

Charting Tools ActiveX Control

WIN-AXC-100

Version 1.0

Quinn-Curtis, Inc.

QUINN-CURTIS, INC. LICENSE AGREEMENT AND LIMITED WARRANTY

READ THE TERMS AND CONDITIONS OF THIS LICENSE AGREEMENT CAREFULLY BEFORE OPENING THE ENVELOPE CONTAINING THE PROGRAM DISKETTES. THIS LICENSE AGREEMENT REPRESENTS THE ENTIRE AGREEMENT CONCERNING THE SOFTWARE BETWEEN YOU AND QUINN-CURTIS, INC. AND IT SUPERSEDES ANY PRIOR PROPOSAL, REPRESENTATION OR UNDERSTANDING BETWEEN THE PARTIES. BY OPENING THE PACKAGE CONTAINING THE DISKETTES, YOU ARE ACCEPTING AND AGREEING TO THE TERMS OF THIS LICENSE AGREEMENT. IF YOU ARE NOT WILLING TO BE BOUND BY THE TERMS OF THIS LICENSE AGREEMENT, YOU SHOULD PROMPTLY RETURN THE PACKAGE WITH THE DISKETTES IN THE UNOPENED ENVELOPE, WITHIN 30 DAYS OF PURCHASE, AND YOU WILL RECEIVE A REFUND OF YOUR MONEY.

LICENSE AGREEMENT

1. Who is the Licensee? The licensee of the associated software (referred to as "SOFTWARE" elsewhere in this agreement) is the individual or company designated in the registration form, either the registration form shipped with the software manual, or by registering on-line at the Quinn-Curtis web site. In the absence of a valid registration form, the licensee is the individual or company that purchased the SOFTWARE. The license for the software (but not customer support privileges) can be transferred on a permanent basis to someone other than the individual or company specified in the original registration form. This is done by notifying Quinn-Curtis in writing of the transfer.

2. Basic Software Developers License Agreement. Quinn-Curtis, Inc. grants the licensee the non-exclusive right to use a single copy of the SOFTWARE on a single computer, and to compile and link the SOFTWARE into an application developed by the licensee. The SOFTWARE can be moved from computer to computer, as long as there is no possibility of the software being used on more than one computer at a time. The software cannot be placed on a network where more than one user can use it at a time, unless a separate license agreement is purchased for each user on the network. Contact Quinn-Curtis about quantity discounts available for network users. Also, the software cannot be placed in a common "library" where it can be checked out by individuals who are not the registered users of the software. Quinn-Curtis documentation is protected from being copied under US copyright law.

3. Distribution of Application Programs Created using the SOFTWARE. The registered user may distribute application programs created using the SOFTWARE royalty free, assuming the following requirements are met.

DOS PROGRAMS - The application program must take the form of an *.EXE file which is a program executable under DOS. Developers who must redistribute software in forms other than *.EXE files will need to contact Quinn-Curtis regarding specific license limitations. Quinn-Curtis runtime support files (*.BGI) can be redistributed with application programs that require them without royalties. Source files (*.C, *.H, *.ASM) cannot be redistributed under any circumstances unless a separate copy the SOFTWARE is purchased for each copy redistributed.

WINDOWS PROGRAMS - The application program must take the form of an *.EXE file which is a program executable under Microsoft Windows. Developers who must redistribute software in forms other than *.EXE files will need to contact Quinn-Curtis regarding specific license limitations. Quinn-Curtis runtime support files (*.DLL) can be redistributed with application programs that require them without royalties. Source files (*.C, *.H, *.RC, *.DLG, *.DEF) cannot be redistributed under any circumstances unless a separate copy the SOFTWARE is purchased for each copy redistributed.

4. Customer Support. Quinn-Curtis maintains the right to restrict customer support to only the registered user (an individual, not a company) of the SOFTWARE. Companies which purchase the SOFTWARE must make sure the registration form is properly filled out, specifying the individual who is to be considered the registered user. Quinn-Curtis maintains the right to limit customer support to a period of 12 months from the date of purchase, or six months from the time the software is obsoleted by a new version, unless appropriate updates to the software have been purchased separately.

LIMITED WARRANTY

1. Limited Warranty. Quinn-Curtis, Inc. warrants that SOFTWARE media (floppy disk) and manual will be free from defects in materials and workmanship under normal use for a period of 60 days from the date of purchase. Quinn-Curtis will replace free of charge the SOFTWARE media and/or manual within the 60 day period if either is found defective. Customers should contact Quinn-Curtis in order to get a return authorization for materials deemed defective.

2. Customer Remedies. Quinn-Curtis' entire liability and your exclusive remedy shall be, at Quinn-Curtis' option, either (a) replacement of the SOFTWARE media and/or manual or (b) return of the purchase price. In no event will Quinn-Curtis be liable for any other damages (including, but not limited to, damages for loss of business profits, business interruption, loss of data, and other special, incidental or consequential damages) arising out of the use, or the inability to use to use the SOFTWARE, even if Quinn-Curtis has been specifically advised of the possibility of such damages.

QUINN-CURTIS, INC. SPECIFICALLY DISCLAIMS ALL OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

GENERAL

1. If any provision in this License and Limited Warranty is found invalid or unenforceable, it will not affect the validity of the balance of this agreement, which shall remain valid and enforceable according to its terms.

2. This License and Limited Warranty may only be modified in writing signed by the Licensee and a specifically authorized representative of Quinn-Curtis.

3. This License and Limited Warranty shall be construed, interpreted and governed by the laws of the State of Massachusetts and any action hereunder shall be brought only in Massachusetts

4. Use, duplication or disclosure by the US Government of the computer software and documentation in this package shall be subject to the restricted rights under DFARS 52.227-7013 applicable to commercial computer software.

5. All rights not specifically granted in this License and Limited Warranty are reserved by Quinn-Curtis, Inc..

07/16/03

Contents

Chapter 1 - Introduction.....	1
REGISTER YOUR SOFTWARE.....	1
Overview	1
Prerequisites	2
Hardware	2
Software	2
Redistributables.....	3
Conventions Used in this Manual	3
Product Upgrades	3
Technical Support.....	3
Chapter 2 - Getting Started	7
Installing the Charting Tools ActiveX Control.	7
Example Programs	8
Document Files	8
Learning to Use the Charting Tools ActiveX Control	9
Chapter 3 - Programming with the <i>Charting Tools ActiveX Control</i>	11
Introduction to ActiveX.....	11
Introduction to the Charting Tools ActiveX Control.....	11
Creating a Chart	11
Basic Concepts	12
Plotting Area	12
Coordinate Systems	12
Data Sets.....	12
Data Plot Types	13
Line Plot.....	13
Scatter Plot.....	13
Line Marker Plot	13
Bar Graph	14
Floating Bars	14
Grouped Bar Graphs.....	15
Stacked Line Plot	15
High-Low-Close.....	16
Error Bars	16
Pie Chart.....	16
Building a CFormView Based Application with AppWizard	17
Generate an MFC Skeleton	17
Edit Project	17
Create a Dialog Resource for CFormView.....	18
Create a Charting Class for the Dialog Resource.....	18
Defining the Chart	18
Recompile and Run.....	19
Building an Application with Visual Basic	19
Generate an Application Program.....	20
Create a Charting Class for the Form	20
Defining the Chart	20
Build and Run.....	22
Building an Application with Delphi.....	22
Generate an Application Program.....	22
Add the Quinn-Curtis ActiveX Control to the Delphi Component Toolbar	22
Create a Charting Class for the Form	23
Defining the Chart	23

Contents

Build and Run.....	24
Chapter 4 - Property Pages.....	27
General Property Page.....	27
Axis Property Page.....	29
Axis Labels Property Page.....	30
Grids Property Page.....	31
Data Property Page.....	32
Simple Plots Property Page.....	33
Group Plots Property Page.....	34
Pie Charts Property Page.....	35
Legends Property Page.....	36
Printing Property Page.....	38
Events Property Page.....	39
Caption Property Description.....	39
Chapter 5 - Charting Tools ActiveX Property Reference.....	41
Parameterized Properties.....	41
Property Reference.....	42
AxisAutoAxis.....	42
AxisColor.....	43
AxisEnable.....	43
AxisGridColor.....	43
AxisGridEnable.....	44
AxisGridLineStyle.....	44
AxisGridLineWidth.....	44
AxisGridType.....	45
AxisIntercept.....	45
AxisInterceptTrack.....	45
AxisLabelColor.....	46
AxisLabelDecs.....	46
AxisLabelFont.....	47
AxisLabelFontSize.....	47
AxisLabelFontStyle.....	47
AxisLabelPos.....	48
AxisLabelsEnable.....	48
AxisLabelStrings.....	48
AxisLabelStringsEnable.....	49
AxisLabelStringsStart.....	49
AxisLineWidth.....	50
AxisMajorTickInterval.....	50
AxisMax.....	50
AxisMin.....	51
AxisMinorTicks.....	51
AxisNumericStyle.....	52
AxisScaleMode.....	52
AxisTickStyle.....	53
AxisTitleColor.....	53
AxisTitleFont.....	54
AxisTitleFontSize.....	54
AxisTitleFontStyle.....	54
AxisTitlePos.....	54
AxisTitleString.....	55
BottomPlotArea.....	55
EventDataCursorType.....	56
EventRetrieveDataValue.....	56
EventShowPropPage.....	56
EventZoom.....	57

EventZoomReset.....	57
EventZoomResetMaxX.....	58
EventZoomResetMaxY.....	58
EventZoomResetMinX.....	58
EventZoomResetMinY.....	59
EventZoomX.....	59
EventZoomXRoundMode.....	59
EventZoomY.....	60
EventZoomYRoundMode.....	60
GPlotAreaFill.....	60
GPlotBarHatch.....	61
GPlotBarJustify.....	61
GPlotBarType.....	62
GPlotBarWidth.....	62
GPlotLineColor.....	62
GPlotLineStyle.....	63
GPlotLineThickness.....	63
GPlotRefAxes.....	63
GPlotScatterShape.....	64
GPlotScatterSize.....	64
GPlotScatterStyle.....	65
GPlotType.....	65
LeftPlotArea.....	65
MajorTickSize.....	66
MinorTickSize.....	66
Pie3DLook.....	66
PieCenterX.....	66
PieCenterY.....	67
PieDiameter.....	67
PieFontColor.....	67
PieFontSize.....	68
PieFontStyle.....	68
PieNumericFont.....	68
PieNumericStyle.....	69
PieNumberPos.....	69
PieSliceBorderColor.....	69
PieSliceExplode.....	70
PieSliceFillColor.....	70
PieSliceHatch.....	70
PieSliceLabel.....	71
PlotBackgroundColor.....	71
PrintBorder.....	71
PrintBottom.....	72
PrintGraphBackground.....	72
PrintLeft.....	72
PrintMaintainAspectRatio.....	73
PrintPlotBackground.....	73
PrintRight.....	73
PrintStyle.....	74
PrintTop.....	74
RightPlotArea.....	74
SDataEnable.....	75
SDataName.....	75
SDataNumGroups.....	75
SDataNumPlotPoints.....	76
SDataType.....	76

Contents

SLegendBackgroundColor	77
SLegendBorderColor	77
SLegendBorderThickness	77
SLegendBottom	78
SLegendEnable	78
SLegendFont	78
SLegendFontColor	78
SLegendFontSize	79
SLegendFontStyle	79
SLegendLeft	79
SLegendOrientation	80
SLegendRight	80
SLegendStrings	80
SLegendTop	81
SLegendType	81
SPlotAreaFill	81
SPlotBarColor	82
SPlotBarHatch	82
SPlotBarJustify	82
SPlotBarType	83
SPlotBarWidth	83
SPlotLineColor	83
SPlotLineStyle	84
SPlotLineThickness	84
SPlotRefAxes	85
SPlotScatterColor	85
SPlotScatterDrop	85
SPlotScatterLine	86
SPlotScatterShape	86
SPlotScatterSize	86
SPlotScatterStyle	86
SPlotSpline	87
SPlotType	87
TitleColor	87
TitleFont	88
TitleFontSize	88
TitleFontStyle	88
TitleString	89
TopPlotArea	89
WindowBackgroundColor	89
WindowBorderColor	90
WindowBorderStyle	90
WindowBorderThickness	90
XDataValues	91
YDataValues	91
Chapter 6 - Charting Tools ActiveX Method Reference	93
CopyToClipboard	93
DefineDataSet	93
DefineGroupDataSet	94
FFTComplexFFT	94
FFTDSPWindow	95
FFTFrequency	96
FFTMagnitude	96
FFTPhase	97
FFTPowerSpectrum	97
FFTRealFFT	98

GetDatasetMaxX	99
GetDatasetMaxY	99
GetDatasetMinX	100
GetMousePos	100
GetMousePosNorm	100
GetDatasetMinY	101
GetNearestPoint	101
GraphZoomReset	102
LoadASCIIDataSet	102
MarkDataPoint	103
MoveDataCursor	103
PrinterSetup	104
PrintGraph	104
ReconnectDataSet	104
SaveASCIIDataSet	105
SavePageMeta	105
SerializeLoadFile	106
SerializeSaveFile	106
SetDataCursor	107
SortDataX	108
SortDataY	108
UpdateGraph	108
Chapter 7 - FFTs	111
FFT Harmonics	111
FFT Normalized Magnitude	113
FFT Phase Shift	113
Typical FFT Results	114
FFT References	114
Index	115

Contents

Chapter 1 - Introduction

REGISTER YOUR SOFTWARE

Register your software at www.quinn-curtis.com/ProductRegistration/ProdReg.aspx. If the user has registered then the associated Quinn-Curtis ActiveX controls, incorporated in application programs (*.EXE files), can be distributed to third parties. Registration will put you on our mailing list ensuring that you will receive update information for this software package and other Quinn-Curtis programming products. If more than one person is going to use this software within your company, you must purchase additional copies of the software or a site license.

Overview

The Quinn-Curtis **Charting Tools ActiveX Control** is built on top of the Quinn-Curtis Charting Tools DLL thereby encapsulating the robust features of the DLL into an easy-to-use ActiveX Control. The **Charting Tools ActiveX Control** is collection of charting routines that solve the most common charting problems encountered in science, engineering, business, and data presentation applications. This product lets you create and incorporate sophisticated charts into your spreadsheet, word processing, database, or other visual containers including: C, Visual Basic or Delphi. The main features of the **Charting Tools ActiveX Control** are:

ActiveX Property Pages

Hundreds of properties and graph attributes can be edited through detailed property pages or set and retrieved programmatically using various visual programming languages, allowing you to rapidly create and import charts into your spreadsheet, word processor, database, or other visual container that supports ActiveX Controls.

