DAQBench®

32-bit ActiveX Controls for Measurement and Automation

User's Guide

Manual Rev: 2.30: November 15, 2001

Part No: 50-10015-103

The information in this document is subject to change without prior notice in order to improve reliability, design and function and does not represent a commitment on the part of the manufacturer.

In no event will the manufacturer be liable for direct, indirect, special, incidental, or consequential damages arising out of the use or inability to use the product or documentation, even if advised of the possibility of such damages.

Trademarks

NuDAQ, NuDAQ, DAQBench series product are registered trademarks of ADLINK Technology Inc. IBM PC is a registered trademark of International Business Machines Corporation. Other product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.

Getting Service from ADLINK

- Customer Satisfaction is 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.adlinktech.com		
Sales & Service	service@adlinktech.com		
Technical	NuDAQ + USBDAQ	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	
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 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.

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**, **"*DAQBench Applications*"**, describes how you can use DAQBench controls to develop your measurement or automation applications.
- **Chapter 3**, **"*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 4**, **"*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 5**, **"*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 6**, ***"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 7**, ***"Distribution of Applications"*,** direct you how and where to know the policy and information of distribution of applications.

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 measurement or automation applications. With DAQBench, you can easily develop custom user interfaces to display your data, analyze data you acquired or received from some other sources, and integrate with popular applications or data sources. Also you can develop automation or distributed applications with its SCADA functionality.

The DAQBench ActiveX controls are designed for use in Visual Basic. However, you can use ActiveX controls in any application that supports them, including Visual C++, Borland C++ Builder, and Delphi.

Currently DAQBench includes the following modules:

- User Interface Controls – Present your data. These controls include graphs, charts, sliders, thermometers, tanks, knobs, seven segment display, meters, and switches.
- Information Integration Controls – Integrate your information with Web, Excel, database; acquire data from OPC servers. These controls include Excel linker, Database access for ODBC, Web snapshot for browser, OPC client for OPC server, and Thermocouple.

- 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.
- SCADA Controls and Utilities

  Tag server and utility – A data center that communicates with OPC servers, processes alarm, and logs data.

  Historical and real-time trends.

  Alarm Controls – Alarm display and Acknowledgement button.

  Report Controls – Alarm or data reporting

  Equipment Controls – Display some popular equipment patterns in industry automation. These patterns are convenient to develop a HMI or SCADA system.

Please use the online help or reference manuals for specific information about the properties, methods, and events of the individual ActiveX controls.

1.2 Installing DAQBench

1.2.1 System Requirements

- Microsoft Windows 95/98/NT/2000 operating system
- 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 64MB of memory
- Minimum of 20MB 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 DAQBench

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

Complete the following steps to install DAQBench or the separate ActiveX modules.

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).

1.2.3 Installed Files

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

ActiveX controls

Directory: Windows system directory
(`\Windows\system` for Windows 95/98)
(`\Windows\system32` for Windows NT/2000)

Example programs

Directory: `\DAQBench\Samples\VB`
`\DAQBench\Samples\VC`
`\DAQBench\Samples\BCB`

PDF Manual Files

Directory: `\DAQBench\Manual`

One-line Help Files

Directory: Windows system directory

Utility Files

Directory: `\DAQBench\Util`

VC++ Data Type Wrapping Library Files

Directory: `\DAQBench\Varpacker`

Tag server and utilities

Directory: `\DAQBench\Tag`

1.3 Getting Help

In addition to this manual, the following sources can provide you with more information about DAQBench:

- DAQBench Function Reference – The manual contains the complete reference information for the DAQBench controls. You can access the PDF file on the ***ADLINK DAQBench CD***, or from the Windows **Start** menu **Programs>>DAQBench>>DAQBench Function Reference**
- DAQBench online reference – The help contains the complete reference information for the DAQBench controls. You can access the help from the Windows **Start** menu **Programs>>DAQBench>>DAQBench Online Help**. You also can open the online reference from within most programming environments by clicking on the **Help** button in the custom property pages of a DAQBench control
- Examples – We provide Visual Basic, Visual C++, and C++ Builder example programs. The installer copies the example programs to `DAQBench\Samples`

1.4 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 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 controls in your particular environment.

1.4.1 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.

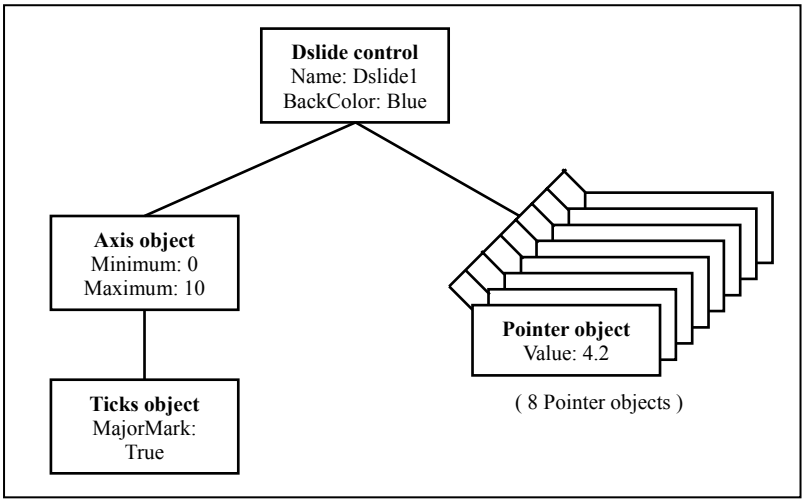
The DGraph control also has a series of methods, or functions, that you can invoke to perform a particular operation. For example, you can use the PlotGraph method to pass an array of data to the DGraph control for drawing.

1.4.2 Object Hierarchy

As described in the previous section, each ActiveX control has properties, methods, and events. Certain ActiveX controls are very complex, containing many different properties. Therefore, 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 properties 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.

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 of its own. Ticks object has properties, such as MajorMark, MajorColor, MinorMark, MinorColor. The DSlide object contains eight Pointer objects. Each Pointer object has its own properties, such as Value, Style, FillColor.

1.5 Setting the Properties of an ActiveX Control

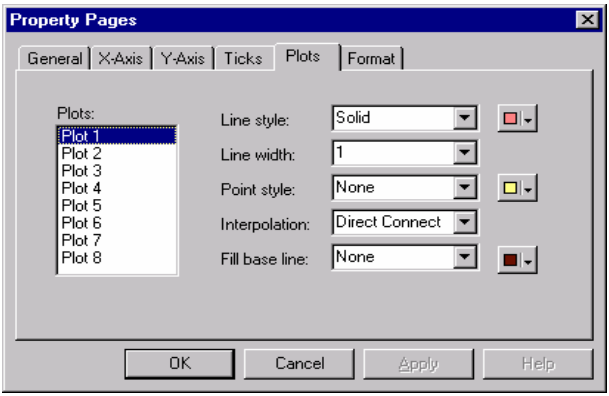
You can modify the properties of an ActiveX control from its property pages or directly from the program.

1.5.1 Using Property Pages

Once you place the control on a form in your programming environment, right click on the control and select **Properties...** A custom property page appears with a variety of properties that you can set to customize the appearance and operation of the control.

Use the property pages to set the property values for each ActiveX control at design time. The property values you select at this point represent the state of the control at the beginning of your application.

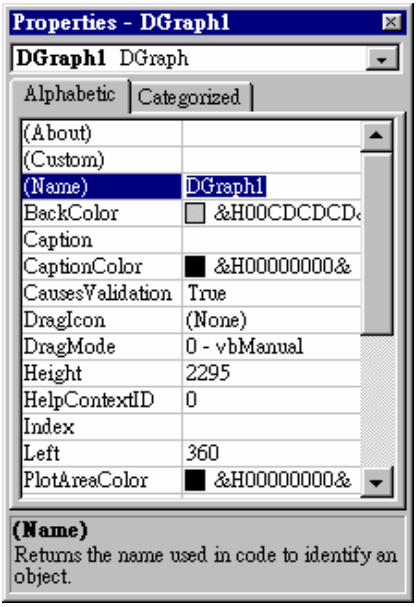
The layout and functionality of the custom property pages vary for different controls. The following illustration shows the custom property page for the DGraph control.



DAQBench Custom Property Pages

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.

