at vertical scale. Users can set the `Maximum` and `Minimum` property to specify the display range at plot window. The YAxis object has many scale format to display scale label. The YAxis object include one Ticks object that will process different style of ticks color ,ticks mark and ticks label.

The DXYGraph object includes eight Plot object. Users can specify the property of each plot object that include line style, line width, pointer style, fill style, line color, fill color, pointer color, interpolation type.



5.9 DIntenGraph Control

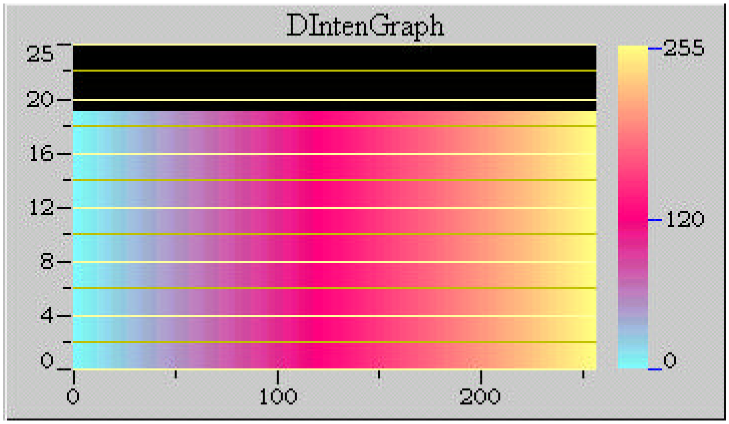
The DIntenGraph control is a control used for drawing color intensity on XY plane. It has one ZAxis that represents the color intensity at one point of XY plane. So, The ZAxis is the 256 color map. Plotting intensity data refers to the process of taking a large XY plane that include a number of points. The DIntenGraph control is made up of a hierarchy of objects, as illustrated in the following figure.



The XAxis object represents the input data points at horizon scale of plane. Users can set the `ViewNumber` property to specify how many data points will display on plot window. The XAxis object can display the time domain scale when the scale format is "Date" or "Time". The XAxis object includes one Ticks object that will process different style of ticks color, ticks mark and ticks label.

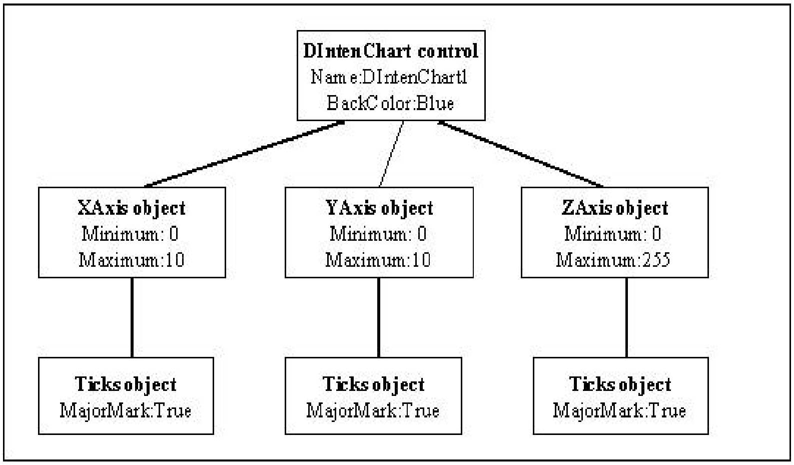
The YAxis object represents the input data points at vertical scale of plane. Users can set the `ViewNumber` property to specify how many data points will display on plot window. The YAxis object has many scale format to display scale label. The YAxis object includes one Ticks object that will process different style of ticks color, ticks mark and ticks label.

The ZAxis object represents the 256 color map. So, the `Maximum` value of scale is fixe7d at 255 and the `Minimum` value is fixed at 0. Users can specify the color value at each color index. The ZAxis object includes one Ticks object that will process different style of ticks color, ticks mark and ticks label.



5.10 DIntenChart Control

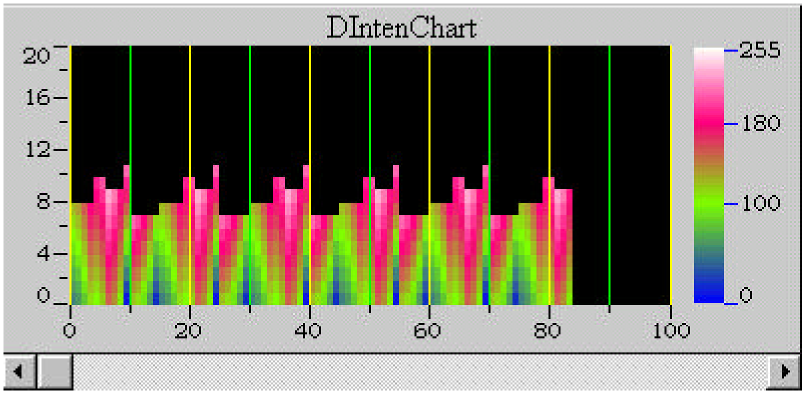
The DIntenChart control is a control used for drawing color intensity on XY plane. It has one ZAxis that represents the color intensity for one point in the XY plane. The ZAxis is a 256 color map. Charting data appends new intensity plane data to plot over time. Charting is used with slow processes where only few plane data per second are added to the graph. When more plane data are added, they also then can be displayed on graph, the graph scrolls and the new plane are added to the right side of the graph while old plane disappear to the left. The DIntenChart control is made up of a hierarchy of objects, as illustrated in the following figure.



The XAxis object represents the input data points at horizon scale of plane. Users can set the `ViewNumber` property to specify how many data points will display on plot window. The XAxis object can display the time domain scale when the scale format is "Date" or "Time". The XAxis object includes one Ticks object that will process different style of ticks color, ticks mark and ticks label.

The YAxis object represents the input data points at vertical scale of plane. Users can set the `ViewNumber` property to specify how many data points will display on plot window. The YAxis object has many scale format to display scale label. The YAxis object includes one Ticks object that will process different style of ticks color, ticks mark and ticks label.

The ZAxis object represents the 256 color map. So, the `Maximum` value of scale is fixed at 255 and the `Minimum` value is fixed at 0. Users can specify the color value at each color index. The ZAxis object includes one Ticks object that will process different style of ticks color, ticks mark and ticks label.