Data Arrays

All data to controls are passed using 32-bit pointers. The theoretical upper limit for a single data array is 4 billion bytes, or 500 million 8-byte doubles. Data can be imported or exported from clipboard, program, or data file.

A single set of data can have an unlimited number of data points. Each set of data displayed in a graph can have a unique number of data points.

Data Display Objects

The data display objects include line, scatter, bar, floating bar, pie, stacked, group, high-low-close, area fill, and error bars. Any combination of data display objects can be added to a chart.

Multiple Axes and Axes Scales

Each chart offers two pairs of x and y axes allowing two separate scaling transformations. Two completely different scaling ranges for each x,y pair can be displayed together. Linear and logarithmic axes are supported. Axes are drawn in either automatic or manual axis scaling modes, and can be labeled using engineering or scientific formats, or with user defined string labels.

Built-in Data Grid Editor

Chapter 1 - Introduction

The data property page includes a data grid editor. Modify data values via data table editor or create new arrays by entering values into the appropriate cells. Arrays can be resized arrays to any dimension. Charts are automatically updated to reflect the modified data set values.

Serialization

Serialization has been designed into every object in the control. Any chart can be serialized regardless of its complexity. A single call to the control will save or restore all of the defining attributes and data for all of the charts and legends.

Integrated Printer Support

Output to any Windows supported printer at the resolution of the device.

Numerical Processing

Our FFT algorithms are available as methods in the control.

Other Features

- On-line Help.
- Integrated zooming and drill down.
- Save graph image as a DIB or copy to clipboard.
- Refresh data without redrawing entire graph.
- Bad Data Point Handling.
- Legends.

Prerequisites

Hardware

Minimum Requirements

- A 486- or Pentium-based (Pentium preferred), PC compatible computer.
- A graphics card compatible with Microsoft Windows NT/98/XP and that supports at least 256 colors.
- 16MB of RAM, 32MB preferred.

Software

To create Microsoft Windows applications using these tools, you must have Microsoft Windows NT/98/XP installed on your computer.

To run applications created using these tools, the target system must have either Microsoft Windows NT/98/XP installed. Microsoft support DLLs, normally installed when a compiler or operating system is installed, must also be present on the target computer. The required support DLLs are:

MFC42.DLL
MSVCRT.DLL
OLE32.DLL
OLEAUT32.DLL

These are standard Microsoft DLLs required for applications written using the DLL (shared) version of MFC.

The Quinn-Curtis Charting Tools DLL, WCT32DX.DLL, is also required. This was installed on your system as part of the **Charting Tools ActiveX Control** installation.

The **Data** property page of the **Charting Tools ActiveX Control** uses the Microsoft FlexGrid ActiveX component. The FlexGrid component is installed as part of the standard controls of Visual C++ and Visual Basic. To distribute applications based on the **Charting Tools ActiveX Control** which include the FlexGrid component, you must own either the Visual C++ and Visual Basic compilers thus providing you a valid license to redistribute the FlexGrid component. The files associated with the FlexGrid component are MSFLXGRD.OCX and MSFLXGRD.DEP. If the FlexGrid components are not present the **Data** property page will not display.

Redistributables

If you plan to ship a product which includes the Quinn-Curtis Charting Tools ActiveX control, you will need to include the following files from this product.

CTWX.OCX The main ActiveX runtime file for the Charting Tools Control.
CTWX.HLP The CTWX help file.
WCT32DX.DLL The Charting Tools DLL file required. It is required. It is slightly different than our regular Charting Tools DLL file in that it is 8-byte aligned in order to make it ActiveX compatible.

Other required files are: MFC42.DLL, MSVCRT.DLL, OLE32.DLL, OLEAUT32.DLL, MSFLXGRD.OCX and MSFLXGRD.DEP. Redistribution rights for these files are granted with the Microsoft Visual C++ and Visual Basic compilers. Borland Delphi users may need to contact Microsoft or Borland about redistribution rights for these files for applications created using Borland Delphi.

Conventions Used in this Manual

- All code listings are printed in equally spaced *Courier font*.
- The names of functions and classes are printed in **bold** type.
- The function arguments appear in *italic*.
- File names are printed in UPPERCASE letters.

Product Upgrades

By registering this software, we are able to notify you when new releases become available. Registered users are eligible for special prices on new releases for a limited time. Notification of new releases generally begins approximately one month after a new release is available.

Technical Support

Technical support is handled through the Quinn-Curtis forums, found at www.quinn-curtis.com. Share your programming questions and solutions by posting to this forum.

You want to reproduce your problem in a simple example program. Sometimes we can only solve a problem if we are able to run and debug a program which fails in our offices. You should be

Chapter 1 - Introduction

prepared to ZIP together all of the files necessary to compile and run your program so that they can be sent to us by e-mail.

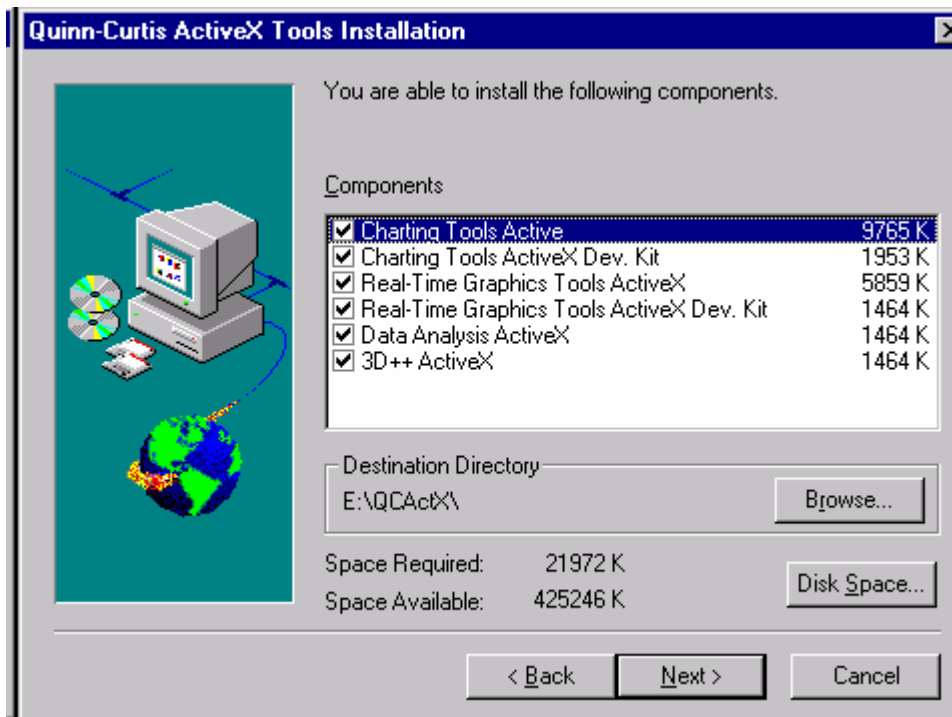
Chapter 2 - Getting Started

Installing the Charting Tools ActiveX Control.

These software tools arrive on one master CD-ROM. If you bought the source code, the source code also is contained on the CD. The product files have been archived and compressed.

To install the ***Charting Tools ActiveX Control for Windows:***

1. If you have a CD, place the supplied Quinn-Curtis CD into your CD-ROM drive. If you downloaded the software, unzip the setup files to a temporary directory.
2. If you have a CD, go to the DRIVE:\CSETUP directory and run the program SETUP.EXE. If you downloaded the software, go to the directory where you unzipped the setup software and run the program SETUP.EXE.
3. Follow the instructions that appear on the screen. You will be prompted for your name, company name, (if any) and a CD key which is found on a sticker on the original CD-ROM case. If you downloaded the software, use the **Install Key** specified on the download screen, and the confirming e-mail. The install key is an 18 character string and looks like: 3200093-3212411231. When you enter the key make sure you also include the '-' hyphen.
4. After you pass the CD key check, the following screen will display.



Only the components that you have purchased will be displayed in the list window. The installation program will install all of the listed components by default. If you do not want to install certain components to save space, uncheck the box next to the component description.

Chapter 2 - Getting Started

The default installation directory for the tools is C:\QCActX, but you can change the default directory that the SETUP program chooses for installing the software, if necessary. Use the Browse button to select the drive and directory where you want the software installed.

If the directory you choose does not exist, the installation program will create it together with several subdirectories. If, for example, you chose the default directory d:\QCActX when installing the tools, the .OCX, .DLL and .HLP files will be moved to the directory d:\QCActX\LIB, and the example programs will be moved to the d:\QCActX\Charting\Examples subdirectory. The \Examples subdirectory contains 3 subdirectories, \CPP, \BASIC and \DELPHI. Within each subdirectory you will find example programs unique to Visual C++, Visual Basic and Delphi.

After the installation is complete, please store the original CD in a safe place.

You should place the directory;

d:\QCActX\LIB

in your system PATH so that it can find the CTWX.OCX ActiveX Control and the WCT32DX.DLL DLL library.

Example Programs

The example programs are installed in the \EXAMPLES subdirectory. Each example program has five or more files associated with it. Workspace or project files are supplied for each demo. Each of the three supported languages, C++, VB and Delphi, has 6 demos, listed below:

AXES	Demonstrates how to create multiple axes for a single graph.
DYNAMIC	Dynamically updates data in real-time without redrawing the entire graph.
GROUP	Demonstrates two types of group plots.
PIE	Displays data in a pie chart format.
SIMPLE	A simple XY chart with three traces with a different number of points for each trace.
ZOOM	Demonstrates the built-in zooming functions.

Document Files

CTWX.HLP Double click on this file to activate the Windows Help system and to load it with the **Charting Tools ActiveX Control** help file. Refer to the Microsoft Developer Studio Extension Help (EXTHELP.HLP) in the product \HELP subdirectory for details on how to customize Microsoft Developer Studio to get F1 help on keywords.

WINAXC100.PDF A PDF version of this manual.

Learning to Use the Charting Tools ActiveX Control

The best way to learn how to use these tools is to study the example programs. Start writing your applications by modifying our example programs. Reading this manual helps greatly. It contains answers to many questions you may have. In particular, it is very important that you carefully read the Tutorial chapter, as it leads you through all of the steps required to build a ***Charting Tools ActiveX Control*** application.

Chapter 3 - Programming with the *Charting Tools ActiveX Control*

Introduction to ActiveX

ActiveX controls, formerly known as OLE controls, let you develop sophisticated controls based on the Component Object Model (COM) that can be installed in dialog boxes or any ActiveX control container application, including pages on the Internet's World Wide Web and Visual Basic applications.

An ActiveX control is a COM-based object that can draw itself in its own window, respond to events (such as mouse clicks), and be managed through an interface that includes properties and methods.

These controls can be developed for many uses, such as database access, data monitoring, or graphing. Besides their portability, ActiveX controls support features previously not available to custom controls, such as compatibility with existing OLE containers and the ability to integrate their menus with the OLE container menus. In addition, an ActiveX control fully supports OLE Automation, allowing the control to expose properties and methods that can be called by the control user.

Introduction to the Charting Tools ActiveX Control

Creating a Chart

A "chart" consists of a set of chart objects. These objects represent the various pieces of a graph including, but not limited to, titles, axes, legends, and data plots.

To define a chart, the user must write a sequence of calls to chart object creation functions. This sequence of calls should occur when the container is initiated. Other chart objects can be added in later functions. Events completed during chart initialization should include:

- Set general attributes for chart.
- Create data sets.
- Establish axes attributes.
- Create data plot types.
- Create Legends.
- Create Miscellaneous Objects.

Visual C++

It is recommended that the initial creation of the chart is completed in the **OnInitialUpdate** member function of **CView** derived view in which the control is placed.

Visual Basic

It is recommended that the initial creation of the chart is completed in the **Form_Load** member function of the form in which the control is placed.

Chapter 3 - Programming

Delphi

It is recommended that the initial creation of the chart is completed in the **FormCreate** (**OnCreate** event) member function of the form in which the control is placed.

Basic Concepts

First, we have to introduce some concepts and define appropriate terms that will be used extensively in this documentation.

Plotting Area

Plotting area is a rectangle placed somewhere in the graph window. All data plots are drawn inside the plotting area and are clipped by the plotting area rectangle. It is typically associated with at least one pair of axes. The extents of the axes are bounded by the plotting area.

Coordinate Systems

Dimensions and positions of windows and graphical objects are expressed in one of two different, device independent, coordinate systems: *Graph Normalized*, and *Physical*. In a *Graph Normalized* coordinate system the point with coordinates (0.0, 0.0) corresponds to the upper-left corner of a graph window. The point with coordinates (1.0, 1.0) corresponds to the lower-right corner. A *Physical* coordinate system positions a graphical object at the location specified by a pair of (X, Y) data values based on a given pair of axes.

Data Sets

All the data plotting functions work with data organized as data sets. A data set combines the x and y values for a plot within a graph into a single entity. A data set is created using the **Data** property page of the control, or by the **DefineDataSet** or **DefineGroupDataSet** member functions.

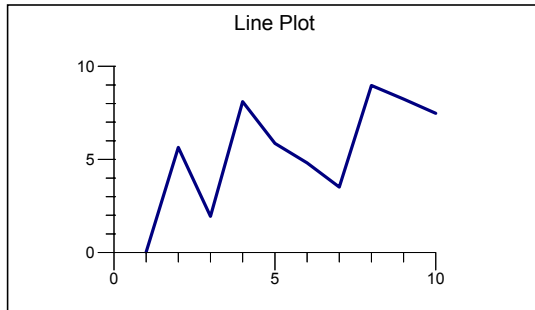
There exists three data set types: *Simple XY*, *Group* and *Pie Chart*. The *Simple XY* type (SIMPLE_XY_DATA_TYPE) consists of a pair of one-dimensional arrays of data, the first array contains the x-values for each point, and the second array contains the corresponding y-values for each point. The *Group* type (GROUP_DATA_TYPE) contains a one dimensional array of x-values and a y-array containing values for each group in the data set. The number of rows in the y-array must equal to the number of data points, and the number of columns equals the number of groups. The *Pie Chart* type (PIECHART_DATA_TYPE) consists of a single array of data containing the values for a pie chart.

A data set is identified by an index in the range of 0 to 31. The total number of data sets existing simultaneously in one instance of a control is 32.