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



Visual Basic Default Property Sheets

1.5.2 Changing Properties Programmatically

You can also set or read the properties of your controls programmatically. The 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 programming languages.

You can set the value of a property on a top-level object with the following syntax.

```
object.property = expression
```

For example, you can change the Value property of a 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:

```
DGraph1.XAxis.ScrollBar = True
```

In the above code, XAxis is a property of the DGraph control and refers to an Axis object. `ScrollBar` is one of Axis properties. The DGraph control also has a YAxis property that refers to a different Axis object.

You can get the value of a property from your program. In most case, to get the value of a property, you use the following syntax:

```
Variable = object.property
```

For example, you can display the `ViewNumber` used by the DGraph control with the following code.

```
Text1.Text = DGraph1.YAxis.ViewNumber
```

1.6 Using Control Methods

ActiveX controls and objects have their own methods, or functions, that you can call from your program. Methods can have arguments that you pass to the method, and return values that pass information back to your program. To call a method, add the name of the method after the name of the control. When a method doesn't take arguments, you call the method using the following syntax:

```
object.method
```

For example, the `ClearPlots` method clears the drawing area of a DChart control.

```
DChart1.ClearPlots
```

Methods can have arguments that you pass to the method, and return values that pass information back to your program. For example, the `PlotGraph` method of the DGraph control has two required arguments -- The array of scaled data to be plotted and the index of plot -- that you must include when you call the method.

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

Depending on your programming environment, the parameters might be enclosed in parentheses.

```
DGraph1.PlotGraph(ScaledData, 0)
```

1.7 Developing Event Handlers

After you configure your controls on a form, you can create event handlers in your program to respond to events generated on the controls. For example, the DSlide control has a Change event that fires (occurs) when the value of the `Value` property changes.

To develop the event routine code, most programming environments generate a skeleton function to handle each event. For example, the Visual Basic environment generates the following function skeleton into which you insert code when the Change event occurs.

```
Private Sub DSlide1_Change(ByVal PointerNo As
Integer,
```

```
        ByVal Value As Variant)

End Sub
```

1.8 Using the Analysis Library

The Analysis Library of DAQBench is packaged as one ActiveX control, named DQAnalysis. After adding the Analysis controls to your programming environment, use the analysis functions like any other method on a control.

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

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

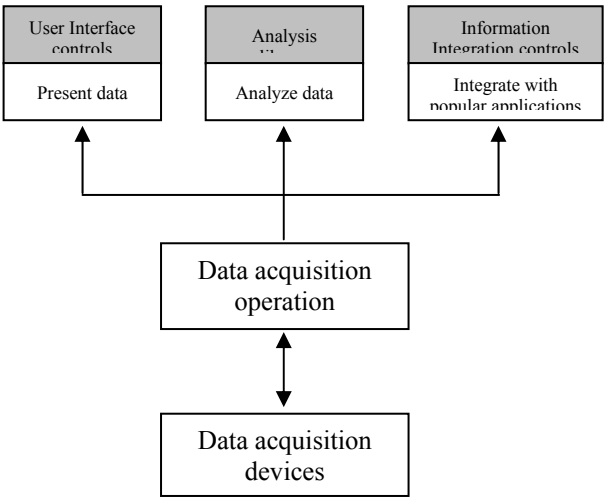
2

DAQBench Applications

This chapter describes how you can use DAQBench controls to build your measurement or automation applications.

2.1 Measurement Applications

DAQBench contains user interface, analysis, and information integration controls that can present and analyze acquired data, integrate the information with popular applications. With the help of DAQBench, you can easily create your measurement applications.



With regards to the data acquisition operations, you can choose one of the following ways:

1. **DLL library**

   You can call the DLL functions to perform the data acquisition task. The acquired data can be passed to DAQBench controls to present, analyze, or integrate with other applications. ADLINK freely provides the DLL libraries for our data acquisition devices. You can find the appropriate DLL library on ADLINK All in One CD.

2. **ActiveX controls**

   We strongly recommend you to use data acquisition ActiveX controls to work with DAQBench. ADLINK provides four data acquisition ActiveX controls packages for our data acquisition devices:

   - PCIS-OCX: NuDAQ PCI cards
   - NDS-OCX: NuDAM modules
   - MOTION-OCX: PCI motion control cards
   - HSL-OCX: High Speed Link modules

   With the data acquisition ActiveX controls, the programming becomes easier and the data can integrate with DAQBench ActiveX controls seamlessly. The data acquisition ActiveX controls are included in DAQBench CD. Or you can find them in ADLINK All in One CD.

2.2 HMI/Automation Applications

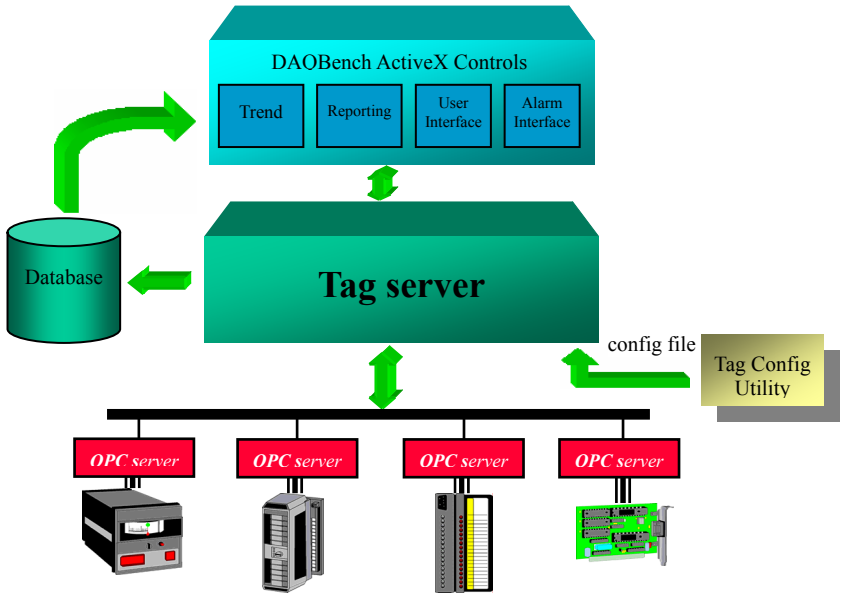
The DAQBench SCADA controls and utilities can help you create automation applications. It is suitable for supervisory control applications that need alarm handling and logging, data logging, trending, etc. Taking advantage of the OPC server networking capability, you can easily develop distributed HMI applications.

The SCADA controls and utilities provide the following capabilities:

- Tag configuration utility
- Monitor and contro tags through Tag Server

- Alarm handling and logging
- Automatic data logging
- Real-time and historical trending
- Data and alarm reporting
- Equipment diagrams

The operation architecture is shown below:



Tag Configuration Utility

You can use Tag Configuration Utility to create tags, connect tags with OPC server items, and configure their properties, including if the tag data is logged to database, how the tag scaled, the alarm levels and priorities, etc. With the tags connection with OPC server items, Tag Server reads data from and writes data to the specified OPC server items.

All of the configuration settings are saved to a configuration file (ADTag.cfg). The Tag Server uses the configuration file to operate.

Tag Server

The Tag Server is the heart of the DAQBench SCADA/HMI function. The Tag Server maintains the defined tags. A tag is a data point that connects to a real-world I/O point through specified OPC server.

The Tag Server performs the following tasks:

- Communicates with specified OPC servers
- Logs historical data and alarms to database (.mdb format)
- Scales data
- Processes alarms

DAQBench ActiveX Controls

DAQBench User Interface and SCADA controls can access and display the real-time tag data or historical data in database.

3

Building DAQBench Applications with Visual Basic

This chapter describes how you can use DAQBench controls with Visual Basic.

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 function reference manual and online help.

3.1 Developing Visual Basic Applications

3.1.1 Adding the DAQBench Controls to a Project's Toolbox

Before building an application using the DAQBench controls, you must add them to the Visual Basic toolbox. The DAQBench ActiveX controls are divided into different groups including user interface controls (Dbui.ocx), graph controls (DBGraph.ocx), equipment controls (DBEquip.ocx), analysis library controls (DQAnalysis.ocx), ExcelLinker control (ExcelLinker.ocx), WebSnapshot control (WebSnapshot.ocx), DBAccess controls (DBAccess.ocx), OPCClient control (OPCClient2.ocx), Thermocouple control (Thermocouple.ocx), Trend controls (Trend.ocx), etc.

