DAQPilot

Task-Oriented DAQ Driver and Wizard

User's Manual

Manual Rev. 2.01

Revision Date: August 30, 2007

Part No: 50-11233-1000

Recycled Paper

Advance Technologies; Automate the World.

Disclaimer

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.

Trademark Information

DAQPilot and **DAQMaster** are registered trademarks of ADLINK Technology Inc.

Product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.

Getting service

Customer satisfaction is our top priority. Contact us should you require any service or assistance.

ADLINK TECHNOLOGY INC.

Web Site	http://www.adlinktech.com
Sales & Service	service@adlinktech.com
Telephone No.	+886-2-8226-5877
Fax No.	+886-2-8226-5717
Mailing Address	9F No. 166 Jian Yi Road, Chungho City, Taipei Hsien 235, Taiwan, ROC

ADLINK TECHNOLOGY AMERICA, INC.

Sales & Service	info@adlinktech.com
Toll-Free	+1-866-4-ADLINK (235465)
Fax No.	+1-949-727-2099
Mailing Address	8900 Research Drive, Irvine, CA 92618, USA

ADLINK TECHNOLOGY EUROPEAN SALES OFFICE

Sales & Service	emea@adlinktech.com
Toll-Free	+49-211-4955552
Fax No.	+49-211-4955557
Mailing Address	Nord Carree 3, 40477 Düsseldorf, Germany

ADLINK TECHNOLOGY SINGAPORE PTE LTD.

Sales & Service	singapore@adlinktech.com
Telephone No.	+65-6844-2261
Fax No.	+65-6844-2263
Mailing Address	84 Genting Lane #07-02A, Cityneon Design Center, Singapore 349584

ADLINK TECHNOLOGY SINGAPORE PTE LTD.
(INDIA Liaison Office)

Sales & Service	india@adlinktech.com
Telephone No.	+91-80-57605817
Fax No.	+91-80-26671806
Mailing Address	No. 1357, Ground Floor, "Anupama", Aurobindo Marg JP Nagar (Ph-1) Bangalore - 560078

ADLINK TECHNOLOGY INC. (KOREA Liaison Office)

Sales & Service	korea@adlinktech.com
Telephone No.	+82-2-20570565
Fax No.	+82-2-20570563
Mailing Address	4F, Kostech Building, 262-2, Yangjae-Dong, Seocho-Gu, Seoul, 137-130, South Korea

ADLINK TECHNOLOGY (BEIJING) CO., LTD.

Sales & Service	market@adlinkchina.com.cn
Telephone No.	+86-10-5885-8666
Fax No.	+86-10-5885-8625
Mailing Address	Room 801, Building E, Yingchuangdongli Plaza, No.1 Shangdidonglu, Haidian District, Beijing, China

ADLINK TECHNOLOGY (SHANGHAI) CO., LTD.

Sales & Service	market@adlinkchina.com.cn
Telephone No.	+86-21-6495-5210
Fax No.	+86-21-5450-0414
Mailing Address	Floor 4, Bldg. 39, Caoheting Science and Technology Park, No.333 Qinjiang Road, Shanghai, China

ADLINK TECHNOLOGY (SHENZEN) CO., LTD.

Sales & Service	market@adlinkchina.com.cn
Telephone No.	+86-755-2643-4858
Fax No.	+86-755-2664-6353
Mailing Address	C Block, 2nd Floor, Building A1, Cyber-tech Zone, Gaoxin Ave. 7.S, High-tech Industrial Park S., Nanshan District, Shenzhen, Guangdong Province, China

Using this manual

Audience and scope

This manual guides you when using the task-oriented DAQPilot development application. This manual also describes how to install and use the DAQPilot when creating programs for your software applications.

How this manual is organized

This manual is organized as follows:

Chapter 1 Introduction: This chapter introduces the DAQPilot application including it main features, highlights, and supported tasks and devices.

Chapter 2 Installation: This chapter provides information on DAQPilot system requirements, installation, and user interface.

Chapter 3 Creating DAQ Tasks: The chapter describes the procedure on how to create a customized DAQ task using the DAQPilot wizard and task manager.

Chapter 4 Programming with DAQPilot: The chapter describes the integration of DAQPilot with mainstream programming languages to streamline the development of DAQ applications.

Chapter 5 APIs Function Reference: The chapter lists the DAQPilot Application Programming Interfaces (APIs) for advanced programming.

Chapter 6 ActiveX Controls and .NET Components Function Reference: The chapter lists the DAQPilot ActiveX Controls and .NET components for advanced programming.

Conventions

Take note of the following conventions used throughout the manual to make sure that you perform certain tasks and instructions properly.

NOTE	Additional information, aids, and tips that help you perform particular tasks.
IMPORTANT	Critical information and instructions that you MUST perform to complete a task.
WARNING	Information that prevents physical injury, data loss, module damage, program corruption etc. when trying to complete a particular task.

Table of Contents

1 Introduction

Thank you for choosing DAQPilot. This task-oriented DAQ driver comes with an intuitive wizard to assist you in developing cutting-edge DAQ applications in an instant. DAQPilot also functions as software development kit (SDK) to assist programmers in developing DAQ applications in various ADE environments including Micosoft® Visual Basic .NET®, Micosoft® Visual C# .NET, Microsoft® Visual Basic®, Microsoft® Visual C++®, Borland Delphi®, and Borland C++ Builder®.

DAQPilot provides easy creation of DAQ task specification, and offers powerful task management, reference code generation including ActiveX control generation, and instant test panel.

With DAQPilot, you don't need to deal with complex functions and parameters so you focus more on the task operations. An intuitive wizard guides you when setting all necessary parameters for the specified task with automatic checking mechanism. After defining a task, you may simply load the task specification to your program, then execute. DAQPilot accelerates your DAQ development 300 times faster than traditional DAQ programming.

When you want to create a DAQ application quickly, the DAQPilot wizard guides you in finishing a DAQ task in just about 45 seconds for instant testing. In addition, if you need to effectively control several DAQ devices, DAQPilot provides universal APIs to help you develop your DAQ application. You may create your DAQ application in minutes with DAQPilot API, ActiveX, or .NET Assembly using a task-oriented approach. Equipped with intuitive built-in components including an instant test panel and automatic reference code generation, DAQPilot lets you land quickly in a variety of DAQ programming environments.

1.1 Features

- Easy-to-use, task-oriented DAQ driver with wizard
- Supports mainstream and new generation OS, including:
  - 32-bit Editions of Windows 98/NT/2000/XP/Server 2003/ Vista
  - 64-bit Editions of Windows XP/Server 2003/Vista
- Supports various application development environments, including:
  - Microsoft Visual C# .NET
  - Microsoft Visual Basic .NET
  - Microsoft Visual Basic
  - Microsoft Visual C++
  - Borland Delphi
  - Borland C++ Builder
- Supports a complete line of ADLINK DAQ cards and is compatible with all ADLINK DAQ hardware functions
- Windows-based utility easily configures and diagnoses the DAQ hardware
- Bundled example programs speed up DAQ programming
- Comprehensive I/O functions including analog input, analog output, digital input/output, timer/counter, and event
- Offers three programming methods:
  - DAQPilot API
  - DAQPilot ActiveX control
  - DAQPilot .NET assembly

1.2 Highlights

Rapid DAQ development

In a traditional way, users must call a series of driver API to control different types of DAQ cards. This is followed by a user-specified algorithm to process the data.

In a task-oriented approach, the task represents the measurement process you want to perform. Based on the DAQ task, you write the code not only for a specific device but also for other devices that will use the same task.

Task manager

The DAQPilot's Task Manager is a powerful tool that creates and manages your DAQ tasks. The DAQPilot wizard may be launched from the task manager for convenient step-by-step DAQ task development. With the task manager, you may visually create, modify, rename, and delete DAQ tasks, and review all related information. The wizard automatically recognizes the hardware, performs hardware classification according to supported tasks, revises the related parameters according to various task conditions, and revises incorrect parameters.

For your convenience, the DAQPilot Task Manager is integrated with the ADLINK DAQMaster to form a unified and an integrated interface for more efficient and faster DAQ programming.

Reference code generation

Through the DAQPilot Task Manager, you may generate the reference code of the selected task for further programming.

Instant test panel

The DAQPilot Task Manager also enables you to instantly test the DAQ task parameters to ensure if these are correct.

DAQPilot Driver Engine

The DAQPilot driver and SDK is a data acquisition DLL library and the device driver for DAQ application development. The driver provides a common set of APIs to control data acquisition cards. DAQPilot's task-oriented concept allows the use of the same code to control various devices performing the same DAQ task. The DAQPilot runtime engine also loads and executes the DAQ task generated by task manager, and performs pre-defined properties for cost-effective DAQ card operations.

New-generation ActiveX control

In addition to DAQPilot APIs, DAQPilot also comes with ActiveX control for developing component-based DAQ applications. The new generation ActiveX control is embedded in the DAQPilot wizard to provide you with a more intuitive interface when setting the DAQ properties and easily configure DAQ devices better than the traditional way. Equipped with four functions — read, write, start, and stop — you may execute any type of DAQ task. Since the ActiveX component technology is popular with many programming languages, it is highly-recommended that you use the DAQPilot ActiveX controls when developing applications in main development environments such as Visual Basic.

High-performance runtime engine

The DAQPilot runtime engine — an invisible component of the DAQPilot — efficiently executes the DAQ task specification. When your program executes a task specification, the runtime engine parses the settings and performs the corresponding hardware operations. The runtime engine guarantees execution of all kinds of data acquisition operations for ADLINK DAQ products.

1.3 Supported Tasks

DAQPilot supports four major task categories and a timer/counter function. These categories and function come with 19 tasks. Refer to the list below.

Class/Function	Icon	Category	
Analog Input		Polling	The single-channel or multi-channel voltage polling acquires the analog input signal from selected channels
		Single-shot Waveform Acquisition	Acquires the analog input signal from selected channels in a period.
		Continuous Waveform Acquisition	Acquires the analog input signal continuously from selected channels.
Analog Output		Voltage Output	Outputs a specific voltage to a selected single or multi-channel configuration.
		Current Output	Outputs a specific current to selected channels in single-channel or multi-channel configuration.
		Single-shot Waveform Generation	Generates analog waveforms to selected channels in a period.
		Continuous Waveform Generation	Generates analog waveforms continuously to selected channels.
		Function Generation	Generates common waveforms such as sine, square, triangle, and sawtooth to selected channels in a period.

Digital Input		Line Input	Acquires a digital line value from a selected line.
		Port Input	Acquires a digital port value from a selected port.
		Single-shot Pattern Acquisition	Acquires a single-shot digital pattern from a selected port.
		Continuous Pattern Acquisition	Acquires continuous digital pattern from a selected port.
Digital Output		Line Output	Outputs a line value to a selected digital line value.
		Port Output	Outputs a port value to a selected port.
		Single-shot Pattern Generation	Outputs a predefined single-shot digital pattern to a selected port.
		Continuous Pattern Generation	Outputs a predefined pattern continuously to a selected port.
Timer/Counter		Simple Counter	Simple event counter
		Timer Interrupt	Generates timer interrupts periodically.
		Mode Operation	Provides mode operations.

1.4 Supported Cards

DAQPilot supports the following ADLINK DAQ cards:

- NuDAQ Series

6208, 6216, 6308, 7200, 7224, 7230, 7233, 7234, 7248, 7250, 7256, 7258, 7260, 7296, 7300, 7348, 7396, 7432, 7433, 7434, 7442, 7443, 7444, 7452, 8554, 9111, 9112, 9113, 9114, 9118, 9221, 9810, 9812, 9820

- DAQ-2000 Series

2005, 2006, 2010, 2016, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

1.5 Where to Get

DAQPilot is available from the DAQPilot Installation CD. You may also download a copy from the ADLINK Test and Measurement website at http://www.adlinktech.com/TM.

2 Installation

This chapter provides information on DAQPilot system requirements, installation, and user interface.

2.1 Before You Proceed

System Requirements

Make sure your system meets the following requirements before you install DAQPilot.

- Windows 98/NT/2000 or 32-/64-bit editions of Windows XP/Server 2003/Vista operating system
- PC with Intel Pentium-class CPU or higher
- VGA display or higher
- Minimum 64 MB of memory
- Minimum 40 MB of free hard disk space
- Mouse

2.2 Installing DAQPilot

This section provides instructions on how to install DAQPilot in your system.

To install DAQPilot:

1. Place the DAQPilot Installation CD to the computer's optical drive, or double-click on the DAQPilot setup file you downloaded from the ADLINK website.
2. When the installation window appears, click on the **Install DAQPilot** button.

NOTE If Autorun is not enabled in your computer, explore the CD, then double-click on the SETUP.EXE to display the installation window.

2.3 Checking the DAQ Card Drivers

To check if the DAQ card is properly installed and detected by the system:

1. Launch the Windows Device Manager.
2. Expand the **NuDAQ Boards** item, then double-click on the listed DAQ device(s).
3. Click the **Resources** tab and check if the device I/O port and IRQ resources are allocated correctly.

NOTE The necessary DASK drivers are automatically installed during the DAQPilot installation. These libraries hold the PCIS-DASK, D2K-DASK, and WD-DASK system files. For more information on these DASK drivers, install the corresponding software packages to view the user's manual.

2.4 Getting to Know DAQPilot

DAQPilot Task Manager

The DAQPilot Task Manager window is divided into two panes: task (left), and task information (right). The task manager is embedded in the ADLINK DAQMaster. The DAQPilot task manager is integrated with the ADLINK DAQMaster to enable efficient and easy management of DAQ tasks and robust configuration and control of DAQ devices using a single interface.

Navigating through the task manager

If you choose to install predefined DAQ task samples during the DAQPilot installation, these tasks appear on the left pane of the DAQPilot Task Manager window. Otherwise, the left pane is empty during initial launch.

Nineteen predefined DAQ tasks samples appear in the task pane. You may use or modify these tasks depending on your specifications. When you select a task, the corresponding task information appears on the right pane.

You can develop a DAQ application based on these predefined tasks. DAQPilot also features a virtual device that allows you to test these predefined DAQ tasks. With virtual device, you can test these functions even before you install the DAQ device.

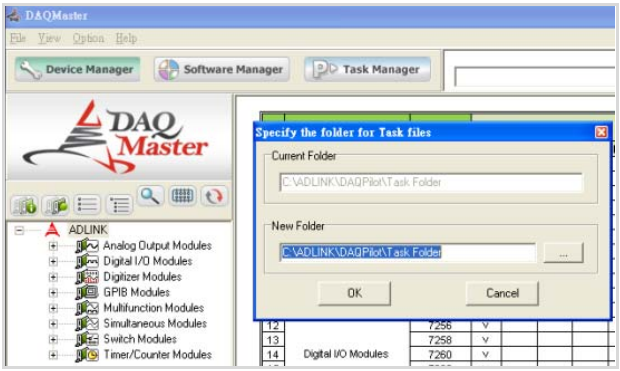
Defining and saving a task file

When you create a DAQ task, all corresponding settings and parameters are recorded in a task file. All task files are stored in this default path and folder:

```
C:\ADLINK\DAQPilot\Task Folder\
```

From DAQMaster, you may change the path/folder for created task files using the DAQPilot task manager. To do this:

1. Click **Option** > **DAQ Task Folder**. A window appears prompting you to specify the path/folder.
2. Click ..., then point to the new path/folder where you want to save all task files.

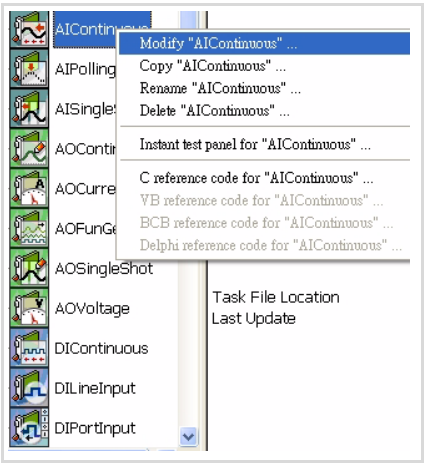


3. Click **OK** when finished.

Task files are identified with a filename that corresponds with the selected task in the DAQPilot task manager window.

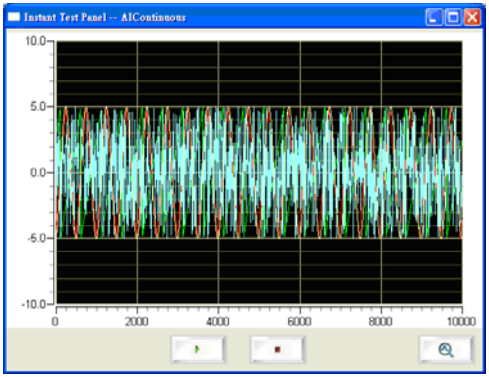
Customizing pre-defined DAQ tasks

You may customize predefined DAQ tasks using the DAQPilot Task Manager. Right-click a task from the left pane, then select the operation you want to perform from the popup menu. You may modify, copy, rename, or delete the selected task. The corresponding task file is automatically updated when you change a DAQ task.



Using the instant test panel

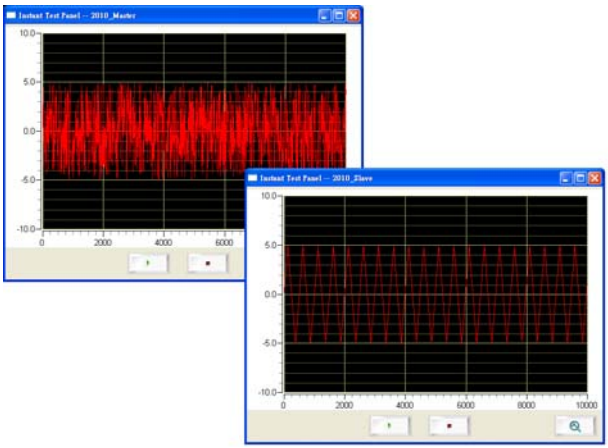
After customizing a DAQ task, right-click the task from the left pane, then select **Instant test panel** from the popup menu. A test panel window appears. Click  to begin testing the task. To stop testing, click  . You may also click  to change the instant test panel window settings.



If you are planning to use two ADLINK DAQ cards with SSI function, the DAQPilot task manager allows you to simultaneously generate two different task specifications and two separate instant test panels to show you the task test result while designing the application.

To display two instant test panels for SSI functionality:

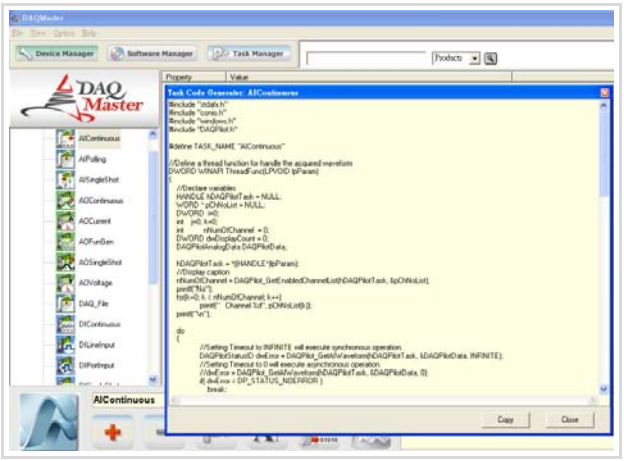
1. Create a DAQ task for each card.
2. Set the SSI parameters for both DAQ tasks.
3. Launch the instant test panels of the DAQ tasks to test.



Generating the reference code

After testing the task using the instant test panel, you may now generate the reference code for further programming. To do this:

1. Right-click the task from the task pane.
2. A window appears with the C reference code.



3. Copy the reference code, then compile and run it in your programming application.

3 Creating DAQ Tasks

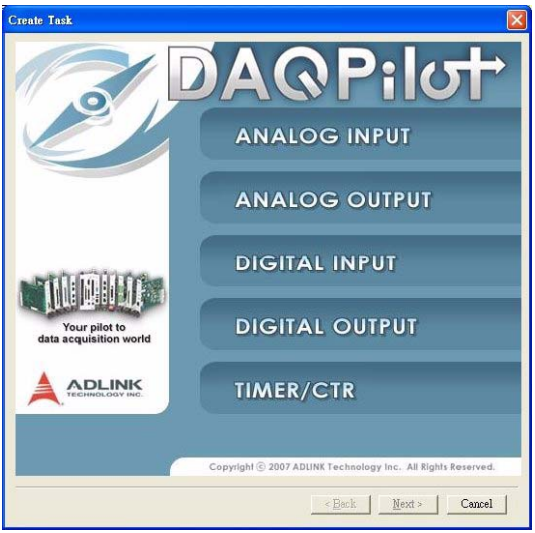
The chapter describes the procedure on how to create a customized DAQ task using the DAQPilot wizard. Creating a DAQ task involves four basic steps:

- Specifying the task
- Setting the parameters
- Testing the task
- Generating the code

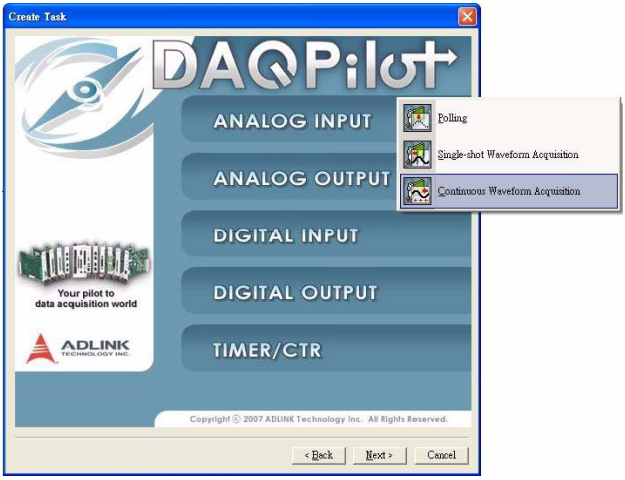
3.1 Specifying the Task

The DAQPilot wizard uses step-by-step instructions to guide you when creating a new task. To launch the wizard from the DAQPilot task manager, right-click a DAQ task from the task pane, then select **Create a new task**.

The Create Task wizard appears.



Right-click on a task class/function, then select the task category from the pop-up menu.



3.2 Setting the Parameters

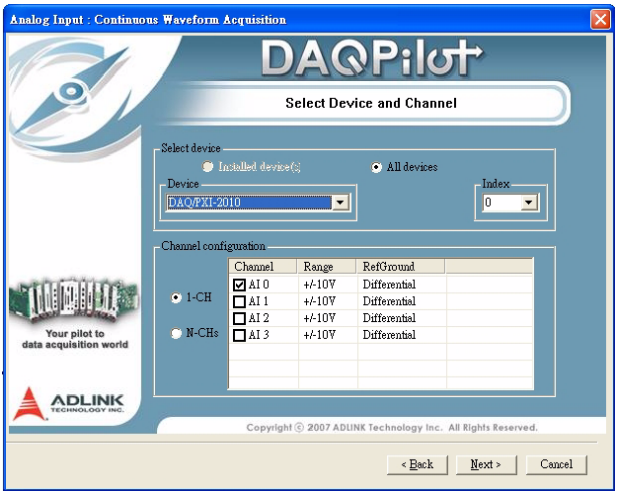
After specifying the DAQ task class, the wizard guides you in setting the task parameters.

Task Name and Description

When prompted assign a name and a description for the new DAQ task. Click **Next** when finished.

Device and Channels

After you have assigned a name and a description for the task, select the device and configure the channels from this window.



The wizard allows you to select which device to use and define the corresponding channel settings. This window also enables you to easily set the Range and RefGround by selecting a supported value from the drop-down combo box.

Click **Next** when finished.