Data Plot Types

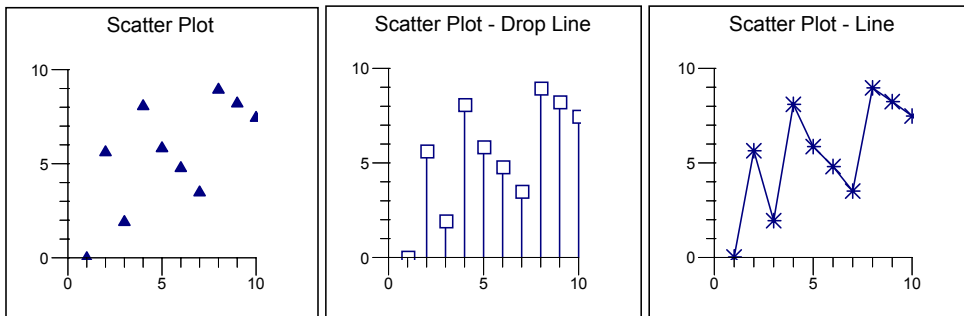
Nine basic data plot types are implemented in this control. They are: line plots, scatter plots, bar graphs, floating bar graphs, grouped bar graphs, high-low-close plots, error bars, line plots with markers, and pie charts. One or more of the basic data plotting functions can be used together in the same graph window. Most of the basic data plotting functions have additional modes which create variations on the basic plot types.

Line Plot



This is a basic plot type in which adjacent data points are connected with straight lines. It has a cubic spline smoothing option and an option to fill the area under the line with a solid color. The line color, line style and line thickness are specified by the user.

Scatter Plot

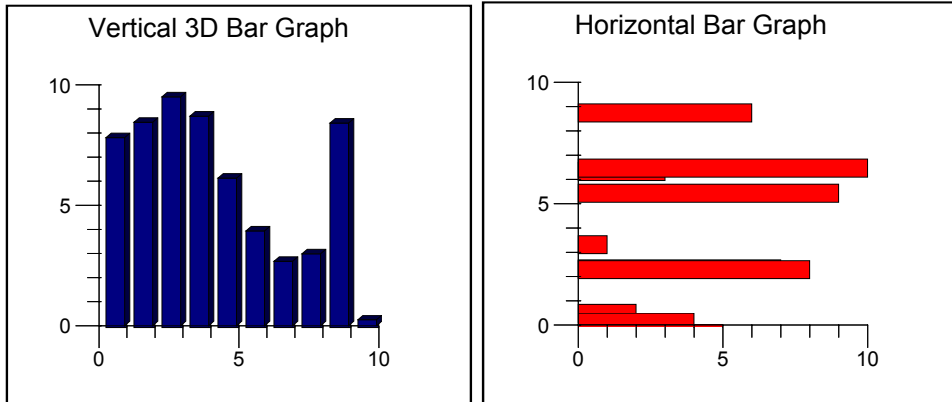


Each data point in a scatter plot is represented as a symbol, or marker. Various symbol shapes are available - dot, square box, asterisk, circle, triangle, cross, etc. The size, color and fill of the symbols are also under user control. Vertical lines can be dropped from the scatter plot symbols to the X-axis.

Line Marker Plot

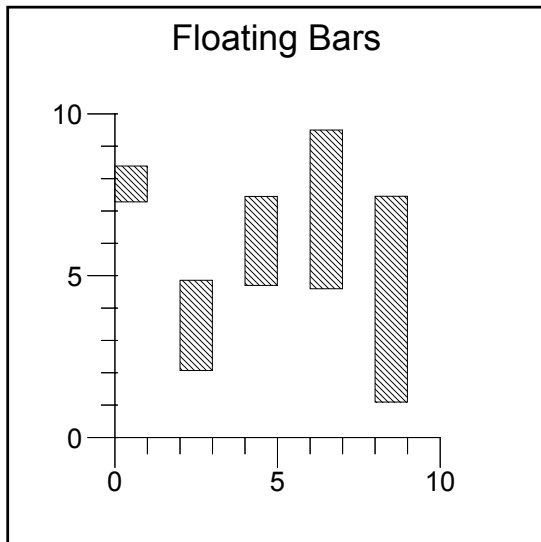
This is a combination of a line plot with a scatter plot.

Bar Graph



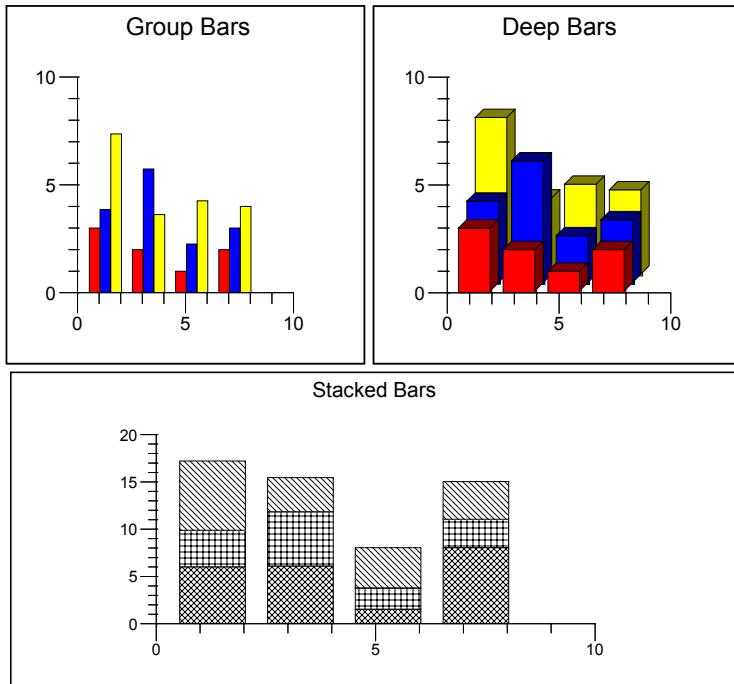
The bar graph function displays a standard bar graph with one end of each bar anchored to either zero or top or bottom of the plotting area. It has 2-D, 3-D, horizontal, and vertical options. Bar graphs are drawn using "dithered" RGB colors and therefore thousands of colors are available on regular VGA systems. Standard Windows hatch styles can be used in bar graphs.

Floating Bars



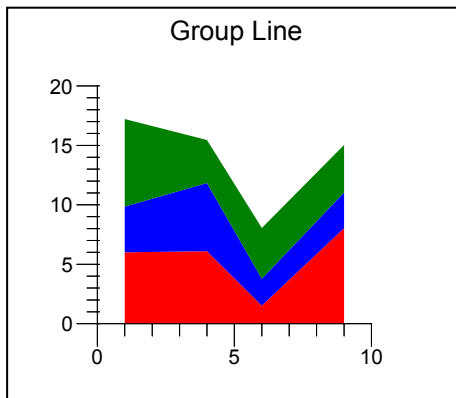
The floating bar graph is similar to the standard bar graph except that both ends of each bar are defined by data values, allowing for bars with different starting points.

Grouped Bar Graphs



Grouped bar graphs can be used for group data sets. They can have one of the three formats - members of the group are represented by bars positioned either side by side ("group" format), behind one another at 45 degrees ("deep" format), or on top of one another (cumulative "stacked" format). The bars can be 2-D, 3-D, horizontal, or vertical. The color and hatching options are the same as for standard bar graphs.

Stacked Line Plot

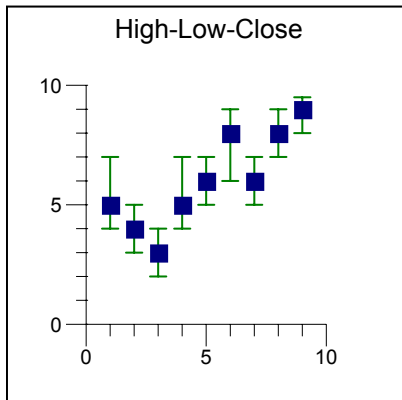


The stacked line plot can be used for group data sets. It is a cumulative set of lines, that is Y-coordinates of each line are calculated relative to the Y-coordinates of the preceding line. The stacked line plot has an option to fill the area between the lines with a solid color. The color, style and thickness of every line are also controllable.

Chapter 3 - Programming

High-Low-Close

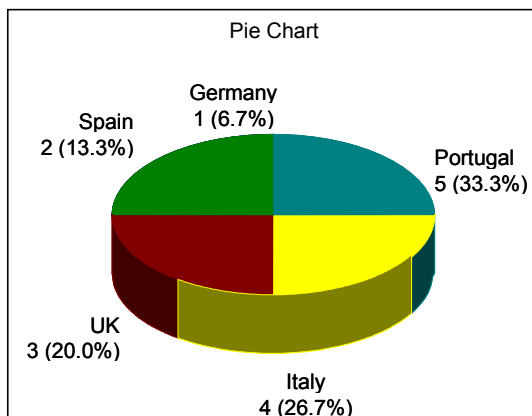
This is the standard "stock market" method of representing data that has three Y-values (high value, low value, and closing value) for each X-value. The Close value can be represented by one of nine different symbol types, while the High and Low values are represented with horizontal lines. The High and Low values are connected to the Close symbol with a vertical line.



Error Bars

The error bars plot is similar to the High-Low-Close plot except that the Close symbol is not used and the High and Low values are not connected with a vertical line.

Pie Chart



Pie charts represent data as pie sections (slices). They can be two- or three-dimensional. Colors, fill patterns, text, exploding of pie sections are all under the user control.

Building a CFormView Based Application with AppWizard

The tutorial creates a simple SDI application with a line plot and a bar graph. The program has a standard toolbar and status bar. The tutorial assumes that the compiler is version 5.0 of Microsoft Visual C++ running under Windows 95/98. The procedure may be slightly different for other versions of Visual C++.

This tutorial creates an application program called Tutorial in the d:\QCActX\Charting\Examples\CPP\TUTORIAL subdirectory. The complete source code to the program is similar to the example found in the \CPP\SIMPLE subdirectory. You can refer to SIMPLE if you wish to copy parts of that program into this one. Make sure you change references to SIMPLE to TUTORIAL.

Generate an MFC Skeleton

The first step in this process is to generate a skeleton application that matches our specification. To do this, use AppWizard to create an application called TUTORIAL in the d:\QCActX\Charting\Examples\CPP\TUTORIAL subdirectory.

The Visual C++ Wizards make creating the skeleton easy. From Visual C++ start AppWizard by selecting File | New | Project

1. Specify MFC AppWizard (exe) as the desired project type.
2. Enter the project name in the Name edit field: Tutorial.
3. Specify the project directory in the Location field. It should be d:\QCActX\Charting\Examples\CPP\TUTORIAL
4. Select the OK button.
5. Specify the application's type as Single Document (SDI).
6. Leave default values in steps 2 - 5 of the AppWizard. A summary of defaults is shown below.

x	No database support
x	No OLE Compound Document support
x	No OLE Automation
x	Support for ActiveX Controls
x	Yes for toolbar, status bar, printing, and 3-D controls
x	No context-sensitive help
x	Yes for source file comments
x	Use MFC in a shared DLL

7. Step 6 of 6 is the last step in the chain of creation dialogs. Select **CTutorialView** from the class list displayed at the top of the form. The *Base class* drop down list at the bottom of the dialog will become active once you select **CTutorialView**. Select **CFormView** from the *Base class* drop down list. This completes the App Wizard's creation dialogs. Select the Finish button followed by OK to allow AppWizard to create for you a fully-functional C++ skeleton.

Edit Project

Perform the following steps:

Add:

```
#include "CTWXDEF.H"
```

to the TutorialView.CPP source file.

Chapter 3 - Programming

To ensure that the compiler finds Quinn-Curtis include files, select Project | Settings| C/C++ and select Preprocessor from the Category drop down list. Add

```
..\..\..\.
```

to the Additional Include Directories field. This directs the compiler to look for include files in the \QCActX directory.

Create a Dialog Resource for CFormView

Select Resource View from the object viewer. Select IDD_TUTORIAL_FORM from the Dialog item in the Resource View. A dialog box should appear to the right.

Select the dialog box with the mouse. Select the “To do...” text inside of the dialog box and press the delete key to remove the item. Click the right mouse button and select the Insert an ActiveX Control menu item. Select the A Quinn-Curtis Charting Control 1.0 from the control list. Press the OK button to add the control to the dialog box. Enlarge the bounding rectangle for the control using the mouse.

This completes the creation of a dialog resource for the program.

Create a Charting Class for the Dialog Resource

Select View | Class Wizard. Go to the Member Variables tab of the ClassWizard. Select the CTutorialView from the class name drop down list. Select IDC_CTWXCTRL1 from the control ID list. Select the Add Member Variable Button from the dialog. The ClassWizard will display a message box notifying you that the Developer’s Studio will generate a wrapper class for the Charting control. Select the OK button to start the wrapper creation. The ClassWizard subsequently displays the Confirm Class dialog for the Charting Class. Select the OK button to create the wrapper. The Add Member Variable for IDC_CTWXCTRL1 is displayed once the Wizard has generated the wrapper. Specify *m_GraphControl* as the member variable name and select the OK button.

Next, go to the Message Maps tab of the Class Wizard. Select **CTutorialView** under Object IDs. Add a member function for the **OnInitialUpdate** message. Code is added to the functions later.

Defining the Chart

To define a chart, the user has to write a sequence of calls to chart object drawing functions. It is recommended that the initial creation of the chart is completed in the **OnInitialUpdate** member function of **CTutorialView**. In this example, the member function **BuildGraph** contains the chart building calls, and is called inside of **OnInitialUpdate**. (The **BuildGraph** code in this example is identical to the **BuildGraph** code in the demo program SIMPLE).

```
void CTutorialView::OnInitialUpdate()
{
    CFormView::OnInitialUpdate();

    BuildGraph();
}

void CTutorialView::BuildGraph()
{
    double pXData0[11] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 };
    double pYData0[11] = { 3, 4, 2, 1, 5, 7, 9, 5, 6, 10, 11 };
    double pXData1[7] = { 2, 3, 4, 5, 6, 7, 8 };
    double pYData1[7] = { 5, 3, 7, 8, 9, 4, 3 };
    double pXData2[5] = { 3, 7, 10, 11, 12 };
}
```

```

double pYData2[5] = { 18, 17, 13, 12, 15};
CString sLegendStrings, CR = " ";

int n0 = 11, n1 = 7, n2 = 5;

m_GraphControl.SetAxisAutoAxis(X_AXIS1, TRUE);
m_GraphControl.SetAxisAutoAxis(Y_AXIS1, TRUE);

m_GraphControl.SetAxisLabelFontSize(X_AXIS1, 7);
m_GraphControl.SetAxisLabelFontSize(Y_AXIS1, 8);

m_GraphControl.SetAxisTitleString(Y_AXIS1, "Y-Axis Title");

m_GraphControl.DefineDataSet(0, "Line Data",
    pXData0,
    pYData0,
    n0);
m_GraphControl.SetSDataEnable(0,TRUE);
m_GraphControl.SetSDataType(0, SIMPLE_XY_DATA_TYPE);
m_GraphControl.SetSPlotType(0,ST_LINEPLOT);
m_GraphControl.SetSPlotLineColor(0, RGB(0,255,255));

m_GraphControl.DefineDataSet(1, "Bar Data",
    pXData1,
    pYData1,
    n1);
m_GraphControl.SetSDataEnable(1,TRUE);
m_GraphControl.SetSDataType(1, SIMPLE_XY_DATA_TYPE );
m_GraphControl.SetSPlotType(1, ST_BARGRAPH);
m_GraphControl.SetSPlotBarColor(1, RGB(255,0,0));

m_GraphControl.DefineDataSet(2, "Scatter Data",
    pXData2,
    pYData2,
    n2);
m_GraphControl.SetSDataEnable(2,TRUE);
m_GraphControl.SetSPlotType(2, ST_SCATTERPLOT);

m_GraphControl.SetTitleString(TITLE1,"Simple Graph Title");
m_GraphControl.SetTitleString(FOOTER,"Simple Graph Footer");

CR.SetAt(0,(char) 13); // make CR a carriage return string;

sLegendStrings = "Line" + CR + "Bar" + CR + "Scatter" + CR;
m_GraphControl.SetSLegendStrings(sLegendStrings);
m_GraphControl.SetSLegendEnable(TRUE);
}

```

Recompile and Run

That is it. At this point you can build the project and run it.