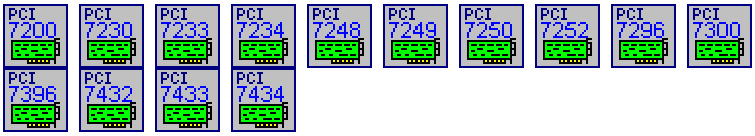


5.11 NuDAQ Controls for NuDAQ PCI Cards

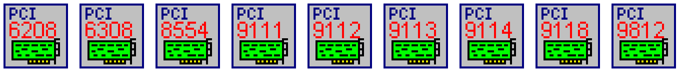
The ActiveX controls of NuDAQ PCI cards are packaged to two OCXs that are Digital.OCX and Multiple.OCX. The Digital.OCX includes digital I/O cards Pci7200, Pci7230, Pci7233, Pci7234, Pci7248, Pci7249, Pci7250, Pci7252, Pci7296, Pci7300, Pci7396, Pci7432, Pci7433, Pci7434. The Multiple.OCX include multiple function I/O cards Pci6208, Pci6308, Pci8554, Pci9111, Pci9112, Pci9113, Pci9114, Pci9118, Pci9812.

Using the NuDAQ controls, user can easily program to drive the NuDAQ PCI cards. Each NuDAQ ActiveX control must set the device name define in NuDAQCfg utility. When the NuDAQ control used at Form window, the NuDAQ control will be one image bitmap that represents the type of NuDAQ PCI card. User can set the bitmap invisible when the application is running.

The image bitmap of Digital I/O ActiveX controls



The image bitmap of Multiple I/O ActiveX controls



User can use the NuDAQ ActiveX object to control the ADLINK NuDAQ H/W operations. In general, the operations can divide into two groups.

1. Polling operation :

- DI operation : user can use the **ReadDIPort** or **ReadDILine** method of the control .
- DO operation : user can use the **WrieDOPort** method of the control .
- AI operation : user can use the **ReadSingleAI** method of the control .
- AO operation : user can use the **WriteSingleAO** method of the control .

2. Continuous operations

- DI operation : user can follow the procedures described below:

  1. Set the related properties according to your requirement.
  2. Use the `StartContDI` method of the control to start the operation.
  3. Receive DI data from DiComplete or DiHalfReady event of control.
  4. Use the `StopContDI` method of control to stop the operation.

- DO operation : user can follow the procedures described below:

  1. Set the related properties according to your requirement.
  2. Use the StartContDO method of the control to start the operation.
  3. When the DO data output operation is complete, the control will fire the DoComplete event to the program.
  4. Use the `StopContDO` method of control to stop the operation.

- AI operation : user can follow the procedures described below :

  1. Set the related properties according to your requirement.
  2. Use the StartContAI method of the control to start the operation.
  3. Receive AI data from AiComplete or AiHalfReady event of control
  4. Use the `StopContAI` method of control to stop the operation.

Within the AI continuous operation, there are two parameters *ScaledData* , *ScaledData* received In the `AiComplete` or `AiHalfReady` event. The two parameters are described below :

ScaledData as Variant.

This is a scaled data that has been translated from the binary code to the current AIrange value. The transformation depends on the feature of card and the `ReturnType` property that users specified.

BinaryCode as Variant.

This is a binary code from the buffer of PCI card. It may include the channel information. The value depends on the feature of card and the `ReturnType` property that users specified.

3. DI/AI continuous data streamToFile operations

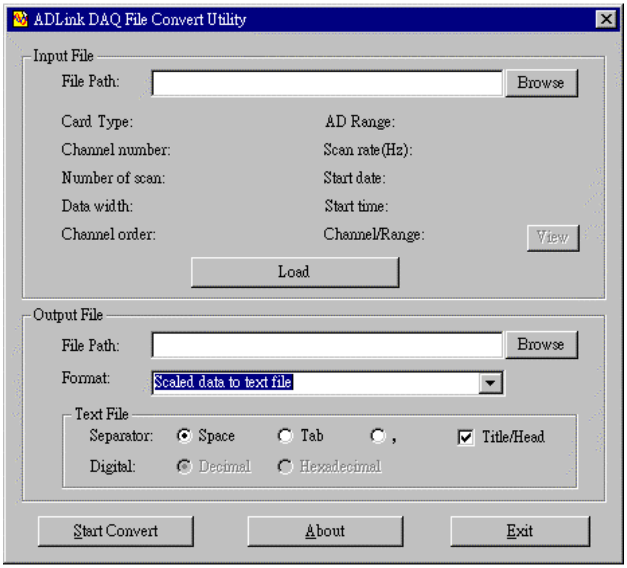
PCI7200, PCI7300, PCI9111, PCI9112, PCI9113, PCI9114, PCI9118 and PCI9812 controls can provide the continuous data streamToFile operation.User has to follow the procedures described below :

1. Set streamToFile property of the control to True.
2. Use the `StartContAI` or `StartContDI` method to start the operation. User has to pass the file name as the argument to the method. For example :

   Pci91111.StartContAI("f:\DAQBench\Samples\record.dat"),

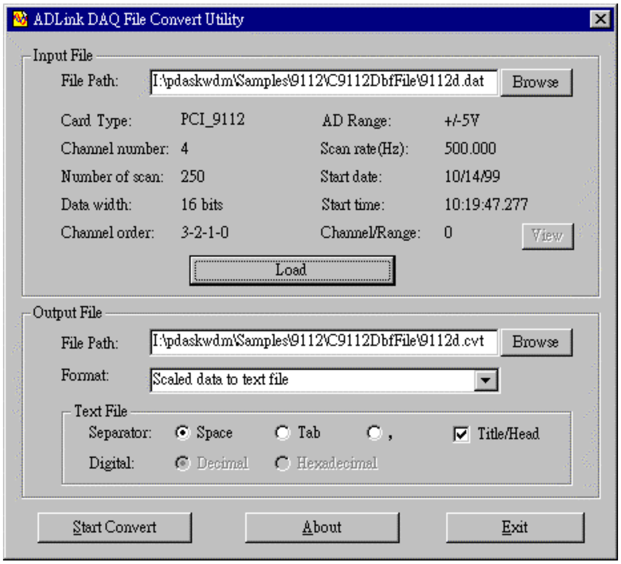
   Pci72001.StartContDI("f:\DAQBench\Samples\record.dat"))
3. The AI or DI continuous data will save to the file.
4. You can also receive the data from `AiComplete/DiComplete` or `AiHalfReady/DiHalfReady` event.
5. Use the `StopContAI` or `StopContDI` method to stop the operation.

The data files is written in binary format. Since a binary file can not be read by the normal text editor and can not be used to analyze the accessed data by Excel, DAQBench provides a convenient tool *DAQCvt* to convert the binary file to the file format read easily. The default location of this utility is <DAQBench InstallDir>\Util directory. The *DAQCvt* main window is as the following figure:



The *DAQCvt* main window includes two frames. The upper frame, *Input File frame* is used for the source data file and the lower frame is used for the destination file.

To **load the source binary data file**, type the binary data file name in *File Path* field or click *Browser* button to select the source file from *Input File frame*, and then click *Load* button. As the file is loaded, the information related to the data file, e.g. *data type*, *data width*, *AD Range*, …etc., are shown in the corresponding fields in "Input File" frame, and the default converted data file path and format are also listed as the figure below.