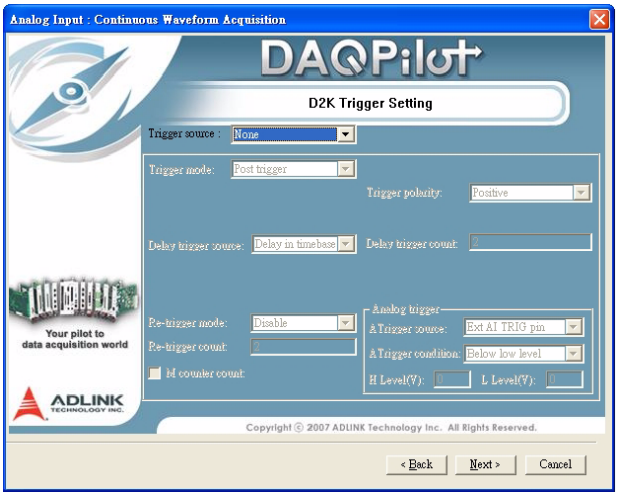
Acquisition Parameters

The next window enables you to set the acquisition parameters for the task. This includes the setting for the clock source, sampling rate, etc.

Click **Next** when finished.

Setting the Trigger

You may define the trigger condition and set the trigger source through this window. Some trigger conditions are automatically adjusted depending on the value of the trigger source.

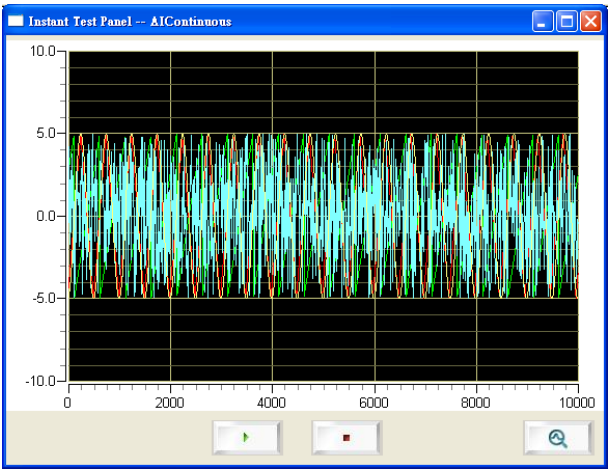


Click **Next** when finished.

3.3 Testing the Task

After setting the task parameters, you may use the built-in test panel in the wizard to know if the task parameters are in range and if the task works properly.

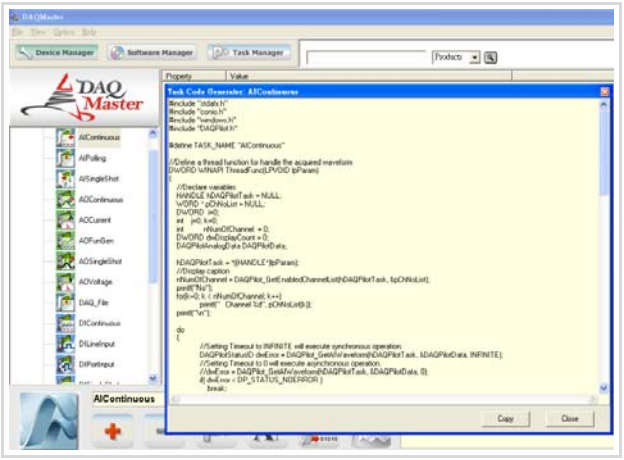
Click  to begin testing the task. To stop testing, click  . You may also click  to change the instant test panel window settings.



After task verification, click **Finished** to close the wizard and return to the DAQPilot task manager.

3.4 Generating the Code

The name of the created task now appears in the list of tasks in the task pane. Right-click on the task to generate the C code.



Another window opens with the generated code. Copy the code, then compile and run it in your programming application.

4 Programming with DAQPilot

The chapter describes the integration of DAQPilot with mainstream programming languages to streamline the development of DAQ applications.

4.1 .NET Component Example for C# .NET

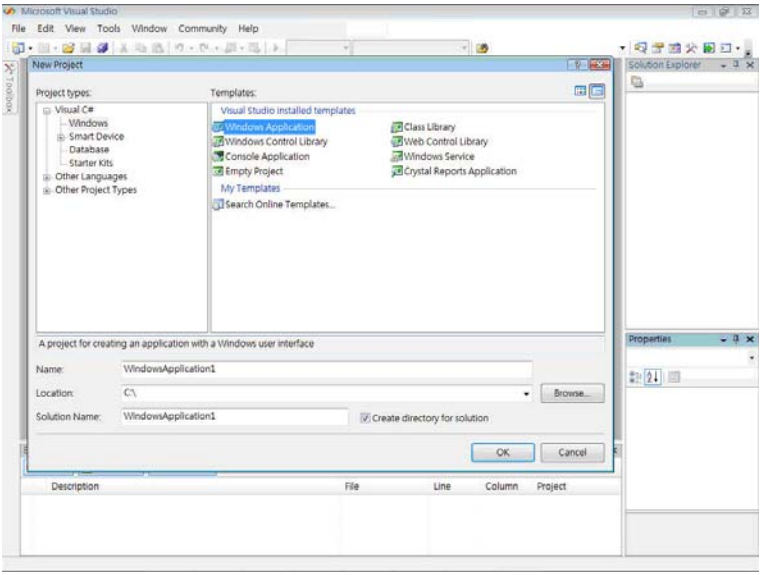
This section provides instructions on how to create a DAQ application using DAQPilot's .NET Assembly.

The process involves seven steps:

- Creating the Visual Studio C# DAQ project
- Adding DAQPilot component to the project's toolbox
- Setting the properties at design time
- Editing the properties at runtime
- Working with control methods
- Developing event handlers
- Running the application

Creating the Visual Studio C# DAQ Project

Use the **New Project** dialog to initialize a new project in Visual Studio development environment. Click **New** > **Project** from the the **File** menu, select **Windows Application**, then press **OK**. Refer to the illustration below.

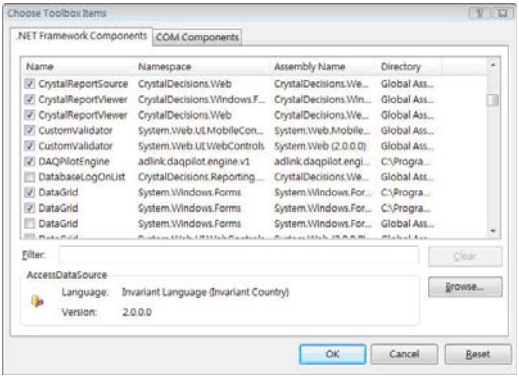


Adding DAQPilot Component to the Project's Toolbox

You need to install the DAQPilot's .NET component to Visual Studio C# before building an application. After installation, you may use the integrated DAQPilot component to conveniently carry out all task functions.

To add the DAQPilot component to the project's toolbox:

1. From a new Visual Studio C# project, right-click on the **Toolbox** menu, then select **Choose Items**. The **Choose Toolbox Items** window appears.



2. Click on **Browse** to add the **DaqPilotEngine** file from this path:

   ```
   Drive\Program Files\Common Files\ADLINK\DAQPilot\
         adlink.daqpilot.engine.v1.dll
   ```

3. Locate the **DAQPilotEngine** component from the list, then click on the checkbox before the component name to select.

4. Select all related components you want to add in your project by clicking on the check boxes.

5. When finished, click **OK** to close the window. All selected components now appear in the toolbox.

Setting Properties at Design Time

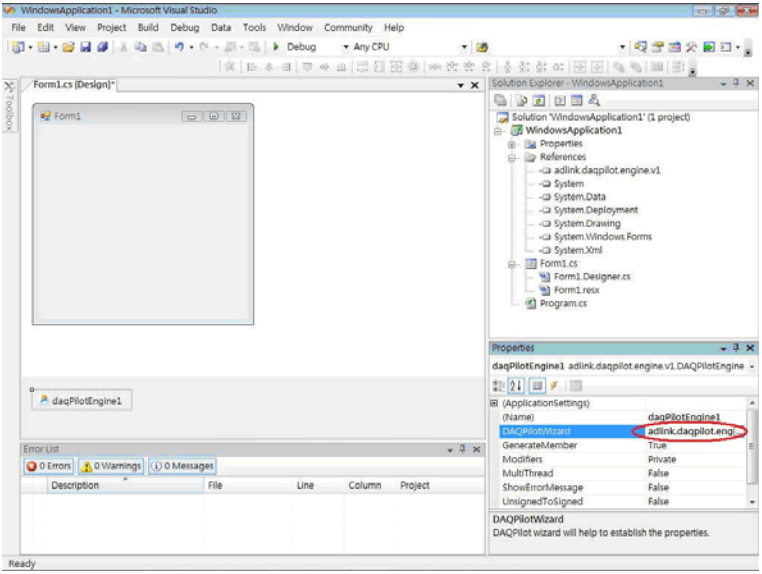
After placing a component on a Visual Studio C# form, you can configure the component by setting its properties with the DAQPilot custom control property pages. Refer to the figure below.

Each component has its own set of default properties such as component name.

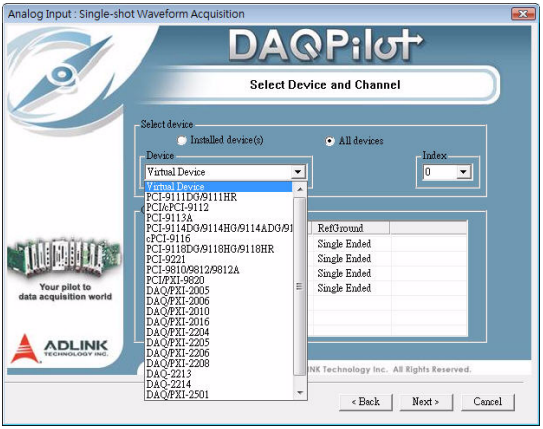
Do any of the following to open the Properties window:

- Select **Properties Window** from the **View** menu, then click the **Properties Window** button from the toolbar.
- Directly launch the **Properties Windows** from the popup menu.

The DAQPilot .NET component provides a custom Property Pages, that is the DAQPilot wizard, to replace the traditional ActiveX Control Property Page and to assist you easily in setting up the properties. You may select the **Property** option from the pop-up menu to launch the DAQPilot wizard.



When the DAQ Pilot wizard appears, set the DAQ parameters. In this example, we would like to perform a single-shot waveform acquisition with a virtual device.



Refer to **Chapter 3: Creating DAQ Tasks** for more information on the DAQPilot wizard parameters and supported tasks.

Edit Properties at Runtime

You can dynamically set and read the properties of the component in Visual Studio C#, by using this syntax to set the property.

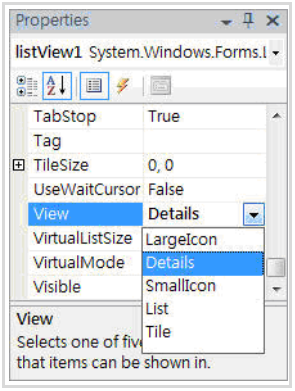
```
object.SetProperty property string, expression
```

For example, if you want to change the values during program execution:

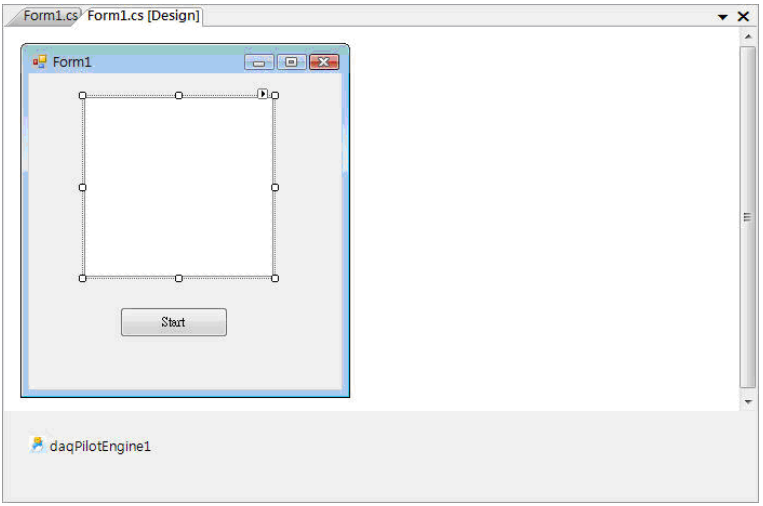
```
//Enable Ch1 by method
daqPilotEngine1.SetChannelProperty(1, "Enable",
   True);

//Without add DAQPilot.cs file
daqPilotEngine1.SetChannelProperty(1, "Range",
   203); // DPV_RANGE_B_2_5_V

//Add DAQPilot.cs file
daqPilotEngine1.SetChannelProperty(1, "Range",
   daqpilotengine.id.DAQPilotValueID.
   DPV_RANGE_B_2_5_V);

//Change number of scan property to 100 by
   method
daqPilotEngine1.SetDProperty("Number of
   scan(s)", 100);

daqPilotEngine1.SetDProperty("Sampling rate per
   channel", 1000);

object vDaqpilotValue;
vDaqpilotValue =
   daqPilotEngine1.GetChannelProperty(1,"Enable"
   );
vDaqpilotValue =
   daqPilotEngine1.GetChannelProperty(1,"Range")
   ;
vDaqpilotValue =
   daqPilotEngine1.GetDProperty("Number of
   scan(s)");
```

Working with Control Methods

Place a ListView .NET component on Form1 with property value **Details** as its View property. Refer to the figure below.



Place a Button .NET component tagged as **Start**. Refer to the figure below.



To call a method, attach the method name to the component. If the method does not need any parameters, you can directly call the method with this syntax.

```
object.method()
```

For example, the **Start** method can launch the task function which is predefined in the component.

```
DAQPilotEngine1.Start();
```

Developing Event Handlers

After configuring the components, you may continue to create event handlers in your program to hook appropriate events generated on the components.

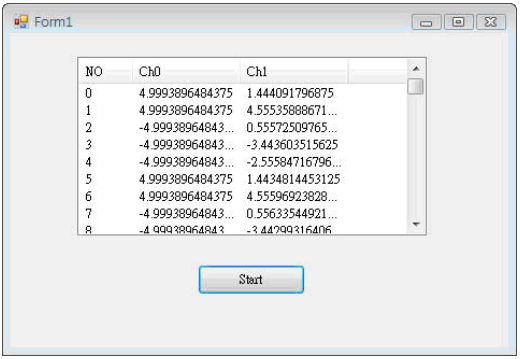
For example, the DAQPilot component has a DataArrival event that is triggered if the hardware DMA is finished. In addition, to develop the event routine code, most programming environments generate a skeleton function to be the code template.

Below is the code slice for processing the event of DMA data arrival based on the skeleton from Visual Studio.

```
private void daqPilotEngine1_DataArrival(object
   sender,
   adlink.daqpilot.engine.v1.DAQPilotEngine.DAQP
   ilotDataEventArgs e)
        {
            int i, j;
            long NumOfScan;
            string strCaption, strValue;
            double[] data = e.Data as double[];

            //Initial column header
            listView1.Columns.Clear();
            listView1.Columns.Add("NO", 100);
            for (i = 0; i < e.ChNumList.Length;
   i++)
            {
                strCaption = "Ch" +
   Convert.ToString(e.ChNumList[i]);
              listView1.Columns.Add(strCaption,
   200);
            }
            NumOfScan = data.Length/
   e.NumberOfChannel ;
//Dump the data
             listView1.Items.Clear();
            for (i = 0; i < NumOfScan ; i++)
            {
                strValue = Convert.ToString(i);
                listView1.Items.Add(strValue);
              for (j = 0; j < e.NumberOfChannel;
   j++)
                {
                    strValue = data[(i *
   e.NumberOfChannel) + j].ToString();

   listView1.Items[i].SubItems.Add(strValue);
                }
            }
         }
```

Running the Application

Press the **Start** button in your sample application to run and display the data.



Source Codes

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using daqpilotengine.id;
     namespace WindowsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void button1_Click(object
  sender, EventArgs e)
        {
```

```
daqPilotEngine1.SetChannelProperty(1, "Enable",
  true);
         daqPilotEngine1.SetChannelProperty(1,
  "Range", 203);
         daqPilotEngine1.SetChannelProperty(1,
  "Range",
  daqpilotengine.id.DAQPilotValueID.DPV_RANGE_B
  _2_5_V);
          daqPilotEngine1.SetDProperty("Number
  of scan(s)", 100);

  daqPilotEngine1.SetDProperty("Sampling rate
  per channel", 10000);
           daqPilotEngine1.Start();

       }
      private void
  daqPilotEngine1_DataArrival(object sender,
  adlink.daqpilot.engine.v1.DAQPilotEngine.DAQP
  ilotDataEventArgs e)
       {
           int i, j;
           long NumOfScan;
           string strCaption, strValue;
           double[] data = e.Data as double[];

           //Initial column header
           listView1.Columns.Clear();
           listView1.Columns.Add("NO", 100);
           for (i = 0; i < e.ChNumList.Length;
  i++)
           {
               strCaption = "Ch" +
  Convert.ToString(e.ChNumList[i]);
             listView1.Columns.Add(strCaption,
  200);
           }
           NumOfScan = data.Length/
```

```
e.NumberOfChannel ;
            //Dump the data
             listView1.Items.Clear();
            for (i = 0; i < NumOfScan ; i++)
            {
                strValue = Convert.ToString(i);
                listView1.Items.Add(strValue);
              for (j = 0; j < e.NumberOfChannel;
   j++)
                {
                    strValue = data[(i *
   e.NumberOfChannel) + j].ToString();

   listView1.Items[i].SubItems.Add(strValue);
                }
            }
         }
        private void Form1_FormClosing(object
   sender, FormClosingEventArgs e)
        {
            daqPilotEngine1.Stop();
        }
    }
}
```

4.2 .NET Component Example for Microsoft Visual Basic .NET

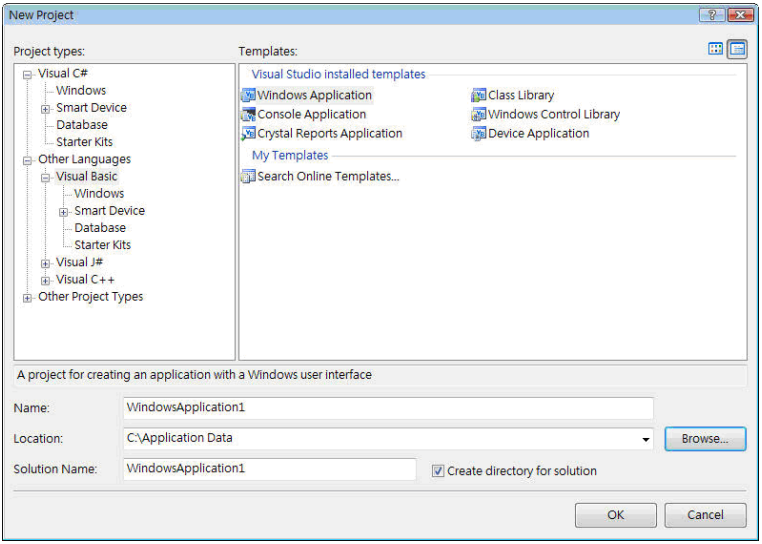
This section provides instructions on how to create a DAQ application using DAQPilot's .NET component.

The process involves seven steps:

- Creating the Visual Studio VB.NET project
- Adding DAQPilot component to the project's toolbox
- Setting the properties at design time
- Editing the properties at runtime
- Working with control methods
- Developing event handlers
- Running the application

Creating the Visual Studio VB.NET DAQ Project

Use the **New Project** dialog to start a new project in the Visual Studio development environment. Click **New** > **Project** from the **File** menu. Select **Visual Basic Language** as the project type, then select **Windows Application** from the templates. Press **OK** to apply. Refer to the figure below.

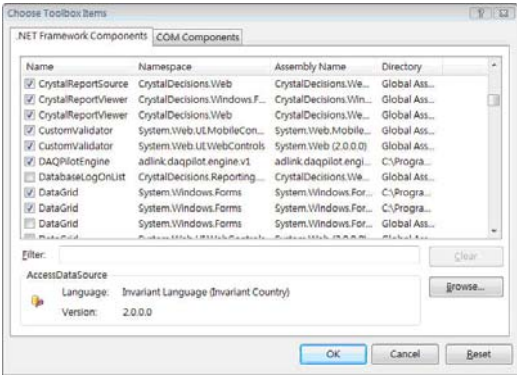


Adding DAQPilot Component to the Project's Toolbox

You need to install the DAQPilot's .NET component to the Visual Studio VB.NET before building an application. After installation, you may use the integrated DAQPilot component to conveniently carry out all task functions.

To add the DAQPilot component to the project's toolbox:

1. From a new Visual Studio VB.NET project, right-click on the **Toolbox** menu, then select **Choose Items**. The **Choose Toolbox Items** window appears.



2. Click on **Browse** to add the **DaqPilotEngine** file from this path:

   ```
   Drive\Program Files\Common Files\ADLINK\DAQPilot\
        adlink.daqpilot.engine.v1.dll
   ```

3. Locate the **DAQPilotEngine** component from the list, then click on the checkbox before the component name to select.

4. Select all related components you want to add in your project by clicking on the check boxes.

5. When finished, click **OK** to close the window. All selected components now appear in the toolbox.

Setting Properties at Design Time

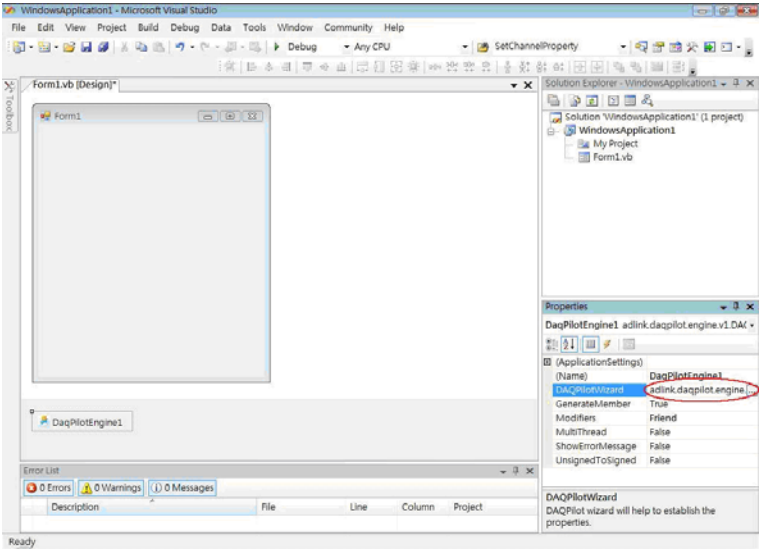
After placing a component on a Visual Basic .NET form, you can configure the component by setting its properties with the DAQPilot custom control property pages. Refer to the figure below.

Each component has its own set of default properties such as component name.

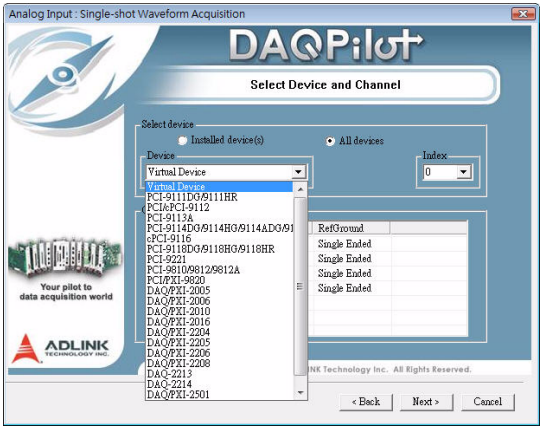
Do any of the following to open the Properties window:

- Select **Properties Window** from the **View** menu, then click the **Properties Window** button from the toolbar.
- Directly launch the **Properties Windows** from the popup menu.

The DAQPilot .NET component provides a custom Property Pages, that is the DAQPilot wizard, to replace the traditional ActiveX Control Property Page and to assist you easily in setting up the properties. You may select the **Property** option from the pop-up menu to launch the DAQPilot wizard.



When the DAQ Pilot wizard appears, set the DAQ parameters. In this example, we would like to perform a single-shot waveform acquisition with a virtual device.



Refer to **Chapter 3: Creating DAQ Tasks** for more information on the DAQPilot wizard parameters and supported tasks.

Edit Properties at Runtime

You can dynamically set and read the properties of the component in Visual Studio VB.NET, by using this syntax to set the property.

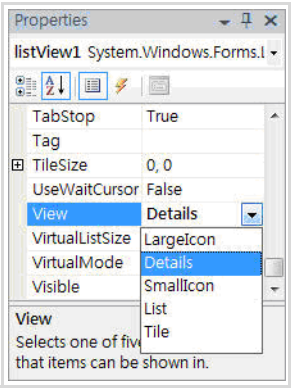
```
object.SetProperty property string, expression
```

For example, if you want to change the values during program execution:

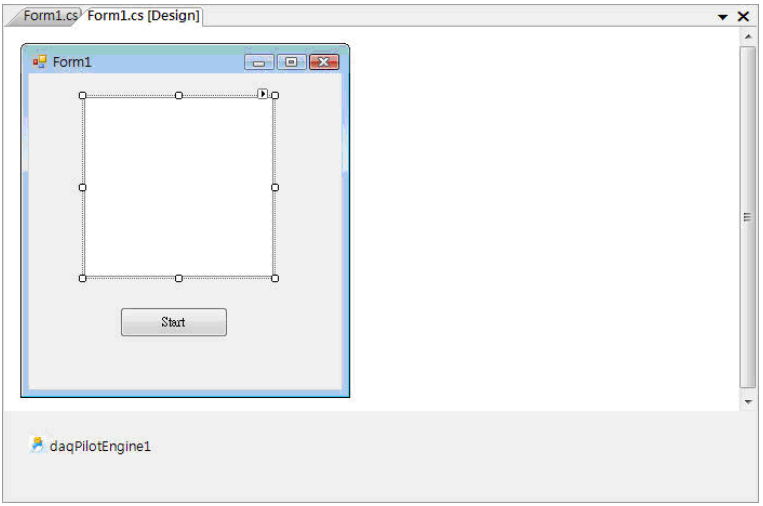
```
//Enable Ch1 by method
DaqPilotEngine1.SetChannelProperty(1, "Enable",
   True)

//Without add DAQPilot.vb file
DaqPilotEngine1.SetChannelProperty(1, "Range",
   203); // DPV_RANGE_B_2_5_V

//Add DAQPilot.vb file
DaqPilotEngine1.SetChannelProperty(1, "Range",
   DAQPilotValueID.DPV_RANGE_B_2_5_V);

//Change number of scan property to 100 by
   method
DaqPilotEngine1.SetDProperty("Number of
   scan(s)", 100);

DaqPilotEngine1.SetDProperty("Sampling rate per
   channel", 1000);

Dim vDaqpilotValue As Object
vDaqpilotValue =
   DaqPilotEngine1.GetChannelProperty(1,
   "Enable")
vDaqpilotValue =
   DaqPilotEngine1.GetChannelProperty(1,
   "Range")
vDaqpilotValue =
   DaqPilotEngine1.GetDProperty("Number of
   scan(s)")
```

Working with Control Methods

Place a ListView .NET component on Form1 with property value **Details** as its View property. Refer to the figure below.



Place a Button .NET component tagged as **Start**. Refer to the figure below.



To call a method, attach the method name to the component. If the method does not need any parameters, you can directly call the method with this syntax.