To add DAQBench controls to the project's toolbox.

1. In a new Visual Basic project, right click on the toolbox and select **Components...**. The *Components* dialog box is displayed.

2. You can find the DAQBench controls which beginning with the "DAQBench".

3. Select the check box to the left of the controls to select the controls you want to use in your project.

4. Choose **OK** to close the *Components* dialog box. All of the ActiveX controls that you selected will now appear in the toolbox.

3.1.2 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. The control appears on the form. You can then move and resize the control by using the mouse. To move a control, use the mouse to drag the control to the desired location on the form. To resize a control, select the control by clicking it with the mouse, and place the mouse pointer on a sizing handle. Drag it to the desired size.

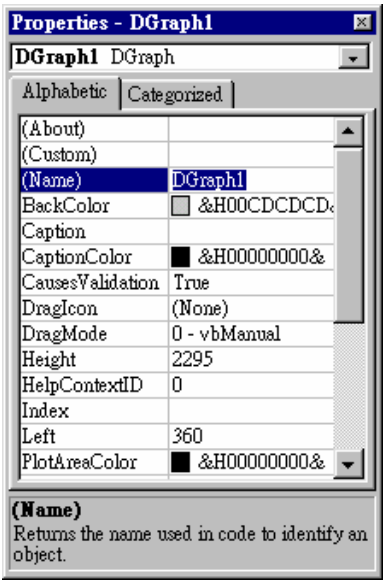
Once ActiveX controls are placed on the form, you can edit their properties using their property sheets or custom property pages. You can also edit the properties from within the Visual Basic program at run time.

3.1.3 Setting Properties at Design Time

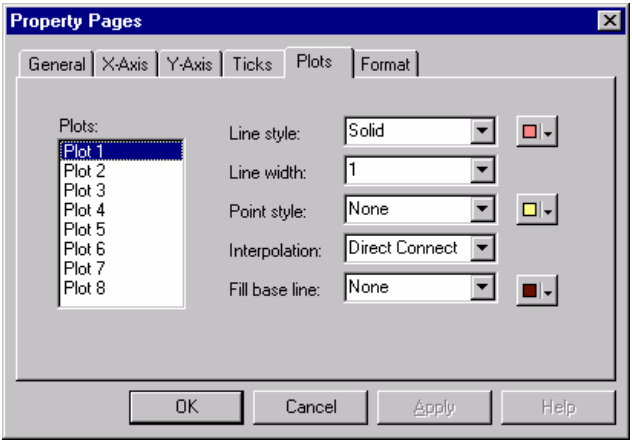
After placing a control on a Visual Basic form, configure the control by setting its properties with the Visual Basic *Properties window* or DAQBench custom control property pages (illustrated below). Visual Basic assigns some default properties, such as the control name and the tab stop. When you create the control, you can edit these stock properties in the

Visual Basic *Properties window*. To open the *Properties window*, select the **Properties Window** command from the **View** menu, click the **Properties Window** button on the toolbar, or use the context menu for the control. To edit a property, from the properties list, select the name of a property. In the right column, type or select the new property setting.

DAQBench controls supply the custom property pages for you to easily set the properties. We suggest you to use custom property pages to set the properties except the stock properties. To open the custom property pages, right click on the control on the form and select **Properties...**.



Visual Basic Properties window



DAQBench custom property pages

3.1.4 Edit Properties at Runtime

You can set and read the properties of your controls programmatically in Visual Basic. To set the value of a property, use the following syntax:

```
object.property = expression
```

For example, if you want to change the state of a DBoolean control during program execution.

```
DBoolean1.Value = 3
```

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 DChart control.

```
DChart1.Xaxis.Interval = 10
```

In the above code, `Interval` is a property of the sub-object `XAxis`.

You can get the value of a property from your program. In most case, to get the value of a property, you use the following syntax:

```
Variable = object.property
```

3.1.5 Working with Control Methods

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. When a method doesn't take arguments, you call the method using the following syntax:

```
object.method
```

For example, the `ClearPlots` method clears the drawing area of a DChart control.

```
DChart1.ClearPlots
```

Methods can have arguments that you pass to the method, and return values that pass information back to your program. For example, the `PlotGraph` method of the DGraph control has two required arguments -- The array of scaled data to be plotted and the index of plot -- that you must include when you call the method. In Visual Basic if you call a method without assigning a return variable, any arguments passed to the method are listed after the method name, separated by commas without parentheses.

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

If you keep the return value of a method, you must enclose the arguments in parentheses. For example, the GetState method returns the state (true or false) of a button of the DBoolean control.

```
result = DBoolean1.GetState(0)
```

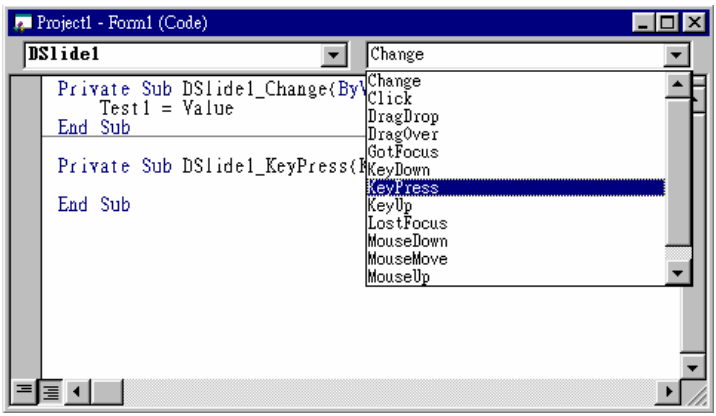
3.1.6 Developing Control Event Procedures

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 procedure code for an ActiveX control, double click the control to open the *Code* window, which automatically generates a default event procedure for the control. The following code is an example of the event procedure generated for the DSlide control. This code is executed when the value of the slide is changed.