ADLink DAQ File Convert Utility

Input File

File Path: I:\pdaskwdm\Samples\9112\C9112DbfFile\9112d.dat

Card Type:	PCI_9112	AD Range:	+/-5V
Channel number:	4	Scan rate (Hz):	500.000
Number of scan:	250	Start date:	10/14/99
Data width:	16 bits	Start time:	10:19:47.277
Channel order:	3-2-1-0	Channel/Range:	0

Output File

File Path: I:\pdaskwdm\Samples\9112\C9112DbfFile\9112d.cvt

Format: Scaled data to text file

Text File

Separator: [✓ Space] Tab , [✓ Title/Head]

Digital: [✓ Decimal] Hexadecimal

The default **destination file** with a *.cvt* extension is located in the same directory as the source one. To change the default setting, type the file path you wish or click the *Browser* button from *Output File* frame to select the destination file location.

DAQCvt provides three types of data format conversion.

Scaled data to text file :

The data in hexadecimal format is scaled to engineering unit (voltage, ample, …etc) according to the card type, data width and data range and then written to disk in text file format. This type is available for the data accessed from continuous AI operation only.

Scaled data to binary file (float) :

The data in hexadecimal is scaled to engineering unit (voltage, ample, …etc) according to the card type, data width and data range and then written to disk in binary file format. This type is available for the data accessed from continuous AI operation only.

Binary codes to text file :

The data in hexadecimal format or converted to a decimal value is written to disk in text file format. If the original data includes channel information, the raw value will be handled to get the real data value. This type is available for the data accessed form continuous AI and DI operations.

The data separator in converted text file is selectable among *space*, *comma* and *Tab*.

If you want to add title/head which includes the card type information at the beginning of file, check the "Title/Head" box.

After setting the properties (File Path, Format, …etc) related to the converted file, you can push *Start Convert* button from the *Output File* frame to perform the file conversion.

5.12 NuDAM Controls for NuDAM Modules

The ActiveX controls of NuDAM modules are package to four OCXs which are NDDigital.OCX , NDAnalog.OCX, NDCounter.OCX and NDHost.OCX. The NDDigital.OCX includes digital I/O module ND6050, ND6052, ND6053, ND6054, ND6056, ND6058, ND6060, ND6063. The NDAnalog.OCX includes multiple function module ND6011, ND6012, ND6013, ND6014, ND6017, ND6018, ND6021, ND6024. The NDCounter.OCX includes counter module ND6080. The NDHost.OCX is NDHost control that is software object for representing the host computer. It would be use to process synchronization and watch dog function.

Using the NuDAM control, Users must us e the MSComm ActiveX control of Microsoft. The MSComm control is public because it is included at Windows 95 / 98 / NT/2000. User can configure the COM port and open/close RS232 COM port by using the MSComm control. When the MSComm control open the COM port, user can assign the `CommID` property of MSComm to the `PortHandle` property of NuDAM control. Then, users can easily program to drive the ADLINK NuDAM modules.

VB Example:

```
MSComm1.CommPort = 1
MSComm1.Setting = "9600,n,8,1"
MSComm1.PortOpen = True
ND60501.PortHandle = MSComm1.CommID
.  .  .  .  .  .  .
ND60501.DigitalOutput = 7
```

The image bitmap of Digital I/O ActiveX controls

6050 6052 6053 6054 6056 6058 6060 6063

The image bitmap of Analog I/O ActiveX controls

6011 6012 6013 6014 6017 6018 6021 6024

The image bitmap of Counter ActiveX controls

6080

The image bitmap of Host ActiveX controls

HOST

5.13 Analysis Control

With the DAQBench analysis control, you can perform operations such as matrix and array calculations, complex number analysis, statistical analysis and Fast-Fouri-Transform. User can receive data from DAQ or NuDAM controls. Then pass data to the DQAnalysis control to process analysis work. The result of analysis can be pass to the DGraph control to display.

The bitmap of DQAnalysis Control

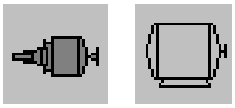
VB Example:

```
Dim tMean As double
Dim data(0 to 99)
For I=0 to 99
    Data( i ) = Rnd
Next
tMean = DQAnalysis.Mean(data)
```

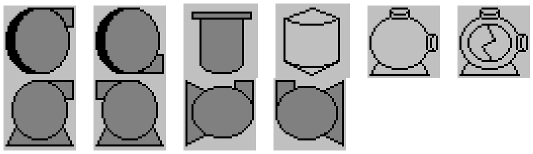
5.14 Equipment Controls

The DBEquip.OCX includes five ActiveX control for some equipment pattern of industry automation such as Pump, Pipe, Motor, Tank, Valve. These controls can be use to represent the equipment when users develop the MMI applications of industry automation. User can select variant style of each equipment control.

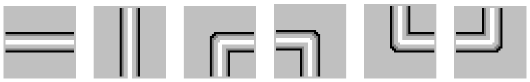
The pattern style of DMoter control



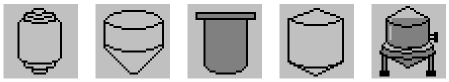
The pattern style of DPump control



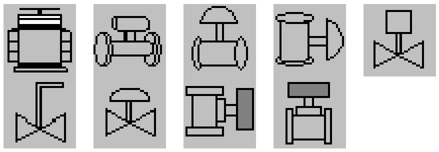
The pattern style of DPipe control



The pattern style of DTank control



The pattern style of DValve control



5.15 ExcelLinker Control

The ExcelLinker.OCX includes one ActiveX control for linking DAQ data to Microsoft Excel Application. The spreadsheet is one of the most commonly used tools among engineering, manufacturing, and management personnel. Using ExcelLinker ActiveX control, scientists and engineers can further increase productivity by integrating DAQ data collection directly into the Microsoft Excel worksheets.

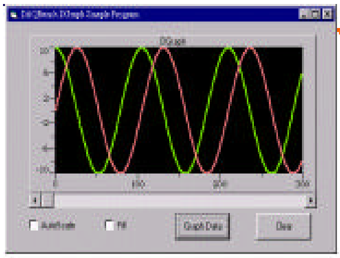
The description of using ExcelLinker control is listed below.

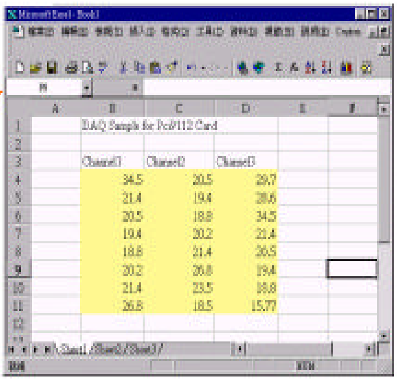
Specification:

1. Specify the file name of Excel, may be a new one or a exist file.
2. Specify the worksheet name in indicated excel file.
3. Specify the cell range for putting DAQ data in indicated worksheet.