```
object.method()
```

For example, the **Start** method can launch the task function which is predefined in the component.

```
DAQPilotEngine1.Start()
```

Developing Event Handlers

After configuring the components, you may continue to create event handlers in your program to hook appropriate events generated on the components.

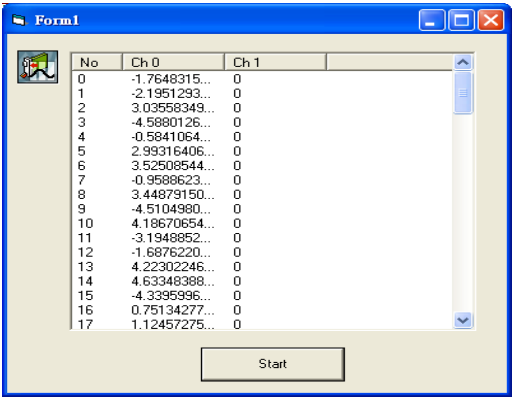
For example, the DAQPilot component has a DataArrival event that is triggered if the hardware DMA is finished. In addition, to develop the event routine code, most programming environments generate a skeleton function to be the code template.

Below is the code slice for processing the event of DMA data arrival based on the skeleton from Visual Studio.

```
Private Sub DaqPilotEngine1_DataArrival(ByVal
   sender As System.Object, ByVal e As
   adlink.daqpilot.engine.v1.DAQPilotEngine.DAQP
   ilotDataEventArgs) Handles
   DaqPilotEngine1.DataArrival
        Dim i, j As Int16
        Dim strCaption, strValue As String

        'Initial column header
        ListView1.Items.Clear()
        ListView1.Columns.Clear()
        ListView1.Columns.Add("No", 70)
        For i = 0 To UBound(e.ChNumList)
            strCaption = "Ch " & e.ChNumList(i)
            ListView1.Columns.Add(strCaption,
   100)
        Next i

        Dim NumOfScan As Long
        NumOfScan = (UBound(e.Data) + 1) /
   e.NumberOfChannel

        'Dump the data
        ListView1.Items.Clear()
        For i = 0 To (NumOfScan - 1)
            strValue = i
            ListView1.Items.Add(strValue)
            For j = 0 To (e.NumberOfChannel - 1)
                strValue = e.Data((i *
   e.NumberOfChannel) + j)

   ListView1.Items(i).SubItems.Add(strValue)
            Next j
        Next i
     End Sub
```

Running the Application

Press the **Start** button in your sample application to run and display the data.



Source Codes

```
Public Class Form1
Private Sub Button1_Click(ByVal sender As
     System.Object, ByVal e As
     System.EventArgs) Handles
     Button1.Click
    DaqPilotEngine1.SetChannelProperty(1,
     "Enable", True)
    DaqPilotEngine1.SetChannelProperty(1,
     "Range", 203)
    DaqPilotEngine1.SetChannelProperty(1,
     "Range",
     DAQPilotValueID.DPV_RANGE_B_2_5_V)
     DaqPilotEngine1.SetDProperty("Number
     of scan(s)", 100)
   DaqPilotEngine1.SetDProperty("Sampling
     rate per channel", 1000)
     Dim vDaqpilotValue As Object
     vDaqpilotValue =
     DaqPilotEngine1.GetChannelProperty(1,
     "Enable")
     vDaqpilotValue =
     DaqPilotEngine1.GetChannelProperty(1,
     "Range")
     vDaqpilotValue =
     DaqPilotEngine1.GetDProperty("Number
     of scan(s)")
     DaqPilotEngine1.Start()
End Sub
Private Sub
     DaqPilotEngine1_DataArrival(ByVal
     sender As System.Object, ByVal e As
     adlink.daqpilot.engine.v1.DAQPilotEng
     ine.DAQPilotDataEventArgs) Handles
     DaqPilotEngine1.DataArrival
        Dim i, j As Int16
        Dim strCaption, strValue As String
```

```
        'Initial column header
               ListView1.Items.Clear()
               ListView1.Columns.Clear()
               ListView1.Columns.Add("No", 70)
               For i = 0 To UBound(e.ChNumList)
                   strCaption = "Ch " &
            e.ChNumList(i)

            ListView1.Columns.Add(strCaption,
            100)
               Next i

               Dim NumOfScan As Long
               NumOfScan = (UBound(e.Data) + 1) /
            e.NumberOfChannel

               'Dump the data
               ListView1.Items.Clear()
               For i = 0 To (NumOfScan - 1)
                   strValue = i
                   ListView1.Items.Add(strValue)
                  For j = 0 To (e.NumberOfChannel
            - 1)
                        strValue = e.Data((i *
            e.NumberOfChannel) + j)

            ListView1.Items(i).SubItems.Add(strVa
            lue)
                   Next j
               Next i

        End Sub
    Private Sub Form1_FormClosing(ByVal sender
         As System.Object, ByVal e As
         System.Windows.Forms.FormClosingEvent
         Args) Handles MyBase.FormClosing
            DaqPilotEngine1.Stop()
        End Sub
    End Class
```

4.3 ActiveX Example for Microsoft Visual Basic

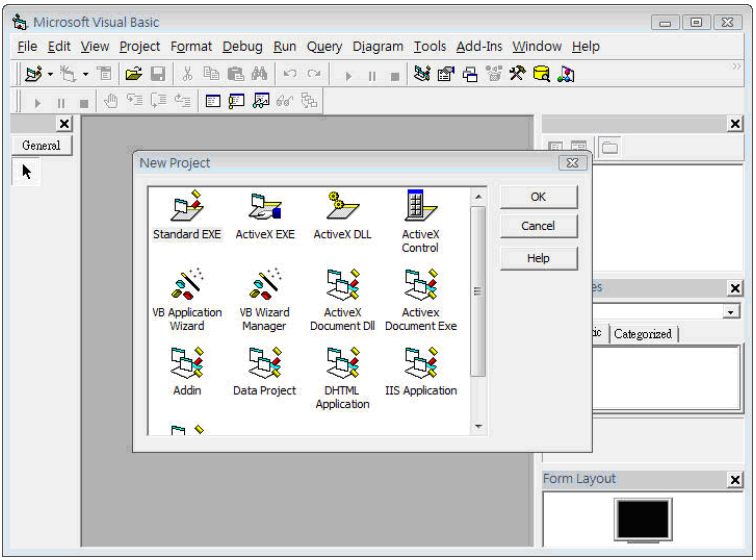
This section provides instructions on how to create a DAQ application using DAQPilot's ActiveX control.

The process involves seven steps:

- Creating the Visual Basic DAQ project
- Adding DAQPilot ActiveX control to the project's toolbox
- Setting the properties at design time
- Editing the properties at runtime
- Working with control methods
- Developing event handlers
- Running the application

Creating the Visual Basic DAQ Project

Use the **New Project** dialog to start a new project in Visual Basic development environment. Click **New Project** from the **File** menu. Select **Standard EXE,** then press **OK** to apply. Refer to the figure below.

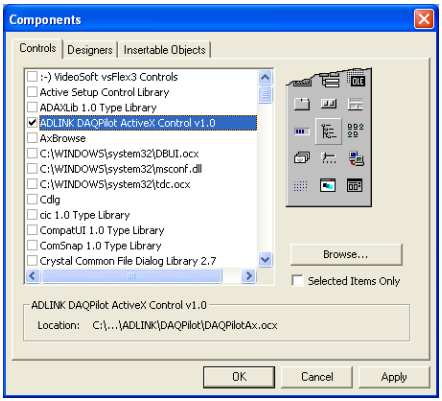


Adding DAQPilot ActiveX Control to the Project's Toolbox

You need to install DAQPilot's ActiveX controls to Visual Basic before building an application. After installation, you may use the integrated DAQPilot ActiveX controls to conveniently carry out all task functions.

To add the DAQPilot component to the project's toolbox:

1. From a new Visual Basic project, right-click on the **Toolbox** menu, then select **Choose Items**. The **Components** window appears.
2. Click the Controls tab, locate the **ADLINK DAQPilot ActiveX Control v1.0** from the list, then click on the checkbox before the component name to select.



3. Select all related components you want to add in your project by clicking on the checkboxes.
4. When finished, click **OK** to close the window. All selected components now appear in the toolbox.

Setting Properties at Design Time

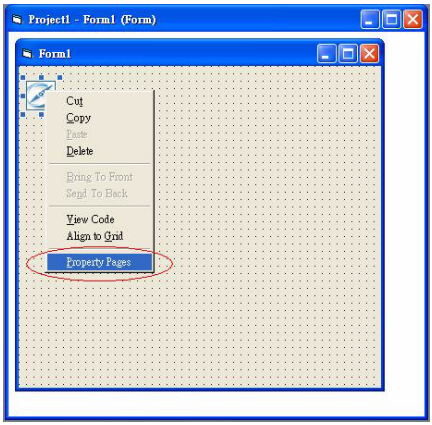
After placing a component on a Visual Basic form, you can configure the component by setting its properties with the DAQPilot custom control property pages. Refer to the figure below.

Each component has its own set of default properties such as component name.

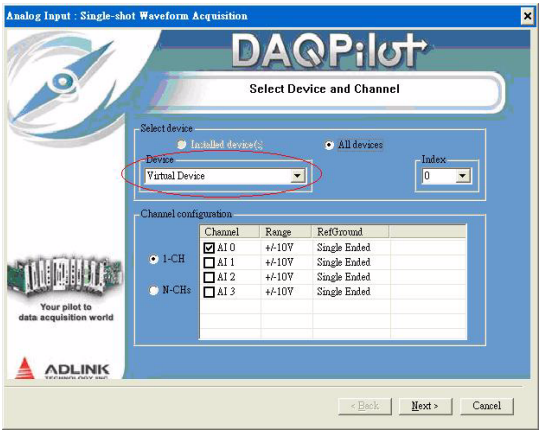
Do any of the following to open the Properties window:

- Select **Properties Window** from the **View** menu, then click the **Properties Window** button from the toolbar.
- Directly launch the **Properties Windows** from the popup menu.

The DAQPilot ActiveX control provides a custom Property Pages, that is the DAQPilot wizard, to replace the traditional ActiveX Control Property Page and to assist you easily in setting up the properties. You may select the **Property Page** item from the pop-up menu to launch the DAQPilot wizard.



When the DAQ Pilot wizard appears, set the DAQ parameters. In this example, we would like to perform a single-shot waveform acquisition with a virtual device.



Refer to **Chapter 3: Creating DAQ Tasks** for more information on the DAQPilot wizard parameters and supported tasks.

Edit Properties at Runtime

You can dynamically set and read the properties of the control in Visual Basic, by using this syntax to set the property.

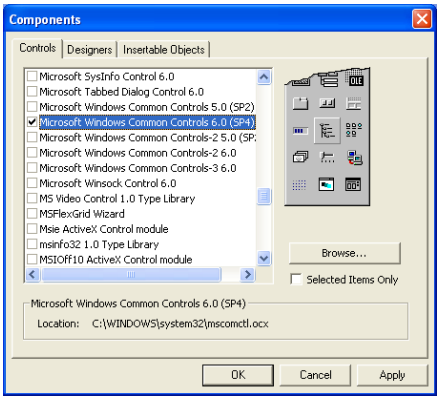
```
object.SetProperty property string, expression
```

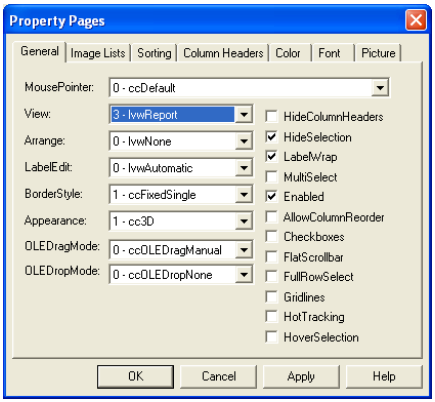
For example, if you want to change the values during program execution:

```
'Enable Ch1 by method
DAQPilot1.SetChannelProperty 1, "Enable", True
 'Without add DAQPilot.bas module file
DAQPilot1.SetChannelProperty 1, "Range", 203
 'DPV_RANGE_B_2_5_V
'Add DAQPilot.bas module file
DAQPilot1.SetChannelProperty 1, "Range",
 DAQPilotValueID.DPV_RANGE_B_2_5_V
 'Change number of scan property to 100 by
 method
DAQPilot1.SetDProperty "Number of scan(s)",
 100
DAQPilot1.SetDProperty "Sampling rate per
 channel", 1000
Dim vDAQPilotValue As Variant
vDAQPilotValue = DAQPilot1.GetChannelProperty
 (1, "Enable")
vDAQPilotValue =
 DAQPilot1.GetChannelProperty(1, "Range")
vDAQPilotValue =
 DAQPilot1.GetDProperty("Number of scan(s)")
```

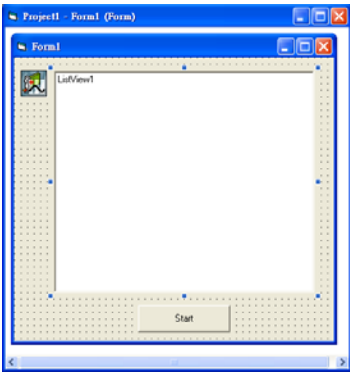
Working with Control Methods

Place a ListView control on Form1 with property value **3 — lvwReport** as its View property. Refer to the figure below.





Place a Button component tagged as **Start**. Refer to the figure below.



To call a method, attach the method name to the component. If the method does not need any parameters, you can directly call the method with this syntax.

```
object.method
```

For example, the **Start** method can launch the task function which is predefined in the component.

```
DAQPilot1.Start
```

Developing Event Handlers

After configuring the components, you may continue to create event handlers in your program to hook appropriate events generated on the components.

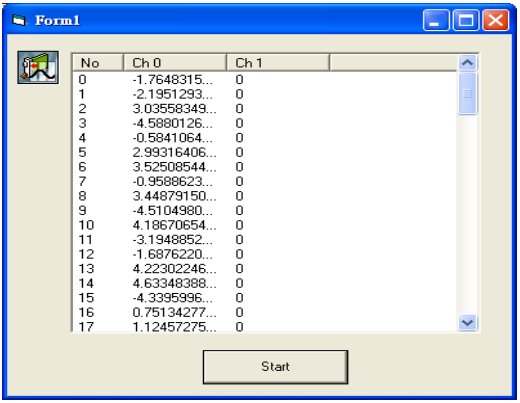
For example, the DAQPilot component has a DataArrival event that is triggered if the hardware DMA is finished. In addition, to develop the event routine code, most programming environments generate a skeleton function to be the code template.

The code slice for processing the event of DMA data arrival based on the skeleton from Visual Basic is illustrated next page.

```
Private Sub DAQPilot1_DataArrival(ByVal
   nNumberOfChannel As Long, ChNumList As
   Variant, Data As Variant)

    Dim i, j As Long
    Dim strCaption, strValue As String

    'Initial column header
    ListView1.ColumnHeaders.Clear
    ListView1.ColumnHeaders.Add , "No", 600
    For i = 0 To UBound(ChNumList)
        strCaption = "Ch" & ChNumList(i)
        ListView1.ColumnHeaders.Add , ,
   strCaption, 1200
    Next i

    Dim NumOfScan As Long
    NumOfScan = (UBound(Data) + 1) /
   nNumberOfChannel

    'Dump the data
    ListView1.ListItems.Clear
    For i = 0 To (NumOfScan - 1)
        strValue = i
        ListView1.ListItems.Add , , strValue
        For j = 0 To (nNumberOfChannel - 1)
          strValue = Data((i * nNumberOfChannel)
   + j)
          ListView1.ListItems.Item(i +
   1).SubItems(j + 1) = strValue
        Next j
    Next i

End Sub
```

Running the Application

Press the **Start** button in your sample application to run and display the data.



Source Codes

```
Option Explicit

Private Sub Command1_Click()
DAQPilotEngine1.SetChannelProperty 2,
     "Enable", True
DAQPilotEngine1.SetChannelProperty 2,
     "Range", 203 ' DPV_RANGE_B_2_5_V
'DAQPilotEngine1.SetChannelProperty 2,
     "Range", DPV_RANGE_B_2_5_V
DAQPilotEngine1.SetProperty "Number of
     scan", 100
DAQPilotEngine1.SetProperty "Sampling rate
     per channel", 10000
DAQPilotEngine1.Start
End Sub

Private Sub DAQPilot1_DataArrival(ByVal
     NumberOfChannel As Long, ChNumList As
     Variant, Data As Variant)
    Dim i, j As Long
    Dim strCaption, strValue As String

    'Initial column header
    ListView1.ColumnHeaders.Clear
    ListView1.ColumnHeaders.Add , , "No",
     600
    For i = 0 To UBound(ChNumList)
        strCaption = "Ch " & ChNumList(i)
        ListView1.ColumnHeaders.Add , ,
     strCaption, 1200
    Next i

    Dim NumOfScan As Long
    NumOfScan = (UBound(Data) + 1) /
     NumberOfChannel
```

```
        'Dump the data
           ListView1.ListItems.Clear
           For i = 0 To (NumOfScan - 1)
               strValue = i
               ListView1.ListItems.Add , ,
            strValue
               For j = 0 To (NumberOfChannel - 1)
                  strValue = Data((i *
            NumberOfChannel) + j)
                  ListView1.ListItems.Item(i +
            1).SubItems(j + 1) = strValue
               Next j
           Next i
       End Sub
```

4.4 API Example for Microsoft Visual Studio C++

This section provides instructions on how to create a DAQ application using DAQPilot's AISingleShot and demonstrates basic AISingleShot usage.

The process involves five steps:

- Creating the configuration (task) file
- Creating the Visual Studio C++ project
- Integrating the library file
- Editing the task files
- Running the application

Creating the Configuration (Task) File

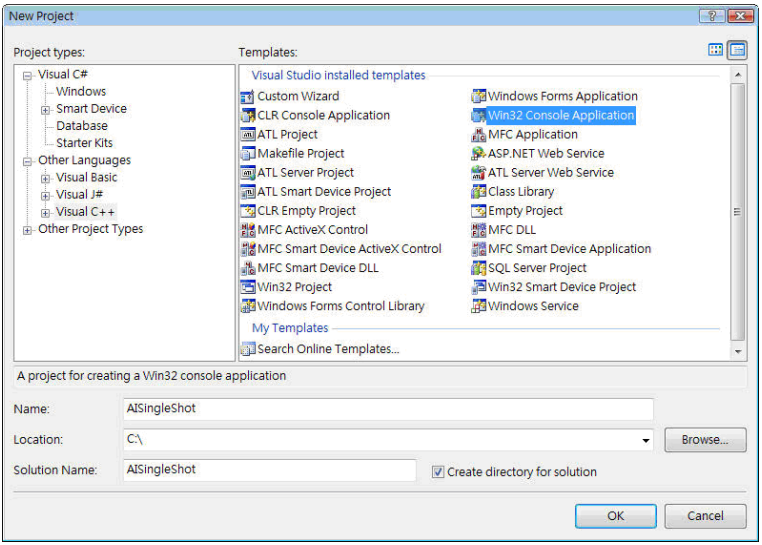
Create a configuration (task) file using DAQPilot. Refer to **Chapter 3: Creating DAQ Tasks** for more information. The created configuration (task) file is stored in this default directory:

```
C:\ADLINK\DAQPilot\Task Folder
```

Take note of the task file location.

Creating the Visual Studio C++ Project

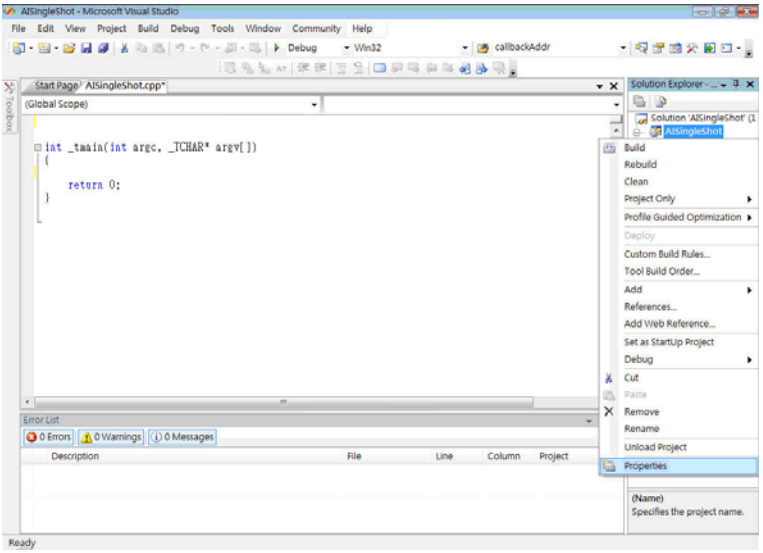
Use the **New Project** dialog to start a new project in Visual Studio C++ development environment. Click **File** from the main menu, then click on **New** > **Project**. Select **Visual C++** from the project type section, then select **Win32 Console Application** from the templates section. Press **OK** to apply. Refer to the figure below.



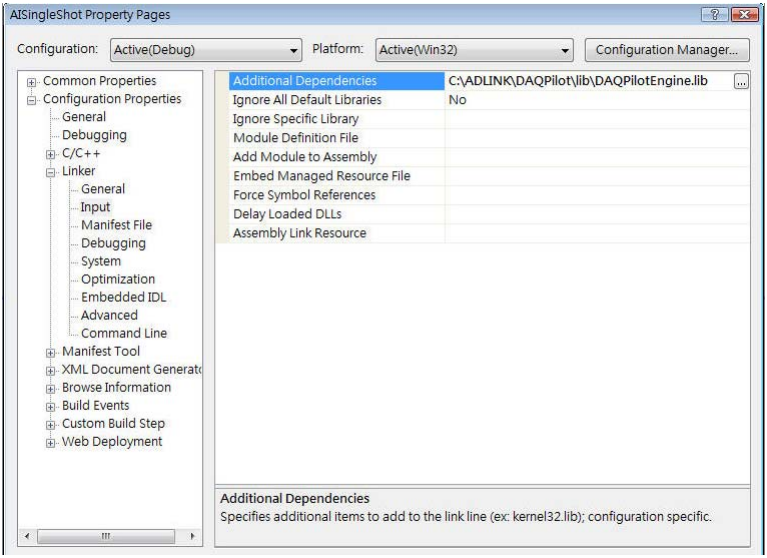
Integrating the Library File

You need to include the DAQPilotEngine.lib to the project. To do this:

1. Click on **AISingleShot**, then select **Properties** from the popup menu.



2. Type the task file path in the **Additional Dependencies** textbox, then press **OK**. Refer to the illustration below.

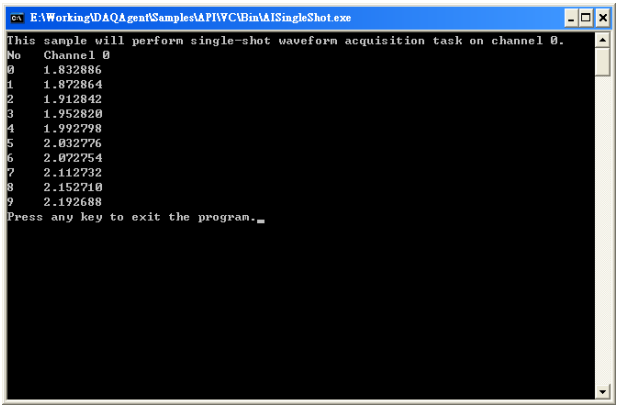


Editing the Task File

Even after you have generated the task file, you may further modify the properties using the DAQPilot Task Manager. Refer to section 2.4.

Running the Application

After you have finished coding, press the **Start Debugging** button to run the application and display the data. Refer to the illustration below.



Source Codes

```
#include "stdafx.h"
#include "conio.h"
#include "windows.h"
#include "DAQPilot.h"


int _tmain(int argc, _TCHAR* argv[])
{
//Declare variables
DAQPilotAnalogData DAQPilotData;
DAQPilotStatusID  dwError =
     DP_STATUS_NOERROR;
DWORD  dwDisplayCount = 0;
HANDLE hDAQPilotTask = NULL;
DWORD  i=0;
int    j=0, k=0;

//Load your DAQPilot task
hDAQPilotTask=DAQPilot_LoadTask("AISingleS
     hot", TRUE);
if(!hDAQPilotTask)
{
     printf("Load task fail\n");
     return 0;
}
printf("This sample will perform single-
     shot waveform acquisition task on
     channel 0.\n");
//Execute your DAQPilot task.
 //For Single-shot Waveform Acquisition
     task, you should use DAQPilot_Start()
     API.
dwError = DAQPilot_Start(hDAQPilotTask);
//LPCSTR lpszMsg =
     DAQPilot_GetErrorMessage(dwError);

//Get the data and set timeout as 10
     seconds (10000 ms).
dwError =
     DAQPilot_GetAIWaveform(hDAQPilotTask,
     &DAQPilotData, 10000);
```

```
        //Setting Timeout to INFINITE will execute
             synchronous operation.
        //dwError =
             DAQPilot_GetAIWaveform(hDAQPilotTask,
             &DAQPilotData, INFINITE);
        //Setting Timeout to 0 will execute
             asynchronous operation.
        //dwError =
             DAQPilot_GetAIWaveform(hDAQPilotTask,
             &DAQPilotData, 0);
        if( dwError==DP_STATUS_NOERROR )
        {
             //Display caption
             printf("No");
             for(k=0; k <
             DAQPilotData.nNumOfChannel; k++)
                  printf("   Channel %d",
             DAQPilotData.lpChannelNoList[k]);
             printf("\n");
             //Now the DAQPilotData.lpScaledData
             contains the waveform.
             //You may process the data here.
             dwDisplayCount = min(10,
             DAQPilotData.dwNumOfScan);
             for(i=0; i < dwDisplayCount; i++)
             {
                  printf("%d", i);
                  for(j=0; j <
             DAQPilotData.nNumOfChannel; j++)
                  {
                  printf(" %f",
             ((double*)DAQPilotData.lpScaledData)[
             (i*DAQPilotData.nNumOfChannel)+j]);
                  }
                  printf("\n");
             }
        }
```

```
        printf("Press any key to exit the
             program.");
        _getch();
             //Finally, close your DAQPilot task
             DAQPilot_EndTask(hDAQPilotTask);
             return 0;
        }
```

4.5 Component Example for Borland C++ Builder

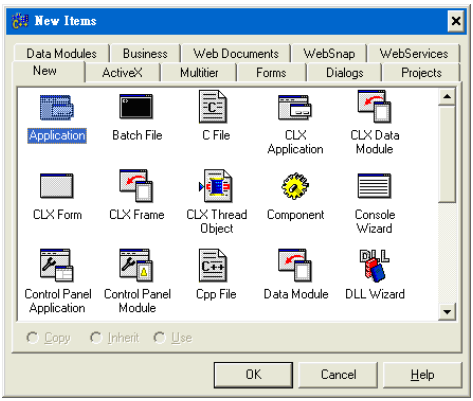
This section provides instructions on how to create a DAQ application using the DAQPilot component for Borland C++ Builder.

The process involves seven steps:

- Creating the Borland C++ Builder project
- Adding the DAQPilot component to the Tool Palette
- Setting the properties at design time
- Editing the properties at runtime
- Working with control methods
- Developing event handlers
- Running the application

Creating the Borland C++ Builder Project

Use the **New Items** dialog to start a new project in Borland C++ Builder development environment. Click **New > Others** from the **File** menu. Select **Application,** then press **OK** to apply. Refer to the figure below.



Adding DAQPilot Component to the Tool Palette

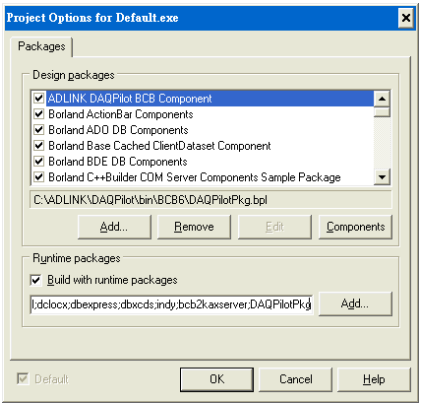
You need to install the DAQPilot component to the Borland C++ Builder tool palette before building an application. After installation, you may use the integrated DAQPilot component to conveniently carry out all task functions.

To add the DAQPilot component to the tool palette:

1. Click **Component** > **Install Packages** from the main menu to display the **Package** dialog box.
2. Click **Add**, to display the **Add Design Package** window, then locate the DAQPilotPkg.bpl file from the following folder:

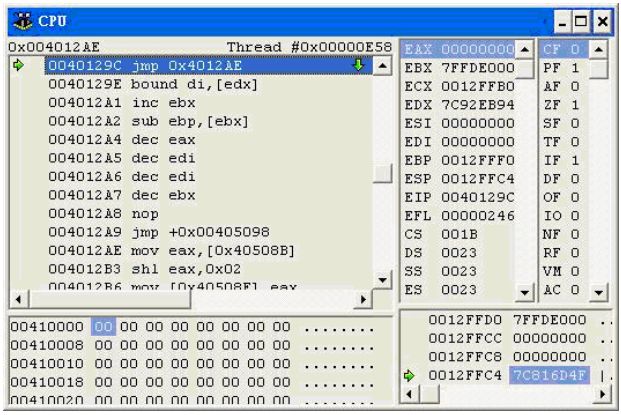
   `C:\ADLINK\DAQPilot\bin\BCB6`

3. Click the checkbox before the ADLINK DAQPilot BCB Component, then click OK button to close. All selected components now appear in the Tool Palette.



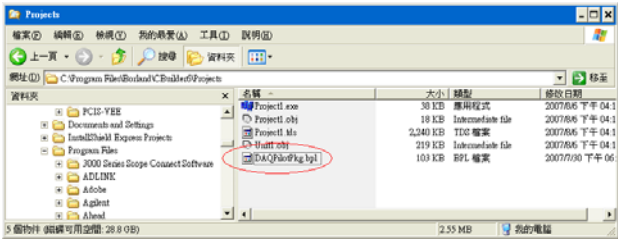
Access Violation

When the application fails to find the package library (DAQPilot-Pkg.bpl), an Access Violation warning message appears when using a DAQPilot component

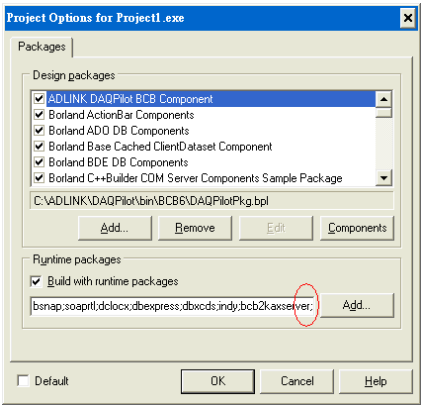


Do any of the following to solve this problem:

- Copy the DAQPilotPkg.bpl file to the project folder. Refer to the illustration below.

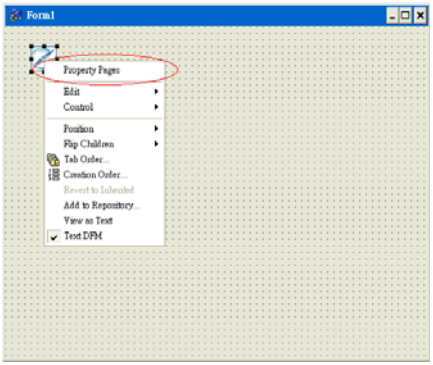


- Remove DAQPilotPkg from the runtime packages. Refer to the illustration below.



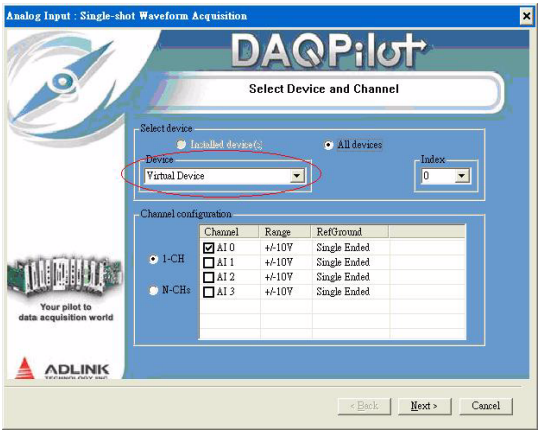
Setting Properties at Design Time

After placing a component on a Borland C++ Builder form, you can configure the component by setting its properties with the DAQPilot custom control property pages. Refer to the figure below.



You may launch the Object Inspector from the View menu. The DAQPilot component embeds a custom Property Pages, that is the DAQPilot Wizard, to replace the traditional ActiveX Control Property Page and to assist you easily in setting up the properties. You may select the **Property Pages** option from the pop-up menu to launch the DAQPilot wizard.

When the DAQ Pilot wizard appears, set the DAQ parameters. In this example, we would like to perform a single-shot waveform acquisition with a virtual device.



Refer to **Chapter 3: Creating DAQ Tasks** for more information on the DAQPilot wizard parameters and supported tasks.

Edit Properties at Runtime

You can dynamically set and read the properties of the DAQPilot component in Borland C++ Builder> For example, use this syntax to set the property:

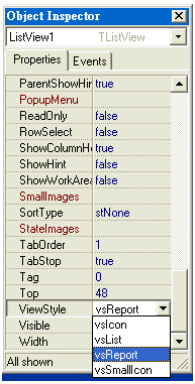
```
Object->SetDProperty (property string, value);
```

For example, if you want to change the values in runtime:

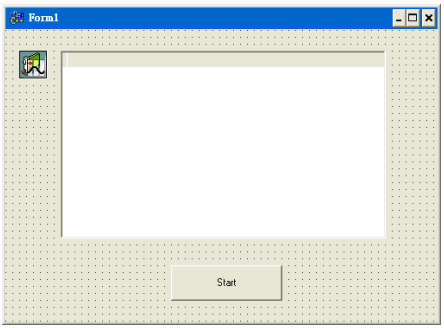
```
//Enable Channel 2 by method
DAQPilot1->SetChannelProperty(2,
   WideString("Enable"), TVariant(true));

DAQPilot1->SetChannelProperty(2,
   WideString("Range"), TVariant(203));//
   DPV_RANGE_B_2_5_V

// DPV_RANGE_B_2_5_V is defined in
   DAQPilotAdvancedProperties.h
DAQPilot1->SetChannelProperty(2,
   WideString("Range"),
   TVariant(DPV_RANGE_B_2_5_V));

//Change number of scan property to 100 by
   following method
DAQPilot1->SetDProperty(WideString("Number of
   scan(s)"), TVariant(100));

DAQPilot1->SetDProperty(WideString("Sampling
   rate per channel"), TVariant(1000));

TVariant vDAQPilotValue;
vDAQPilotValue = DAQPilot1-
   >GetChannelProperty(2, WideString("Enable"));
vDAQPilotValue = DAQPilot1-
   >GetChannelProperty(2, WideString("Range"));
vDAQPilotValue = DAQPilot1-
   >GetDProperty(WideString("Number of
   scan(s)"));
```

Working with Control Methods

Place a ListView component on Form1 with property value **vsreport** as its View property. Refer to the figure below.



Place a Button component named **Start**. Refer to the figure below.



To call a method, attach the method name to the component. If the method does not need any parameters, you can directly call the method with this syntax.

```
object->method();
```

For example, the **Start** method can launch the task function which is predefined in the component.

```
DAQPilot1->Start();
```

Developing Event Handlers

After you have configured the components, you may continue creating event handlers in your application to hook appropriate functions and process the received data.

For example, the DAQPilot component has a DataArrival event that is triggered when the hardware DMA function is finished.

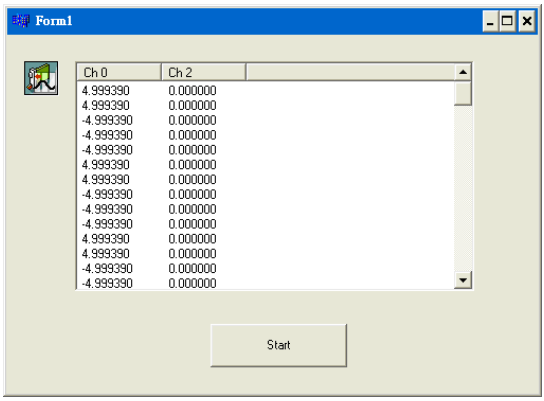
To develop the event routine code, most programming tools can generate the skeleton code as the code template. A code slice to process data is shown below for your reference.

```
void
   __fastcallTForm1::DAQPilot1DataArrival(TObjec
   t *Sender,
      int NumberOfChannel, Variant *ChNumList,
   Variant *Data)
{
        //Initial Column header
        ListView1->Items->Clear();
             ListView1->Columns->Clear();
        SAFEARRAY sa  = (ChNumList->vt &
   VT_BYREF) ? (**ChNumList->pparray) :
   (*ChNumList->parray);
        DWORD dwElements =
   sa.rgsabound[0].cElements;
        ::SafeArrayLock(&sa);
        for(UINT i=0; i< dwElements; i++)
        {
            AnsiString strColumn;
            strColumn.sprintf("Ch %d",
   ((short*)sa.pvData)[i]);
            ListView1->Columns->Add();
          ListView1->Columns->Items[i]->Caption
   = strColumn;
            ListView1->Columns->Items[i]->Width
   = 80;
        }
        ::SafeArrayUnlock(&sa);
```

```
  //Display the data
        sa  = (Data->vt & VT_BYREF) ? (**Data-
    >pparray) : (*Data->parray);
        dwElements = sa.rgsabound[0].cElements;

              DWORD dwDisplayCount = dwElements/
    NumberOfChannel;
        ::SafeArrayLock(&sa);
        for(UINT i=0; i< dwDisplayCount; i++)
        {
           AnsiString strValue;
           strValue.sprintf("%f",
    ((double*)sa.pvData)[(i*NumberOfChannel)+0]);
           TListItem  *ListItem;
           ListItem = ListView1->Items->Add();
           ListItem->Caption = strValue;
                for(int j=1; j <
    NumberOfChannel; j++)
           {
              strValue.sprintf("%f",
    ((double*)sa.pvData)[(i*NumberOfChannel)+j]);
              ListItem->SubItems->Add(strValue);
                }
        }
        ::SafeArrayUnlock(&sa);
  }
```

Running the Application

Press the **Start** button in your sample application to run and display the data.



Source Codes

```
//---------------------------------------
     #include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include
     "C:\ADLINK\DAQPilot\include\DAQPilotA
     dvancedProperties.h"
//---------------------------------------
     #pragma package(smart_init)
#pragma link "DAQPILOTAXLib_OCX"
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------
     __fastcall TForm1::TForm1(TComponent*
     Owner)
        : TForm(Owner)
{
}
//---------------------------------------
     void __fastcall
     TForm1::Button1Click(TObject *Sender)
{

   DAQPilot1->SetChannelProperty(2,
     WideString("Enable"),
     TVariant(true));
   DAQPilot1->SetChannelProperty(2,
     WideString("Range"), TVariant(203));
   DAQPilot1->SetChannelProperty(2,
     WideString("Range"),
     TVariant(DPV_RANGE_B_2_5_V));

   DAQPilot1-
     >SetDProperty(WideString("Number of
     scan(s)"), TVariant(100));

   DAQPilot1-
     >SetDProperty(WideString("Sampling
     rate per channel"), TVariant(1000));
```

```
 TVariant vDAQPilotValue;
   vDAQPilotValue = DAQPilot1-
     >GetChannelProperty(2,
     WideString("Enable"));
   vDAQPilotValue = DAQPilot1-
     >GetChannelProperty(2,
     WideString("Range"));
   vDAQPilotValue = DAQPilot1-
     >GetDProperty(WideString("Number of
     scan(s)"));

   DAQPilot1->Start();
}
//---------------------------------------
void __fastcall
     TForm1::DAQPilot1DataArrival(TObject
     *Sender,
      int NumberOfChannel, Variant
     *ChNumList, Variant *Data)
{
        //Initial Column header
        ListView1->Items->Clear();
        ListView1->Columns->Clear();
        SAFEARRAY sa  = (ChNumList->vt &
     VT_BYREF) ? (**ChNumList->pparray) :
     (*ChNumList->parray);
        DWORD dwElements =
     sa.rgsabound[0].cElements;
        ::SafeArrayLock(&sa);
        for(UINT i=0; i< dwElements; i++)
        {
            AnsiString strColumn;
            strColumn.sprintf("Ch %d",
     ((short*)sa.pvData)[i]);
            ListView1->Columns->Add();
            ListView1->Columns->Items[i]-
     >Caption = strColumn;
            ListView1->Columns->Items[i]-
     >Width = 80;
        }
        ::SafeArrayUnlock(&sa);
```

```
         //Display the data
                sa  = (Data->vt & VT_BYREF) ?
            (**Data->pparray) : (*Data->parray);
               dwElements =
            sa.rgsabound[0].cElements;

               DWORD dwDisplayCount = dwElements/
            NumberOfChannel;
               ::SafeArrayLock(&sa);
               for(UINT i=0; i< dwDisplayCount;
            i++)
               {
                  AnsiString strValue;
                  strValue.sprintf("%f",
            ((double*)sa.pvData)[(i*NumberOfChann
            el)+0]);
                  TListItem  *ListItem;
                  ListItem = ListView1->Items-
            >Add();
                  ListItem->Caption = strValue;
                for(int j=1; j < NumberOfChannel;
            j++)
                  {
                     strValue.sprintf("%f",
            ((double*)sa.pvData)[(i*NumberOfChann
            el)+j]);
                     ListItem->SubItems-
            >Add(strValue);
                  }
               }
               ::SafeArrayUnlock(&sa);
         }
         //---------------------------------------
            void __fastcall
            TForm1::FormCloseQuery(TObject
            *Sender, bool &CanClose)
         {
            DAQPilot1->Stop();
         }
         //---------------------------------------
```

5 APIs Function Reference

The chapter lists the DAQPilot Application Programming Interfaces (APIs).

NOTE For additional information on function usage variations for different DAQ devices, refer to the user and function reference manuals of PCIS-DASK, D2K-DASK, and WD-DASK.

5.1 DAQPilot_LoadTask

Loads a DAQPilot task specification from a DAQ task file. Use this function before any other operations.

```
HANDLE DAQPilot_LoadTask (
   LPCSTR lpszTaskName,
   BOOL bShowErrorMessage
);
```

Parameters

lpszTaskName The task filename to load. This function loads the task file (*.tsk) from the default folder: $INSTALLDIR\Task Folder\. If DAQPilot does not install in the default target folder, you must store the application file and the task file in the same folder.

bShowErrorMessage

TRUE	The detailed error message will be reported by a message box.
FALSE	No detailed error message will be reported.

Return Values

The function returns the handle of the new task when properly executed. When the function fails with a NULL return, you will receive the relative error information if bShowErrorMessage is set to TRUE.

Remarks

1. The lpszTaskName parameter does not need a full file-name with extension. For example, AISingleShot represent AISingleShot.tsk.
2. DAQPilot_LoadTask() or DAQPilot_CreateTask() only loads or creates the specified task. If you want to apply the task configuration to the hardware, call DAQPilot_Config() to complete the configuration.
3. Call DAQPilot_Close() to release the DAQPilot task.
4. The order of search for the task file is as follows:
   - ▷ Default task files' folder: $INSTALLDIR\Task Folder\
   - ▷ Current directory

Example

```
HANDLE hDAQPilotTask = NULL;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("AISingleShot",
     TRUE);
DAQPilot_EndTask(hDAQPilotTask);
```

5.2 DAQPilot_CreateTask

Creates the DAQPilot task handle with simple parameters.

```
HANDLE DAQPilot_CreateTask (
int nSubTaskID,
AllDeviceTypeID eDeviceID,
int nBoardIndex,
BOOL bShowErrorMessage
);
```

Parameters

nSubTaskID These Sub Task IDs are defined in the DAQPilotAdvancedProperties.h. For more information on DAQPilot supported task classes and categories, see section 1.3.

eDeviceID The type of card that performs the task. The Device IDs are defined in the DAQPilotAdvancedProperties.h.

nBoardIndex Each installed card has its own sequence number for identification. For similar or same type cards (as defined in argument eDeviceID) or cards that belong to the same series (except PCI- 7300A_Rev.A and PCI-7300A Rev.B), the assigned sequence number depends on the PCI slot order.

For example, if there is a PCI-9111DG card installed in the first PCI slot, one PCI-9111HR card, and two PCI-9112 cards installed on other PCI slots, the PCI-9111DG will be registered with nBoardIndex 0, and the PCI-9111HR card with nBoardIndex 1.

The PCI-9112 card in the first slot will be registered with nBoardIndex 0 while the second PCI-9112 card will be registered with nBoardIndex 1.

For PCI-7256, PCI-7258, PCI-7260, PCI-7442, PCI-7443, PCI-7444, and PCI-7452 series cards, you may use the onboard switch to set the board index.

bShowErrorMessage

TRUE	The detailed error message will be reported by a message box.
FALSE	No detailed error message will be reported.

Return Values

The function returns the handle of the new task when properly executed. When the function fails with a NULL return, you will receive the relative error information if bShowErrorMessage is set to TRUE.

Remarks

1. This function generates a simple task with default configuration.
2. DAQPilot_LoadTask() or DAQPilot_CreateTask() only loads or creates the specified task. If you want to apply the task configuration to the hardware, call DAQPilot_Config() to complete the configuration.
3. Call DAQPilot_Close() to release the DAQPilot task.
4. You may launch the DAQPilot wizard to know more about function usage.
5. See **Chapter 4: Programming with DAQPilot** for more information.

Example

```
HANDLE hDAQPilotTask = NULL;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_CreateTask
(DAQPilot_TASK_AI_ONESHOT,
     DEVICEID_VIRTUAL_DEVICE, 0, TRUE);
DAQPilot_EndTask(hDAQPilotTask);
```

5.3 DAQPilot_EndTask

Releases the DAQPilot's memory, threads, and related system resource. The function must be called at the end of a DAQPilot application.