Building an Application with Visual Basic

The tutorial creates a simple Visual Basic standard application with a line plot and a bar graph. The program has a standard toolbar and status bar. The tutorial assumes that the compiler is version 5.0 of Microsoft Visual Basic running under Windows 95/98. The procedure may be slightly different for other versions of Visual Basic.

This tutorial creates an application program called Tutorial in the d:\QCActX\Charting\Examples\VB\TUTORIAL subdirectory. The complete source code to the

Chapter 3 - Programming

program is similar to the example found in the \VB\SIMPLE subdirectory. You can refer to SIMPLE if you wish to copy parts of that program into this one. Make sure you change references to SIMPLE to TUTORIAL.

Generate an Application Program

The first step in this process is to generate a skeleton application that matches our specification. To do this, use Visual Basic to create an application called TUTORIAL in the d:\QCActX\Charting\Examples\VB\TUTORIAL subdirectory.

Visual Basic does not automatically create a folder when a new application is generated. Create a new folder via Windows Explorer called "Tutorial" in d:\QCActX\Charting\Examples\VB.

The Visual Basic environment make creating the skeleton easy. From Visual Basic select File | New | Project.

1. Specify Standard EXE as the desired project type.
2. Select the OK button.
3. Save the project by selecting File | Save As
4. Specify the project directory in the Location field. It should be d:\QCActX\Charting\Examples\VB\TUTORIAL
5. Specify the project name as Tutorial.

Add the module CTWXDEF.BAS to the project. This module contains all of the predefined constants unique to the CTWX ActiveX control.

Create a Charting Class for the Form

To add an instance of the Charting Control to the applications select Project | Components and select the "A Quinn-Curtis Charting ActiveX Control 1.0" from the Controls dialog. The Charting control icon should appear in the toolbox. Select the Charting control icon from the Toolbox and add it to the Tutorial form.

Defining the Chart

To define a chart, the user has to write a sequence of calls to chart object drawing functions. It is recommended that the initial creation of the chart is completed in the **Form_Load** member function of **tutorial.frm**. The code in this example is identical to the **Form_Load** code in the demo program SIMPLE).

```
Private Sub Form_Load()  
  
    Dim pXData0(11) As Double  
    Dim pYData0(11) As Double  
    Dim pXData1(7) As Double  
    Dim pYData1(7) As Double  
    Dim pXData2(5) As Double  
    Dim pYData2(5) As Double  
    Dim sLegendStrings As String  
    Dim CR As String  
    Dim n0, n1, n2 As Integer  
  
    pXData0(0) = 1  
    pXData0(1) = 2  
    pXData0(2) = 3  
    pXData0(3) = 4  
    pXData0(4) = 5  
    pXData0(5) = 6
```

```

pXData0(6) = 7
pXData0(7) = 8
pXData0(8) = 9
pXData0(9) = 10
pXData0(10) = 11

pYData0(0) = 3
pYData0(1) = 4
pYData0(2) = 2
pYData0(3) = 1
pYData0(4) = 5
pYData0(5) = 7
pYData0(6) = 9
pYData0(7) = 5
pYData0(8) = 6
pYData0(9) = 10
pYData0(10) = 11

pXData1(0) = 2
pXData1(1) = 3
pXData1(2) = 4
pXData1(3) = 5
pXData1(4) = 6
pXData1(5) = 7
pXData1(6) = 8

pYData1(0) = 5
pYData1(1) = 3
pYData1(2) = 7
pYData1(3) = 8
pYData1(4) = 9
pYData1(5) = 4
pYData1(6) = 3

pXData2(0) = 3
pXData2(1) = 7
pXData2(2) = 10
pXData2(3) = 11
pXData2(4) = 12

pYData2(0) = 18
pYData2(1) = 17
pYData2(2) = 13
pYData2(3) = 12
pYData2(4) = 15
n0 = 11
n1 = 7
n2 = 5

CTWX1.AxisAutoAxis(X_AXIS1) = True
CTWX1.AxisAutoAxis(Y_AXIS1) = True

CTWX1.AxisLabelFontSize(0) = 7
CTWX1.AxisLabelFontSize(1) = 8

CTWX1.AxisTitleString(1) = "Y-Axis Title"

Call CTWX1.DefineDataSet(0, "Line Data", pXData0(0),
                        pYData0(0), n0)

CTWX1.SDataEnable(0) = True
CTWX1.SDataType(0) = SIMPLE_XY_DATA_TYPE
CTWX1.SPlotType(0) = ST_LINEPLOT
CTWX1.SPlotLineColor(0) = RGB(0, 255, 255)

Call CTWX1.DefineDataSet(1, "Bar Data", pXData1(0), pYData1(0), n1)

CTWX1.SDataEnable(1) = True
CTWX1.SDataType(1) = SIMPLE_XY_DATA_TYPE
CTWX1.SPlotType(1) = ST_BARGRAPH
CTWX1.SPlotBarColor(1) = RGB(255, 0, 0)
Call CTWX1.DefineDataSet(2, "Scatter Data", pXData2(0),

```

Chapter 3 - Programming

```
                pYData2(0), n2)
CTWX1.SDataEnable(2) = True
CTWX1.SPlotType(2) = ST_SCATTERPLOT

CTWX1.TitleString(TITLE1) = "Simple Graph Title"
CTWX1.TitleString(FOOTER) = "Simple Graph Footer"

'carriage return
CR = Chr$(13)

sLegendStrings = "Line" + CR + "Bar" + CR + "Scatter" + CR
CTWX1.sLegendStrings = sLegendStrings
CTWX1.SLegendEnable = True

End Sub
```

Build and Run

That is it. At this point you can build the project and run it.

Building an Application with Delphi

The tutorial creates a simple Delphi standard application with a line plot and a bar graph. The program has a standard toolbar and status bar. The tutorial assumes that the compiler is version 2.0 or 3.0 of Delphi running under Windows 95/98.

This tutorial creates an application program called Tutorial in the d:\QCActX\Charting\Examples\Delphi\TUTORIAL subdirectory. The complete source code to the program is similar to the example found in the \DELPHI\SIMPLE subdirectory. You can refer to SIMPLE if you wish to copy parts of that program into this one. Make sure you change references to SIMPLE to TUTORIAL.

Generate an Application Program

The first step in this process is to generate a skeleton application that matches our specification. To do this, use Delphi to create an application called TUTORIAL in the d:\QCActX\Charting\Examples\Delphi\TUTORIAL subdirectory.

The Delphi environment makes creating the skeleton easy. From the Delphi File menu select New Application.

1. Save the project by selecting File | Save As
2. Specify the project directory in the Location field. It should be d:\QCActX\Charting\Examples\Delphi\TUTORIAL
3. Specify the form unit name as TUTORIALU and the project name as TUTORIAL.

Add the Quinn-Curtis ActiveX Control to the Delphi Component Toolbar

Select Component | Install from the main menu. Select the OCX button. In the Import OLE dialog you will see a list of registered OCX controls. Select the Quinn-Curtis Charting Tools ActiveX Control. Select OK to exit the Import OLE control dialog. Select OK again to exit the Install Components dialog. Select the OCX tab of the components tool bar and the Charting Tools ActiveX icon should be present.

Create a Charting Class for the Form

Select the Charting Tools ActiveX control icon from the toolbar and add it to the form. Size and position the graph the way you want. Place

Defining the Chart

First, place a reference to CTWXDEF in a uses clause in the implementation.

```
implementation
uses CTWXDEF;
```

To define a chart, the user has to write a sequence of calls to chart object drawing functions. It is recommended that the initial creation of the chart is completed in the **FormCreate (OnCreate event)** member function of the TForm1 class. The code in this example is identical to the **FormCreate** code in the demo program SIMPLE).

```
procedure TForm1.FormCreate(Sender: TObject);
const
  pXData0: ARRAY [0..10] of double = ( 1,2,3,4,5,6,7,8,9, 10, 11 );
  pYData0: ARRAY [0..10] of double = ( 3,4,2,1,5,7,9, 5,6,10,11);
  pXData1: ARRAY [0..6] of double = ( 2,3,4,5,6,7,8 );
  pYData1: ARRAY [0..6] of double = ( 5, 3, 7, 8, 9, 4,3);
  pXData2: ARRAY [0..4] of double = ( 3,7,10,11,12 );
  pYData2: ARRAY [0..4] of double = ( 18, 17, 13, 12, 15);
var
  CR: String;
  sLegendStrings: String;
  n0, n1, n2: INTEGER;
begin
  n0 := 11;
  n1 := 7;
  n2 := 5;

  CTWXCtrl1.AxisAutoAxis[X_AXIS1] := TRUE;
  CTWXCtrl1.AxisAutoAxis[Y_AXIS1] := TRUE;

  CTWXCtrl1.AxisLabelFontSize[X_AXIS1] := 7;
  CTWXCtrl1.AxisLabelFontSize[Y_AXIS1] := 8;

  CTWXCtrl1.AxisTitleString[Y_AXIS1] := 'Y-Axis Title';

  CTWXCtrl1.DefineDataSet(0, 'Line Data',
    pXData0[0],
    pYData0[0],
    n0);
  CTWXCtrl1.SDataEnable[0] := TRUE;
  CTWXCtrl1.SDataType[0] := SIMPLE_XY_DATA_TYPE ;
  CTWXCtrl1.SPlotType[0] := ST_LINEPLOT;
  CTWXCtrl1.SPlotLineColor[0] := RGB(0,255,255);

  CTWXCtrl1.DefineDataSet(1, 'Bar Data',
    pXData1[0],
    pYData1[0],
    n1);
  CTWXCtrl1.SDataEnable[1] := TRUE;
  CTWXCtrl1.SDataType[1] := SIMPLE_XY_DATA_TYPE;
  CTWXCtrl1.SPlotType[1] := ST_BARGRAPH;
  CTWXCtrl1.SPlotBarColor[1] := RGB(255,0,0);

  CTWXCtrl1.DefineDataSet(2, 'Scatter Data',
    pXData2[0],
    pYData2[0],
    n2);
  CTWXCtrl1.SDataEnable[2] := TRUE;
  CTWXCtrl1.SPlotType[2] := ST_SCATTERPLOT;
```

Chapter 3 - Programming

```
CTWXCtrl1.TitleString[TITLE1] := 'Simple Graph Title';
CTWXCtrl1.TitleString[FOOTER] := 'Simple Graph Footer';

CR := CHR (13);
sLegendStrings := 'Line' + CR + 'Bar' + CR + 'Scatter' + CR;
CTWXCtrl1.SLegendStrings := sLegendStrings;
CTWXCtrl1.SLegendEnable := TRUE;
end;
```

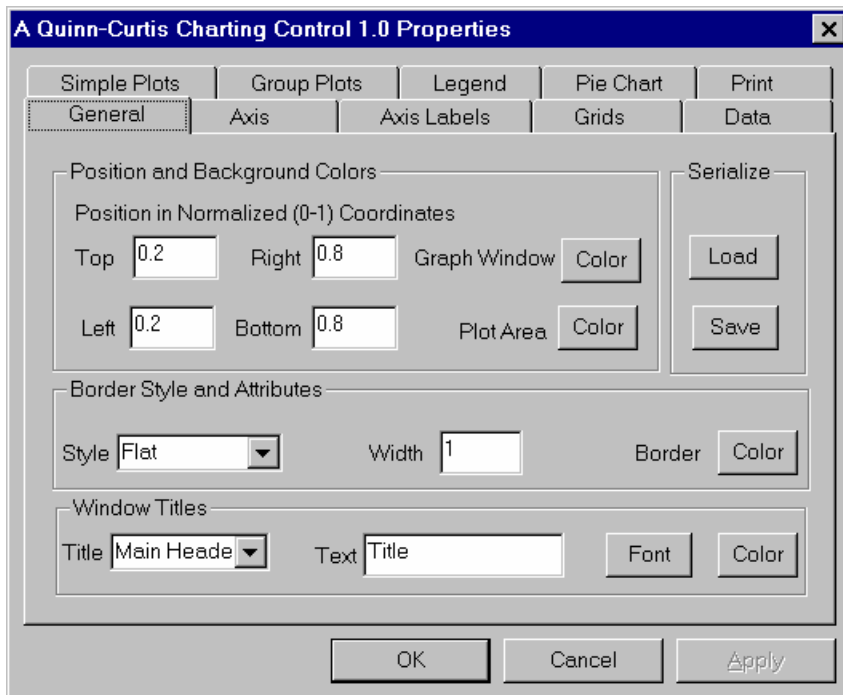
Build and Run

That is it. At this point you can build the project and run it.

Chapter 4 - Property Pages

This chapter documents the Charting Tools ActiveX control property pages and provides a cross reference to the individual properties documented in Chapter 5.

General Property Page



Caption	Property	Description
Position and Background Colors		
Top	TopPlotArea	Position of the top of the plotting area relative to the graph height. Value of 0.0 indicates that the top of the plotting area coincides with the top of the graph window.
Left	LeftPlotArea	Position of the left side of the plotting area relative to the graph width. Value of 0.0 indicates that the left side of the plotting area coincides with the left side of the graph window.
Bottom	BottomPlotArea	Position of the bottom of the plotting area relative to the graph height. Value of 1.0 indicates that the bottom of the plotting area coincides with the bottom of the graph window.
Right	RightPlotArea	Position of the right side of the plotting area relative to the graph width. Value of 1.0 indicates that the right side of the plotting area coincides with the right side of the graph window.
Plot Area Color	PlotBackgroundColor	Background color code of the plotting area. The plotting area is the area bounded by the axes.
Graph Window Color	WindowBackgroundColor	Background color of the entire graph area.