In run time:

1. Retrieve DAQ data form DAQ ActiveX control of DAQBench.
2. Call ExcelLinking(Data) method of ExcelLinker control to linking Excel application. If excel have not been run then will be automatically invoked.
3. ExcelLinker will select the specified worksheet and put DAQ data into the specified cells.
4. Last, ExcelLinker will command Excel application to recalculate theFormulas in worksheet.





	A	B	C	D	E	F
1		DAQ Sample for Pci9112 Card				
2						
3		Channel1	Channel2	Channel3		
4		34.5	20.5	29.7		
5		21.4	19.4	28.6		
6		20.5	18.8	34.5		
7		19.4	20.2	21.4		
8		18.8	21.4	20.5		
9		20.2	26.8	19.4		
10		21.4	23.5	18.8		
11		26.8	18.5	15.77		
12						

5.16 WebSnapshot Control

The WebSnapshot.OCX includes one ActiveX control that can snapshot the image of application and export the image to web through http protocol. The Internet browser is common and public tool on Internet. Using WebSnapshot ActiveX control, user can easily use Internet browser to remote monitoring the application image because the WebSnapshot ActiveX control can automatically create template HTML file that would refresh the JPG file of application image.

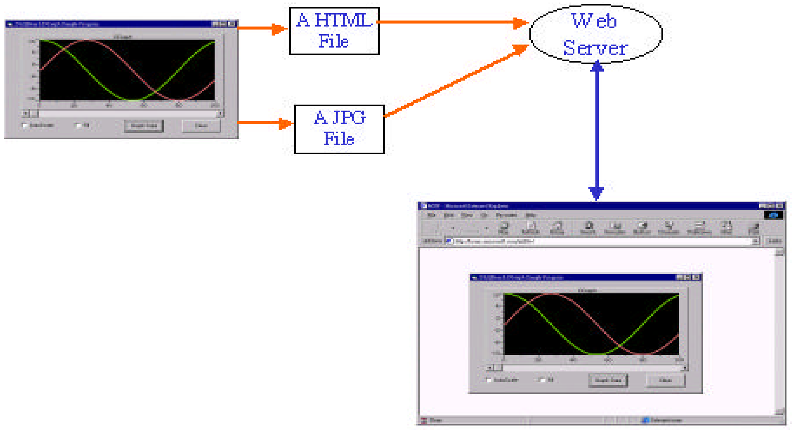
The description of using WebSnapshot control is listed below.

Specification:

1. Specify the file name of HTML, may be a new one or a exist file.
2. Specify the file name of JPG for storing the image of application.
3. Specify the operate mode, may be automatic or manual updated.
4. Specify the interval time of refresh image in automatic updated.
5. Create HTML file of refresh JPG file.

In run time:

1. Automatically capture image of application to the JPG file.
2. Manually, Call CaptureImage() method to capture image of application to the JPG file.
3. User can use Internet browser to browse the specified HTML file in remote machine.



5.17 DBAccess Controls

The DBAccess.OCX includes three ActiveX controls that can access database through ODBC. Open Database Connectivity (ODBC) is a standard or open application programming interface (API) for accessing a database. By using ODBC statements in a program, you can access datas in a number of different databases, including Access, dBase, DB2, Excel, and Text. Using the ActiveX controls of DBAccess, programmers don' t need to understand the detail ODBC API and only need to specify some information by using friendly property page, then user can easily write data to, read data from and delete data from Database.

The processes of using DBAccess controls are listed below.

DBWrite control

Specification:

1. Specify the data source name (DSN) of Database on ODBC.
2. Specify the tables and columns for writing data in specified Database

In run time:

1. Retrieve DAQ data form DAQ ActiveX control of DAQBench.
2. Call ExecuteWrite(*DataArray*) method to write data to specified Database.

DBRead control

Specification:

1. Specify the data source name (DSN) of Database on ODBC.
2. Specify the tables and columns for reading data in specified Database.
3. Specify the query condition.

In run time:

1. Call ExecuteRead(*DataArray*) method to read data from specified Database.
2. Uses can pass *DataArray* to DGraph control to display

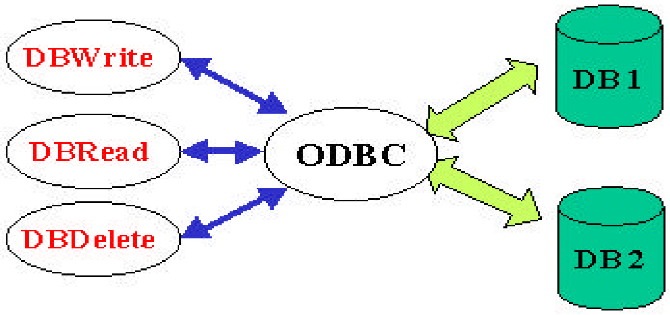
DBDelete control

Specification:

1. Specify the data source name (DSN) of Database on ODBC.
2. Specify the table for removing data in specified Database.
3. Specify the remove condition

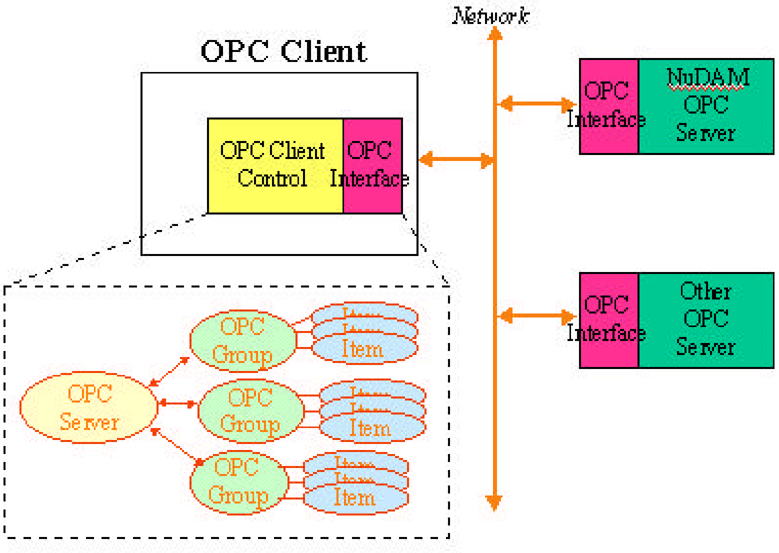
In run time:

1. Call ExecuteDelete() method to remove data from specified Database.



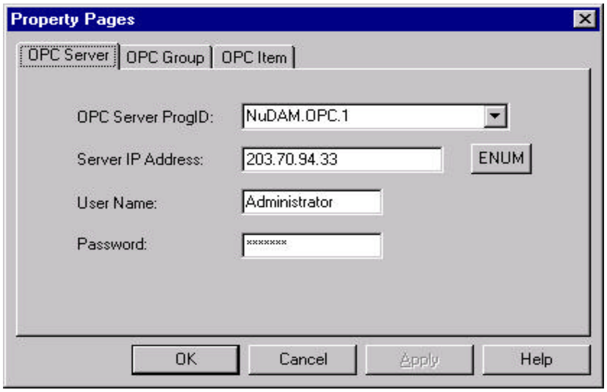
5.18 OPCClient Control

The OPCClient.OCX includes one ActiveX control that can connect, access data and disconnect the OPC Server. The OPC (OLE for Process Control) is constituted by OPC Foundation. It will be a standard interface for accessing process control data in industry automation. The OPC is the Client/Server model and based on the technology of COM/DCOM. Using OPC interface you can easily access process control data across Internet and can fulfill the integration between manufacture system and business system. OPC Server is supplied by hardware vendor. The OPCClient control use OPC interface to connect, access and disconnect to various OPC servers. Using the OPCClient control, You don' t need to know and program the OPC interface and only need to specify some information by a friendly user interface.



The following steps show how to use OPCClient control.

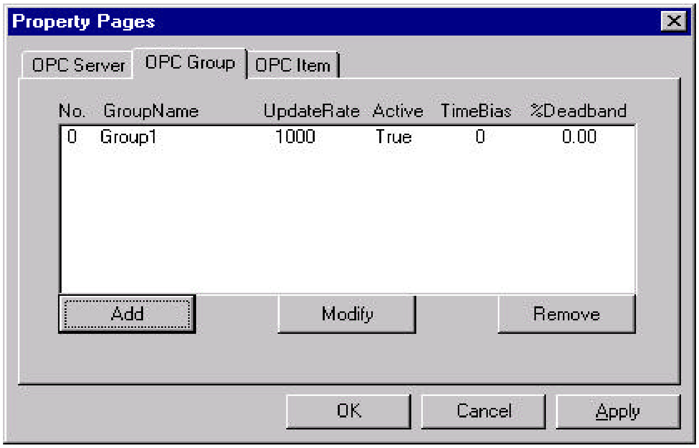
1. Specify the OPC Server ProgID and Server IP Address of OPC Server and us er account. If the OPC server is on the local machine then you do not care the IP Address and user account. About the OPC Server ProgID, user can use " ENUM" button to browser the OPC server ProgID or just keyin the ProgID of the OPC server(You can find the OPC server ProgID in each OPC server documents).



OPC Server ProgID: NuDAM.OPC.1
Server IP Address: 203.70.94.33
User Name: Administrator
Password: *******

Property page "OPC Server" of OPCClient ActiveX control

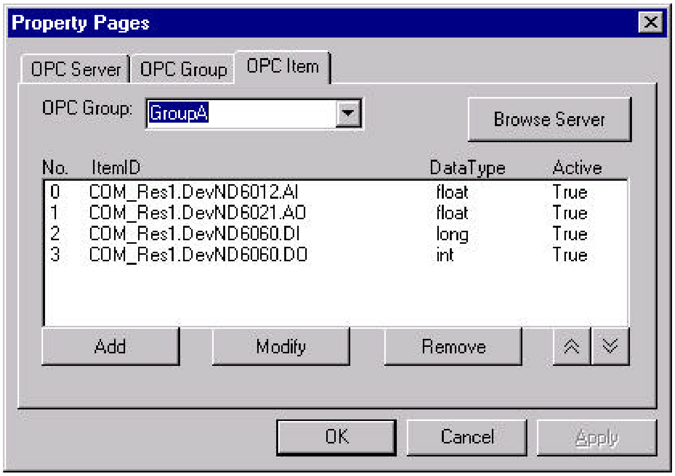
2. Create the OPC Groups that would own some data items and some attributes (ex. Update rate…)



No.	GroupName	UpdateRate	Active	TimeBias	%Deadband
0	Group1	1000	True	0	0.00

Property page "OPC Group" of OPCClient ActiveX control

3. Create the OPC Items for OPC Groups.



OPC Group: GroupA

No.	ItemID	DataType	Active
0	COM_Res1.DevND6012.AI	float	True
1	COM_Res1.DevND6021.AO	float	True
2	COM_Res1.DevND6060.DI	long	True
3	COM_Res1.DevND6060.DO	int	True

Property page "OPC Item" of OPCClient ActiveX control

4. In run time, you must first connect OPC server.

```
Result = OPCClient1.Connect()
```

5. Then, you can directly access OPC item as read/write variables.

```
//Read
OPCClient1.Group(0).ReadItems
Value1 = OPCClient1.Group(0).Item(0).Value
Value2 = OPCClient1.Group(0).Item(2).Value
//Write
OPCClient1.Group(0).Item(1).Value = 5.6
OPCClient1.Group(0).Item(3).Value = 3
```

5. Last, you must disconnect OPC server.

OPCClient1.Disconnect

5.19 Thermocouple Control

The ADLINK Thermocouple control supports three types of Thermocouple. They are the J-type, K-type and T-type Thermocouple. User can just assign the voltage value as the control method's input parameter, then the Thermocouple control converts the voltage value to the temperature value. A Thermocouple control example is described as below:

```
Dim Seekback_Temperature as Variant
Dim Temperature as Variant
Seekback_Temperature = Thermocouple1.SeebeckTemperature(298.3, 0)
Temperature=Thermocouple1.Temperature(34521.11,
                                   Seekback_Temperature, 1)
```

5.20 DDE/NetDDE Function

The DAQBench User Interface objects (except DGraph, DXYGraph and DIntenGraph objects) and Equipment objects now support the DDE (Dynamic Data Exchange) client capability. Therefore they can connect with DDE server applications for exchanging data. For example, DAQBench now can animate graphics with values coming from any DDE server or share data with DDE server via the DDE protocol. (Example : ISaGRAF target)

In order to connect with DDE server, user first must assign the appropriate properties values for the LinkTopic, LinkItem and LinkMode items in the DAQBench property page. These items are used to identify the DDE conversion. After identify it, user then can use the UI or equipment object's DDE methods to control the communication between DDE server and DAQBench.

The DAQBench DDE property setting example for UI and equipments object is described as below :

Control.LinkTopic= Application|topic (Application_name|topic_name);

Control.LinkItem=item (item_name)

Control.LinkMode=1 (Automatic) or others

There are three different link modes supported– 1(automatic), 2(manual), and 3(notify). If you set the LinkMode property to automatic, whenever the data specified by the combination of the LinkTopic and LinkItem changes, the control receives the new data automatically. For the controls having Change event, the event occurs. If you set the LinkMode property to manual or notify, the control is not update automatically and you must use the LinkRequest method to obtain new data from the DDE server. The difference of the two modes is that with notify link, the LinkNotify event occurs whenever the source has new data to supply to the control. You can also stop the conversation at any time by setting the LinkMode property to 0(None).