```
BOOL DAQPilot_EndTask (
   HANDLE hTask
);
```

Parameters

hTask Task handle.

Return Values

TRUE	DAQPilot closed successfully.
FALSE	DAQPilot failed to close.

Example

```
HANDLE hDAQPilotTask = NULL;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("AISingleShot",
     TRUE);
DAQPilot_EndTask(hDAQPilotTask);
```

5.4 DAQPilot_SetChannelProperty

DAQPilot provides a series of APIs to set all channel properties. You may directly modify the parameters to suit a specific task.

```
DAQPilotStatusID
     DAQPilot_SetChannelStringProperty (
HANDLE hTask,
int nChannelNum,
LPCSTR lpszChannelPropertyID,
VARIANT vValue
);

DAQPilotStatusID DAQPilot_SetChannelProperty (
HANDLE hTask,
int nChannelNum,
ChannelPropertyID eChannelPropertyID,
VARIANT vValue
);

DAQPilotStatusID DAQPilot_SetChannelPropertyBool
     (
HANDLE hTask,
int nChannelNum,
ChannelPropertyID eChannelPropertyID,
BOOL bValue
);

DAQPilotStatusID DAQPilot_SetChannelPropertyEnum
     (
HANDLE hTask,
int nChannelNum,
ChannelPropertyID eChannelPropertyID,
DAQPilotValueID nAgentID
);

DAQPilotStatusID DAQPilot_SetChannelPropertyI4 (
HANDLE hTask,
int nChannelNum,
ChannelPropertyID eChannelPropertyID,
int nValue
);
```

```
DAQPilotStatusID DAQPilot_SetChannelPropertyU4 (
HANDLE hTask,
int nChannelNum,
ChannelPropertyID eChannelPropertyID,
DWORD dwValue
);

DAQPilotStatusID DAQPilot_SetChannelPropertyF64 (
HANDLE hTask,
int nChannelNum,
ChannelPropertyID eChannelPropertyID,
double dbValue
);

DAQPilotStatusID DAQPilot_SetChannelPropertySTR (
HANDLE hTask,
int nChannelNum,
ChannelPropertyID eChannelPropertyID,
LPCSTR lpszValue
);
```

Parameters

hTask	Task handle
nChannelNum	Channel number
lpszChannelPropertyID	Channel property string
eChannelPropertyID	Channel property ID defined in DAQPilotAdvancedProperties.h.
vValue, bValue, nAgentID, nValue, dwValue, dbValue, lpszValue	Channel property value.

Return Values

The return value is the DAQPilot status ID.

Remarks

1. Different devices support different channel parameters.
2. For various data type, you may use the appropriate API to set the value. For example, you may use the DAQPilot_SetChannelPropertyI4() to configure data that ranges from -100 to +100. Refer to the edit controls below.
   - ▷ Edit Control: Use DAQPilot_SetChannelPropertyI4() function to set the data that ranges from -2147483648 to +2147483647.
   - ▷ Edit Control: Use DAQPilot_SetChannelPropertyU4() function to set the data that ranges from 0 to + 4294967295.
   - ▷ Edit Control: Use DAQPilot_SetChannelPropertyF64() function to set the data that ranges from -1.79769313486232E308 to +1.79769313486232E308
   - ▷ Edit Control: Use DAQPilot_SetChannelPropertySTR() function to set string data.
   - ▷ Combo Box Control: Use DAQPilot_SetChannelPropertyEnum() function to set Enum data.
   - ▷ For Check Box Control: Use the DAQPilot_SetChannelPropertyBool() function to set the Boolean data.
3. You may launch the DAQPilot wizard to know more about function usage.
4. See **Chapter 4: Programming with DAQPilot** for more information.
5. For quick and minor modifications, you may use the DAQPilot_SetStringProperty to directly configure specific parameters with the appropriate caption in the **Channel configuration** frame. However, it is suggested that you finish the main task using the DAQPilot wizard.

For example, use

```
DAQPilot_SetChannelStringProperty(0, "Range",…);
```

to modify the range, and use

```
DAQPilot_SetChannelStringProperty(0,
     "RefGround",…);
```

to assign the reference base.

Example

```
HANDLE hDAQPilotTask = NULL;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("AISingleShot",
     TRUE);
DAQPilot_SetChannelPropertyEnum(hDAQPilotTask,
     0, CP_Range,
DPV_RANGE_B_2_5_V);
DAQPilot_EndTask(hDAQPilotTask);
```

5.5 DAQPilot_GetChannelProperty

DAQPilot provides a series of APIs to get all channel properties. You can query the parameters from a specific task.

```
VARIANT DAQPilot_GetChannelStringProperty (
HANDLE hTask,
int nChannelNum,
LPCSTR lpszChannelPropertyID
);

VARIANT DAQPilot_GetChannelProperty (
HANDLE hTask,
int nChannelNum,
ChannelPropertyID eChannelPropertyID
);

BOOL DAQPilot_GetChannelPropertyBool (
HANDLE hTask,
int nChannelNum,
ChannelPropertyID eChannelPropertyID
);

DAQPilotValueID DAQPilot_GetChannelPropertyEnum (
HANDLE hTask,
int nChannelNum,
ChannelPropertyID eChannelPropertyID
);

int DAQPilot_GetChannelPropertyI4 (
HANDLE hTask,
int nChannelNum,
ChannelPropertyID eChannelPropertyID
);

DWORD DAQPilot_GetChannelPropertyU4 (
HANDLE hTask,
int nChannelNum,
ChannelPropertyID eChannelPropertyID
);
```

```
double DAQPilot_GetChannelPropertyF64 (
HANDLE hTask,
int nChannelNum,
ChannelPropertyID eChannelPropertyID
);

LPCSTR DAQPilot_SetChannelPropertySTR (
HANDLE hTask,
int nChannelNum,
ChannelPropertyID eChannelPropertyID
);
```

Parameters

hTask	Task handle
nChannelNum	Channel number
lpszChannelPropertyID	Channel property string
eChannelPropertyID	Channel property ID defined in the DAQPilotAdvancedProperties.h.

Return Values

The return value is the channel property.

Remarks

1. Different devices support different channel parameters.
2. For various data type, you may use the appropriate API to query the value. For example, you may use the DAQPilot_GetChannelPropertyI4() to configure data that ranges from -100 to +100. Refer to the edit controls below.
   - ▷ Edit Control: Use DAQPilot_GetChannelPropertyI4() function to set the data that ranges from -2147483648 to +2147483647.
   - ▷ Edit Control: Use DAQPilot_GetChannelPropertyU4() function to set the data that ranges from 0 to + 4294967295.

- ▷ Edit Control: Use DAQPilot_GetChannelPropertyF64() function to set the data that ranges from -1.79769313486232E308 to +1.79769313486232E308
- ▷ Edit Control: Use DAQPilot_GetChannelPropertySTR() function to set string data.
- ▷ Combo Box Control: Use DAQPilot_GetChannelPropertyEnum() function to set Enum data.
- ▷ For Check Box Control: Use the DAQPilot_GetChannelPropertyBool() function to set the Boolean data.

3. You may launch the DAQPilot wizard to know more about function usage.
4. See **Chapter 4: Programming with DAQPilot** for more information.

Example

```
HANDLE hDAQPilotTask = NULL;
DAQPilotValueID eValue = DPV_UNDEF;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("AISingleShot",
     TRUE);
eValue =
     DAQPilot_GetChannelPropertyEnum(hDAQPilotTa
     sk , 0, CP_Range);
DAQPilot_EndTask(hDAQPilotTask);
```

5.6 DAQPilot_SetProperty

DAQPilot uses a series of APIs to set all properties. You may directly modify the parameters to a specific task.

```
DAQPilotStatusID DAQPilot_SetStringProperty (
HANDLE hTask,
LPCSTR lpszPropertyID,
VARIANT vValue);

DAQPilotStatusID DAQPilot_SetProperty(
HANDLE hTask,
DAQPilotPropertyID ePropertyID,
VARIANT vValue);

DAQPilotStatusID DAQPilot_SetPropertyBool(
HANDLE hTask,
DAQPilotPropertyID ePropertyID,
BOOL bValue);

DAQPilotStatusID DAQPilot_SetPropertyEnum(
HANDLE hTask,
DAQPilotPropertyID ePropertyID,
DAQPilotValueID nAgentID);

DAQPilotStatusID DAQPilot_SetPropertyI4(
HANDLE hTask,
DAQPilotPropertyID ePropertyID,
int nValue);

DAQPilotStatusID DAQPilot_SetPropertyU4(
HANDLE hTask,
DAQPilotPropertyID ePropertyID,
DWORD dwValue);

DAQPilotStatusID DAQPilot_SetPropertyF64(
HANDLE hTask,
DAQPilotPropertyID ePropertyID,
double dbValue);

DAQPilotStatusID DAQPilot_SetPropertySTR(
HANDLE hTask,
DAQPilotPropertyID ePropertyID,
LPCSTR lpszValue);
```

Parameters

hTask	Task handle
lpszPropertyID	Property string
ePropertyID	Property ID

Return Values

The return value is the DAQPilot status ID.

Remarks

1. Different devices support different channel parameters.
2. For various data type, you may use the appropriate API to configure various data . For example, you may use the DAQPilot_SetPropertyI4() to set the data that ranges from -100 to +100. Refer to the edit controls below.
   - ▷ Edit Control: Use DAQPilot_SetPropertyI4() function to set the data that ranges from -2147483648 to +2147483647.
   - ▷ Edit Control: Use DAQPilot_SetPropertyU4() function to set the data that ranges from 0 to + 4294967295.
   - ▷ Edit Control: Use DAQPilot_SetPropertyF64() function to set the data that ranges from -1.79769313486232E308 to +1.79769313486232E308
   - ▷ Edit Control: Use DAQPilot_SetPropertySTR() function to set string data.
   - ▷ Combo Box Control: Use DAQPilot_SetPropertyEnum() function to set Enum data.
   - ▷ For Check Box Control: Use the DAQPilot_SetPropertyBool() function to set the Boolean data.

3. You may launch the DAQPilot wizard to know more about function usage.
4. See **Chapter 4: Programming with DAQPilot** for more information.
5. For quick and minor modifications, you may use the DAQPilot_SetStringProperty to directly configure specific parameters with the appropriate caption in the **Channel configuration** frame. However, it is suggested that you finish the main task using the DAQPilot wizard.

For example, use

```
DAQPilot_SetStringProperty("Clock source
     (Conversion source)",…);
```

to modify the clock source, and use

```
DAQPilot_SetStringProperty("Sampling rate per
     channel",…);
```

to modify the sampling rate per channel.

Example

```
HANDLE hDAQPilotTask = NULL;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("AISingleShot",
     TRUE);
DAQPilot_SetPropertyU4 (hDAQPilotTask,
     DP_TIMING_NUM_OF_SCAN, 1000);
DAQPilot_EndTask(hDAQPilotTask);
```

5.7 DAQPilot_GetProperty

DAQPilot uses a series of APIs to get all properties. You can query the parameters from a specific task.

```
VARIANT DAQPilot_GetStringProperty (
HANDLE hTask,
LPCSTR lpszPropertyID);

VARIANT DAQPilot_GetProperty (
HANDLE hTask,
DAQPilotPropertyID ePropertyID
);

BOOL DAQPilot_GetPropertyBool (
HANDLE hTask,
DAQPilotPropertyID ePropertyID
);

DAQPilotValueID DAQPilot_GetPropertyEnum (
HANDLE hTask,
DAQPilotPropertyID ePropertyID);

DAQPilotStatusID DAQPilot_GetPropertyI4 (
HANDLE hTask,
DAQPilotPropertyID ePropertyID,
int nValue);

DWORD DAQPilot_GetPropertyU4 (
HANDLE hTask,
DAQPilotPropertyID ePropertyID);

double DAQPilot_GetPropertyF64 (
HANDLE hTask,
DAQPilotPropertyID ePropertyID
);

LPCSTR DAQPilot_GetPropertySTR (
HANDLE hTask,
DAQPilotPropertyID ePropertyID
);
```

Parameters

hTask	Task handle
lpszPropertyID	Property string
ePropertyID	Property ID

Return Values

The return value is the DAQPilot property value.

Remarks

1. Different devices support different channel parameters.
2. For various data type, you may use the appropriate API to query various data types. For example, you may use the DAQPilot_GetPropertyI4() to configure data that ranges from -100 to +100. Refer to the edit controls below.
   - ▷ Edit Control: Use DAQPilot_GetPropertyI4() function to set the data that ranges from -2147483648 to +2147483647.
   - ▷ Edit Control: Use DAQPilot_GetPropertyU4() function to set the data that ranges from 0 to +4294967295.
   - ▷ Edit Control: Use DAQPilot_GetPropertyF64() function to set the data that ranges from -1.79769313486232E308 to +1.79769313486232E308
   - ▷ Edit Control: Use DAQPilot_GetPropertySTR() function to set string data.
   - ▷ Combo Box Control: Use DAQPilot_GetPropertyEnum() function to set Enum data.
   - ▷ For Check Box Control: Use the DAQPilot_GetPropertyBool() function to set the Boolean data.
3. You may launch the DAQPilot wizard to know more about function usage.
4. See **Chapter 4: Programming with DAQPilot** for more information.

Example

```
HANDLE hDAQPilotTask = NULL;
DWORD dwValue = 0;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("AISingleShot",
     TRUE);
dwValue = DAQPilot_GetPropertyU4 (hDAQPilotTask,
     P_TIMING_NUM_OF_SCAN);
DAQPilot_EndTask(hDAQPilotTask);
```

5.8 DAQPilot_GetTaskStatus

Gets the status of specific DAQ task.

```
DAQPilotStatusID DAQPilot_GetTaskStatus (
   HANDLE hTask
);
```

Parameters

hTask Task handle

Return Values

```
DP_STATUS_DEVICE_STOP_AND_DIRTY
```

The device stops. If you call DAQPilot_Start() to restart the task, DAQPilot_Start() automatically calls the DAQPilot_Config() function before running.

```
DP_STATUS_DEVICE_STOP
```

The device stops. You can call DAQPilot_Start() to restart the task but the DAQPilot_Start() function does not automatically call DAQPilot_Config() function before running.

```
DP_STATUS_DEVICE_RUNING
```

The device is running.

Example

```
HANDLE hDAQPilotTask = NULL;
hDAQPilotTask=DAQPilot_LoadTask("AOSingleShot",
     TRUE);
DAQPilot_Start(hDAQPilotTask);
printf("Wait for AO output finish.");
while(DP_STATUS_DEVICE_RUNING==DAQPilot_GetTaskS
     tatus(hDAQPilotTask));
DAQPilot_EndTask(hDAQPilotTask);
```

5.9 DAQPilot_GetEnabledChannelList

Gets the number of enabled channels and the channels list.

```
int DAQPilot_GetEnabledChannelList (
   HANDLE hTask,
   WORD **ppChNoList
);
```

Parameter

hTask Task handle

ppChNoList WORD pointer address for obtaining enabled channel number list.

Return Values

Numbers of all enabled channels.

Example

```
HANDLE hDAQPilotTask = NULL;
WORD * pChNoList = NULL;
int nNumOfChannel  = 0;
int i = 0;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("AISingleShot",
     TRUE);
//Display channel number
nNumOfChannel =
     DAQPilot_GetEnabledChannelList(hDAQPilotTas
     k, &pChNoList);
for(i=0; i < nNumOfChannel; i++)
printf("Channel %d", pChNoList[i]);
printf("\n");
DAQPilot_EndTask(hDAQPilotTask);
```

5.10 DAQPilot_EnableSingleChannel

Enables a single-channel only configuration.

```
DAQPilotStatusID DAQPilot_EnableSingleChannel (
   HANDLE hTask,
int nChannelNum
);
```

Parameters

hTask Task handle

nChannelNum Channel number

Return Values

The return value is the DAQPilot status ID.

Remarks

This function disables all channels, then enables the specified channel.

Example

```
HANDLE hDAQPilotTask = NULL;
WORD * pChNoList = NULL;
int nNumOfChannel = 0;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("AISingleShot",
     TRUE);
//Enable channel 0 and channel 1 multiple
     channels
     DAQPilot_SetChannelPropertyBool(hDAQPilotTa
     sk, 0, CP_Enable, TRUE);
DAQPilot_SetChannelPropertyBool(hDAQPilotTask,1,
     CP_Enable, TRUE);
nNumOfChannel =
     DAQPilot_GetEnabledChannelList(hDAQPilotTas
     k, &pChNoList); //nNumOfChannel == 2,
     pChNoList[0]==0, pChNoList[1]==1
//Enable channel 2 only.
DAQPilot_EnableSingleChannel (hDAQPilotTask, 2);
nNumOfChannel =
     DAQPilot_GetEnabledChannelList(hDAQPilotTas
     k, &pChNoList); // nNumOfChannel == 1,
     pChNoList[0]==2
DAQPilot_EndTask(hDAQPilotTask);
```

5.11 DAQPilot_GetErrorMessage

Gets the error message from status ID.

```
LPCSTR DAQPilot_GetErrorMessage (
   DAQPilotStatusID eStatusCode
);
```

Parameters

eStatusCode DAQPilot status ID

Return Value

The return value is the DAQPilot status string.

Example

```
HANDLE hDAQPilotTask = NULL;
DAQPilotStatusID eStatus = DP_STATUS_NOERROR;
LPCSTR lpszMessage = NULL;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("AISingleShot",
     FALSE);
eStatus = DAQPilot_Start(hDAQPilotTask);
lpszMessage = DAQPilot_GetErrorMessage(eStatus);
printf("DAQPilot start result:%s \n",
     lpszMessage);
DAQPilot_EndTask(hDAQPilotTask);
```

5.12 DAQPilot_Config

Pre-initializes the system resource before executing any task job. If you do not execute DAQPilot_Config before DAQPilot_Start, DAQPilot_Start automatically initializes all related configurations and does not allow a task application to immediately start.

```
DAQPilotStatusID DAQPilot_Config (
   HANDLE hTask
);
```

Parameters

hTask Task handle

Return Values

The return value may be a status, a warning, or an error.

NOTE DAQPilot_Config() is a time-consuming function to initialize hardware, allocate memory, generate thread, and so on. So, it is better to complete this function before executing DAQPilot_Start.

If you have changed channel property by DAQPilot_SetChannelProperty and DAQPilot_SetProperty respectively, you may call DAQPilot_Config() function to apply the modification.

Example

```
HANDLE hDAQPilotTask = NULL;
DAQPilotStatusID eStatus = DP_STATUS_NOERROR;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("AISingleShot",
     TRUE);
eStatus = DAQPilot_GetTaskStatus(hDAQPilotTask);
     //eStatus ==
     DP_STATUS_DEVICE_STOP_AND_DIRTY
DAQPilot_Config(hDAQPilotTask);
eStatus = DAQPilot_GetTaskStatus(hDAQPilotTask);
     //eStatus == DP_STATUS_DEVICE_STOP, device
     is ready to run
DAQPilot_Start(hDAQPilotTask);//Start task
     immediately.
DAQPilot_EndTask(hDAQPilotTask);
```

5.13 DAQPilot_Start

Performs a DAQ task start.

```
DAQPilotStatusID DAQPilot_Start (
   HANDLE hTask
);
```

Parameters

hTask Task handle

Return Values

The return value may be a status, a warning, or an error.

Remarks

1. This function is available only for the following tasks:

   ```
   DAQPilot_TASK_AI_ONESHOT
   DAQPilot_TASK_AI_CONTINUE
   DAQPilot_TASK_AO_ONESHOT
   DAQPilot_TASK_AO_CONTINUE
   DAQPilot_TASK_AO_FUNCTION_GEN
   DAQPilot_TASK_DI_ONESHOT
   DAQPilot_TASK_DI_CONTINUE
   DAQPilot_TASK_DO_ONESHOT
   DAQPilot_TASK_DO_CONTINUE
   DAQPilot_TASK_TC_COUNTER
   DAQPilot_TASK_TC_TIMER_INTERRUPT
   DAQPilot_TASK_TC_MODE_OPERATION
   ```

2. To improve the function performance, refer to the DAQPilot_Config() function.

Example

```
HANDLE hDAQPilotTask = NULL;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("AISingleShot",
     TRUE);
DAQPilot_Start(hDAQPilotTask);
DAQPilot_EndTask(hDAQPilotTask);
```

5.14 DAQPilot_Stop

Stops a DAQ task

```
DAQPilotStatusID DAQPilot_Start (
   HANDLE hTask
);
```

Parameters

hTask Task handle

Return Values

The return value is the DAQPilot status ID.

Remarks

This function is available only for the following task categories:

```
DAQPilot_TASK_AI_ONESHOT
DAQPilot_TASK_AI_CONTINUE
DAQPilot_TASK_AO_ONESHOT
DAQPilot_TASK_AO_CONTINUE
DAQPilot_TASK_AO_FUNCTION_GEN
DAQPilot_TASK_DI_ONESHOT
DAQPilot_TASK_DI_CONTINUE
DAQPilot_TASK_DO_ONESHOT
DAQPilot_TASK_DO_CONTINUE
DAQPilot_TASK_TC_COUNTER
DAQPilot_TASK_TC_TIMER_INTERRUPT
DAQPilot_TASK_TC_MODE_OPERATION
```

Example

```
HANDLE hDAQPilotTask = NULL;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("AISingleShot",
     TRUE);
DAQPilot_Start(hDAQPilotTask);
DAQPilot_Stop(hDAQPilotTask);
DAQPilot_EndTask(hDAQPilotTask);
```

5.15 DAQPilot_AI_ReadChannel

Acquires an analog value from a specific channel.

```
DWORD DAQPilot_AI_ReadChannel (
   HANDLE hTask,
   double *pdbValue
);
```

Parameters

hTask Task handle

pdbValue Acquired analog value

Return Values

The return value is the DAQPilot status ID.

Remarks

This function is available only for:

```
DAQPilot_TASK_AI_VOLTAGE_POLLING.
```

Example

```
HANDLE hDAQPilotTask = NULL;
double dbVoltageValue = 0;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("AIPolling",
     TRUE);
//Switch to Channel 0
DAQPilot_EnableSingleChannel(hDAQPilotTask, 0);
DAQPilot_AI_ReadChannel(hDAQPilotTask,
     &dbVoltageValue);
//Now the dbVoltageValue contains voltage value
     of input signal.
printf("Voltage value=%f\n", dbVoltageValue);
DAQPilot_EndTask(hDAQPilotTask);
```

5.16 DAQPilot_AI_ReadChannels

Acquires analog values from multiple channels.

```
DAQPilotStatusID DAQPilot_AI_ReadChannels (
   HANDLE hTask,
   DAQPILOTAnalogData *pstDataInfo
);
```

Parameters

hTask Task handle

pstDataInfo A pointer to the DAQPILOTAnalogData structure memory space which will receive the analog data information.

Return Values

The return value is DAQPilot status ID.

Remarks

This function is available only for:

```
DAQPilot_TASK_AI_VOLTAGE_POLLING
```

Example

```
HANDLE hDAQPilotTask = NULL;
DAQPilotAnalogData DAQPilotData;
DAQPilotStatusID eError = DP_STATUS_NOERROR;
int j=0, k=0;

//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("AIPolling",
     TRUE);
//Enable both channel 0 and channel 1.
DAQPilot_SetChannelPropertyBool(hDAQPilotTask,
     0, CP_Enable, TRUE);
DAQPilot_SetChannelPropertyBool(hDAQPilotTask,
     1, CP_Enable, TRUE);
eError = DAQPilot_AI_ReadChannels(hDAQPilotTask,
     &DAQPilotData);
if( eError==DP_STATUS_NOERROR )
{
//Display caption
for(k=0; k < DAQPilotData.nNumOfChannel; k++)
```

```
        printf("Channel %d",
             DAQPilotData.lpChannelNoList[k]);
        printf("\n");
        //Now the DAQPilotData.lpScaledData contains the
             data.
        //You may process the data here.
        for(j=0; j < DAQPilotData.nNumOfChannel; j++)
        printf("%f",
             ((double*)DAQPilotData.lpScaledData)[j]);
        printf("\n");
        }
        DAQPilot_EndTask(hDAQPilotTask);
```

5.17 DAQPilot_GetAIWaveform

Acquires continuous analog input signal from a single channel or from multiple channels. Acquisition mode may be in one-shot or continuous mode.