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

End Sub
```

To generate an event procedure for a different event of the same control, you can select the desired event from the right pull-down menu in the code window.



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.

3.1.7 Learning to Use Specific DAQBench Controls

Each DAQBench control and its use are described in more detail in other sections of this manual. However, these sections do not discuss every property, method, and feature of every control. The DAQBench function reference manual or online help contains detailed information about each control and all its associated properties, events, and methods. Refer to them to find descriptions of the different features of a particular control.

4

Building DAQBench Applications with Visual C++

This chapter describes how you can use DAQBench controls with Visual C++.

At this point you should be familiar with the general structure of ActiveX controls as well as C++ programming and the Visual C++ environment. The individual DAQBench controls are described in the function reference manual and online help.

4.1 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++. To create a project compatible with the DAQBench ActiveX controls, use the Visual C++ MFC AppWizard to create a skeleton project and program.

2. Add the ActiveX controls you need to the controls toolbar. From the toolbar, you can add the controls to the application itself.

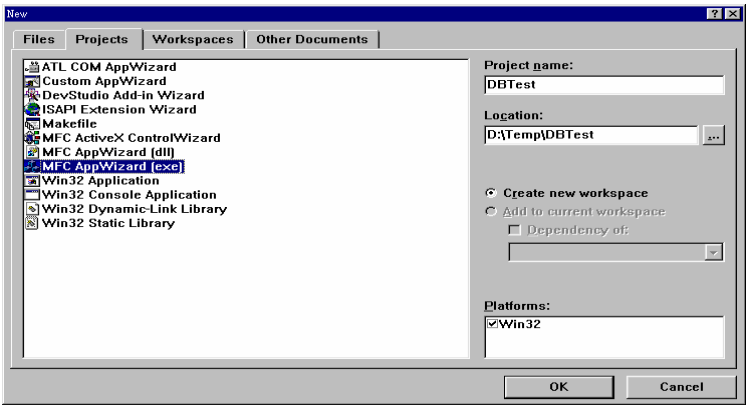
3. After adding a control to your application, you can configure its properties by its property pages.

4. While developing your program code, use the control properties and methods and create event handlers to process different events generated by the control.

4.1.1 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 clicking **New...** from the **File** menu. The **New** dialog box opens.



New Dialog Box

2. On the **Projects** tab, select the **MFC AppWizard (exe)** and click **OK** to launch the wizard. If necessary, specify the directory where the project workspace files are stored by using the *Location* box.

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

Before building an application using the DAQBench controls, you must add the controls into your Visual C++ project. To add controls to the project, use the following procedure:

1. On the Project menu, point to Add To Project, then click Components and Controls.
2. In the *Components and Controls Gallery* dialog box (the Gallery), expand the *Registered ActiveX Controls* folder and select the ActiveX controls you want to add to your project. Click **Insert**. Please notice that all DAQBench controls start with `DAQBench`.
3. The *Confirm Classes* dialog box appears. Confirm the class information using the *Class Confirmation* dialog.

4.1.3 Building the User Interface Using DAQBench Controls

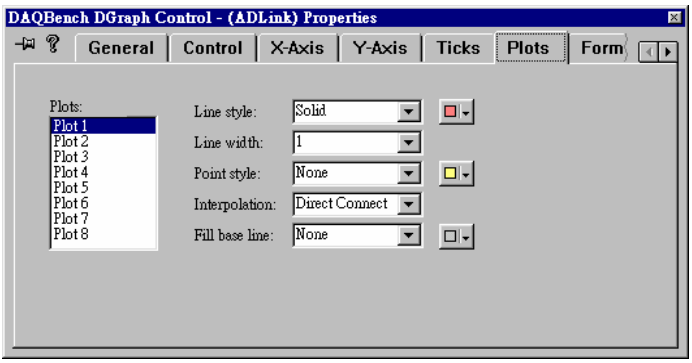
After adding the controls to the project, use the controls in the design of the application user interface. Place the controls on the dialog form using the dialog editor. You can size and move individual controls to customize the interface. Use the custom property pages to set the value of properties.

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.

The fastest way to add controls to a dialog box, reposition existing controls, or move controls from one dialog box to another is to use the drag-and-drop method. The control's position is outlined in a dotted line until it is dropped

into the dialog box. When you add a control to a dialog box with the drag-and-drop method, the control is given a standard height appropriate to that type of control.

Once you add a DAQBench control to a dialog box, you can change its properties by right clicking on the control and selecting **Properties…** to display its custom property pages.



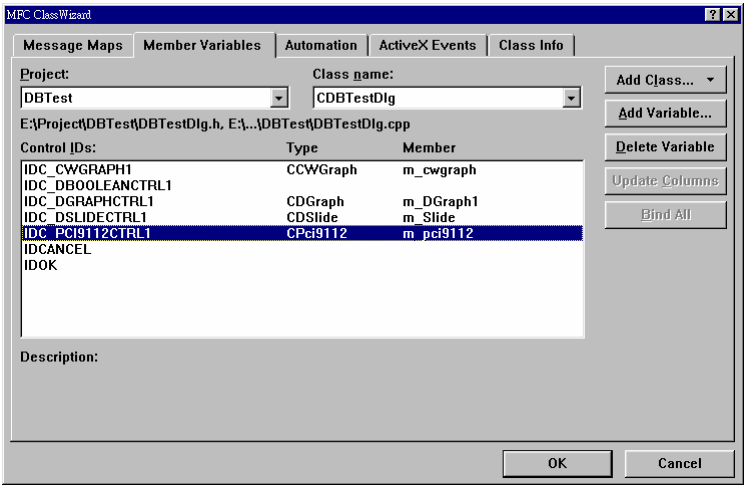
DGraph Control Property Sheets

4.1.4 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, click the **Member Variables** tab.



MFC ClassWizard -- Member Variable Tab

Select the control in the *Control IDs* field to which you want to add a variable. Click **Add Variable...** button. The Add Member Variable dialog box appears. In the *Member variable name* text box, type the name of the variable and click **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.

4.1.5 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 DSlide object, use the `SetValue` function of the wrapper class for the DSlide control. In the source code, the function call is preceded by the member variable name of the control to which it applies.

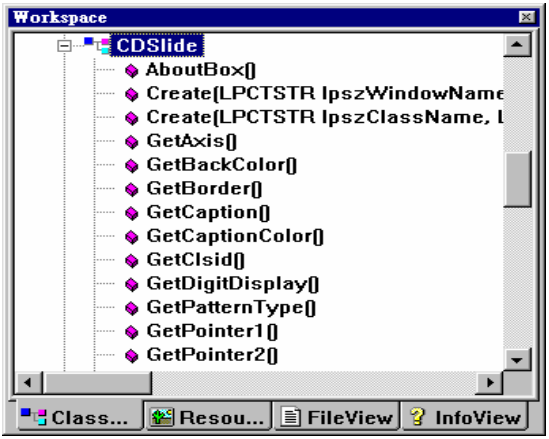
```
m_DSlide.SetValue(COleVariant(5.0));
```

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 wrapping library function. (Please refer to section 4.1.8.)

Use the `GetValue()` function to read the value of a control. For example, pass the value of a DSlide control to a DMeter control.

```
m_DMeter.SetValue(m_DSlide.Pointer1.GetValue()
);
```

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 DSlide object as listed in the Workspace. These are created automatically when you add a control to you project.



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 DSlide control.

```
#include daxis.h
CDAxis  Axis1;
Axis1 = m_DSlide.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_DSlide.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 y-axis on a graph, use the following code.

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

4.1.6 Using Methods

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 arguments. When a method doesn't take arguments, use a pair of empty parentheses.

```
m_DGraph.Refresh();
```

Most methods take some arguments as variants. You must convert any such argument to a variant before passing it to the method. You can use the DAQBench type wrapping library to do so. (Please refer to section 4.1.8) You can convert most scalar values to variants with COleVariant(). For example, the first argument of `PlotGraph` method of the DGraph control is variant type.

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

4.1.7 Using Events

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 control.

Use the following procedure to create an event handler.

1. Right click on a control and select **ClassWizard**.

2. In the *MFC Class Wizard* window, click the **Message Maps** tab and the desired control in the *Object IDs* field. The *Messages* field displays the available events for the selected control.

3. Select the event and click the **Add Function..**. button to add the event handler.

4. To switch directly to the source code for the event handler, click 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 is an example of an event handler generated for the `Change` event of a DKnob. 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
}
```

4.1.8 DAQBench Enhancements 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 controls**

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(tru
e);
```

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 or online help for the details of the enhancement methods.

- **The data type wrapping 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 install dir>\VarPacker 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 install dir>\VarPacker 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 or online help for the details of the data type wrapping functions in **VarPacker.dll.**

5

Building DAQBench Applications with Delphi

This chapter describes how you can use DAQBench controls with Delphi.

At this point you should be familiar with the general structure of ActiveX controls. The individual DAQBench controls are described in the function reference manual and online help.

5.1 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. Click Install Packages... from the Component menu.
2. A list of available packages appears under *Design packages*, 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 **Remove**.

5. Click on Compile to rebuild the package.

6. Close the package editor.

5.2 Developing Delphi Applications

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.

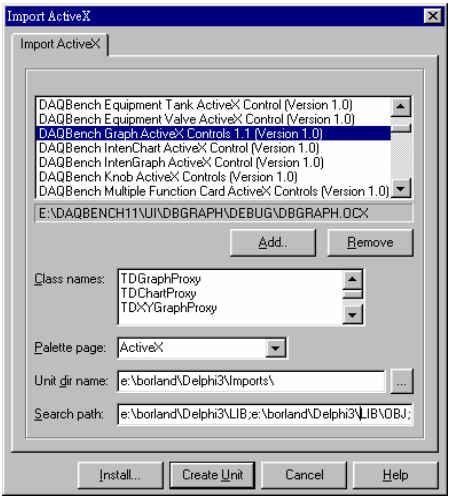
5.2.1 Loading the DAQBench Controls into the Component Palette

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. Choose **Import ActiveX Control...** from the **Component** menu to open the *Import ActiveX Control* dialog box. The dialog box displays a list of currently registered ActiveX controls.



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...**.

4. In the **Install** dialog, click **OK** to add the control to the user package, which makes the control available on the Palette.

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.

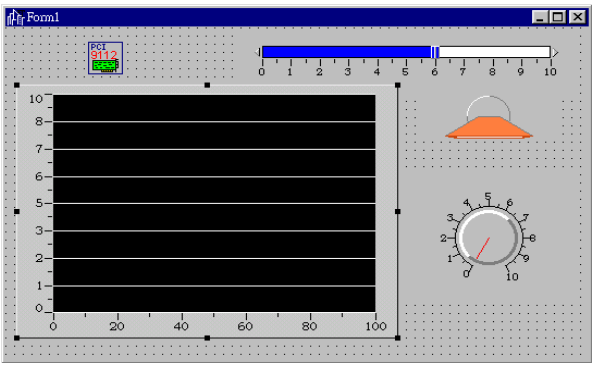
b. From the **Import ActiveX** tab, select the ActiveX control you want to add.