Chapter 4 Property Pages

Serialize

Load	SerializeLoadFile	Load a previously saved graph serialization file and initialize the current graph with it.
Save	SerializeSaveFile	Save the current graph as a serialization file.

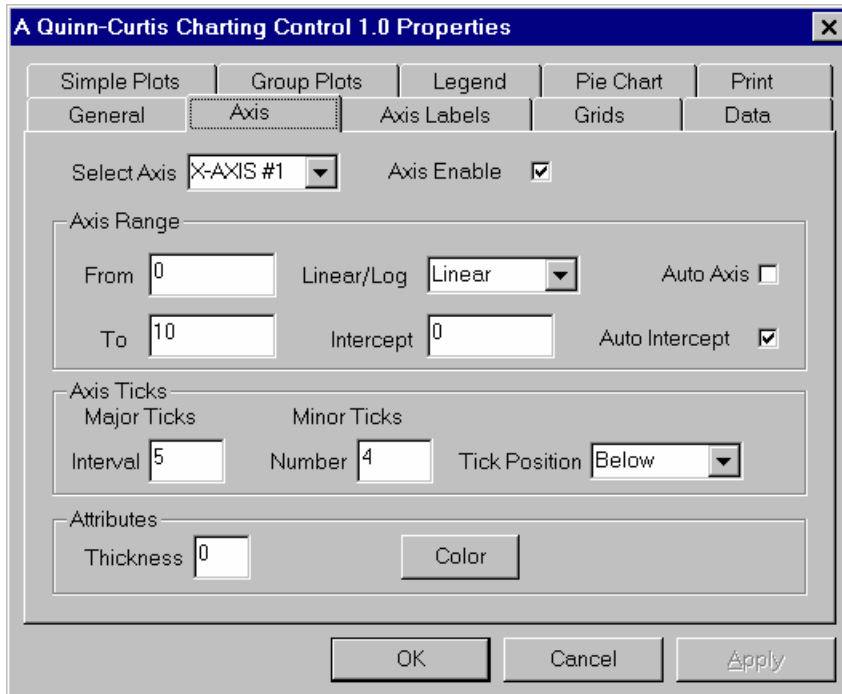
Window Titles

Color	TitleColor	The color of the graph titles.
Font	TitleFont	The ASCII name of the font typeface for the graph titles.
Font	TitleFontSize	The font size in points (1/72") of the graph titles.
Font	TitleFontStyle	The text font style of the graph titles.
Text	TitleString	Text associated with the selected title.

Border Style and Attributes

Border Color	WindowBorderColor	Border color for the border around the graph area.
Style	WindowBorderStyle	Border style for the border around the graph area.
Width	WindowBorderThickness	Thickness of the lines used in drawing the border around the graph area.

Axis Property Page



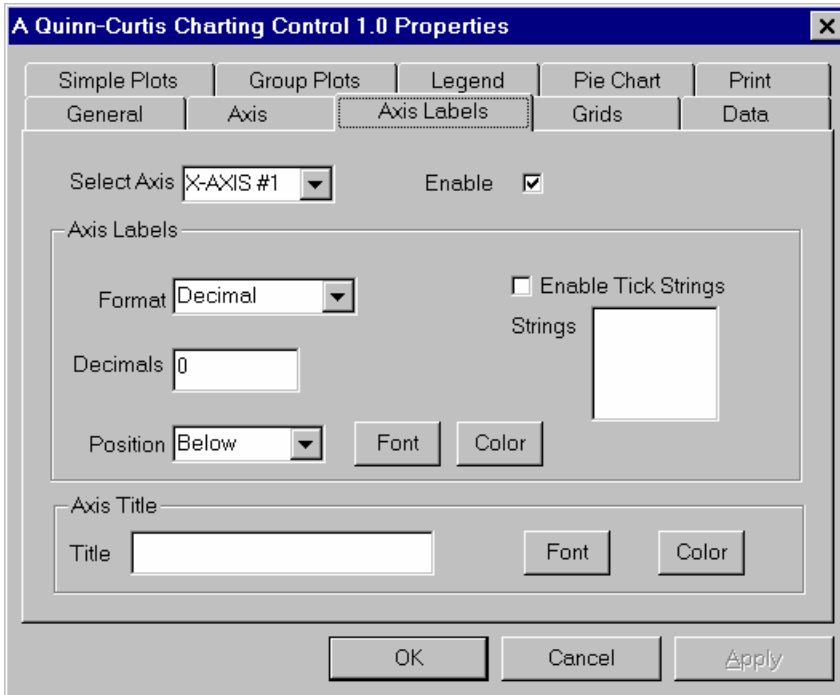
Caption	Property	Description
Select Axis	Not a property	Index used to select which axis the other parameters apply to: X_AXIS1, Y_AXIS1, X_AXIS2 or Y_AXIS2.
Axis Enable	AxisEnable	Enable the selected axis.
Axis Range		
From	AxisMax	The value corresponding to the right (x-axis) or top (y-axis) of the graph plotting area.
To	AxisMin	The value corresponding to the left (x-axis) or bottom (y-axis) of the graph plotting area.
Linear/Log	AxisScaleMode	Specifies axis scaling mode, either linear or logarithmic.
Intercept	AxisIntercept	Value where the selected axis intercepts the axis at right angles to it.
AutoAxis	AxisAutoAxis	Set the auto-axis mode for a given axis. Set to TRUE to have the axis range automatically calculated based on the data.
Auto Intercept	AxisInterceptTrack	Forces the intercept of an axis to be placed at the axis extreme of the axis at right angles to it. This will position the selected axis on the outside edge of the graph.
Axis Ticks		
Interval	AxisMajorTickInterval	Distance between major tick marks, in physical units of the x-axis.
Number	AxisMinorTicks	Number of minor tick marks between adjacent major ones. If it is zero, minor tick marks are not drawn.
Tick Position	AxisTickPos	Tick direction with respect to axis.

Chapter 4 Property Pages

Attributes

Thickness	AxisLineWidth	Specifies the width, in screen pixels, of selected axis. When a graph is sent to the printer, the width of a line relative to the graph width remains constant unless the value of this parameter is 0. In this case the width of line in device units is always one pixel, resulting in the thinnest possible line.
Color	AxisColor	Set the color of the selected axis.

Axis Labels Property Page



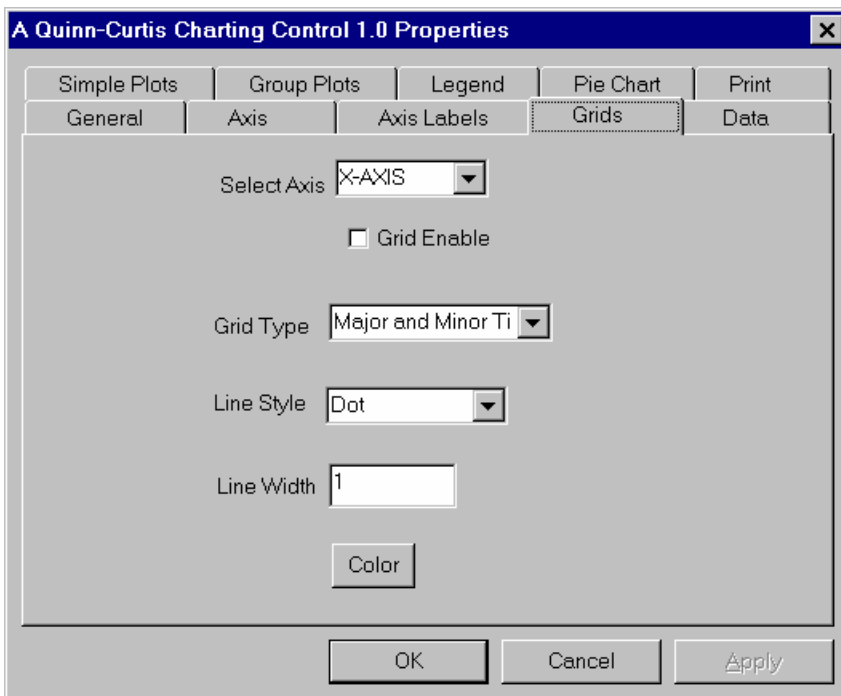
Caption	Property	Description
Select Axis	Not a property	Index used to select which axis the other parameters apply to: X_AXIS1, Y_AXIS1, X_AXIS2 or Y_AXIS2.
Enable	AxisLabelsEnable	Enable the labels for the selected axis.
Axis Labels		
Format	AxisNumericStyle	Defines the axis numeric label format.
Decimals	AxisLabelDecs	The number of digits after the decimal point used when labeling the axis tick marks.
Position	AxisLabelPos	Controls the justification of the labels' position relative to the axis.
Font	AxisLabelFont	The ASCII name of the axis labels font typeface.
Font	AxisLabelFontSize	The axis labels font size in points (1/72").
Font	AxisLabelFontStyle	The axis labels text font style.
Color	AxisLabelColor	The color of the axis.
Enable Axis Strings	AxisLabelStringsEnable	Enable string labels for the selected axis. If this parameter is FALSE the labels for an axis will default to numeric labeling.

Strings	AxisLabelStrings	A string which contains sub string used in place for numeric labels for the given axis.
(N/A)	AxisLabelStringsStart	Sequential number of the major tick where the first label is displayed.

Axis Title

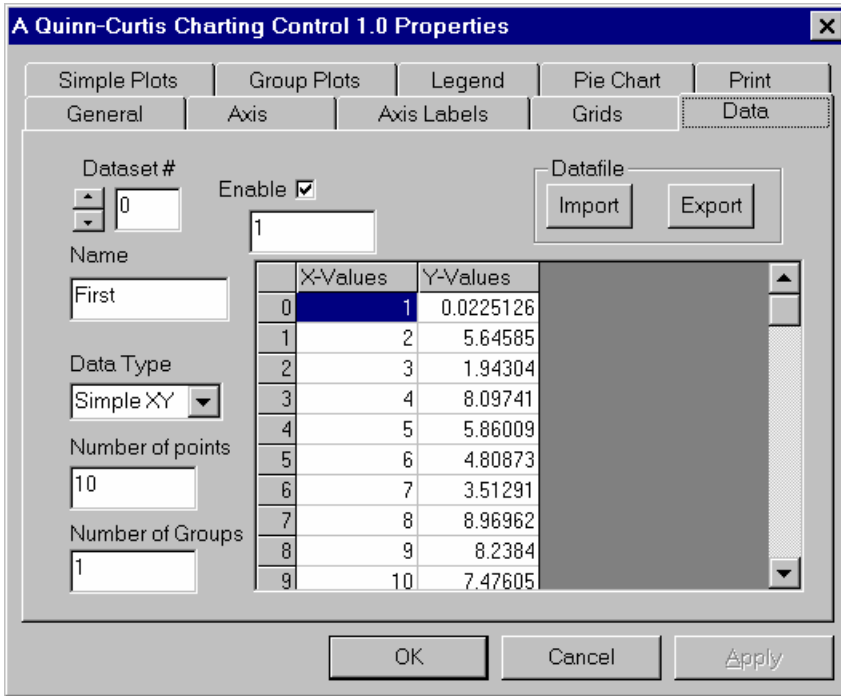
Title	AxisTitleString	Text string to be used as the axis title.
Font	AxisTitleFont	Syntax
Font	AxisTitleFontSize	The axis title font size in points (1/72").
Font	AxisTitleFontStyle	The axis title text font style.
Color	AxisTitleColor	Color of the selected title.

Grids Property Page



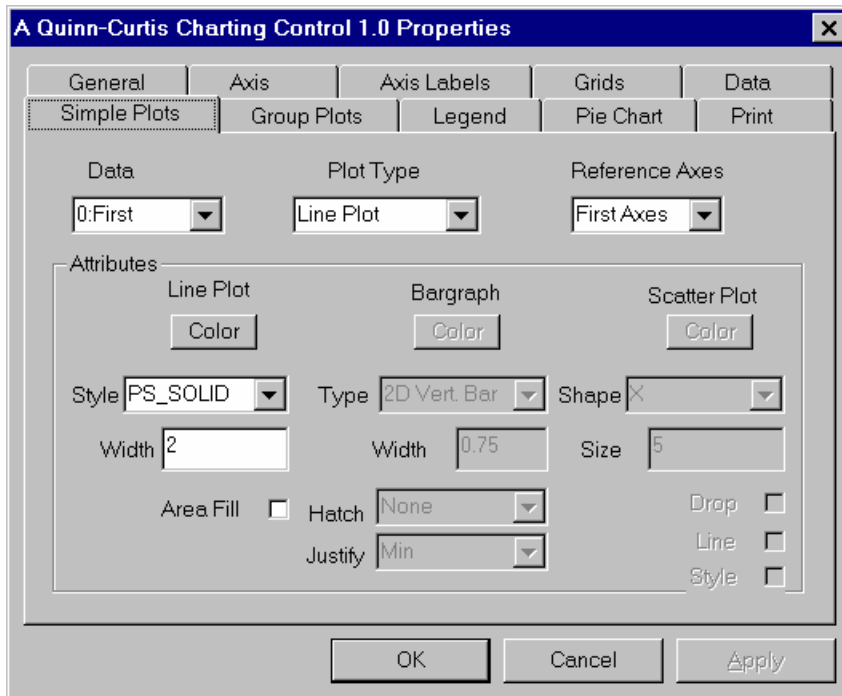
Caption	Property	Description
Select Axis	Not a property	Index used to select which axis the other parameters apply to: X_AXIS1, Y_AXIS1, X_AXIS2 or Y_AXIS2.
Grid Enable	AxisGridEnable	Enable the grid for the selected axis.
Grid Type	AxisGridType	Determines if the grid lines are positioned at major or minor tick marks on the axis, or both.
Line Style	AxisGridLineStyle	Line style for the grid associated with the selected axis.
Line Width	AxisGridLineWidth	Specifies the width, in screen pixels, of the grid associated with the selected axis.
Color	AxisGridColor	Color of the grid associated with the given axis.