```
DAQPilotStatusID DAQPilot_GetAIWaveform (
   HANDLE hTask,
   DAQPILOTAnalogData *pstDataInfo,
   DWORD dwTimeOut
);
```

Parameters

hTask Task handle

pstDataInfo A pointer to the DAQPILOTAnalogData structure memory space that will receive the analog data.

dwTimeOut The timeout interval in milliseconds. If dwTimeOut is 0, the function tests the object's state, then returns immediately. If dwTimeOut is INFINITE, the function continues until data acquisition is completed.

Return Values

The return value may be a status, a warning, or an error.

Remarks:

This function is available only for the following tasks:

```
DAQPilot_TASK_AI_ONESHOT
DAQPilot_TASK_AI_CONTINUE
```

Example

```
HANDLE hDAQPilotTask = NULL;
DAQPilotAnalogData DAQPilotData;
DAQPilotStatusID   eError = DP_STATUS_NOERROR;
int j=0, k=0;

//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("AISingleShot",
     TRUE);
eError = DAQPilot_Start(hDAQPilotTask);
//Get the data and set timeout as 10 seconds
     (10000 ms).
eError = DAQPilot_GetAIWaveform(hDAQPilotTask,
     &DAQPilotData, 10000);
if( eError==DP_STATUS_NOERROR )
{
//Display caption
printf("No");
for(k=0; k < DAQPilotData.nNumOfChannel; k++)
printf("Channel %d",
     DAQPilotData.lpChannelNoList[k]);
printf("\n");
for(i=0; i < DAQPilotData.dwNumOfScan; i++)
{
printf("%d", i);
for(j=0; j < DAQPilotData.nNumOfChannel; j++)
{
printf("%f",
     ((double*)DAQPilotData.lpScaledData)[(i*DAQ
     PilotData.nNumOfChannel)+j]);
}
printf("\n");
}
}
DAQPilot_EndTask(hDAQPilotTask);
```

5.18 DAQPilot_AO_WriteChannel

Outputs an analog value to a specific channel.

```
DAQPilotStatusID DAQPilot_AO_WriteChannel (
   HANDLE hTask,
   double dbValue
);
```

Parameters

hTask Task handle

dbValue Output analog value

Return Values

The return value is DAQPilot status ID.

Remarks

This function is available only for the following tasks:

```
DAQPilot_TASK_AO_VOLTAGE_OUTPUT
DAQPilot_TASK_AO_CURRENT_OUTPUT
```

Example

```
HANDLE hDAQPilotTask = NULL;
double dbVoltageValue = 1.0;

//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("AOVoltage",
     TRUE);
//Switch to Channel 0
DAQPilot_EnableSingleChannel(hDAQPilotTask, 0);
DAQPilot_AO_WriteChannel(hDAQPilotTask,
     dbVoltageValue);
DAQPilot_EndTask(hDAQPilotTask);
```

5.19 DAQPilot_AO_WriteChannels

Outputs analog data to multiple channels.

```
DAQPilotStatusID DAQPilot_AO_WriteChannels (
   HANDLE hTask,
   DAQPILOTAnalogData *pstDataInfo
);
```

Parameters

hTask	Task handle
pstDataInfo	A pointer to the DAQPILOTAnalogData structure memory space for data to output.

Return Values

The return value is DAQPilot status ID.

Remarks

This function is available only for the following tasks:

```
DAQPilot_TASK_AO_VOLTAGE_OUTPUT
DAQPilot_TASK_AO_CURRENT_OUTPUT
```

Example

```
HANDLE hDAQPilotTask = NULL;
DAQPilotAnalogData DAQPilotData;
DAQPilotStatusID eError = DP_STATUS_NOERROR;
int i=0;
float fVoltageValue = 0;

//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("AOVoltage",
     TRUE);
//Call DAQPilot_GetAOBuffer() function to get the
     default DAQPilotAnalogData.
eError = DAQPilot_GetAOBuffer(hDAQPilotTask,
     &DAQPilotData, 0);
if( eError==DP_STATUS_NOERROR )
{
//fill AO Buffer
for(i=0; i < DAQPilotData.nNumOfChannel; i++)
{
printf("Channel %d, Voltage value=",
     DAQPilotData.lpChannelNoList[i]);
scanf("%f", &fVoltageValue);
DAQPilotData.lpScaledData[i] = fVoltageValue;
printf("\n");
}
}
eError = DAQPilot_AO_WriteChannels(hDAQPilotTask,
     &DAQPilotData);
DAQPilot_EndTask(hDAQPilotTask);
```

5.20 DAQPilot_GetAOBuffer

Obtains a data buffer for analog output and performs analog output to single channel or multiple channels. Output mode may be in one-shot or continuous mode.

```
DAQPilotStatusID DAQPilot_GetAOBuffer (
   HANDLE hTask,
   DAQPILOTAnalogData *pstDataInfo,
   DWORD dwTimeOut
);
```

Parameters

hTask Task handle

pstDataInfo Pointer to the DAQPILOTAnalogData structure. The analog data will output to multiple channels.

dwTimeOut The timeout interval in milliseconds. If dwTimeOut is 0, the function tests the object's state, then returns immediately. If dwTimeOut is INFINITE, the function continues until data acquisition is completed.

Return Values

The return value is the DAQPilot status ID.

Remarks

1. This function is available only for the following tasks:

   ```
   DAQPilot_TASK_AO_VOLTAGE_OUTPUT
   DAQPilot_TASK_AO_CURRENT_OUTPUT
   DAQPilot_TASK_AO_ONESHOT
   DAQPilot_TASK_AO_CONTINUE
   ```

2. Without this function, you need to manually allocate memory space to the DAQPILOTAnalogData structure.

Example

```
HANDLE hDAQPilotTask = NULL;
DAQPilotAnalogData DAQPilotData;
DAQPilotStatusID eError = DP_STATUS_NOERROR;
int i=0;
float fVoltageValue = 0;

//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("AOVoltage",
     TRUE);
//Call DAQPilot_GetAOBuffer() function to get the
     default DAQPilotAnalogData.
eError = DAQPilot_GetAOBuffer(hDAQPilotTask,
     &DAQPilotData, 0);
if( eError==DP_STATUS_NOERROR )
{
//fill AO Buffer
for(i=0; i < DAQPilotData.nNumOfChannel; i++)
{
printf("Channel %d, Voltage value=",
     DAQPilotData.lpChannelNoList[i]);
scanf("%f", &fVoltageValue);
DAQPilotData.lpScaledData[i] = fVoltageValue;
printf("\n");
}
}
eError = DAQPilot_AO_WriteChannels(hDAQPilotTask,
     &DAQPilotData);
DAQPilot_EndTask(hDAQPilotTask);
```

5.21 DAQPilot_SetAOWaveform

Performs an analog waveform output for single channel or multiple channels. Output mode may be in one-shot or continuous mode.

```
DAQPilotStatusID DAQPilot_SetAOWaveform (
   HANDLE hTask,
   DAQPILOTAnalogData *pstDataInfo
);
```

Parameters

hTask Task handle

pstDataInfo Pointer to the DAQPILOTAnalogData structure memory space, where analog data is used to output.

Return Values

The return value is DAQPilot status ID.

Remarks

1. This function is available only for the following tasks:

   ```
   DAQPilot_TASK_AO_ONESHOT
   DAQPilot_TASK_AO_CONTINUE
   ```

2. Use DAQPilot_Start() to output the waveform data.

Example

```
HANDLE hDAQPilotTask = NULL;
DAQPilotAnalogData DAQPilotData;
DAQPilotStatusID eStatus = DP_STATUS_NOERROR;
DAQPilotAnalogData stGetDataInfo;
double dbAmplitude = 3;
DWORD  i=0, j=0, k=0;
DWORD dwCyclEnumOfSamples = 0;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("AOSingleShot
     ", TRUE);
//Configure parameters and build default AO
     buffer
eStatus = DAQPilot_Config(hTask);
//Get default AO Buffer info
DAQPilot_GetAOBuffer(hTask, &stGetDataInfo, 0);
dwCyclEnumOfSamples = stGetDataInfo.dwNumOfScan;
for(i=0; i < stGetDataInfo.dwNumOfScan ; i++)
{
for(j=0; j<(DWORD)stGetDataInfo.nNumOfChannel;
     j++)
{
k = (i*stGetDataInfo.nNumOfChannel) + j;
if(k % stGetDataInfo.nNumOfChannel)
stGetDataInfo.lpScaledData[k] = (-1 *
     dbAmplitude) + ((2 * i * dbAmplitude)/
     dwCyclEnumOfSamples);
else
stGetDataInfo.lpScaledData[k] = (sin( ((double)i/
     (double)dwCyclEnumOfSamples) * 2 * 3.14159)
     * dbAmplitude);
}
}
//Put AO Buffer to DAQPilot driver
DAQPilot_SetAOWaveform(hTask, &stGetDataInfo);
eStatus = DAQPilot_Start(hDAQPilotTask);
DAQPilot_EndTask(hDAQPilotTask);
```

5.22 DAQPilot_DI_ReadPort

Acquires the digital value from a digital port.

```
DAQPilotStatusID DAQPilot_DI_ReadPort (
   HANDLE hTask,
   DWORD *pdwValue
);
```

Parameters

hTask Task handle

pdwValue Pointer to the DWORD memory space for acquiring digital port.

Return Values

The return value is the DAQPilot status ID.

Remarks

This function is available only for:

```
DAQPilot_TASK_DI_PORT_INPUT
```

Example

```
HANDLE hDAQPilotTask = NULL;
DWORD dwValue = FALSE;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("DIPortInput",
     TRUE);
//Switch to Channel 0
DAQPilot_EnableSingleChannel(hDAQPilotTask, 0);
DAQPilot_DI_ReadPort(hDAQPilotTask, &dwValue);
//Now the dwValue contains port value of input
     signal.
printf("Port value=%u\n", dwValue);
DAQPilot_EndTask(hDAQPilotTask);
```

5.23 DAQPilot_DI_ReadLine

Acquires a digital value from a digital line.

```
DAQPilotStatusID DAQPilot_DI_ReadLine (
   HANDLE hTask,
   BOOL *pbValue
);
```

Parameters

hTask Task handle

pbValue Pointer to the Boolean memory space for acquiring digital line.

Return Values

The return value is the DAQPilot status ID.

Remarks

This function is available only for:

```
DAQPilot_TASK_DI_LINE_INPUT
```

Example

```
HANDLE hDAQPilotTask = NULL;
BOOL bValue = FALSE;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("DILineInput",
     TRUE);
//Switch to Channel 0
DAQPilot_EnableSingleChannel(hDAQPilotTask, 0);
DAQPilot_DI_ReadLine(hDAQPilotTask, &bValue);
//Now the bValue contains line value of input
     signal.
printf("Line
     value=%s\n",(bValue?"TRUE":"FALSE"));
DAQPilot_EndTask(hDAQPilotTask);
```

5.24 DAQPilot_DI_ReadChannels

Acquires digital values from multiple channels.

```
DAQPilotStatusID DAQPilot_DI_ReadChannels (
   HANDLE hTask,
   DAQPilotDigitalData *pstDataInfo
);
```

Parameters

hTask Task handle

pstDataInfo Pointer to the DAQPilotDigitalData structure memory space that will receive the digital data.

Return Values

The return value is the DAQPilot status ID.

Remarks

This function is available only for the following tasks:

```
DAQPilot_TASK_DI_LINE_INPUT
DAQPilot_TASK_DI_PORT_INPUT
```

Example

```
HANDLE hDAQPilotTask = NULL;
DAQPilotDigitalData DAQPilotData;
DAQPilotStatusID eError = DP_STATUS_NOERROR;
int j=0, k=0;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("DILineInput ",
     TRUE);
//Enable both channel 0 and channel 1.
DAQPilot_SetChannelPropertyBool(hDAQPilotTask,
     0, CP_Enable, TRUE);
DAQPilot_SetChannelPropertyBool(hDAQPilotTask,
     1, CP_Enable, TRUE);
eError = DAQPilot_DI_ReadChannels(hDAQPilotTask,
     &DAQPilotData);
if( eError==DP_STATUS_NOERROR )
{
//Display caption
for(k=0; k < DAQPilotData.nNumOfChannel; k++)
printf("   Line %d",
     DAQPilotData.lpPortOrLineList[k]);
printf("\n");
//Now the DAQPilotData.lpRawData contains the
     data.
//You may process the data here.
for(j=0; j < DAQPilotData.nNumOfChannel; j++)
printf("%s",
     ((LPBYTE)DAQPilotData.lpRawData)[j]? "TRUE"
     : "FALSE");
printf("\n");
}
DAQPilot_EndTask(hDAQPilotTask);
```

5.25 DAQPilot_GetDIPattern

Acquires a digital pattern from a digital port. This performs a one-shot or a continuous digital data acquisition.

```
DAQPilotStatusID DAQPilot_GetDIPattern (
   HANDLE hTask,
   DAQPILOTDigitalData *pstDataInfo,
   DWORD dwTimeOut
);
```

Parameters

hTask Task handle

pstDataInfo Pointer to the DAQPilotDigitalData structure memory space that will receive the digital data.

dwTimeOut The timeout interval in milliseconds. If dwTimeOut is 0, the function tests the object's state, then returns immediately. If dwTimeOut is INFINITE, the function continues until data acquisition is completed.

Return Values

The return value may be a status, a warning, or an error.

Remarks

This function is available only for the following tasks:

```
DAQPilot_TASK_DI_ONESHOT
DAQPilot_TASK_DI_CONTINUE
```

Example

```
HANDLE hDAQPilotTask = NULL;
DAQPilotAnalogData DAQPilotData;
DAQPilotStatusID eError = DP_STATUS_NOERROR;
int j=0, k=0;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("DISingleShot",
     TRUE);
eError = DAQPilot_Start(hDAQPilotTask);
//Get the data and set timeout as 10 seconds
     (10000 ms).
eError = DAQPilot_GetDIPattern(hDAQPilotTask,
     &DAQPilotData, 10000);
if( eError==DP_STATUS_NOERROR )
```

```
{
//Display caption
nNumOfPort = DAQPilotData.nNumOfChannel;
printf("No");
for(k=0; k < nNumOfPort; k++)
printf("Port %d",
     DAQPilotData.lpPortOrLineList[k]);
printf("\n");
//Now the DAQPilotData. lpRawData contains the
     digital data.
//You may process the data here.
for(i=0; i < DAQPilotData.dwNumOfScan; i++)
{
printf("%d", i);
for(j=0; j < nNumOfPort; j++)
{
k = (i*nNumOfPort) + j;
switch(DAQPilotData.nRawDataBits)
{
case 8:
printf("%c",
     ((BYTE*)DAQPilotData.lpRawData)[k]);
break;
case 16:
printf("%u",
     ((WORD*)DAQPilotData.lpRawData)[k]);
break;
case 32:
printf("%lu",
     ((DWORD*)DAQPilotData.lpRawData)[k]);
break;
}
}
printf("\n");
}
}
DAQPilot_EndTask(hDAQPilotTask);
```

5.26 DAQPilot_DO_WritePort

Outputs a digital value to a digital port.

```
DAQPilotStatusID DAQPilot_DO_WritePort (
   HANDLE hTask,
   DWORD dwValue
);
```

Parameters

hTask Task handle

dwValue The digital port value to output.

Return Values

The return value is DAQPilot status ID.

Remarks

This function is available only for:

```
DAQPilot_TASK_DO_PORT_OUTPUT
```

Example

```
HANDLE hDAQPilotTask = NULL;
DWORD dwValue = 1;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("DOPortOutput",
     TRUE);
//Switch to Channel 0
DAQPilot_EnableSingleChannel(hDAQPilotTask, 0);
DAQPilot_DO_WritePort(hDAQPilotTask, dwValue);
DAQPilot_EndTask(hDAQPilotTask);
```

5.27 DAQPilot_DO_WriteLine

Outputs a digital value to a digital line.

```
DAQPilotStatusID DAQPilot_DO_WriteLine (
   HANDLE hTask,
   BOOL bValue
);
```

Parameters

hTask Task handle

bValue Digital line value (Bollean) to output

Return Values

The return value is the DAQPilot status ID.

Remarks

This function is available only for:

```
DAQPilot_TASK_DO_LINE_OUTPUT
```

Example

```
HANDLE hDAQPilotTask = NULL;
BOOL bValue = TRUE;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("DOLineOutput",
     TRUE);
//Switch to Channel 0
DAQPilot_EnableSingleChannel(hDAQPilotTask, 0);
DAQPilot_DO_WriteLine(hDAQPilotTask,
     bValue);DAQPilot_EndTask(hDAQPilotTask);
```

5.28 DAQPilot_DO_WriteChannels

Outputs digital values to multiple channels.

```
DAQPilotStatusID DAQPilot_DO_WriteChannels (
   HANDLE hTask,
   DAQPilotDigitalData *pstDataInfo
);
```

Parameters

hTask Task handle

pstDataInfo Pointer to the DAQPilotDigitalData structure that will output the digital data to multiple channels.

Return Values

The return value is DAQPilot status ID.

Remarks

1. This function is available only for the following tasks:

```
DAQPilot_TASK_DO_LINE_OUTPUT
DAQPilot_TASK_DO_PORT_OUTPUT
```

Example

```
HANDLE hDAQPilotTask = NULL;
DAQPilotDigitalData DAQPilotData;
DAQPilotStatusID eError = DP_STATUS_NOERROR;
int nValue = 0;
int i=0;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("DOPortOutput",
     TRUE);
//Enable both channel 0 and channel 1.
DAQPilot_SetChannelPropertyBool(hDAQPilotTask,
     0, CP_Enable, TRUE);
DAQPilot_SetChannelPropertyBool(hDAQPilotTask,
     1, CP_Enable, TRUE);
//Call DAQPilot_GetDOBuffer() function to get the
     default DAQPilotDigitalData.
eError = DAQPilot_GetDOBuffer(hDAQPilotTask,
     &DAQPilotData, 0);
if( eError==DP_STATUS_NOERROR )
{
//fill DO Buffer
```

```
for(i=0; i < DAQPilotData.nNumOfChannel; i++)
{
printf("Port %d, output value=",
     DAQPilotData.lpPortOrLineList[i]);
scanf("%d", &nValue);
switch(DAQPilotData.nRawDataBits)
{
case 8:
((BYTE*)DAQPilotData.lpRawData)[i]  =
     (BYTE)nValue;
break;
case 16:
((WORD*)DAQPilotData.lpRawData)[i]  =
     (WORD)nValue;
break;
case 32:
((DWORD*)DAQPilotData.lpRawData)[i] =
     (DWORD)nValue;
break;
}
printf("\n");
}
}
eError = DAQPilot_DO_WriteChannels(hDAQPilotTask,
     &DAQPilotData);
DAQPilot_EndTask(hDAQPilotTask);
```

5.29 DAQPilot_DO_ReadBackPort

Reads the current port value from digital output port.

```
DAQPilotStatusID DAQPilot_DO_ReadBackPort (
   HANDLE hTask,
   DWORD *pdwValue
);
```

Parameters

hTask Task handle

pdwValue The pointer to the DWORD memory space to save the digital port value.

Return Values

The return value is the DAQPilot status ID.

Remarks

This function is available only for:

```
DAQPilot_TASK_DO_PORT_OUTPUT
```

Example

```
HANDLE hDAQPilotTask = NULL;
DWORD dwValue = 0;

//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("DOPortOutput",
     TRUE);
//Switch to Channel 0
DAQPilot_EnableSingleChannel(hDAQPilotTask, 0);
DAQPilot_DO_ReadBackPort(hDAQPilotTask,
     &dwValue);
//Now the dwValue contains value of digital
     output port.
printf("Port value=%u\n", dwValue);
DAQPilot_EndTask(hDAQPilotTask);
```

5.30 DAQPilot_DO_ReadBackLine

Reads the current line value from digital output line.

```
DAQPilotStatusID DAQPilot_DO_ReadBackLine (
   HANDLE hTask,
   BOOL *pbValue
);
```

Parameters

hTask Task handle

pbValue The pointer to the Boolean memory space to save the digital output line.

Return Values

The return value is the DAQPilot status ID.

Remarks

This function is available only for:

```
DAQPilot_TASK_DO_LINE_OUTPUT
```

Example

```
HANDLE hDAQPilotTask = NULL;
BOOL bValue = FALSE;

//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("DOLineOutput",
     TRUE);
//Switch to Channel 0
DAQPilot_EnableSingleChannel(hDAQPilotTask, 0);
DAQPilot_DO_ReadBackLine (hDAQPilotTask,
     &bValue);
//Now the bValue contains value of digital output
     line.
printf("Line
     value=%s\n",(bValue?"TRUE":"FALSE"));
     DAQPilot_EndTask(hDAQPilotTask);
```

5.31 DAQPilot_DO_ReadBackChannels

Reads the counter values from multiple counters.

```
DAQPilotStatusID DAQPilot_DO_ReadBackChannels (
   HANDLE hTask,
   DAQPilotDigitalData *pstDataInfo
);
```

Parameters

hTask Task handle

pstDataInfo The pointer to the DAQPilotDigitalData structure memory space to save the digital data.

Return Values

The return value is DAQPilot status ID.

Remarks

This function is available only for these tasks:

```
DAQPilot_TASK_DO_LINE_OUTPUT
DAQPilot_TASK_DO_PORT_OUTPUT
```

Example

```
HANDLE hDAQPilotTask = NULL;
DAQPilotDigitalData DAQPilotData;
DAQPilotStatusID eError = DP_STATUS_NOERROR;
int  j=0, k=0;

//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("DOPortOutput",
     TRUE);
//Enable channel 0 and channel 1
     DAQPilot_SetChannelPropertyBool(hDAQPilotTa
     sk, 0, CP_Enable, TRUE);
DAQPilot_SetChannelPropertyBool(hDAQPilotTask,
     1, CP_Enable, TRUE);

eError = DAQPilot_DO_ReadBackChannels
     (hDAQPilotTask, &DAQPilotData);
if( eError==DP_STATUS_NOERROR )
{
//Display caption
for(k=0; k < DAQPilotData.nNumOfChannel; k++)
```

```
printf("Port %d",
     DAQPilotData.lpPortOrLineList[k]);
printf("\n");
/Now the DAQPilotData.lpRawData contains the
     data.
//You may process the data here.
for(j=0; j < DAQPilotData.nNumOfChannel; j++)
{
switch(DAQPilotData.nRawDataBits)
{
case 8:
printf("%c",
     ((BYTE*)DAQPilotData.lpRawData)[j]);
break;
case 16:
printf("%u",
     ((WORD*)DAQPilotData.lpRawData)[j]);
break;
case 32:
printf("%lu",
     ((DWORD*)DAQPilotData.lpRawData)[j]);
break;
}
}
printf("\n");
}
DAQPilot_EndTask(hDAQPilotTask);
```

5.32 DAQPilot_GetDOBuffer

Prepare a buffer for digital output. The space is used to perform a digital pattern output includes one-shot and continuous acquisition mode.