Please refer to the DAQBench function reference manual for the details of DDE properties, events and methods of **User Interface and equipment objects.**

In DAQBench, the UI and equipment objects can provide the DDE capability. The detail connect capabilities are described as below :

- In DBoolean object , the DDE conversion link with the value of the Object.
- In D7Segment object , the DDE conversion link with the value of the Object.
- In DLEDMeter object , the DDE conversion link with the value of the Object.
- In DSlide object , the DDE conversion link with the Pointer value of the Object. (The pointer1 to pointer8 can support DDE.)
- In DKnob object , the DDE conversion link with the Pointer value of the Object. (The pointer1 to pointer8 can support DDE.)
- In DChart object , the DDE conversion link with the Plot value of the object (The polt1 to polt8 can support DDE)
- In DMotor object , the DDE conversion link with the On/Off state of the Object.
- In DPipe object , the DDE conversion link with the Fill state of the Object.
- In DPump object , the DDE conversion link with the FanMode state of the Object.
- In DTank object , the DDE conversion link with the Value of the Object.
- In DValve object , the DDE conversion link with the State of the Object.

Based on the DAQBench DDE functions, DAQBench also can provide the NetDDE function. The NetDDE provides DAQBench the additional capabilities to get / set data of DDE server through the Microsoft Network. It is the remote control capability. User can use this capability on Win 95/98/NT/2000 OS platform.

The NetDDE property setting example is described as below :

Control.LinkTopic= \\Node\Application|topic

(\\ Node_name\Application_name|topic_name; Node_name is the name of the node(computer) which in the Microsoft network neighborhood. The DDE server is inside this machine.)

Control.LinkItem=item (item_name)

Control.LinkMode=1 (Automatic)

Depend on the Window OS platform, there are different ways to start the DDE Service.

- Window NT:

If you are using the NetDDE service on Windows NT, you need to set up DDE share for these nodes.

About the configuring of DDE Share, please follow the procedures described below:

◆ **To add a DDE Share on Windows NT operating systems.**

1. 1.On the **Start** menu of the Windows Taskbar, point to **Run**. In the **Run** dialog box that appears, type DDESHARE and then click OK. The DDE Share program' s main window appears.
2. From the **DDE Shares** menu, click **DDE** Shares. The **DDE Shares** dialog box appears.
3. Click **Add a Share.** The **DDE Share Properties** dialog box appears.
4. In the **Share Name** box, enter the name of the DDE server application and " |*" for the Share name. For example, if your server application name is ADLDDE, enter ADLDDE|*.
5. In the **Application Name** box, enter the name of the application again.
6. In the **Topic Name** box, enter " *" .
7. Click **Permissions**. The **DDE Share Name Permissions** dialog box appear.
8. Select " Everyone" in the Name list and " Full Control" as the Type of Access.
9. Click **OK** to exit the DDE Share Name Permissions dialog box and return to the DDE Share Properties dialog box.
10. Click **OK** to return to the DDE Shares dialog box.

Now the Share you created will be included in the DDE Shares list.

(For more information on using the DDE Share program, see your Microsoft documentation.)

◆ **To configure trusted DDE share.**

1. From the **DDE Shares** menu, click **DDE** Shares.
2. In the **DDE Shares** dialog box that appears, select the DDE share for which you want to set up a trust relationship.
3. Click **Trust Share**.
4. The **Trusted Share Properties** dialog box appears.
5. Click the **Start Application Enable** and **Initiate to application Enable** options.
6. Click **OK**.

- Window 95 :

To run NetDDE program in Windows95, you must add a shortcut for Netdde.exe to the Startup group(The Netdde.exe is in the Window95 directory.). To do so, use the following four steps:

1. Use the right mouse button to click an empty space on the taskbar, and then click Properties on the menu that appear.
2. On the Start Menu Program tab, click Add.
3. Use the Create Shortcut Wizard to create a shortcut for Netdde.exe in the Windows folder.
4. After you create the shortcut, restart your computer.

6

NuDAQ Configuration

Before you begin your NuDAQ application development, you must configure your NuDAQ devices. NuDAQ needs the device configuration information to program your hardware properly.

In most cases you follow the same general steps:

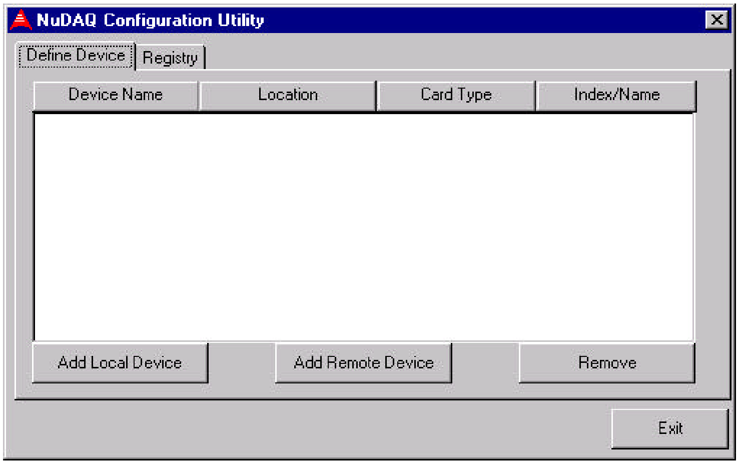
1. Install your application software.
2. If your platform is Windows 98 or Windows 2000, you have to install DAQ hardware divice when you play NuDAQ card and enter Windows. Please refer to “NuDAQ PCI and NuIPC CompactPCI DAQ cards software Installation Guide”.
3. Configure your device using the NuDAQ Configuration Utility.
4. Define your device using the NuDAQ Configuration Utility.

6.1 Using NuDAQ Configuration Utility

Using the NuDAQ Configuration Utility, you can:

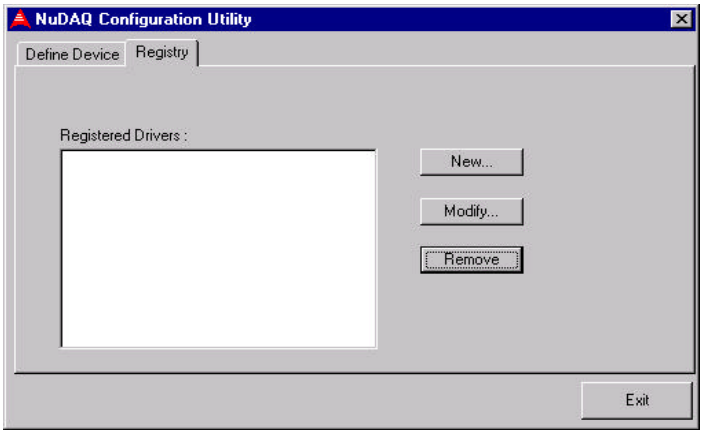
1. Registry NuDAQ device drivers to Windows in your system (NT only).
2. Configure the Continuous AI/AO/DI/DO buffers of NuDAQ cards.
3. Define NuDAQ devices that may be local or remote device in your system.
4. Save the NuDAQ device configuration to the configuration file.