Data Property Page



Caption	Property	Description
Dataset #	Not a property	Index used to select which data set all other parameters apply to. Data sets are indexed from 0 to 31.
Enable	SDataEnable	Enable a data set and the associated plot.
Name	SDataName	A text string used to identify a data set.
Data Type	SDataType	The data set type. There are three data set types: <i>Simple XY</i> , <i>Group</i> and <i>Pie Chart</i> .
Number of Points	SDataNumPlotPoints	The number of values (rows) for one column of a data set.
Number of Groups	SDataNumGroups	The number of groups in a group data set. If a data set is of the group type (GROUP_DATA_TYPE), it can have up to 16 sets of y-values for one set of x-values. The number of columns of y-values is the number of groups. Simple data sets (SIMPLE_XY_DATA_TYPE) always have one group.
[The Grid]	XDataValues	Set and retrieve individual x-data values for a data set.
[The Grid]	YDataValues	Set and retrieve individual y-data values for a data set.

Simple Plots Property Page

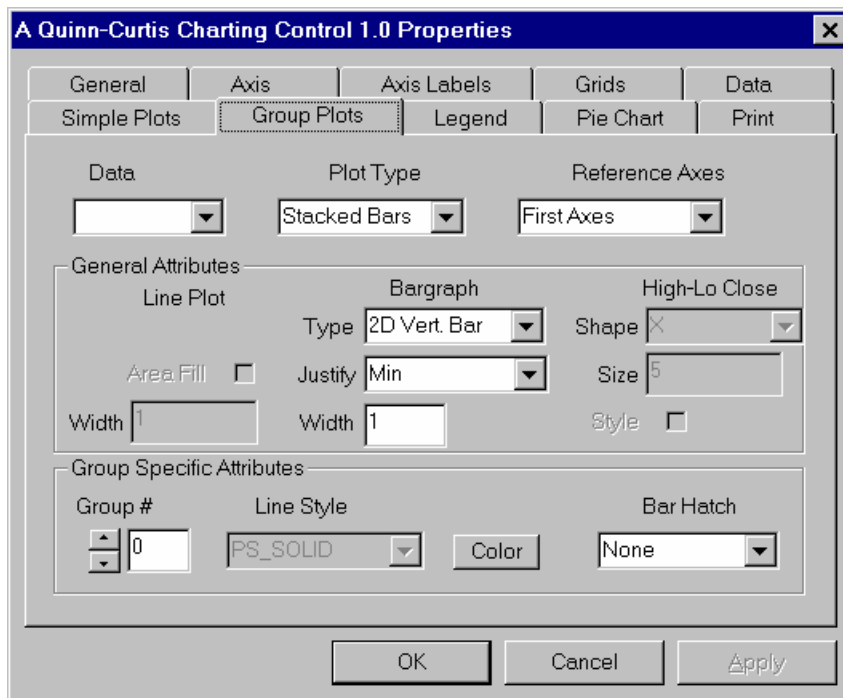


Caption	Property	Description
Data	Not a property	Index used to select which data set the other parameters apply to. Data sets are indexed from 0 to 31. If a name was assigned to the data set it is displayed along with its index. Only data sets of the <i>Simple XY</i> type will show up in this drop down list box.
Plot Type	SPlotType	Sets the plot type for the selected <i>Simple XY</i> data set. Simple plot types include: line plots, bar graphs and scatter plots.
Reference Axes	SPlotRefAxes	Specifies the reference axes for the selected plot.
Attributes		
Line Plot Color	SPlotLineColor	The RGB color of the line plot for the selected data set.
Line Plot Style	SPlotLineStyle	Specifies the line style of the line associated with the selected data set.
Line Plot Width	SPlotLineThickness	Specifies the width, in screen pixels, of the line associated with the selected data set.
Line Plot Area Fill	SPlotAreaFill	If this flag is TRUE, the area between line plot and the x-axis is filled with color.
Bargraph Color	SPlotBarColor	The RGB color of the bar plot for the selected data set.
Bargraph Type	SPlotBarType	Controls the direction and type of the bars.
Bargraph Hatch	SPlotBarHatch	Specifies the hatch style codes for bars of the selected plot.
Bargraph Justify	SPlotBarJustify	Controls placement of bars. Vertical bars can be left-, center-, or right-justified, horizontal bars can be top-, center-, or bottom-justified, relative to the actual values of the independent variable.

Chapter 4 Property Pages

Bargraph Width	SPlotBarWidth	Bar width in the units of the axis that the bar width is parallel to.
Scatter Plot Color	SPlotScatterColor	Specifies the color of the symbols in the selected scatter plot.
Scatter Plot Shape	SPlotScatterShape	Determines the shape of symbols used in the scatter plot chart type.
Scatter Plot Size	SPlotScatterSize	Determines the size of in points (1/72") of the symbols used in the scatter plot chart type.
Scatter Plot Drop	SPlotScatterDrop	If TRUE, a line is dropped from each marker to the X-axis.
Scatter Plot Line	SPlotScatterLine	If TRUE, all of the scatter plot points are connected with a contiguous line.
Scatter Plot Style	SPlotScatterStyle	Specifies if the symbols in a scatter plot are drawn as filled with color or empty.
(N/A)	SPlotSpline	Turns on and off spline smoothing for the associated line plot. Spline smoothing has no effect on data sets larger than 600 values.

Group Plots Property Page



Caption	Property	Description
Data	Not a property	Index used to select which data set the other parameters apply to. Data sets are indexed from 0 to 31. If a name was assigned to the data set it is displayed along with its index. Only data sets of the <i>Group</i> type are displayed in this drop down list box.
Plot Type	GPlotType	Determines the group plot graph format. Available group plot types are: stacked lines, stacked bars, 3D deep bars, group (side by side) bars, floating bars, high-low-close and error bars.
Reference Axes	GPlotRefAxes	Specifies the reference axes for the selected group plot.

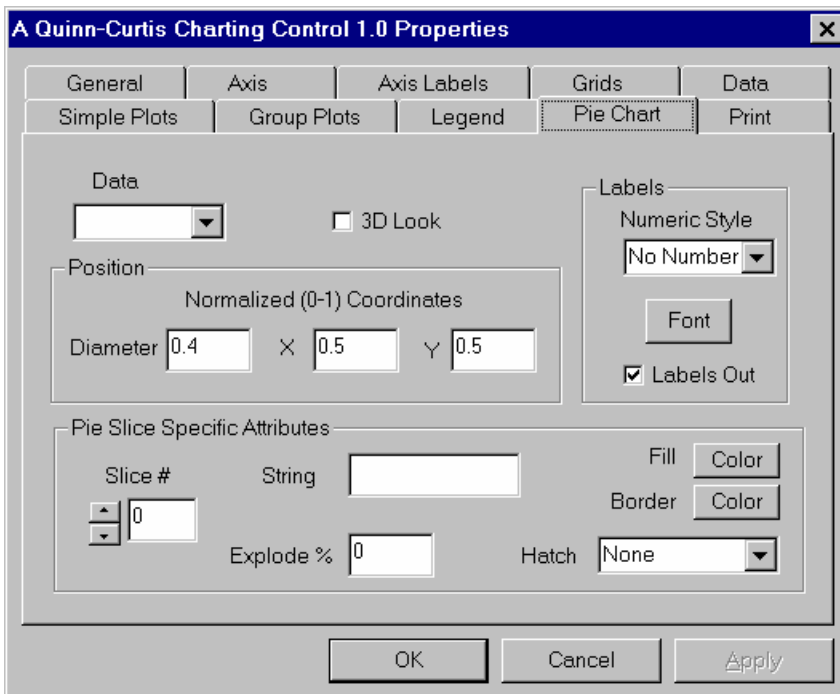
General Attributes

Line Plot Area Fill	GPlotAreaFill	If this flag is TRUE, the area between line plot and the x-axis is filled with color.
Line Plot Width	GPlotLineThickness	Specifies the width, in screen pixels, of the line associated with the selected group line.
Bargraph Type	GPlotBarType	Controls the direction and type of the bars.
Bargraph Justify	GPlotBarJustify	Controls placement of bars. Vertical bars can be left-, center-, or right-justified, horizontal bars can be top-, center-, or bottom-justified, relative to the actual values of the independent variable.
Bargraph Width	GPlotBarWidth	Bar width in the units of the axis the bar width is parallel to.
High-Low-Shape	GPlotScatterShape	Determines the shape of symbols used in the High-Low-Close group chart type.
High-Low Size	GPlotScatterSize	Determines the size of in points (1/72") of the symbols used in the High-Low-Close group chart type.
High-Low-Style	GPlotScatterStyle	Specifies if the symbols in a High-Low-Close group plot are drawn as filled with color or empty.

Group Specific Attributes

Line Plot Color	GPlotLineColor	The RGB color of the group line for a given data set and group.
Line Plot Style	GPlotLineStyle	Specifies the line style of the line associated with the selected group line.
Bar Hatch	GPlotBarHatch	Specifies the hatch style of the bar associated with the selected group bar.

Pie Charts Property Page



Caption	Property	Description
Data	Not a property	Index used to select which data set the other parameters apply to. Data sets are indexed from 0 to 31. If a name was assigned to the data set it is displayed along with its index.

Chapter 4 Property Pages

3D Look	Pie3Dlook	Only data sets of the <i>Pie Chart</i> type are shown in this drop down list box. Determines if the pie chart "look" is two- or three-dimensional.
Position		
X	PieCenterX	Position of the center of the pie relative to the graph width.
Y	PieCenterY	Position of the center of the pie relative to the graph height.
Diameter	PieDiameter	Pie diameter expressed in <i>Graph Normalized</i> coordinates.
Labels		
Numeric Style	PieNumericStyle	Specifies pie slice numeric labeling. Pie slices can be labeled with actual values, percent values, both or neither.
Font	PieFontColor	Color of the numeric text used to label the pie chart.
Font	PieFontSize	Size of the numeric text used to label the pie chart, in points (1/72").
Font	PieFontStyle	Font style for the pie chart numeric text.
Font	PieNumericFont	ASCII name of the pie font typeface.
Labels Out	PieNumberPos	Specifies the placement of the pie slice labels. Labels can be placed inside or outside of the pie slices.
Pie Slice Specific Attributes		
String	PieSliceLabel	Text string to be used as a label for a pie slice.
Explode %	PieSliceExplode	Sets the explosion value for a pie slice relative to the radius (diameter/2) of the pie.
Fill Color	PieSliceFillColor	Color of a pie slice.
Border Color	PieSliceBorderColor	Color of the border for the specified pie slice.
Hatch	PieSliceHatch	Hatch style for a pie slice.

Legends Property Page

A Quinn-Curtis Charting Control 1.0 Properties

General | Axis | Axis Labels | Grids | Data
Simple Plots | Group Plots | **Legend** | Pie Chart | Print

Type: **Simple Legend** | Enable:

Legend Position and Background Colors

Position in Normalized (0-1) Coordinates

Top: | Right:
Left: | Bottom:

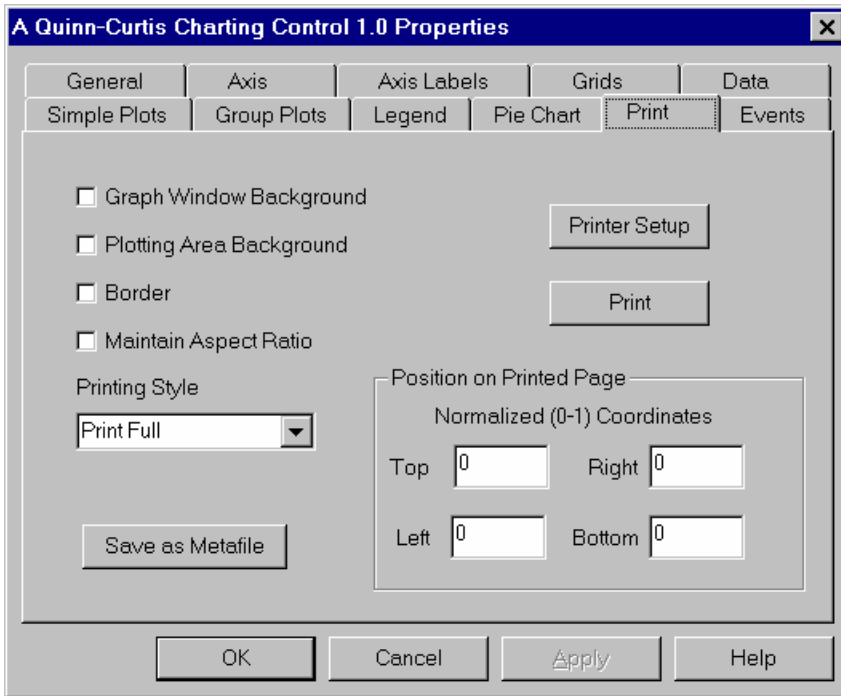
Background:
Border:
Border Thickness:

Legend Strings

Orient Horizontal:

Caption	Property	Description
Type	SLegendType	A legend can either be a simple legend (SIMPLE_LEGEND), or a group legend (GROUP_LEGEND).
Enable	SLegendEnable	Enable legends for the current graph.
Legend Position and Background Colors		
Top	SLegendTop	Position of the top of the legends' bounding rectangle relative to the graph height. Value of 0.0 indicates that the top of the legend rectangle coincides with the top of the graph window.
Left	SLegendLeft	Position of the left side of the legends' bounding rectangle relative to the graph width. Value of 0.0 indicates that the left side of the legend rectangle coincides with the left side of the graph window.
Bottom	SLegendBottom	Position of the bottom of the legends' bounding rectangle relative to the graph height. Value of 1.0 indicates that the bottom of the legend rectangle coincides with the bottom of the graph window.
Right	SLegendRight	Position of the right side of the legends' bounding rectangle relative to the graph width. Value of 1.0 indicates that the right side of the legend rectangle coincides with the right side of the graph window.
Background	SLegendBackgroundColor	Color of the legend rectangle border.
Border	SLegendBorderThickness	Line width of the legend rectangle border, in screen pixels.
Legend Strings		
Strings	SLegendStrings	A string holding the text substrings used to label the legend symbols.
Orient Horizontal	SLegendOrientation	Flag specifying the legend strings layout inside the bounding rectangle.
Font	SLegendFont	ASCII name of the font typeface.
Font	SLegendFontColor	Color of the text in a legend.
Font	SLegendFontSize	Font size in points (1/72").
Font	SLegendFontStyle	Font style for the legend text.