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

5.2.2 Building the User Interface

Placing Controls

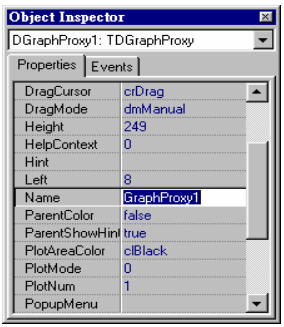
To add a control on the form, select the control on the palette, then clicking on the form where you want to place it. You can also double-click on the control to put it in the middle of the form. Use the mouse to move and resize controls. You can change their default property values by using the Object Inspector and custom property pages.



DAQBench Controls on a Delhi Form

Object Inspector

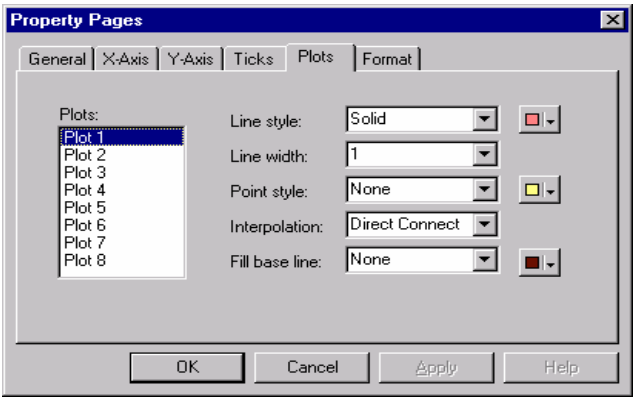
When you select a control on a form, the Object Inspector displays its published properties and allows you to edit it. To open the Object Inspector, select Object Inspector from the View menu or press <F11>. Under the Properties tab of the Object Inspector, you can set properties of the control.



Delphi Object Inspector

Custom Property Pages

DAQBench controls supply the custom property pages for you to easily set the properties. We suggest you to use custom property pages to set the properties except the stock properties. To open the custom property pages, double-click on the control or right click on the control on the form and select **Properties...**. The following figure shows the DAQBench DGraph control property pages.



DAQBench DGraph Control Property Page

5.2.3 Programming with DAQBench

After placing controls on the form, you can use their methods in your code and create event handler to process events generated by the controls at run time.

Setting Properties at Runtime

Any writable property can be set at runtime in your program. To set the value of a property, use the following syntax:

```
object.property := expression;
```

For example, if you want to change the state of a DBoolean control during program execution.

```
DBoolean1.Value := 3;
```

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 DChart control.

```
DChart1.Xaxis.Interval := 10;
```

In the above code, `Interval` is a property of the sub-object `XAxis`.

You can get the value of a property from your program. In most case, to get the value of a property, you use the following syntax:

```
Variable := object.property;
```

For example, you can assign the value of a DBoolean control to a text box on the user interface.

```
Edit1.Text := DBoolean1.Value;
```

Using Methods

Methods are called just like ordinary procedures and functions. To call a method, add the name of the method after the name of the control. When a method doesn't take arguments, you call the method using the following syntax:

```
object.method;
```

For example, the `ClearPlots` method clears the drawing area of a DChart control.

```
DChart1.ClearPlots;
```

Methods can have arguments that you pass to the method. For example, the `PlotGraph` method of the DGraph control has two required arguments -- The array of scaled data to be plotted and the index of plot -- that you must include when you call the method. You must enclose the arguments in parentheses.

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

In most cases, arguments passed to a method are of type variant. Simple scalar values can be automatically converted to variants. 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 handlers in your program to respond to and process events generated by the DAQBench controls. Delphi can generate skeleton event handlers for controls. To create the event handler.

1. Select a control.

2. Click the Events tab in the Object Inspector. The Event page displays all events for the selected control.

3. Select the event you want, then double-click the `Value` column. Delphi generates the event handler in the code editor.

4. Inside the **`begin`**...**`end`** block, type the code that you want to execute when the event occurs.

6

Introducing the DAQBench ActiveX Controls

6.1 User Interface Controls

6.1.1 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

Toggle Switch

Push Button

Switch

Round Button

Slide Switch

6.1.2 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

6.1.3 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

6.1.4 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

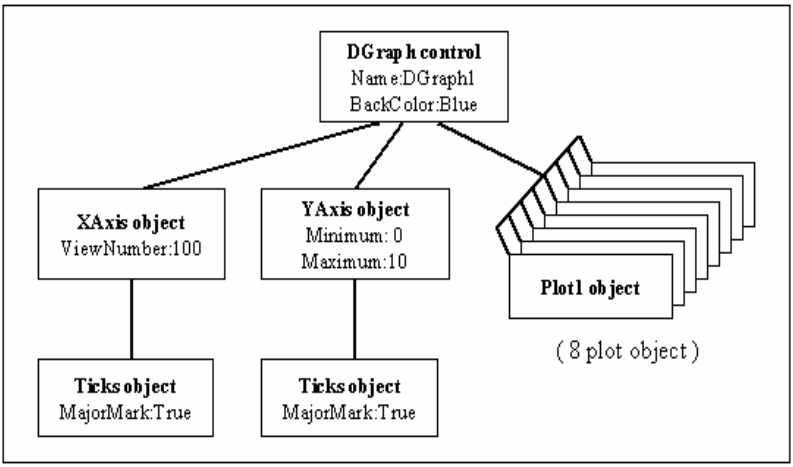
6.1.5 DLEDMeter Control

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

6.1.6 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.

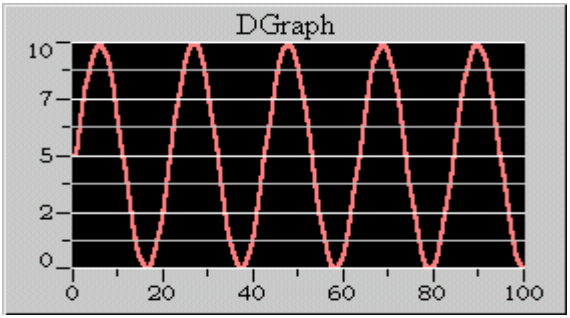


The XAxis object represents the input data points at horizon scale. Users can set the ViewNumber property to specify the DGraph object how many data points will display on plot window. The XAxis object can display the time domain scale when the scale format is “Date” or “Time”. The XAxis object include 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.

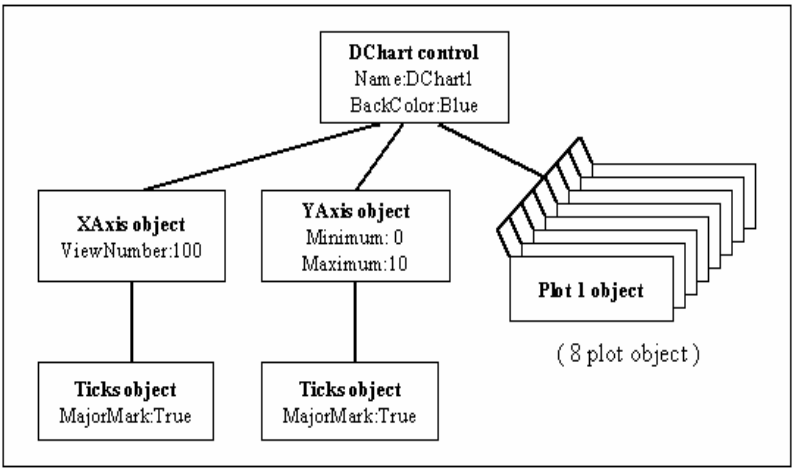
The DGraph object includes eight Plot objects. Users can specify the property of each plot object that include line style, line width, pointer style, fill style, line color, fill color, pointer color, interpolation type.

Example



6.1.7 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.