The utility, NuDAQCfg.EXE, is installed in your DAQBench\PCIDAQ directory.

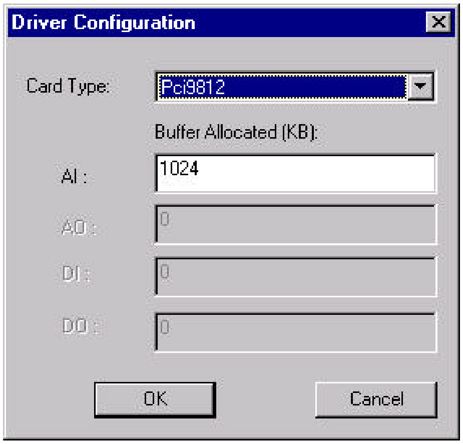


6.1.1 Register NuDAQ cards for Windows NT

The NuDAQ devices must be registered at Window Registry before the NuDAQ applications are run. You can use NuDAQ Configuration Utility to do the registry of NuDAQ cards. On NuDAQ Configuration Utility window, Select "**Registry**" panel and view the window as below.



This Panel is used for users to ***make the registry*** of local NuDAQ PCI device drivers, ***remove*** installed drivers and ***modify*** the allocated buffer sizes of AI, AO, DI and DO. Click "**New**" or "**Modify**" button and popup a Driver Configuration dialog for specifying the allocated buffers as below.


Driver Configuration

Card Type: Pci9812

Buffer Allocated (KB):

AI : 1024

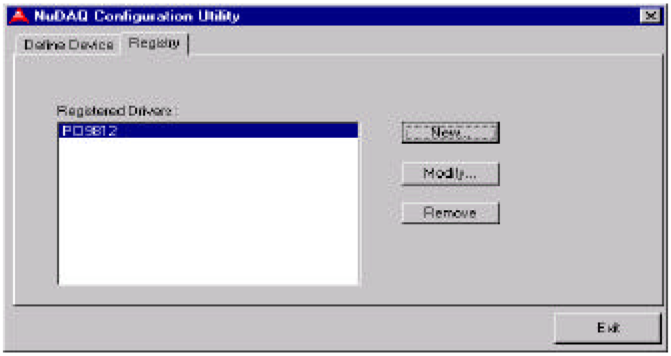
AO : 0

DI : 0

DO : 0

The allocated buffer sizes of AI, AO, DI, DO represent the sizes of contiguous Initially Allocated memory for continuous analog input, analog output, digital input, digital output respectively. Its unit is *KB*, i.e. 1024 bytes. Device driver will try to allocate these sizes of memory at system startup time. The size of initially allocated memory is the maximum memory size that continuous AI/AO/DI/DO can be performed on this type of cards plugged on this local machine. It will induce an unexpected result in that the data size of continuous operation exceeds the initially allocated size.

After the device configurations of the driver you select is finished, click “OK” to register the driver and return to the *NuDAQCfg* main window. The driver you just registered will be shown on the registered driver list as the following figure:

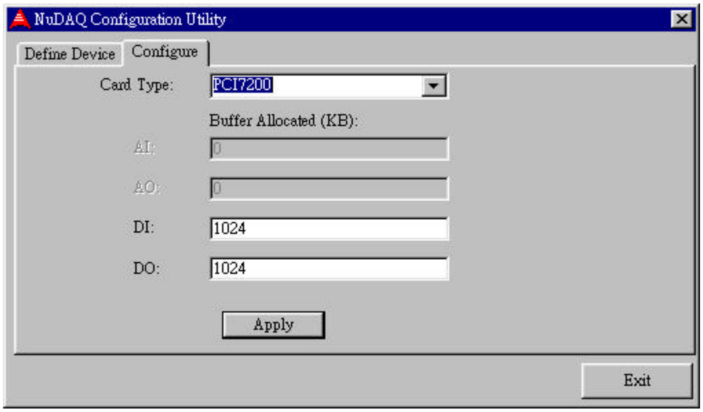


Registered Drivers:
PCI9812

Then you can Click “Exit” button to exit the driver registry utility. To make the registered drivers work, you have to restart Windows NT system.

6.1.2 Configure NuDAQ cards for Windows 98 or Windows 2000

Windows 98/2000 and NuDAQ PCI cards work very well together because Windows 98/2000 includes Plug and Play capabilities and standard drivers for PCI card devices. On Windows 98/2000, NuDAQ cards don’t need to do registry work but they must allocate memory buffer for continuous operation. You can use NuDAQ Configuration Utility to specify the size of contiguous Initially Allocated Memory for analog input, analog output, digital input and digital output. On NuDAQ Configuration Utility window, Select “**Configure**” panel and view the window as below.



Card Type: PCI7200

Buffer Allocated (KB):

AI: 0

AO: 0

DI: 1024

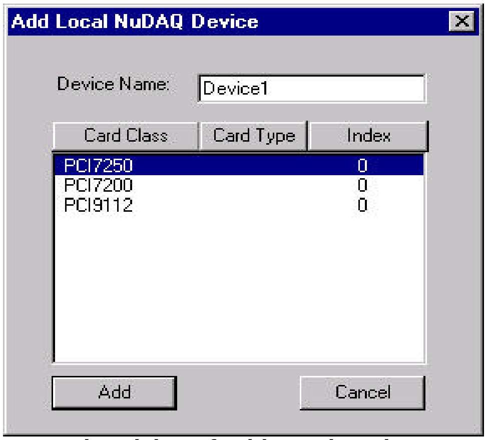
DO: 1024

The allocated buffer sizes of AI, AO, DI, DO represent the sizes of contiguous Initially Allocated memory for continuous analog input, analog output, digital input, digital output respectively. Its unit is *KB*, i.e. 1024 bytes. Device driver will try to allocate these sizes of memory at system startup time. The size of initially allocated memory is the maximum memory size that continuous AI/AO/DI/DO can be performed on this type of cards plugged on this local machine. It will induce an unexpected result in that the data size of continuous operation exceeds the initially allocated size.

After the device configurations of the driver you select is finished, click " Apply" to register the driver.

6.1.3 Define local device

You can click **Add Local Device** button and will popup one dialog box. In the list box of Add Local NuDAQ Device dialog, you can find some ADLINK NuDAQ cards that are currently installed on this machine. Then, you can select one card and enter its device name. Click **Add** button and this local NuDAQ card is defined and addded in the list box of Define Device Panel.