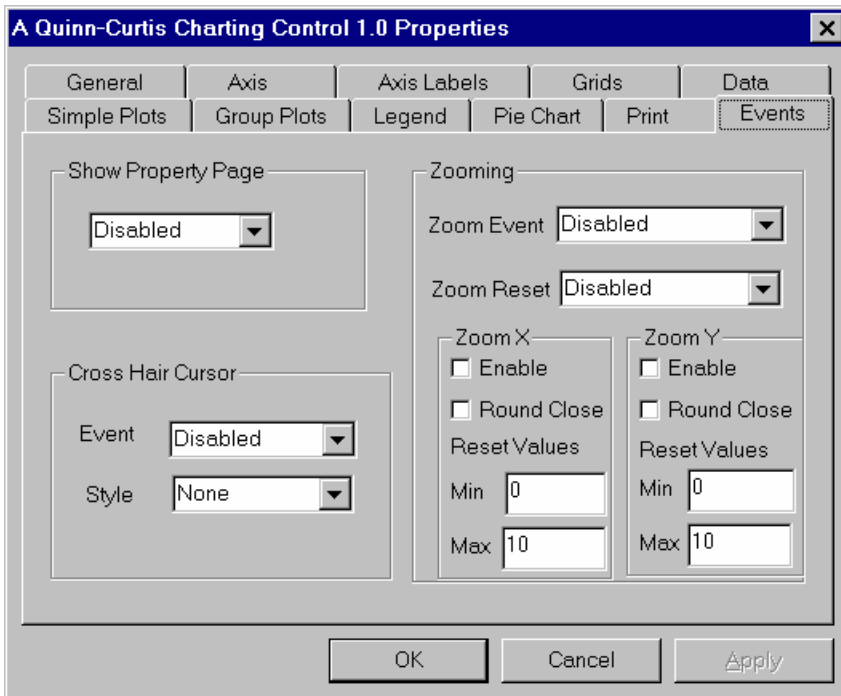
Printing Property Page



Caption	Property	Description
Graph Window Background	PrintGraphBackground	If this flag is TRUE, the background color of a graph is printed, otherwise it is ignored. Default value of this parameter is FALSE.
Plotting Area Background	PrintPlotBackground	If this flag is TRUE, the background color of a graph's plotting area is printed, otherwise it is ignored. Default value of this parameter is FALSE.
Maintain Aspect Ratio	PrintMaintainAspectRatio	If this flag is TRUE, the aspect ratio is maintained when printing with PRT_FULL or PRT_PROP styles. Otherwise the page window is mapped to the full printed page, and the aspect ratio may change.
Border	PrintBorder	If this flag is TRUE, a graph or a page is printed with the rectangular border around it. Default value of this parameter is FALSE.
Print Style	PrintStyle	Specifies the manner in which a page or a graph is printed. The graph can be printed to the full page, or it can be printed
Save as Metafile	SavePageMeta	Save the current graph as a placeable metafile.
Position on Printed Page		
Top	PrintTop	Top edge of the print rectangle, relative to the full size of the printed page. Used when the PrintStyle is set to PRT_EXACT and PRT_POSITIONONPAGE.
Left	PrintLeft	Left edge of the print rectangle, relative to the full size of the printed page. Used when the PrintStyle is PRT_EXACT and PRT_POSITIONONPAGE.

Bottom	PrintBottom	Bottom edge of the print rectangle, relative to the full size of the printed page. Used when the PrintStyle is PRT_POSITIONONPAGE.
Right	PrintRight	Right edge of the print rectangle, relative to the full size of the printed page. Used when the PrintStyle is set to PRT_POSITIONONPAGE.
Printer Setup	PrinterSetup	A methods which invokes the printers setup dialog box.
Print	PrintGraph	Prints the current graph.

Events Property Page



Caption

Show Property Page

Property

EventShowPropPage

Description

Enables “self” display of the property pages. Select either left button double click or right button double click.

Cross Hair Cursor

Event

EventRetrieveDataValue

Enables the data cursor. Select either left button drag or right button drag.

Style

EventDataCursorType

Sets the style of the data cursor. Select from three different types of cross hairs.

Zooming

Zoom Event

EventZoom

Enables zooming. Zooming can be connected to either the left mouse button or the right mouse button.

Chapter 4 Property Pages

Zoom Reset

EventZoomReset

The graph can be reset back to pre-established scaling values. This event can be connected to either a left or right mouse button double click.

Zoom X

Enable

Round Close

Reset Min Value

Reset Max Value

EventZoomX

EventZoomXRoundMode

EventZoomResetMinX

EventZoomResetMaxX

Enables x-axis zooming.

Set zoom rounding of y-axis to far or close.

Set the zoom reset value for the x-axis minimum.

Set the zoom reset value for the x-axis maximum.

Zoom Y

Enable

Round Close

Reset Min Value

Reset Max Value

EventZoomY

EventZoomYRoundMode

EventZoomResetMinY

EventZoomResetMaxY

Enables y-axis zooming.

Set zoom rounding of y-axis to far or close.

Set the zoom reset value for the y-axis minimum.

Set the zoom reset value for the y-axis maximum.

Chapter 5 - Charting Tools ActiveX Property Reference

Properties are specific attributes of an ActiveX control that are exposed to any container. Properties describe a feature of the object, such as the title of the chart or axis extremes. The **Charting Tools ActiveX Control** provides a large number of custom properties allowing the user to modify the appearance, change certain values, or access information about the chart.

The syntax for the custom properties for the **Charting Tools ActiveX Control** are described as Visual Basic standard accessible properties.

```
return = Object.Property           ' no parameters
return = Object.Property(param1, param2) ' includes parameters
```

For example, **WindowBackgroundColor** is the background color property for the graph area. The syntax is :

```
WindowBackgroundColor[= value ]
```

To call a property in Visual Basic, use the following syntax:

```
Long rgb
CTWX1.WindowBackgroundColor
= RGB(255,0,0)
rgb = CTWX1.WindowBackgroundColor
```

Accessing custom properties in Visual C++ is handled a bit differently. When an instance of the **Charting Tools ActiveX Control** is added to the container, a wrapper class for the control is generated. The ClassWizard implements properties by adding a pair of Get/Set functions that allow the control user to access the property.

To call a property in Visual C++, use the following syntax:

```
m_GraphControl.SetWindowBackgroundColor( RGB(255,0,0) );
long rgb = m_GraphControl.GetWindowBackgroundColor();
```

Parameterized Properties

Many of the properties available in this control are “parameterized properties”. A parameterized property exposes a homogeneous collection of values as a single property of the control. The **Charting Tools ActiveX Control** uses parameterized properties to expose axes and data sets as properties.

Visual Basic and Delphi

In Visual Basic and Delphi, a parameterized property is accessed using array notation. To call a parameterized property, use the following syntax:

VB

```
Object.Property(param1) = value 'Set a parameterized property
[return =] Object.Property(param) 'Get a parameterized property
```

Chapter 5 Properties

For example:

```
Dim bStatus as Boolean
CTWX1.AxisAutoAxis(X_AXIS1) = True      'Set the auto axis mode for
                                         X-Axis1
bStatus = CTWX1.AxisAutoAxis(X_AXIS1)'Get the auto axis mode
```

Delphi

```
Object.Property[param1] := value;      //Set a parameterized property
[return :=] Object.Property[param];    //Get a parameterized property
```

For example:

```
bStatus: Boolean;
CTWX1.AxisAutoAxis[X_AXIS1] := TRUE; //Set the auto axis mode for
                                         X-Axis1
bStatus := CTWX1.AxisAutoAxis[X_AXIS1]; //Get the auto axis mode
```

Visual C++

To call a parameterized property use the following syntax:

```
pObject.SetProperty (param, value)
value = pObject.GetProperty (param);
```

For example:

The AxisAutoAxis property has the following syntax:

```
AxisAutoAxis( axis ) [= value ]

m_GraphControl.SetAxisAutoAxis(X_AXIS, True);
BOOL bStatus = m_GraphControl.GetAxisAutoAxis(X_AXIS);
```

Property Reference

AxisAutoAxis

Set the auto-axis mode for the specified axis. Set to TRUE to have the axis range automatically calculated based on the data.

Syntax

```
AxisAutoAxis( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Boolean (BOOL). If set to TRUE, the axis is automatically scaled, otherwise manual scaling is used.

Remarks

1. If **AxisAutoAxis** does not produce axes scaling to your liking, you can explicitly set the axes scaling using the **AxisMin**, **AxisMax**, **AxisMajorTickInterval** and **AxisMinorTicks**.
2. **AxisAutoAxis** will not display “reversed” axes. It will only display axes with the minimum X-value at the left edge and the minimum Y-value at the bottom edge of the plotting area.

3. The endpoints for a logarithmically scaled axis are always at decade intervals, i.e., 0.1, 1, 10, 100, 1000, etc.

See also **AxisMin**, **AxisMax**, **AxisMajorTickInterval** and **AxisMinorTicks**.

AxisColor

Set the color of the selected axis.

Syntax

```
AxisColor( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Long (long). The RGB color of the axis.

Remarks

The color is specified using an RGB color value. When displayed on 256 color displays, the RGB color will be converted to the nearest of 20 pure colors. When displayed on 24-bit color displays (True Color) the color is the exact RGB color specified.

AxisEnable

Enable the selected axis.

Syntax

```
AxisEnable( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Boolean (BOOL). If set to TRUE, the axis is enabled, otherwise the axis is disabled.

AxisGridColor

Color of the grid associated with the given axis.

Syntax

```
AxisGridColor( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.

Chapter 5 Properties

value Long (long). The RGB color of the axis.

Remarks

The color is specified using an RGB color value. When displayed on 256 color displays, the RGB color will be converted to the nearest of 20 pure colors. When displayed on 24-bit color displays (True Color) the color is the exact RGB color specified.

AxisGridEnable

Enable the grid for the selected axis.

Syntax

```
AxisGridEnable( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Boolean (BOOL). If set to TRUE the grid is enabled, otherwise the grid is disabled.

AxisGridLineStyle

Line style for the grid associated with the selected axis.

Syntax

```
AxisGridLineStyle( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Long (long). The line style of the axis.

Remarks

Line styles use the Windows API line style constants: PS_SOLID, PS_DASH, PS_DOT, PS_DASHDOT, and PS_DASHDOTDOT. Only the PS_SOLID linestyle is available for lines with a width greater than 1.

AxisGridLineWidth

Specifies the line width of the grid associated with the selected axis.

Syntax

```
AxisGridLineWidth( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Long (long). The line width of the grid, in screen pixels, .

Remarks

When a graph is sent to the printer, the width of a line relative to the graph width remains constant unless the value of this parameter is 0. In this case, the width of line in device units is always one pixel, resulting in the thinnest possible line.

AxisGridType

Determines if the grid lines are positioned at major or minor tick marks of the axis, or both.

Syntax

```
AxisGridType( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Long (long). The type of the grid.

Remarks

The possible values for the grid type are constants: GRID_MAJOR, GRID_MINOR, GRID_ALL.

AxisIntercept

Value at which the specified axis crosses (intercepts) the axis it is perpendicular to.

Syntax

```
AxisIntercept( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Double (double). The value of the axis intercept.

Remarks

If the selected axis is the Y_AXIS1, then the value of **AxisIntercept(Y_AXIS1)** is the value of X where Y_AXIS1 intercepts X_AXIS1.

AxisInterceptTrack

Forces the intercept of the axis to be placed at the axis extreme of the axis at right angles to it. This will position the selected axis on the outside edge of the graph, regardless of the scaling.

Chapter 5 Properties

Syntax

```
AxisInterceptTrack( axis ) [= value ]
```

Parameter	Description
-----------	-------------

<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Boolean (BOOL). If set to TRUE the auto intercept tracking is enabled, otherwise the auto intercept tracking is disabled.

Remarks

See **AxisIntercept**.

AxisLabelColor

The color of the axis.

Syntax

```
AxisLabelColor( axis ) [= value ]
```

Parameter	Description
-----------	-------------

<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Long (long). The RGB color of the axis labels.

Remarks

The color is specified using an RGB color value. When displayed on 256 color displays, the RGB color will be converted to the nearest of 20 pure colors. When displayed on 24-bit color displays (True Color) the color is the exact RGB color specified.

AxisLabelDecs

The number of digits after the decimal point for labeling the axis tick marks.

Syntax

```
AxisLabelDecs( axis ) [= value ]
```

Parameter	Description
-----------	-------------

<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Long (long). The number of digits after the decimal point.

Remarks

If set to -1 the precision is determined automatically.

AxisLabelFont

The ASCII name of the font typeface for axis labels.

Syntax

```
AxisLabelFont( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	String (CString). The name of the font used for the axis labels.

AxisLabelFontSize

The font size for axis labels specified in points (1/72").

Syntax

```
AxisLabelFontSize( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Long (long) The size in points (1/72") of the font used for the axis labels.

AxisLabelFontStyle

The font style for axis labels.

Syntax

```
AxisLabelFontStyle( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Long (long) The style (bold, italics, underline) of the font used for the axis labels.

Remarks

The font style can be zero, if no special text attributes are required, or a logical OR of the constants TEXT_BOLD, TEXT_ITAL, and TEXT_UNDERLINE.

Chapter 5 Properties

AxisLabelPos

Controls the justification of the labels' position relative to the axis.

Syntax

```
AxisLabelPos( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Long (double). The position of the axis labels relative to the axis tick marks. Use the axis position constants described under Remarks.

Remarks

For vertical axes it must be one of the following values:

<u>Value</u>	<u>Meaning</u>
POS_LEFT	Labels are displayed immediately to the left of the axis.
POS_LEFT_PLOT	Labels are displayed to the left of the plotting area.
POS_RIGHT	Labels are displayed immediately to the right of the axis.
POS_RIGHT_PLOT	Labels are displayed to the right of the plotting area.

For horizontal axes it must be one of the following values:

<u>Value</u>	<u>Meaning</u>
POS_ABOVE	Labels are displayed immediately above the axis.
POS_BELOW	Labels are displayed immediately below the axis.
POS_ABOVE_PLOT	Labels are displayed above the plotting area.
POS_BELOW_PLOT	Labels are displayed below the plotting area.

AxisLabelsEnable

Enable the labels for the selected axis.

Syntax

```
AxisLabelsEnable( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Boolean (BOOL). When TRUE the axis labels are enabled, otherwise the axis labels are disabled.

AxisLabelStrings

A string containing the sub strings used in place of numeric labels for the given axis.

Syntax

```
AxisLabelStrings( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	String (CString). A string which contains the sub strings used in place of numeric labels for the given axis.

Remarks

These string labels are only used if the **AxisLabelStringsEnable** flag is TRUE. The *value* holds n strings, where n is the number of major tick marks defined for the axis. Individual substrings within the **AxisLabelStrings** string are delimited using the CR (ASCII 13) character.

AxisLabelStringsEnable

Enable string labels for the selected axis. If this parameter is FALSE the labels for an axis will default to numeric labeling.

Syntax

```
AxisLabelStringsEnable( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Boolean (BOOL). If TRUE, the AxisLabelStrings string is used to label the major tick marks of the axis, if FALSE the major tick marks are labeled with numbers.