```
DAQPilotStatusID DAQPilot_GetDOBuffer (
   HANDLE hTask,
   DAQPILOTDigitalData *pstDataInfo,
   DWORD dwTimeOut
);
```

Parameters

hTask Task handle

pstDataInfo The pointer to the DAQPILOTDigitalData structure memory space for data to output

dwTimeOut The time-out interval (milliseconds). If dwTimeOut is 0 the function tests the object's state and returns immediately. If dwTimeOut is INFINITE, the function will keep running until data acquisition completed.

Return Values

The return value is DAQPilot status ID.

Remarks

1. This function is available only for the following tasks:

   ```
   DAQPilot_TASK_DO_LINE_OUTPUT
   DAQPilot_TASK_DO_PORT_OUTPUT
   DAQPilot_TASK_DO_ONESHOT
   DAQPilot_TASK_DO_CONTINUE
   ```

2. Without this function, user needs to prepare the DAQPILOTDigitalData data.

Example

```
HANDLE hDAQPilotTask = NULL;
DAQPilotDigitalData DAQPilotData;
DAQPilotStatusID eError = DP_STATUS_NOERROR;
int i=0;
int nValue = 0;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("DOLineOutput",
     TRUE);
//Call DAQPilot_GetDOBuffer() function to get the
     default DAQPilotDigitalData.
eError = DAQPilot_GetDOBuffer(hDAQPilotTask,
     &DAQPilotData, 0);
if( eError==DP_STATUS_NOERROR )
{
//fill DO Buffer
for(i=0; i < DAQPilotData.nNumOfChannel; i++)
{
printf("Line %d, output value=",
     DAQPilotData.lpPortOrLineList[i]);
scanf("%d", &nValue);
DAQPilotData. lpRawData[i] = nValue;
printf("\n");
}
}
eError = DAQPilot_DO_WriteChannels(hDAQPilotTask,
     &DAQPilotData);
DAQPilot_EndTask(hDAQPilotTask);
```

5.33 DAQPilot_SetDOPattern

Prepares the output pattern for one-shot and continuous DO modes.

```
DAQPilotStatusID DAQPilot_SetDOPattern (
   HANDLE hTask,
   DAQPILOTDigitalData *pstDataInfo
);
```

Parameters

hTask Task handle

pstDataInfo The pointer to the DAQPILOTDigitalData structure memory space for data to output.

Return Values

The return value is DAQPilot status ID.

Remarks

1. This function is available only for the following tasks:

   ```
   DAQPilot_TASK_DO_ONESHOT
   DAQPilot_TASK_DO_CONTINUE
   ```

2. Use DAQPilot_SetDOPattern to prepare the output pattern. Then use DAQPilot_Start() to output the data.

Example

```
DAQPilotDigitalData DAQPilotData;
DAQPilotStatusID eError = DP_STATUS_NOERROR;
DWORD dwCyclEnumOfSamples = 0;
DWORD dwNumOfPort = 0;
double dbAmplitude = 1;
DWORD  i=0, j=0, k=0;
//Load your DAQPilot task
hDAQPilotTask = DAQPilot_LoadTask("DOSingleShot",
     TRUE);
//Configure parameters and build default DO
     buffer
eError = DAQPilot_Config(hDAQPilotTask);
//Get default DO Buffer info
DAQPilot_GetDOBuffer(hDAQPilotTask,
     &DAQPilotData, 0);
dwCyclEnumOfSamples = DAQPilotData.dwNumOfScan;
```

```
dwNumOfPort=DAQPilotData.nNumOfChanne;
for(i=0; i < DAQPilotData.dwNumOfScan ; i++)
{
for(j=0; j<(DWORD)dwNumOfPort; j++)
{
k = (i*dwNumOfPort) + j;
switch(DAQPilotData.nRawDataBits)
{
case 8:
((BYTE*)DAQPilotData.lpRawData)[k] =
     (BYTE)(sin(((double)i/
     (double)dwCyclEnumOfSamples)*2*3.14159)*dbA
     mplitude);
break;
case 16:
((WORD*)DAQPilotData.lpRawData)[k] = (WORD)(sin(
     ((double)i/(double)dwCyclEnumOfSamples) * 2
     * 3.14159) * dbAmplitude);
break;
case 32:
((DWORD*)DAQPilotData.lpRawData)[k] =
     (DWORD)(sin( ((double)i/
     (double)dwCyclEnumOfSamples) * 2 * 3.14159)
     * dbAmplitude);
break;
}
}
}
//Put AO Buffer to DAQPilot driver
DAQPilot_SetDOPattern(hTask, DAQPilotData);
eError = DAQPilot_Start(hDAQPilotTask);
DAQPilot_EndTask(hDAQPilotTask);
```

5.34 DAQPilot_TC_GetValue

Gets a counter value. DAQPilot comes with two counter modes: **Simple Counter** or **Mode Operation**.

```
DAQPilotStatusID DAQPilot_TC_GetValue (
   HANDLE hTask,
   DWORD *pdwValue
);
```

Parameters

hTask Task handle

pdwValue The pointer to a WORD memory space to save the counter value.

Return Values

The return value is the DAQPilot status ID.

Remarks

1. This function is available only for the following tasks:

   ```
   DAQPilot_TASK_TC_COUNTER
   DAQPilot_TASK_TC_MODE_OPERATION
   ```

2. Default counter bit width is 16. But DAQPilot "Simple Counter" mode can count to 32-bits width by software. Therefore, if the application's sampling rate is higher than 100Hz, we suggest user to use "Mode Operation" instead.

Example

```
HANDLE hDAQPilotTask = NULL;
DWORD dwValue = FALSE;
//Load your DAQPilot task
hDAQPilotTask =
     DAQPilot_LoadTask("TCSimpleCounter", TRUE);
//Switch to Channel 0
DAQPilot_EnableSingleChannel(hDAQPilotTask, 0);
DAQPilot_Start(hDAQPilotTask);
DAQPilot_TC_GetValue (hDAQPilotTask, &dwValue);
//Now the dwValue contains current counter value.
printf("Counter value=%u\n", dwValue);
DAQPilot_Stop(hDAQPilotTask);
DAQPilot_EndTask(hDAQPilotTask);
```

5.35 DAQPilot_GetNotifyEvent

Gets the notification event. This function is for DAQPilot's **Timer Interrupt** function mode.

```
HANDLE DAQPilot_GetNotifyEvent (
   HANDLE hTask
);
```

Parameters

hTask Task handle

Return Values

Returns the notification handle object of event.

Remarks

This function is available only for:

```
DAQPilot_TASK_TC_TIMER_INTERRUPT
```

Example

```
HANDLE hDAQPilotTask = NULL;
//Load your DAQPilot task
hDAQPilotTask =
     DAQPilot_LoadTask("TCTimerInterrupt ",
     TRUE);
DAQPilot_Start(hDAQPilotTask);
//Get DAQPilot notify event
hNotifyObject =
     DAQPilot_GetNotifyEvent(hDAQPilotTask);
do
{
dwReturn =
     WaitForSingleObject(hNotifyObject,10000);
if(dwReturn == WAIT_OBJECT_0)
printf("Get timer interrupt!\n");
else
break;
} while(TRUE);
DAQPilot_Stop(hDAQPilotTask);
DAQPilot_EndTask(hDAQPilotTask);
```

5.36 DAQPilot_TC_ReadCounters

Reads the counter values from multiple counters. This function performs the DAQPilot's **Mode Operation** mode.

```
DAQPilotStatusID DAQPilot_TC_ReadCounters (
   HANDLE hTask,
   DAQPilotDigitalData *pstDataInfo
);
```

Parameters

hTask	Task handle
pstDataInfo	Pointer to the DAQPilotDigitalData structure memory space to save the counter data.

Return Values

The return value is the DAQPilot status ID.

Remark

This function is available only for:

```
DAQPilot_TASK_TC_MODE_OPERATION
```

Example

```
HANDLE hDAQPilotTask = NULL;
DAQPilotDigitalData DAQPilotData;
DAQPilotStatusID eError = DP_STATUS_NOERROR;
int j=0, k=0;
//Load your DAQPilot task
hDAQPilotTask =
     DAQPilot_LoadTask("TCModeOperation", TRUE);
//Enable channel 0
DAQPilot_SetChannelPropertyBool(hDAQPilotTask,
     0, CP_Enable, TRUE);
DAQPilot_Start(hDAQPilotTask);
eError = DAQPilot_TC_ReadCounters(hDAQPilotTask,
     &DAQPilotData);
if( eError==DP_STATUS_NOERROR )
{
//Display caption
for(k=0; k < DAQPilotData.nNumOfChannel; k++)
printf("Counter %d",
     DAQPilotData.lpPortOrLineList[k]);
printf("\n");
```

```
//Now the DAQPilotData.lpRawData contains the
     data.
//You may process the data here.
for(j=0; j < DAQPilotData.nNumOfChannel; j++)
{
switch(DAQPilotData.nRawDataBits)
{
case 16:
printf("%u",
     ((WORD*)DAQPilotData.lpRawData)[j]);
break;
case 32:
printf("%lu",
     ((DWORD*)DAQPilotData.lpRawData)[j]);
break;
}
}
printf("\n");
}
DAQPilot_Stop(hDAQPilotTask);
DAQPilot_EndTask(hDAQPilotTask);
```

6 ActiveX Controls and .NET Component Function Reference

This section lists all DAQPilot ActiveX controls and .NET components for your reference. Unless otherwise specified, the following programming applications are abbreviated for brevity:

Programming application	Abbreviation
Microsoft Visual Basic	VB
Microsoft Visual C++	VC++
Microsoft Visual Basic .NET	VB.NET
Microsoft Visual C# .NET	VC#
Borland C++ Builder	BCB
Borland Delphi	Delphi

6.1 AutoSize Property

Returns a Boolean value to indicate whether or not the control will automatically resize to fit the original bitmap resolution.

Data Type

VB	`Boolean`
VC++	`BOOL`
BCB	`bool`
Delphi	`WordBool`

Syntax

VB	`Dim bAutoSize As Boolean`
	`bAutoSize = object.AutoSize  'get AutoSize`
	`object.AutoSize = bAutoSize  'set AutoSize`

VC++	`BOOL bAutoSize;`
	`bAutoSize = object. GetAutoSize();  //get AutoSize`
	`object. SetAutoSize(bAutoSize);  //set AutoSize`

BCB	`bool bAutoSize;`
	`bAutoSize = object->AutoSize;  //get AutoSize`
	`object->AutoSize = bAutoSize;  //set AutoSize`

Delphi	`bAutoSize : WordBool;`
	`bAutoSize := object. AutoSize;  //get AutoSize`
	`object. AutoSize := bAutoSize;  //set AutoSize`

Settings

Value	Description
`True`	The control automatically resizes to fit original bitmap resolution. (Default)
`False`	User can set the width and height at design time.

6.2 EnabledChNumList Property

Gets the enabled channels list.

Data Type

VB	`Variant (Integer array)`
VC++	`VARIANT (short array)`
BCB	`TVariant (short array)`
Delphi	`OleVariant (SmallInt array)`
VB.NET	`Object(short array)`
VC#	`Array(short array)`

Syntax

VB	`Dim vEnabledChNumList As Variant` `vEnabledChNumList = object. EnabledChNumList  'get EnabledChNumList`

VC++	`VARIANT vEnabledChNumList;`
	`vEnabledChNumList = object.GetEnabledChNumList()`
	`'get EnabledChNumList`

BCB	`TVariant vEnabledChNumList;` `vEnabledChNumList = object->EnabledChNumList  'get EnabledChNumList`
Delphi	`vEnabledChNumList : OleVariant;` `vEnabledChNumList = object. EnabledChNumList  'get EnabledChNumList`
VB.NET	`Dim vEnabledChNumList As object` `vEnabledChNumList = object. EnabledChNumList  'get EnabledChNumList`
VC#	`Short[ ] vEnabledChNumList;` `vEnabledChNumList = object.EnabledChNumList()  'get EnabledChNumList`

Sample

```
Dim vChNumList As Variant
'Get channel no list
vChNumList = DAQPilot1.EnabledChNumList
For i = 0 To UBound(vChNumList)
        nEnabledChNo = vChNumList (i)
Next i
```

6.3 MultiThread Property

Gets or sets a value that determines whether or not to use multi-thread to trigger event.

Data Type

VB	`Boolean`
VC++	`BOOL`
BCB	`bool`
Delphi	`WordBool`
BCB	`Boolean`
Delphi	`bool`

Syntax

VB	`Dim bMultiThread As Boolean`
	`bMultiThread = object.MultiThread  'get MultiThread`
	`object.MultiThread = bMultiThread  'set MultiThread`

VC++	`BOOL bMultiThread;`
	`bMultiThread = object. GetMultiThread();  //get MultiThread`
	`object. SetMultiThread(bMultiThread);  //set MultiThread`

BCB	`bool bMultiThread;`
	`bMultiThread = object->MultiThread;  //get MultiThread`
	`object->MultiThread = bMultiThread;  //set MultiThread`

Delphi	`bMultiThread : WordBool;`
	`bMultiThread := object. MultiThread;  //get MultiThread`
	`object. MultiThread := bMultiThread;  //set MultiThread`

VB.NET	`Dim bMultiThread As Boolean`
	`bMultiThread = object.MultiThread  'get MultiThread`
	`object.MultiThread = bMultiThread  'set MultiThread`

VC#	`bool bMultiThread;`
	`bMultiThread = object. GetMultiThread();  //get MultiThread`
	`object. SetMultiThread(bMultiThread);  //set MultiThread`

Settings

Value	Description
`True`	Use multiple threads to trigger event.
`False`	Use single-thread (main thread) to trigger event. (Default)

Remarks

1. When accessing data, multi-threaded applications require rigid handling than single-threaded applications. Since there are multiple, independent paths of execution used simultaneously in a multi-threaded application, the MFC and Windows UI objects are not thread-safe at the object level for size and performance reasons. This means that you can have two separate threads manipulating two different MFC objects, but not two threads manipulating the same MFC object.
2. To use the DAQPilot ActiveX control, set MultiThread Property to False. There is no more need to look after complex multi-threading programming with critical section mechanism.
3. Some IDE, such as in VB6 and VEE, cannot handle multiple threaded components properly in terms of memory allocation while using these continuous I/O functions:

   ```
   DAQPilot_TASK_AI_CONTINUE
   DAQPilot_TASK_AO_CONTINUE
   DAQPilot_TASK_DI_CONTINUE
   DAQPilot_TASK_DO_CONTINUE
   ```

4. If you are a multiple thread program expert, enable the MultiThread property to enhance task performance. However, provide access to the same object with appropriate Win32 synchronization mechanisms.

6.4 ShowErrorMessage Property

A flag that determines whether or not to report an error message.

Data Type

VB	`Boolean`
VC++	`BOOL`
BCB	`bool`
Delphi	`WordBool`
BCB	`Boolean`
Delphi	`bool`

Syntax

VB	`Dim bShowErrorMessage As Boolean`
	`bShowErrorMessage = object.ShowErrorMessage 'get ShowErrorMessage`
	`object.ShowErrorMessage = bShowErrorMessage 'set ShowErrorMessage`

VC++	`BOOL bShowErrorMessage;`
	`bShowErrorMessage = object. GetShowErrorMessage();`
	`//get ShowErrorMessage`
	`object. SetShowErrorMessage(bShowErrorMessage);`
	`//set ShowErrorMessage`

BCB	`bool bShowErrorMessage;`
	`bShowErrorMessage = object->ShowErrorMessage; //get ShowErrorMessage`
	`object->ShowErrorMessage = bShowErrorMessage; //set ShowErrorMessage`

Delphi	`bShowErrorMessage : WordBool;`
	`bShowErrorMessage := object. ShowErrorMessage; //get ShowErrorMessage`
	`object. ShowErrorMessage := bShowErrorMessage; //set ShowErrorMessage`

VB.NET	`Dim bShowErrorMessage As Boolean`
	`bShowErrorMessage = object.ShowErrorMessage 'get ShowErrorMessage`
	`object.ShowErrorMessage = bShowErrorMessage 'set ShowErrorMessage`

VC#	`bool bShowErrorMessage;`
	`bShowErrorMessage = object. GetShowErrorMessage();`
	`//get ShowErrorMessage`
	`object. SetShowErrorMessage(bShowErrorMessage);`
	`//set ShowErrorMessage`

Settings

Value	Description
`True`	The detailed error message is reported by message box.
`False`	No detailed error message is reported. (Default)

Remarks

Regardless of the ShowErrorMessage status, the component triggers a DAQPilotError event when the function fails.

6.5 Status Property

Obtains the status of a specific DAQ task.

Data Type

VB	`Long`
VC++	`long`
BCB	`long`
Delphi	`LongInt`
BCB	`Integer`
Delphi	`Int`

Syntax

VB	`Dim lStatus As Long` `lStatus = object.Status  'get Status`
VC++	`long lStatus;` `lStatus = object. GetStatus();  //get Status`
BCB	`long lStatus;` `lStatus = object->Status;  //get Status`
Delphi	`lStatus : LongInt;` `object. Status := lStatus;  //get Status`
VB.NET	`Dim lStatus As Integer` `lStatus = object.Status  'get Status`
VC#	`Int lStatus;` `lStatus = object. GetStatus();  //get Status`

Return Values

`DP_STATUS_DEVICE_STOP_AND_DIRTY(1)`

The device stopped. You may call Start() to restart the task, Start() automatically calls Config() method before running. This is the default return value.

`DP_STATUS_DEVICE_STOP(2)`

The device stopped. You may call Start() to restart the task but the Start() method does not automatically call Config() method before running.

`DP_STATUS_DEVICE_RUNING(3)`

The device is running.

6.6 UnsignedToSigned Property

This flag determines whether or not to convert an unsigned value.

Data Type

VB	`Boolean`
VC++	`BOOL`
BCB	`bool`
Delphi	`WordBool`
BCB	`Boolean`
Delphi	`bool`

Syntax

VB	`Dim bUnsignedToSigned As Boolean`
	`bUnsignedToSigned = object.UnsignedToSigned  'get UnsignedToSigned`
	`object.UnsignedToSigned = bUnsignedToSigned  'set UnsignedToSigned`

VC++	`BOOL bUnsignedToSigned;`
	`bUnsignedToSigned = object. GetUnsignedToSigned();`
	`//get UnsignedToSigned`
	`object. SetUnsignedToSigned(bUnsignedToSigned);  //set UnsignedToSigned`

BCB	`bool bUnsignedToSigned;`
	`bUnsignedToSigned = object->UnsignedToSigned;  //get UnsignedToSigned`
	`object->UnsignedToSigned = bUnsignedToSigned;  //set UnsignedToSigned`

Delphi	`bUnsignedToSigned : WordBool;`
	`object. UnsignedToSigned := bUnsignedToSigned;  //get UnsignedToSigned`
	`bUnsignedToSigned := object. UnsignedToSigned;  //set UnsignedToSigned`

VB.NET	`Dim bUnsignedToSigned As Boolean`
	`bUnsignedToSigned = object.UnsignedToSigned  'get UnsignedToSigned`
	`object.UnsignedToSigned = bUnsignedToSigned  'set UnsignedToSigned`

VC#	`bool bUnsignedToSigned;`
	`bUnsignedToSigned = object. GetUnsignedToSigned();`
	`//get UnsignedToSigned`
	`object. SetUnsignedToSigned(bUnsignedToSigned);  //set UnsignedToSigned`

Settings

Value	Description
`True`	Display tick above graph
`False`	Do not display tick about graph. (Default)

Remarks

This property is intended for languages which do not support unsigned data type, such as VB and VEE. When you use DAQPilot ActiveX component with these IDE tool and gets a “Variant/ Unsupported variant type” error message, set this property to True.

6.7 LoadTask Method

Loads the DAQPilot task specification from the DAQ task file. This function must be called before calling other operations.

Syntax

VB	`Function object.LoadTask( lpszTaskName As String ) As Boolean`
VC++	`BOOL object.GetLoadTask( LPCTSTR lpszTaskName );`
BCB	`bool object->LoadTask( BSTR lpszTaskName );`
Delphi	`function object.LoadTask( const lpszTaskName : WideString ) : WordBool;`
VB.NET	`Function object.LoadTask( lpszTaskName As String ) As Boolean`
VC#	`bool object.LoadTask(string lpszTaskName);`

Arguments

lpszTaskName As String

The task filename to load. This function loads the task file (*.tsk) from the default folder: $INSTALL-DIR\Task Folder\. If DAQPilot does not install in the target folder, you must store the application file and the task file in the same folder.

Return Value

When the function succeeds, the return value is True. If it fails, the return value is False. You may obtain the relative error information when ShowErrorMessage Property is True.

Remarks

1. The lpszTaskName parameter does not need a full filename with extension. For example, AISingleShot represent AISingleShot.tsk.
2. DAQPilot_LoadTask() or DAQPilot_CreateTask() only loads or creates the specified task. If you want to apply the task configuration to the hardware, call DAQPilot_Config() to complete the configuration.
3. Call DAQPilot_Close() to release the DAQPilot task.
4. The order of search for the task file is as follows:
   - ▷ Default task files' folder: $INSTALLDIR\Task Folder\
   - ▷ Current directory

6.8 CreateTask Method

Creates a DAQPilot task with simple parameters.

Syntax

VB	`Function object.CreateTask( nSubTaskID As Long, eDeviceID As Long, nDaskIndex As Long ) As Boolean`
VC++	`BOOL object.GetCreateTask( long nSubTaskID, long eDeviceID, long nDaskIndex );`
BCB	`bool object->CreateTask( long nSubTaskID, long eDeviceID, long nDaskIndex );`
Delphi	`function object.CreateTask( nSubTaskID : LongInt; eDeviceID : LongInt; nDaskIndex : LongInt ) : WordBool;`
VB.NET	`Function object.CreateTask( nSubTaskID As Integer, eDeviceID As Integer, nDaskIndex As Integer ) As Boolean`
VC#	`bool object.GetCreateTask( int nSubTaskID, int eDeviceID, int nDaskIndex );`

Arguments

nSubTaskID As Long

These Sub Task IDs are defined in DAQPilotAdvancedProperties.h. For more information on DAQPilot supported task classes and categories, see section 1.3.

eDeviceID As Long

The type of card that performs the task. The Device IDs are defined in the DAQPilotAdvancedProperties.h.

nDaskIndex As Long

Each installed card has its own sequence number for identification. For similar or same type cards (as defined in argument eDeviceID) or cards that belong to the same series (except PCI- 7300A_Rev.A and PCI-7300A Rev.B), the assigned sequence number depends on the PCI slot order.

For example, if there is a PCI-9111DG card installed in the first PCI slot, one PCI-9111HR card, and two

PCI-9112 cards installed on other PCI slots, the PCI-9111DG will be registered with nBoardIndex 0, and the PCI-9111HR card with nBoardIndex 1.

The PCI-9112 card in the first slot will be registered with nBoardIndex 0 while the second PCI-9112 card will be registered with nBoardIndex 1.

For PCI-7256, PCI-7258, PCI-7260, PCI-7442, PCI-7443, PCI-7444, and PCI-7452 series cards, you may use the onboard switch to set the board index.

Return Value

When the function succeeds, the return value is True. If it fails, the return value is False. You may obtain the relative error information when ShowErrorMessage Property is True.

Remarks

1. This function generates a simple task with default configuration.
2. DAQPilot_LoadTask() or DAQPilot_CreateTask() only loads or creates the specified task. If you want to apply the task configuration to the hardware, call DAQPilot_Config() to complete the configuration.
3. Call DAQPilot_Close() to release the DAQPilot task.
4. You may launch the DAQPilot wizard to know more about function usage.
5. See **Chapter 4: Programming with DAQPilot** for more information.

6.9 EndTask Method

This method is used at the end of a DAQPilot application to release related system memory, threads, and resource.

Syntax

VB	`Sub object.EndTask( )`
VC++	`object.GetEndTask( );`
BCB	`object->EndTask( );`
Delphi	`procedure object.EndTask( );`
VB.NET	`Sub object.EndTask( )`
VC#	`object.EndTask( );`

6.10 SetChannelProperty Method

DAQPilot provides a single method of setting all channel properties. You may directly modify the parameters in a specific task.

Syntax

VB	`Function object.SetChannelProperty( ChannelNum As Long, ChannelProperty As String, Value As Variant ) As Long`
VC++	`long object.GetSetChannelProperty( long ChannelNum , LPCTSTR ChannelProperty, const VARIANT& Value );`
BCB	`long object->SetChannelProperty( long ChannelNum , BSTR ChannelProperty, TVariant Value );`
Delphi	`function object.SetChannelProperty( ChannelNum : LongInt; const ChannelProperty : WideString; Value : OleVariant ) : LongInt;`
VB.NET	`Function object.SetChannelProperty( ChannelNum As Integer, ChannelProperty As String, Value As Object ) As Long`
VC#	`long object.SetChannelProperty( int ChannelNum , string ChannelProperty,object value);`

Arguments

ChannelNum As Long	Channel number
ChannelProperty As String	Channel property string
Value As Variant	Channel property value

Return Value

The return value is the DAQPilot status ID, as defined in the DAQPilotAdvancedProperties.h

Remarks

1. Different devices support different channel parameters.
   - ▷ Edit Control: Data ranges from -1.79769313486232E308 to + 1.79769313486232E308
   - ▷ Combo Box Control: Sets the Enum data
   - ▷ Check Box Control: Sets the Boolean data.
2. You may launch the DAQPilot wizard to know more about function usage.
3. See **Chapter 4: Programming with DAQPilot** for more information.
4. For quick and minor modifications, you may use the DAQPilot_SetChannelProperty to directly configure specific parameters with the appropriate caption in the **Channel configuration** frame. However, it is suggested that you finish the main task using the DAQPilot wizard.

For example, use