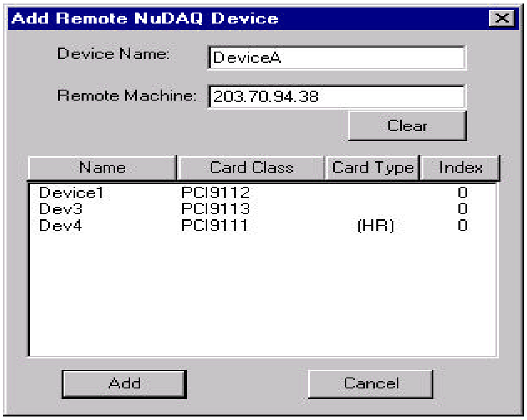
Device Name: Device1

Card Class	Card Type	Index
PCI7250		0
PCI7200		0
PCI9112		0

The Dialog of Add Local Device

6.1.4 Define remote device

You can click **Add Remote Device** button and will popup one dialog box. You must first connect to the NuDAQ RDA Server. So, you would specify the IP address or name of remote machine then click **Connect** button. If the connection is successful then the defined devices of RDA Server will be retrieved and adding in the list box of dialog. Otherwise, An error dialog will be popup that notify the connection is failed for RDA Server. You can select one remote device and enter its device name. Click **Add** button and this remote NuDAQ card is defined and added in the list box of Define Device Panel.

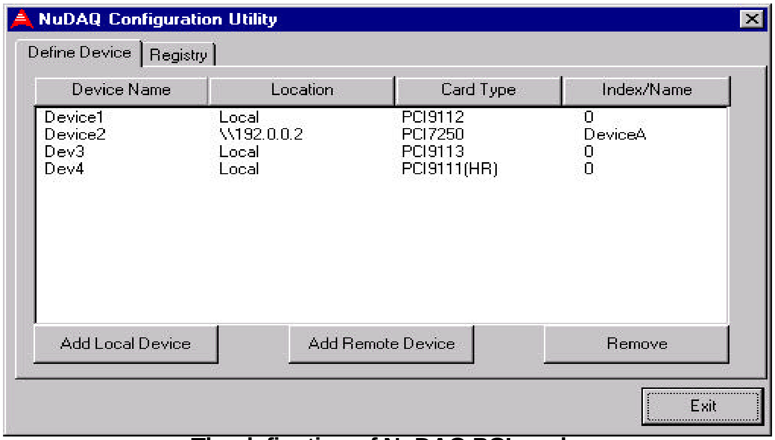


Device Name: DeviceA

Remote Machine: 203.70.94.38

Name	Card Class	Card Type	Index
Device1	PCI9112		0
Dev3	PCI9113		0
Dev4	PCI9111	(HR)	0

The Dialog of Add Remote Device



Device Name	Location	Card Type	Index/Name
Device1	Local	PCI9112	0
Device2	\\192.0.0.2	PCI7250	DeviceA
Dev3	Local	PCI9113	0
Dev4	Local	PCI9111(HR)	0

The defination of NuDAQ PCI cards

6.2 NuDAQ Remote Device Access

The Remote Device Access (RDA) feature lets you use ADLINK data acquisition (DAQ) devices plugged into other computers on your network. This feature can works with programs using NuDAQ PCI ActiveX controls of DAQBench. Your programs need no modifications to use this feature. Access to the remote devices are transparent to your program.

Using NuDAQ Remote Device Access:

1. Create one or more NuDAQ RDA servers.

a. Install DAQBench.

b. Install and configure your data acquisition (DAQ) devices.

Make sure your device drivers are starting properly by viewing the drivers status at Devices dialog of Control Panel. Open the NuDAQ RDA Server Utility from **Start»Programs» ADLINK DAQBench** for Windows menu. If you want this computer to become an RDA server every time it starts up, place a shortcut to the NuDAQ RDA Server Utility in your Startup folder. The RDA server must remain running; your computer cannot act as a n RDA server if the server program is not running.

2. Create one or more NuDAQ RDA clients.

a. Install DAQBench.

Open the NuDAQ Configuration Utility from **Start»Programs»ADLINK DAQBench** for Windows menu. Click **Add Remote Device**. Locate the computer that you have designated as the RDA server and connect to the server. You must enter the IP address or name of the RDA server into the **Remote Machine** field and click on the **Connect** button.

3. Select the device you want to use and enter the device name.

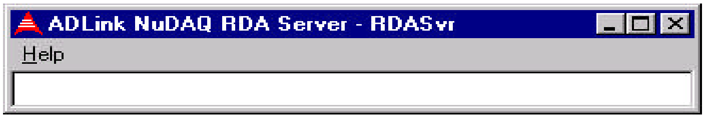
A list of remote devices installed and defined on that server appears in the list box. Select one of these devices by clicking once on its item row and enter a unique device name into **Device Name** field, then click **Ok** button. The new defined device will now appear in the list box and its location will be the IP address or name of the RDA server. The device name you have defined is the one you will use in your local programs. To connect to a different computer and its devices, repeat steps 2 and 3.

4. Undo your device name assignment to a remote device.

To undo the device name assigned to a remote device. highlight the device you want to undo and click **Remove** button.

6.2.1 NuDAQ RDA Server

An NuDAQ RDA server is any computer on which DAQBench is installed, on which one or more DAQ devices are installed and configured, and on which the NuDAQ RDA Server Utility is running. The NuDAQ RDA Server Utility is the program on your RDA server that enables its RDA server capability. The utility, RDASvr.EXE, is installed in your DAQBench\PCIDAQ directory. The “NuDAQ RDA server” window is shown as below.



6.2.2 RDA Considerations

Just as two different applications running on the same computer can use the same data acquisition (DAQ) device, two different NuDAQ RDA clients can use the same remote device on an RDA server. While this flexibility is usually a good thing, nothing prevents one RDA client from interfering with another client’s use of a remote device.

If you have assigned a local device name to a remote device, DAQBench attempts to connect to the RDA server computer each time it loads in your client computer. If your RDA server is not turned on, your client computer will appear to stall for a while during this process. This stalling is the timing out of the attempt to connect to the server computer. When the connection fails, any attempt to use the remote device will return FALSE.

A computer can be both an RDA client and an RDA server at the same time. Clients and servers can run any mix of Windows 98 and Windows NT 4.0 and Windows 2000. A program running on the RDA server can use a device local to that server while a program running on an RDA client is using the same device. You can open and close the NuDAQ ActiveX control of DAQBench on RDA client computers without affecting the state of devices on RDA servers to which the client has assigned local device name.

7

Distribution of Applications

About the distribution of application with DAQBnech ActiveX control objects, please contact ADLINK for the ADLINK DAQBenech object distribution policy.

e-mail : service@ADLINK.com.tw

e-mail : sw@ADLINK.com.tw