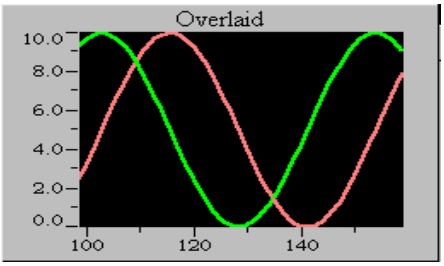
The XAxis object represents the input data points at horizon scale. Users can set the ViewNumber property to specify how many data points will display on plot window. The XAxis object can display the time domain scale when the scale format is “Date” or “Time”. The XAxis object includes one Ticks object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

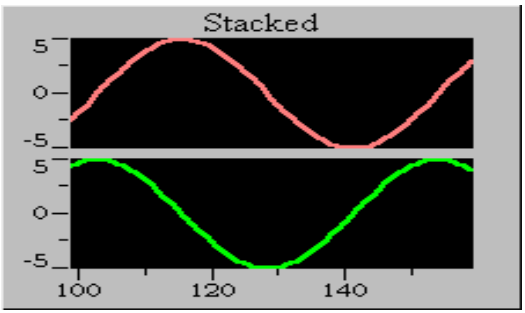
The YAxis object represent the value of data points at vertical scale. Users can set the `Maximum` and `Minimum` property to specify the display range at plot window. The YAxis object has many scale format to display scale label. The YAxis object includes one Ticks object that will process different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

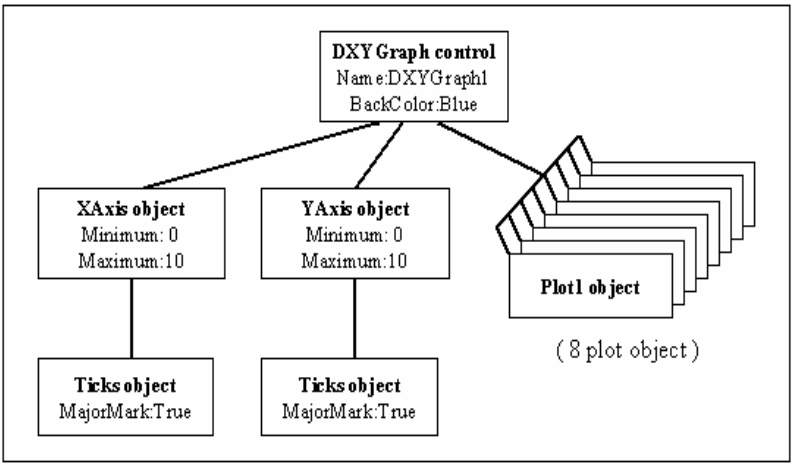
Example





6.1.8 DXYGraph Control

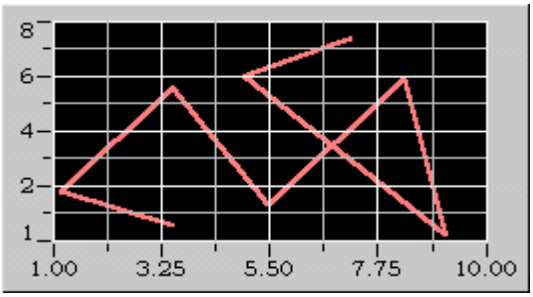
The DXYGraph control is a flexible control used for drawing XY data. It can display multiple plots(maximum eight plots). Plotting XY graph data is drawing the curve of a (x,y) data array. The DXYGraph control is made up of a hierarchy of objects, as illustrated in the following figure.



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.

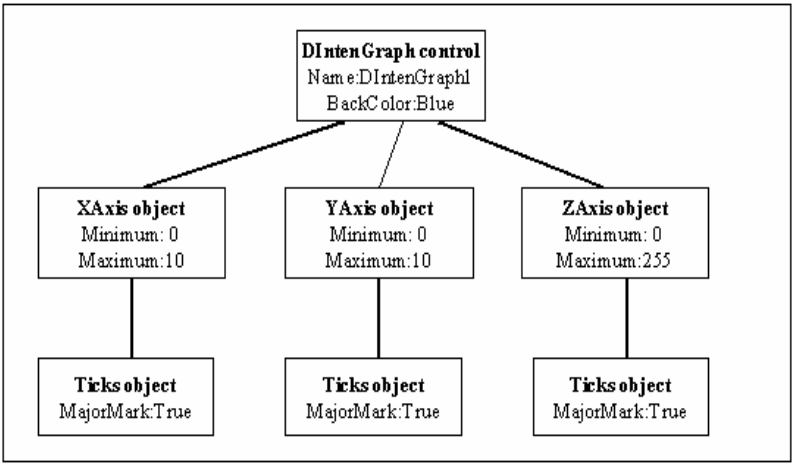
The YAxis object represents the value of data points at vertical scale. Users can set the `Maximum` and `Minimum` property to specify the display range at plot window. The YAxis object has many scale format to display scale label. The YAxis object include one Ticks object that will process different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



6.1.9 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.

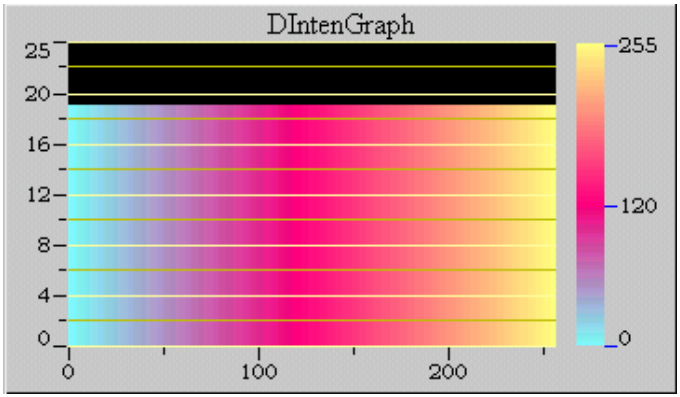


The XAxis object represents the input data points at horizon scale of plane. Users can set the `ViewNumber` property to specify how many data points will display on plot window. The XAxis object can display the time domain scale when the scale format is “Date” or “Time”. The XAxis object includes

one Ticks object that will process different style of ticks color, ticks mark and ticks label.

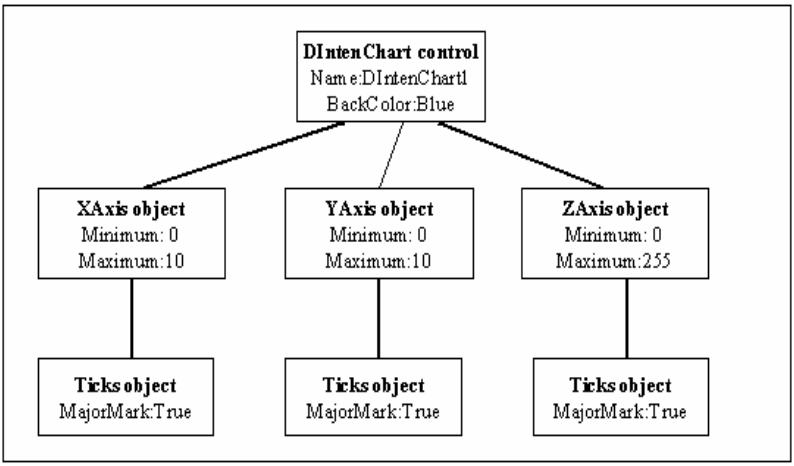
The YAxis object represents the input data points at vertical scale of plane. Users can set the `ViewNumber` property to specify how many data points will display on plot window. The YAxis object has many scale format to display scale label. The YAxis object includes one Ticks object that will process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



6.1.10 DIntenChart Control

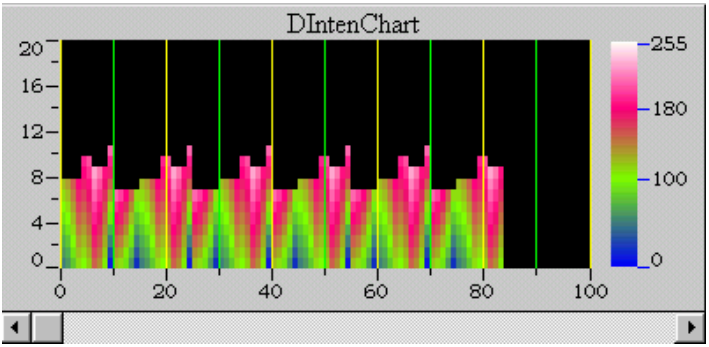
The DIntenChart control is a control used for drawing color intensity on XY plane. It has one ZAxis that represents the color intensity for one point in the XY plane. The ZAxis is a 256 color map. Charting data appends new intensity plane data to plot over time. Charting is used with slow processes where 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.