```
object.SetSetChannelProperty(0, “Range”,…);
```

to modify the range, and use

```
object.SetSetChannelProperty (0, “RefGround”,…);
```

to assign the reference base.

6.11 GetChannelProperty Method

Gets all channel properties. You may query the parameters from a specific task.

Syntax

VB	`Function object.GetChannelProperty( ChannelNum As Long, ChannelProperty As String ) As Variant`
VC++	`VARIANT object.GetGetChannelProperty( long ChannelNum, LPCTSTR ChannelProperty );`
BCB	`TVariant object->GetChannelProperty( long ChannelNum, BSTR ChannelProperty );`
Delphi	`function object.GetChannelProperty( ChannelNum : LongInt; const ChannelProperty : WideString ) : OleVariant;`
VB.NET	`Function object.GetChannelProperty( ChannelNum As Integer, ChannelProperty As String ) As Object`
VC#	`object object.GetGetChannelProperty( int ChannelNum, string ChannelProperty );`

Arguments

ChannelNum As Long Channel number

ChannelProperty As String Channel property string

Return Value

The return value is the channel property value.

Remarks

1. Different devices support different channel parameters.
2. You may launch the DAQPilot wizard to know more about function usage.
   - ▷ Edit Control: data range is from -1.79769313486232E308 to + 1.79769313486232E308
   - ▷ Combo Box Control: get Enum data
   - ▷ Check Box Control: get Boolean data
3. See **Chapter 4: Programming with DAQPilot** for more information.

6.12 SetDProperty Method

DAQPilot provides a method to set all properties. You may directly modify the parameters in a specific task.

Syntax

VB	`Function object.SetDProperty ( PropertyName As String, Value As Variant ) As Long`
VC++	`long object.GetSetDProperty ( LPCTSTR PropertyName, const VARIANT& Value );`
BCB	`long object->SetDProperty ( BSTR PropertyName, TVariant Value );`
Delphi	`function object.SetDProperty ( const PropertyName : WideString; Value : OleVariant ) : LongInt;`
VB.NET	`Function object.SetDProperty ( PropertyName As String, Value As Object ) As Long`
VC#	`long object.GetSetDProperty ( string PropertyName, Object vValue );`

Arguments

PropertyName As String Property string

Value As Variant Property value

Return Value

The return value is the DAQPilot status ID as defined in DAQPilotAdvancedProperties.h.

Remarks

1. Different devices support different parameters.
2. See **Chapter 4: Programming with DAQPilot** for more information.
   - ▷ Edit Control: data range from -1.79769313486232E308 to +1.79769313486232E308
   - ▷ Combo Box Control: configure Enum data
   - ▷ Check Box Control: configure Boolean data

3. For quick and minor modifications, you may use the SetDProperty method to directly configure specific parameter with the appropriate label in the **Configuration frame**. However, it is suggested that you finish the main task using the DAQPilot Wizard property page.

For example, use

```
object.SetDProperty("Clock source (Conversion
     source)",…);
```

to modify the clock source, and use

```
object.SetDProperty("Sampling rate per
     channel",…);
```

to modify the sampling rate per channel.

6.13 GetDProperty Method

DAQPilot provides a method to get all properties. You may query the parameters from a specific task.

Syntax

VB	`Function object.GetDProperty ( PropertyName As String ) As Variant`
VC++	`VARIANT object.GetGetDProperty ( LPCTSTR PropertyName );`
BCB	`TVariant object->GetDProperty ( BSTR PropertyName );`
Delphi	`function object.GetDProperty ( const PropertyName : WideString ) : OleVariant;`
VB.NET	`Function object.GetDProperty ( PropertyName As String ) As Object`
VC#	`object object.GetGetDProperty ( string PropertyName );`

Arguments

PropertyName As String Property string

Return Value

The return value is the property value.

Remarks

1. Different devices support different parameters.
2. You may launch the DAQPilot wizard property page to get additional property information.
   - ▷ Edit Control: data range from -1.79769313486232E308 to + 1.79769313486232E308
   - ▷ Combo Box Control: get Enum data
   - ▷ Check Box Control: get Boolean data
3. See **Chapter 4: Programming with DAQPilot** for more information.

6.14 Config Method

Pre-initializes the system resource before executing any tasks. If you do not execute Config() before Start(), the Start() method automatically initializes the relative configuration, and you may not immediately start the task application.

Syntax

VB	`Function object.Config( ) As Long`
VC++	`long object.GetConfig( );`
BCB	`long object->Config( );`
Delphi	`function object.Config( ) : LongInt;`
VB.NET	`Function object.Config( ) As Long`
VC#	`long object.Config( );`

Return Value

The return value may be a status, a warning, or an error. These status IDs are defined in the DAQPilotAdvancedProperties.h.

Remarks

1. Since Config() is a time-consuming method of initializing hardware, allocating memory, generating thread, etc., it is recommended that you complete this function before executing Start().
2. If you changed the channel property or property, call Config() method to apply the modification(s).

6.15 EnableSingleChannel Method

Enables only a single channel.

Syntax

VB	`Function object.EnableSingleChannel( nChannelNum As Long ) As Long`
VC++	`long object.GetEnableSingleChannel( long nChannelNum );`
BCB	`long object->EnableSingleChannel( long nChannelNum );`
Delphi	`function object.EnableSingleChannel( nChannelNum : LongInt ) : LongInt;`
VB.NET	`Function object.EnableSingleChannel( nChannelNum As Integer) As Long`
VC#	`long object.GetEnableSingleChannel( int nChannelNum );`

Arguments

nChannelNum As Long Channel number

Return Value

The return value is DAQPilot status ID, These status IDs are defined in DAQPilotAdvancedProperties.h.

Remarks

This function disables all channels, then enables the specified channel.

6.16 Read Method

Acquires the values from multiple channels.

Syntax

VB	`Function object.Read( ) As Variant`
VC++	`VARIANT object.GetRead( );`
BCB	`TVariant object->Read( );`
Delphi	`function object.Read( ) : OleVariant;`
VB.NET	`Function object.Read( ) As Object`
VC#	`object object.GetRead( );`

Return Value

Acquired values from multiple channels, wrapped in a VARIANT variable.

Remarks

1. This function is available only for the following tasks:

   ```
   DAQPilot_TASK_AI_VOLTAGE_POLLING
   DAQPilot_TASK_DI_LINE_INPUT
   DAQPilot_TASK_DI_PORT_INPUT
   DAQPilot_TASK_DO_LINE_OUTPUT
   DAQPilot_TASK_DO_PORT_OUTPUT
   DAQPilot_TASK_TC_COUNTER
   DAQPilot_TASK_TC_MODE_OPERATION
   ```

2. The return data type depends on the task type, enabled number of channels, and UnsignedToSigned Property. When only channel is enabled, the return data type is VT_UI1 or VT_UI2 or VT_UI4 or VT_R8. When the UnsignedToSigned Property is enabled, the return data type is VT_I1 or VT_I2 or VT_I4 or VT_R8. When multiple channels are enabled, the return data type is VT_UI1|VT_ARRAY or VT_I1|VT_ARRAY or VT_UI2|VT_ARRAY or VT_I2|VT_ARRAY or VT_UI4|VT_ARRAY or VT_I4|VT_ARRAY or VT_R8|VT_ARRAY.

Example:

```
Private Sub btnPolling_Click()
    Dim i As Integer
    Dim vData As Variant
    Dim a As VbVarType

    vData = DAQPilot1.Read
    'Is it a multi channel array?
    If (vbArray And VarType(vData)) Then
        For i = 0 To UBound(vData)
           ListData.List(i) = vData(i)
        Next i
    Else
       ListData.List(0) = vData
    End If
End Sub
```

6.17 Write Method

Writes the values to multiple channels.

Syntax

VB	`Function object.Write( Data As Variant ) As Long`
VC++	`long object.GetWrite( const VARIANT& Data );`
BCB	`long object->Write( TVariant Data );`
Delphi	`function object.Write( Data : OleVariant ) : LongInt;`
VB.NET	`Function object.Write( vData As Object ) As Long`
VC#	`long object.Write( object vData );`

Arguments

Data As Variant Write values to multiple channels, Output data must be wrapped in a VARIANT variable.

Return Value

The return value is DAQPilot status ID, These status IDs are defined in DAQPilotAdvancedProperties.h.

Remarks

1. This function is available only for the following task

```
DAQPilot_TASK_AO_VOLTAGE_OUTPUT
DAQPilot_TASK_AO_CURRENT_OUTPUT
DAQPilot_TASK_AO_ONESHOT
DAQPilot_TASK_AO_CONTINUE
DAQPilot_TASK_DO_LINE_OUTPUT
DAQPilot_TASK_DO_PORT_OUTPUT
DAQPilot_TASK_DO_ONESHOT
DAQPilot_TASK_DO_CONTINUE
```

2. The output data type depends on the task type, the enabled number of channel, and UnsignedToSigned Property. When only one channel is enabled, the output data type is VT_UI1 or VT_UI2 or VT_UI4 or VT_R8. When the UnsignedToSigned Property is enabled, the output data type is VT_I1 or VT_I2 or VT_I4 or VT_R8. When multiple channels are enabled, the output data type is VT_UI1|VT_ARRAY or VT_I1|VT_ARRAY or VT_UI2|VT_ARRAY or VT_I2|VT_ARRAY or VT_UI4|VT_ARRAY or VT_I4|VT_ARRAY or VT_R8|VT_ARRAY.

Example:

```
Private Sub btnUpdate_Click()
    Dim vChNumList As Variant
    Dim bytArray() As Byte
    'Get channel no list
    vChNumList = DAQPilot1.EnabledChNumList

    If (UBound(vChNumList) = 0) Then 'Single
     channel
        DAQPilot1.Write 1
    Else 'Multiple channels
        ReDim bytArray(UBound(vChNumList))
        For i = 0 To UBound(vChNumList)
           bytArray(i) = i
        Next i
        DAQPilot1.Write bytArray
    End If
End Sub
```

6.18 Start Method

Perform a DAQ task start.

Syntax

VB	`Function object.Start( ) As Long`
VC++	`long object.GetStart( );`
BCB	`long object->Start( );`
Delphi	`function object.Start( ) : LongInt;`
VB.NET	`Function object.Start( ) As Long`
VC#	`long object.GetStart( );`

Return Value

The return value may be a status, a warning, or an error. These status IDs are defined in DAQPilotAdvancedProperties.h.

Remarks

1. This function is available only for the following tasks:

   ```
   DAQPilot_TASK_AI_ONESHOT
   DAQPilot_TASK_AI_CONTINUE
   DAQPilot_TASK_AO_ONESHOT
   DAQPilot_TASK_AO_CONTINUE
   DAQPilot_TASK_AO_FUNCTION_GEN
   DAQPilot_TASK_DI_ONESHOT
   DAQPilot_TASK_DI_CONTINUE
   DAQPilot_TASK_DO_ONESHOT
   DAQPilot_TASK_DO_CONTINUE
   DAQPilot_TASK_TC_COUNTER
   DAQPilot_TASK_TC_TIMER_INTERRUPT
   DAQPilot_TASK_TC_MODE_OPERATION
   ```

2. To improve the function performance, refer to the Config() method.

6.19 Stop Method

Stops a DAQ task.

Syntax

VB	`Function object.Stop( ) As Long`
VC++	`long object.GetStop( );`
BCB	`long object->Stop( );`
Delphi	`function object.Stop( ) : LongInt;`
VB.NET	`Function object.Stop( ) As Long`
VC#	`long object.Stop( );`

Return Value

The return value is the DAQPilot status ID. The status IDs are defined in the DAQPilotAdvancedProperties.h.

Remarks

This function is available only for the following tasks:

```
DAQPilot_TASK_AI_ONESHOT
DAQPilot_TASK_AI_CONTINUE
DAQPilot_TASK_AO_ONESHOT
DAQPilot_TASK_AO_CONTINUE
DAQPilot_TASK_AO_FUNCTION_GEN
DAQPilot_TASK_DI_ONESHOT
DAQPilot_TASK_DI_CONTINUE
DAQPilot_TASK_DO_ONESHOT
DAQPilot_TASK_DO_CONTINUE
DAQPilot_TASK_TC_COUNTER
DAQPilot_TASK_TC_TIMER_INTERRUPT
DAQPilot_TASK_TC_MODE_OPERATION
```

6.20 ShowPropertyPage Method

Displays the DAQPilot wizard property page.

Syntax

VB	`Sub object.ShowPropertyPage( )`
VC++	`object.GetShowPropertyPage( );`
BCB	`object->ShowPropertyPage( );`
Delphi	`procedure object.ShowPropertyPage( );`
VB.NET	`Sub object.ShowPropertyPage( )`
VC#	`object.GetShowPropertyPage( );`

Return Value

Null

Remarks

You may use this method to change the properties in runtime period.

6.21 ShowInstantTestPanel Method

Displays a standard test panel.

Syntax

VB	`Function object.ShowInstantTestPanel( IsModalDialog As Boolean, Caption As String ) As Boolean`
VC++	`BOOL object.GetShowInstantTestPanel( BOOL IsModalDialog, LPCTSTR Caption );`
BCB	`bool object->ShowInstantTestPanel( bool IsModalDialog, BSTR Caption );`
Delphi	`function object.ShowInstantTestPanel( IsModalDialog : WordBool; const Caption : WideString ) : WordBool;`
VB.NET	`Function object.ShowInstantTestPanel( IsModalDialog As Boolean, Caption As String ) As Boolean`
VC#	`bool object.GetShowInstantTestPanel( bool IsModalDialog, string Caption );`

Arguments

IsModalDialog As Boolean

`True`	A modal dialog box prompts you to enter information or cancel the dialog box before allowing the application to continue.
`False`	A modeless dialog box allows you to enter information and return to a previous task without closing the dialog box.

Caption As String Standard test panel caption

Return Values

- TRUE: Function is successfully excuted.
- FALSE: Function failed.

6.22 DataArrival Event

DAQPilot ActiveX control gets the data in analog/digital input task.

Syntax

VB	`Sub ControlName_DataArrival( NumberOfChannel As Long, ChNumList As Variant, Data As Variant )`
VC++	`afx_msg void OnDataArrival( long NumberOfChannel, VARIANT FAR* ChNumList, VARIANT FAR* Data );`
BCB	`void __fastcall DataArrival( long NumberOfChannel, TVariant ChNumList, TVariant Data );`
Delphi	`procedure DataArrival( NumberOfChannel : LongInt; ChNumList : OleVariant; Data : OleVariant );`
VB.NET	`Sub ControlName_DataArrival(ByVal sender As System.Object, ByVal e As DAQPilotEngine.DAQPilotDataEventArgs)` `Note: e.NumberOfChannel As Int32, e.ChNumList As Int16, e.Data As Object`
VC#	`void ControlName_DataArrival(object sender, DAQPilotEngine.DAQPilotDataEventArgs e)` `Note: int e.NumberOfChannel, short[ ] e.ChNumList, ref object e.Data`

Arguments

NumberOfChannel As Long Number of channel

ChNumList As Variant(VT_I2|VT_ARRAY)

Enabled channel number list

Data As Variant DAQ acquisition data

Remarks

1. This event is available only for the following tasks:

   ```
   DAQPilot_TASK_AI_ONESHOT
   DAQPilot_TASK_AI_CONTINUE
   DAQPilot_TASK_DI_ONESHOT
   DAQPilot_TASK_DI_CONTINUE
   ```

2. When you run DAQPilot_TASK_AI_ONESHOT or DAQPilot_TASK_AI_CONTINUE task, the return data type depends on the **Data format** property. When you run DAQPilot_TASK_DI_ONESHOT or DAQPilot_TASK_DI_CONTINUE task, the return data type depends on the **Port width** property.
3. The return data type also depends on the UnsignedTo-Signed property. When this property is enabled, the return data type is VT_I1|VT_ARRAY or VT_I2|VT_ARRAY or VT_I4|VT_ARRAY or VT_R8|VT_ARRAY. When disabled, the return data type is VT_UI1|VT_ARRAY or VT_UI2|VT_ARRAY or VT_UI4|VT_ARRAY or VT_R8|VT_ARRAY.
4. Data is a one-dimensional array wrapped in a VARIANT. For example, Data = [a1, b1, c1; a2, b2, c2; a3, b3, c3… … a100, b100, c100], NumberOfChannel = 3. a# series represents 1st channel data. b# series represents 2nd channel data, and c# series represents the channel data.

Example:

```
Private Sub DAQPilot1_DataArrival(ByVal
     NumberOfChannel As Long, ChNumList As
     Variant, Data As Variant)
    Dim NumOfScan As Long
    NumOfScan = (UBound(Data) + 1) /
     NumberOfChannel

    'Dump the data
    Dim i, j As Long
    Dim strValue As String
    vDataList.ListItems.Clear
    For i = 0 To (NumOfScan - 1)
        strValue = i
        vDataList.ListItems.Add , , strValue
        For j = 0 To (NumberOfChannel - 1)
           strValue = Data((i * NumberOfChannel)
     + j)
           vDataList.ListItems.Item(i +
     1).SubItems(j + 1) = strValue
        Next j
    Next i
End Sub
```

6.23 UpdateData Event

The DAQPilot ActiveX control requires the data for analog/digital output task.

Syntax

VB	`Sub ControlName_UpdateData( NumberOfChannel As Long, ChNumList As Variant, Data As Variant )`
VC++	`afx_msg void OnUpdateData( long NumberOfChannel, VARIANT FAR* ChNumList, VARIANT FAR* Data );`
BCB	`void __fastcall UpdateData( long NumberOfChannel, TVariant ChNumList, TVariant Data );`
Delphi	`procedure UpdateData( NumberOfChannel : LongInt; ChNumList : OleVariant; Data : OleVariant );`
VB.NET	`Sub ControlName_UpdateData(ByVal sender As Object, ByVal e As DAQPilotEngine.DAQPilotDataEventArgs)` `Note: e.NumberOfChannel As Int32, e.ChNumList As Int16, e.Data As Object`
VC#	`void ControlName_UpdateData(object sender,DAQPilotEngine.DAQPilotDataEventArgs e)` `Note: int e.NumberOfChannel, short[ ] e.ChNumList, ref object e.Data`

Arguments

NumberOfChannel As Long Number of channels

ChNumList As Variant(VT_I2|VT_ARRAY)

Enabled channel number list

Data As Variant Data is output to a waveform pattern.

Remarks

1. This event is available only for the following tasks

   ```
   DAQPilot_TASK_AO_ONESHOT
   DAQPilot_TASK_AO_CONTINUE
   DAQPilot_TASK_DO_ONESHOT
   DAQPilot_TASK_DO_CONTINUE
   ```

2. When you run DAQPilot_TASK_AO_ONESHOT or DAQPilot_TASK_AO_CONTINUE task, the update data type depends on the Data **format** property. When you run DAQPilot_TASK_DO_ONESHOT or

DAQPilot_TASK_DO_CONTINUE task, the return data type depends on the **Port width** property.

3. The update data type also depends on the UnsignedTo-Signed property. When this property is enabled, the update data type is VT_I1|VT_ARRAY or VT_I2|VT_ARRAY or VT_I4|VT_ARRAY or VT_R8|VT_ARRAY. When the property is disabled, the update data type is VT_UI1|VT_ARRAY or VT_UI2|VT_ARRAY or VT_UI4|VT_ARRAY or VT_R8|VT_ARRAY.
4. Data is a one-dimensional array wrapped in a VARIANT. For example, Data = [a1, b1, c1; a2, b2, c2; a3, b3, c3… … a100, b100, c100], NumberOfChannel = 3 a# series represents 1st channel data, b# series represents 2nd channel data, and c# series represents channel data.

Example:

```
Private Sub DAQPilot1_UpdateData(ByVal
     NumberOfChannel As Long, ChNumList As
     Variant, Data As Variant)
    'Update the output buffer in here
    Dim dwNumOfScan, dwCycleNumOfSamples As Long
    Dim dbAmplitude As Double
    Dim i, j, k As Long
    dwNumOfScan = (UBound(Data) + 1) /
     NumberOfChannel

    dwCycleNumOfSamples = dwNumOfScan
    dbAmplitude = 5
    For i = 0 To (dwNumOfScan - 1)
       For j = 0 To (NumberOfChannel - 1)
            k = (i * NumberOfChannel) + j
            If j Mod NumberOfChannel Then
               Data(k) = (-1 * dbAmplitude) + ((2
     * i * dbAmplitude) / dwCycleNumOfSamples)
            Else
                Data(k) = (Sin((CDbl(i) /
     CDbl(dwCycleNumOfSamples)) * 2 * 3.14159) *
     dbAmplitude)
            End If
       Next j
    Next i
End Sub
```

6.24 SendComplete Event

When analog/digital output task finish, the DAQPilot activex control will fire SendComplete event.

Syntax

VB	`Sub ControlName_SendComplete( )`
VC++	`afx_msg void OnSendComplete( );`
BCB	`void __fastcall SendComplete( );`
Delphi	`procedure SendComplete( );`
VB.NET	`Sub ControlName_SendComplete(ByVal sender As Object, ByVal e As EventArgs)`
VC#	`Void ControlName_SendComplete(object sender, EventArgs e);`

Remarks

This event is available only for the following tasks:

```
DAQPilot_TASK_AO_ONESHOT
DAQPilot_TASK_DO_ONESHOT
```

6.25 TimerInterrupt Event

When the TimerInterrupt task gets the interrupt, the DAQPilot ActiveX control initiates a TimerInterrupt event

Syntax

VB	`Sub ControlName_TimerInterrupt( )`
VC++	`afx_msg void OnTimerInterrupt( );`
BCB	`void __fastcall TimerInterrupt( );`
Delphi	`procedure TimerInterrupt( );`
VB.NET	`Sub ControlName_TimerInterrupt(ByVal sender As Object, ByVal e As EventArgs)`
VC#	`void ControlName_TimerInterrupt(object sender, EventArgs e);`

Remarks

This event is available only for this task:

```
DAQPilot_TASK_TC_TIMER_INTERRUPT
```

6.26 DAQPilotError Event

When a DAQPilot ActiveX control encounters an error, the DAQPilot ActiveX control initiates a DAQPilotError event.

Syntax

VB	`Sub ControlName_DAQPilotError( ErrorCode As Long, ErrorString As String )`
VC++	`afx_msg void OnDAQPilotError( long ErrorCode, LPCTSTR ErrorString );`
BCB	`void __fastcall DAQPilotError( long ErrorCode, BSTR ErrorString );`
Delphi	`procedure DAQPilotError( ErrorCode : LongInt; const ErrorString : WideString );`
VB.NET	`Sub ControlName_DAQPilotError( ErrorCode As System.Int32, ErrorString As System.String )`
VC#	`void ControlName_DAQPilotError (int ErrorCode, string ErrorString );`

Arguments

ErrorCode As Long — The DAQPilot status ID. The status ID may be a status, warning, or an error. These status IDs are defined in the DAQPilotAdvancedProperties.h.

ErrorString As String — DAQPilot status string

Example:

```
Private Sub DAQPilot1_Error(ByVal ErrorCode As
     Long, ByVal ErrorString As String)
    MsgBox ErrorString
End Sub
```