6.1.11 DDE/NetDDE Function

The User Interface objects (except DGraph, DXYGraph and DIntenGraph objects) now support the DDE (Dynamic Data Exchange) client capability. Therefore they can connect with DDE server applications for exchanging data. You 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 property values for the LinkTopic, LinkItem and LinkMode properties. These properties are used to identify the DDE conversion. User then can use the DDE methods to control the communication between DDE server and User Interface controls.

The DDE property setting example is described below:

```
Control.LinkTopic=Application|topic
(Application_name|topic_name)
Control.LinkItem=item (item_name)
Control.LinkMode=1 (Automatic) or others
```

There are three 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 data do 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 objects.

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).

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

The NetDDE property setting example is described 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)
```

Depending on the Windows 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. 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 appears.

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.

6.2 Information Integration Controls

6.2.1 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.

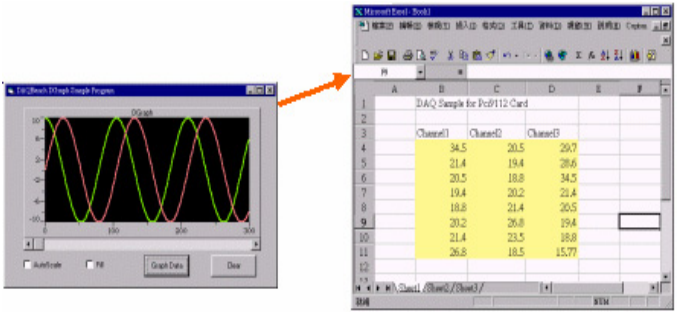
The description of using ExcelLinker control is listed below.

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.

At runtime:

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.


DGraph
Graph Data
Clear
DAQ Sample for Pcl9112 Card
Channel1
Channel2
Channel3
34.5
20.5
29.7
21.4
19.4
28.6
20.5
18.8
34.5
19.4
20.2
21.4
18.8
21.4
20.5
20.2
26.8
19.4
21.4
23.5
18.8
26.8
18.5
15.77

6.2.2 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.

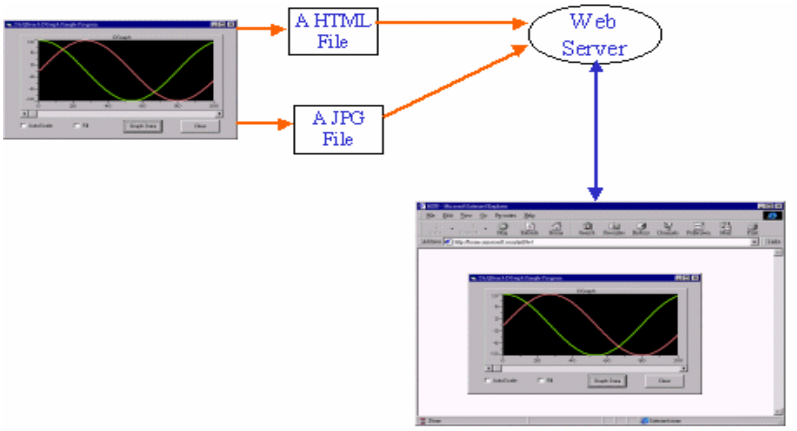
The description of using WebSnapshot control is listed below.

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.

At runtime:

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.


A HTML File
Web Server
A JPG File

6.2.3 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 data 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.

At runtime:

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.

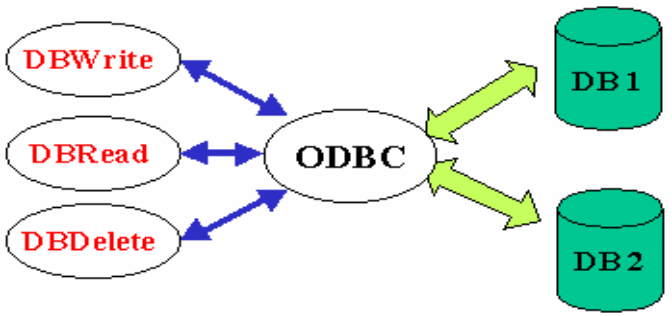
DBDelete control

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.

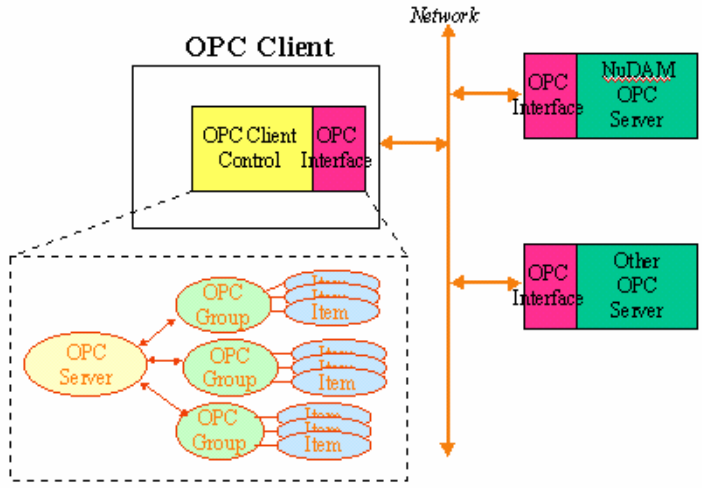
At runtime:

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



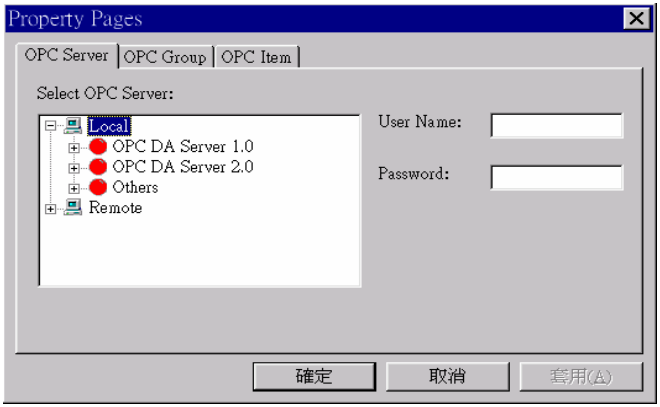
6.2.4 OPCClient Control

The OPCClient2.OCX includes one ActiveX control that can connect, access data and disconnect the OPC Server. The OPC (OLE for Process Control) is established by OPC Foundation. It is a standard interface for accessing process control data in industry automation. The OPC is in client/server model and based on the COM/DCOM technology. Using OPC interface you can easily access control data across Internet and can fulfill the integration between manufacture system and business system. The OPCClient control uses OPC interface to connect, access and disconnect to 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.



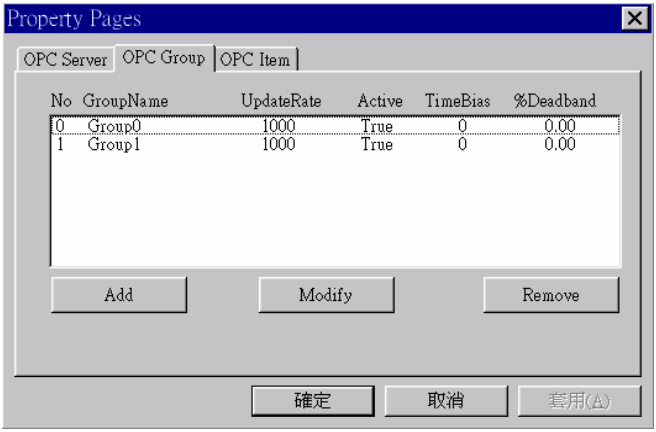
The following steps show how to use OPCClient control.

1. Select the OPC server on local machine or remote machine. If the OPC server is on the remote machine then you have to input the user name and password to log on remote machine.



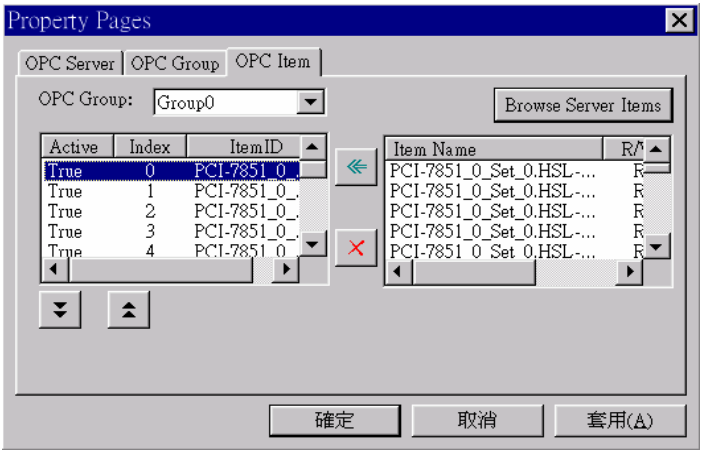
Property page "OPC Server" of OPCClient ActiveX control

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



Property page "OPC Group" of OPCClient ActiveX control

3. Create the OPC Items for OPC Groups.



Property page "OPC Item" of OPCClient ActiveX control

4. At runtime, you must first connect OPC server.

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

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

```
//Read all items in group(0)
OPCClient1.Group(0).ReadItems
Value1 = OPCClient1.Group(0).Item(0).Value
Value2 = OPCClient1.Group(0).Item(2).Value

//Write item(3) in group(0)
OPCClient1.Group(0).Item(1).Value = 5.6
OPCClient1.Group(0).Item(3).Write
```

6. If you don't need to access the OPC server any more, you must disconnect OPC server.

```
OPCClient1.Disconnect
```

6.2.5 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.Seebeck
                                        Temperature(298.3, 0)
Temperature=Thermocouple1.Temperature(34521.11,
                                        Seekback_Temperature,
                                        1)
```

6.3 Analysis Control

With the 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:

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

6.4 SCADA Controls and Utilities

6.4.1 Tag Server and Tag Configuration Utility

The Tag Server is the heart of DAQBench SCADA/HMI function. The Tag Server maintains the defined tags. A tag is a data point that connects to a real-world I/O point through specific OPC server.

A Tag Configuration Utility is provided for you to configure and management tags. You can access the utility from the Windows **Start** menu **Programs>>DAQBench>>Tag Configuration Utility**. With Tag Configuration Utility, you can configure the tag attributes, including if the tag data is logged to database, how the tag scaled, the alarm levels and priorities, etc.

DAQBench User Interface and SCADA controls can access and display the real-time tag data or historical data in database.

6.4.2 Alarm Controls

AlarmDisplay ActiveX control is an UI component that particularly designed for cooperating with ADLINK TagServer to display the alarms happens in the Tag Server. You may pre-define some alarm situations in the Tag Server (for example, an analog tag can have alarms such as LoLo, Lo, Hi, Hi, major deviation, minor deviation, and rate of change). When the alarm(s) occur(s), Tag Server will inform the AlarmDisplay to display the currently happened alarm message(s). Also when the user acknowledges the happened alarm or it returns to the normal state, Tag Server will inform AlarmDisplay again to erase the shown alarm, or display a RTN or ACK message.

The AckButton can acknowledge the alarm(s) that defined in Tag Server. You can use AckButton to acknowledge the alarm of a single tag, alarms in a selected area, or all tags in the Tag Server. After receiving the acknowledge from AckButton, the Tag Server will modify the alarm status and write the change of status into alarm logging file.

6.4.3 Trend Controls

HistTrend ActiveX control is an UI component that particularly designed for retrieving data from the database created by ADLINK Tag Server. You may simply specify the tag names and time interval, and the HistTrend will retrieve data from database and draw them using the specified plot. In addition, the on-screen display feature of HistTrend ActiveX control provides you a simple way to read the data value and timestamp using mouse cursor.

RTTrend ActiveX control is an UI component that particularly designed for cooperating with ADLINK TagServer. RTTrend retrieves data from Tag Server, and plots the data using its value as y-coordinate and its timestamp as x-coordinate. Therefore you can see the “real-time” trend of data. You may also use particular method to plot your own data rather than the data retrieved from Tag Server.

6.4.4 Report Controls

AlarmReport ActiveX control is an UI component that particularly designed for cooperating with the database that created by ADLINK TagServer. Using the AlarmReport control, you can easily retrieve the

alarm information stored in the database. You may simply use mouse click to set the filtering condition such as the time interval, the type of alarms, or the priority of alarms. We also provide some additional methods to let you print the alarm information and save the alarm information to file.

DataReport ActiveX control is an UI component that particularly designed for cooperating with the database that created by ADLINK TagServer. Using the DataReport control, you can easily retrieve the tag data stored in the database. You may simply use mouse click to set the filtering condition such as the time interval and the selected tags. We also provide some additional methods to let you print the tag data and save the tag data to file.

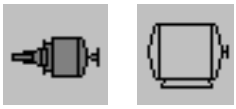
6.4.5 Tag Control

The Tag ActiveX control is a component that particularly designed for cooperating with ADLINK Tag Server. A Tag is a abstract symbol that represents a real data source (i.e. a analog input channel, a digital output channel, etc.), and physically connects to the real data source. Your manipulation to a Tag ActiveX control will reflect to the physical device, hence, you can read/write a value from/to a Tag like you read/write it from/to the physical device. The Tag ActiveX control provides an easy way to manipulate your hardware regardless of what they are or who made them.

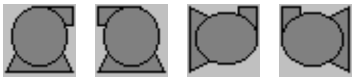
6.4.6 Equipment Controls

The DBEquip.OCX includes five ActiveX controls 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.

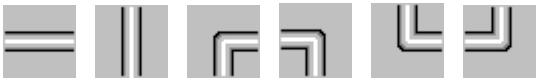
The pattern style of DMoter control



The pattern style of DPump control

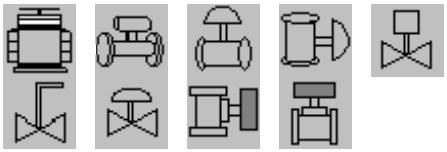


The pattern style of DPump control



The pattern style of DPump control

The pattern style of DPump control



DDE/NetDDE Function

The Equipment objects now support the DDE (Dynamic Data Exchange) client capability.

The connect capabilities are described as below :

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.

Please refer to section 6.1.11 for the details of the usage of DDE conversion.

7

Distribution of Applications

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

e-mail : service@adlinktech.com

e-mail : sw@adlink.com.tw

Warranty Policy

Thank you for choosing ADLINK. To understand your rights and enjoy all the after-sales services we offer, please read the following carefully.

1. Before using ADLINK's products please read the user manual and follow the instructions exactly. When sending in damaged products for repair, please attach an RMA application form which can be downloaded from: http://rma.adlinktech.com/policy/.
2. All ADLINK products come with a limited two-year warranty, one year for products bought in China.
    - The warranty period starts on the day the product is shipped from ADLINK's factory.
    - Peripherals and third-party products not manufactured by ADLINK will be covered by the original manufacturers' warranty.
    - For products containing storage devices (hard drives, flash cards, etc.), please back up your data before sending them for repair. ADLINK is not responsible for any loss of data.
    - Please ensure the use of properly licensed software with our systems. ADLINK does not condone the use of pirated software and will not service systems using such software. ADLINK will not be held legally responsible for products shipped with unlicensed software installed by the user.
    - For general repairs, please do not include peripheral accessories. If peripherals need to be included, be certain to specify which items you sent on the RMA Request & Confirmation Form. ADLINK is not responsible for items not listed on the RMA Request & Confirmation Form.
3. Our repair service is not covered by ADLINK's guarantee in the following situations:
    - Damage caused by not following instructions in the User's Manual.
    - Damage caused by carelessness on the user's part during product transportation.

- Damage caused by fire, earthquakes, floods, lightening, pollution, other acts of God, and/or incorrect usage of voltage transformers.
- Damage caused by inappropriate storage environments such as with high temperatures, high humidity, or volatile chemicals.
- Damage caused by leakage of battery fluid during or after change of batteries by customer/user.
- Damage from improper repair by unauthorized ADLINK technicians.
- Products with altered and/or damaged serial numbers are not entitled to our service.
- This warranty is not transferable or extendible.
- Other categories not protected under our warranty.

4. Customers are responsible for all fees necessary to transport damaged products to ADLINK.

For further questions, please e-mail our FAE staff: service@adlinktech.com