Remarks

See **AxisLabelStrings**.

AxisLabelStringsStart

Sequential number of the major tick where the first label is to be displayed.

Syntax

```
AxisLabelStringsStart( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Long (long). Sequential number of the major tick where the first label is displayed.

Chapter 5 Properties

Remarks

If 0, the first label is displayed at the major first tick, if 1 - at the second major tick, etc. See **AxisLabelStrings**.

AxisLineWidth

Specifies the width, in screen pixels, of selected axis.

Syntax

```
AxisLineWidth( axis ) [= value ]
```

Parameter	Description
-----------	-------------

<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Long (long). The line width of the axis.

Remarks

When a graph is sent to the printer, the width of a line relative to the graph width remains constant unless the value of this parameter is 0. In this case the width of line in device units is always one pixel, resulting in the thinnest possible line.

AxisMajorTickInterval

Distance between major tick marks, in physical units of the X-axis.

Syntax

```
AxisMajorTickInterval( axis ) [= value ]
```

Parameter	Description
-----------	-------------

<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Double (double). The distance between major tick marks, in physical units of the x-axis.

Remarks

Tick marks are drawn in both directions, starting at the Y-intercept of the X-axis. The X-distance between major tick marks for a linear axis is specified by the property **AxisMajorTickInterval**, while the major tick marks for a logarithmically scaled axis are always drawn at decade intervals, i.e., 0.1, 1, 10, 100, 1000, etc.

See **AxisMin**, **AxisMax** and **AxisMinorTicks**.

AxisMax

The value corresponding to the right (x-axis) or top (y-axis) of the graph plotting area.

Syntax

```
AxisMax( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Double (double). The scaling value representing the right (x-axis) or the top (y-axis) of the graph plotting area.

Remarks

The endpoints for a logarithmically scaled axis are always at decade intervals, i.e., 0.1, 1, 10, 100, 1000, etc.

See **AxisMin**.

AxisMin

The value corresponding to the left (x-axis) or bottom (y-axis) of the graph plotting area.

Syntax

```
AxisMin( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Double (double). The scaling value representing the left (x-axis) or bottom (y-axis) of the graph plotting area.

Remarks

The endpoints for a logarithmically scaled axis are always at decade intervals, i.e., 0.1, 1, 10, 100, 1000, etc.

See **AxisMax**.

AxisMinorTicks

Number of minor tick marks between adjacent major ones.

Syntax

```
AxisMinorTicks( axis ) [= value ]
```

Chapter 5 Properties

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Long (long). The number of minor tick marks between adjacent major ones. If it is zero, minor tick marks are not drawn.

Remarks

See **AxisMajorTickInterval**.

AxisNumericStyle

Defines the format of the numeric labels for the axis.

Syntax

```
AxisNumericStyle( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Long (long). The axis labels numeric style. Use the numeric style constants described under Remarks.

Remarks

<u>Value</u>	<u>Meaning</u>
NF_DECIMAL	Labels are displayed as numbers in decimal form with a fixed point.
NF_SCIENTIFIC	Labels are displayed in scientific exponential form.
NF_ENG	For numbers below 1000 this format is equivalent to NF_DECIMAL. Numbers above 1000 are represented by a decimal number between 1 and 999 followed by one of the letters K, M, G, T, which stand for $x10^3$, $x10^6$, $x10^9$, $x10^{12}$ respectively.

AxisScaleMode

Specifies axis scaling mode, either linear or logarithmic.

Syntax

```
AxisScaleMode( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Long (long). The axis scale mode. It can be set to ST_LINEAR or ST_LOG.

Remarks

The endpoints for a logarithmically scaled axis are always at decade intervals, i.e., 0.1, 1, 10, 100, 1000, etc.

See **AxisMin**, **AxisMax** and **AxisMajorTickInterval**, **AxisAutoAxis**.

AxisTickStyle

Position of the axis ticks relative to the axis.

Syntax

```
AxisTickStyle( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Long (long). Ticks position relative to the axis. Use one of the tick positioning constants listed under remarks.

Remarks

Positioning Constants

<u>Value</u>	<u>Meaning</u>
X-AXIS	
POS_LEFT	Position tick mark to the left of the axis.
POS_MIDDLE	Position the tick mark centered vertically on the axis.
POS_RIGHT	Position the tick mark to the right of the axis.
Y-AXIS	
POS_BELOW	Position the tick mark below the axis.
POS_MIDDLE	Position the tick mark centered horizontally on the axis.
POS_ABOVE	Position the tick mark above the axis.

AxisTitleColor

Color of the selected title.

Syntax

```
AxisTitleColor( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Long (long). The RGB color of the axis title.

Remarks

The color is specified using an RGB color value. When displayed on 256 color displays, the RGB color will be converted to the nearest of 20 pure colors. When displayed on 24-bit color displays (True Color) the color is the exact RGB color specified.

Chapter 5 Properties

AxisTitleFont

The ASCII font name for the axis title.

Syntax

```
AxisTitleFont( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	String (CString). The name of the font used for the axis title.

AxisTitleFontSize

The font size for the axis title in points (1/72").

Syntax

```
AxisTitleFontSize( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Long (long). The size in points (1/72") of the font used for the axis title.

AxisTitleFontStyle

The font style for the axis title text.

Syntax

```
AxisTitleFontStyle( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Long (long) The style (bold, italics, underline) of the font used for the axis title.

Remarks

The font style can be zero, if no special text attributes are required, or a logical OR of the constants TEXT_BOLD, TEXT_ITAL, and TEXT_UNDERLINE.

AxisTitlePos

Position of the axis title relative to the plotting area of the graph.

Syntax

```
AxisTitlePos( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	Long (long). Title position relative to the plotting area. Use one of the positioning constants listed under remarks.

Remarks

Positioning Constants

<u>Value</u>	<u>Meaning</u>
X-AXIS	
POS_LEFT	Position the title to the left of the plotting area.
POS_RIGHT	Position the title to the right of the plotting area.
Y-AXIS	
POS_BELOW	Position the title below the plotting area.
POS_ABOVE	Position the title above the plotting area.

AxisTitleString

Text string to be used as the axis title.

Syntax

```
AxisTitleString( axis ) [= value ]
```

Parameter	Description
<i>axis</i>	Long (long). The ID (X_AXIS1, X_AXIS2, Y_AXIS1, or Y_AXIS2) of the axis to modify.
<i>value</i>	String (CString). The axis title.

BottomPlotArea

Position of the bottom of the plotting area relative to the graph height. Value of 1.0 indicates that the bottom of the plotting area coincides with the bottom of the graph window.

Syntax

```
BottomPlotArea [= value ]
```

Parameter	Description
<i>value</i>	Double (double). Position of the bottom of the plotting area relative to the graph height.

Chapter 5 Properties

Remarks

Used in conjunction with **TopPlotArea**, **LeftPlotArea** and **RightPlotArea**.

EventDataCursorType;

Cursor type used for the data cursor. The user can position a data cursor over the plotting area of a graph and retrieve the scaled x and y values for that point.

Syntax

```
EventDataCursorType [= value ]
```

Parameter	Description
<i>value</i>	Long (long). Use one of the data cursor type constants: CROSSHAIR_NONE, CROSSHAIR_GRAPHAREA, CROSSHAIR_PLOTAREA, CROSSHAIR_SMALL.

Remarks

Used in conjunction with **EventRetrieveDataValue**.

EventRetrieveDataValue

Enables and sets the mouse button mode for the data cursor.

Syntax

```
EventRetrieveDataValue [= value ]
```

Parameter	Description
<i>value</i>	Long (long). Use one of the data cursor event constants DATACURSOR_DISABLED, DATACURSOR_LEFTBUTTON, DATACURSOR_RIGHTBUTTON.

Remarks

Used in conjunction with **EventDataCursorType**.

EventShowPropPage

The **EventShowPropPage** property can enable the “self” display of the control’s property pages.

Syntax

```
EventShowPropPage[= value ]
```

Parameter	Description
<i>value</i>	Long (long). Set to 0 disable “self” display of the property pages. Set to WM_LBUTTONDOWNBLCLK to have a left button double click display the property pages, or WM_RBUTTONDOWNBLCLK to have a right button double click display the property pages.

Remarks

It is usually up to the container to invoke the property pages of an ActiveX control. Some containers invoke it with a double click, some with a right button pop-up menu item and some from a main menu item. Some containers have no way of displaying the property pages. It is possible to have the control display its own property pages. The **EventShowPropPage** property can enable the “self” display of the control’s property pages.

EventZoom

The **EventZoom** property will enable zooming.

Syntax

```
EventZoom[= value ]
```

Parameter	Description
<i>value</i>	Long (long). Set to 0 disable zooming. Set to WM_LBUTTONDOWN to enable left button zooming, or WM_RBUTTONDOWN to enable right button zooming.

Remarks

Used in conjunction with **EventZoomReset**, **EventZoomX** and **EventZoomY**.

EventZoomReset

This event can reset the graph back to pre-established values after zooming has altered the scaling of the x and y axes.

Syntax

```
EventZoomReset[= value ]
```

Parameter	Description
<i>value</i>	Long (long). Set to 0 to disable the zooming reset event. Set to WM_LBUTTONDOWNBLCLK to have a left button double click reset the graph, or WM_RBUTTONDOWNBLCLK to have a right button double click reset the graph.

Remarks

Used in conjunction with **EventZoomResetMaxX**, **EventZoomResetMinX**, **EventZoomResetMaxY**, **EventZoomResetMinY**.

EventZoomResetMaxX

Set the value for the x-axis maximum after a zoom reset event.

Syntax

```
EventZoomResetMaxX[= value ]
```

Parameter	Description
<i>value</i>	Double (double). Value for the x-axis maximum after a zoom reset event

Remarks

Used in conjunction with **EventZoomReset**, **EventZoomResetMinX**, **EventZoomResetMaxY**, **EventZoomResetMinY**.

EventZoomResetMaxY

Set the value for the y-axis maximum after a zoom reset event.

Syntax

```
EventZoomResetMaxY[= value ]
```

Parameter	Description
<i>value</i>	Double (double). Value for the y-axis maximum after a zoom reset event

Remarks

Used in conjunction with **EventZoomReset**, **EventZoomResetMinX**, **EventZoomResetMaxX**, **EventZoomResetMinY**.

EventZoomResetMinX

Set the value for the x-axis minimum after a zoom reset event.

Syntax

```
EventZoomResetMinX[= value ]
```

Parameter	Description
<i>value</i>	Double (double). Value for the x-axis minimum after a zoom reset event

Remarks

Used in conjunction with **EventZoomReset**, **EventZoomResetMaxX**, **EventZoomResetMaxY**, **EventZoomResetMinY**.

EventZoomResetMinY

Set the value for the y-axis minimum after a zoom reset event.

Syntax

```
EventZoomResetMinY[= value ]
```

Parameter	Description
<i>value</i>	Double (long). Value for the y-axis minimum after a zoom reset event

Remarks

Used in conjunction with **EventZoomReset**, **EventZoomResetMaxX**, **EventZoomResetMaxY**, **EventZoomResetMinX**.

EventZoomX

Zooming can take place in one or two dimensions. **EventZoomX** enables zooming for the x-axis.

Syntax

```
EventZoomX[= value ]
```

Parameter	Description
<i>value</i>	Boolean (BOOL). Enables zooming for the x-axis.

Remarks

Used in conjunction with **EventZoom** and **EventZoomY**.

EventZoomXRoundMode

The values captured by the zooming rectangle are rounded to produce “cleaner” x and y axes that have minimum and maximum values on even values. The degree of rounding for the x-axis is controlled by the **EventZoomXRoundMode** property

Syntax

```
EventZoomXRoundMode[= value ]
```

Chapter 5 Properties

Parameter	Description
<i>value</i>	Long (long). Set to AS_ROUND_CLOSE for close zoom rounding and AS_ROUND_FAR for wide zoom rounding.

Remarks

Used in conjunction with **EventZoom**, and **EventZoomYRoundMode**

EventZoomY

Zooming can take place in one or two dimensions. **EventZoomY** enables zooming for the y-axis.

Syntax

```
EventZoomY [= value ]
```

Parameter	Description
<i>value</i>	Boolean (BOOL). Enables zooming for the y-axis.

Remarks

Used in conjunction with **EventZoom** and **EventZoomX**.

EventZoomYRoundMode

The values captured by the zooming rectangle are rounded to produce “cleaner” x and y axes that have minimum and maximum values on even values. The degree of rounding for the y-axis is controlled by the **EventZoomYRoundMode** property

Syntax

```
EventZoomYRoundMode [= value ]
```

Parameter	Description
<i>value</i>	Long (long). Set to AS_ROUND_CLOSE for close zoom rounding and AS_ROUND_FAR for wide zoom rounding.

Remarks

Used in conjunction with **EventZoom**, and **EventZoomXRoundMode**

GPlotAreaFill

If this flag is TRUE, the area between line plot and the x-axis is filled with color.

Syntax

```
GPlotAreaFill( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Boolean (BOOL) If this flag is TRUE, the area between line plot and the x-axis is filled with color.

Remarks

The color is specified using an RGB color value. When displayed on 256 color displays, the RGB color will be converted to the nearest of 20 pure colors. When displayed on 24-bit color displays (True Color) the color is the exact RGB color specified.

GPlotBarHatch

Specifies the hatch style codes for bars of the selected plot number and group.

Syntax

```
GPlotBarHatch( dataset, group ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>group</i>	The group number within the plot to modify.
<i>value</i>	Long (long). The hatch style codes for bars of the selected <i>dataset</i> and <i>group</i> .

Remarks

Possible values are the same as Windows hatch styles: HS_HORIZONTAL, HS_VERTICAL, S_FDIAGONAL, HS_BDIAGONAL, HS_CROSS, and HS_DIAGCROSS. A value of -1 means that no hatching is used for the particular group member.

GPlotBarJustify

Controls placement of bars. Vertical bars can be left-, center-, or right-justified, horizontal bars can be top-, center-, or bottom-justified, relative to the actual values of the independent variable.

Syntax

```
GPlotBarJustify( dataset, group ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>group</i>	The group number within the plot to modify.
<i>value</i>	Long (long). The justification value for bars of the selected <i>dataset</i> and <i>group</i> .

Remarks

The possible values of this parameter are constants POS_LEFT, POS_RIGHT, POS_MIDDLE, POS_ABOVE, POS_BELOW.

Chapter 5 Properties

GPlotBarType

Controls the direction and type of the bars.

Syntax

```
GPlotBarType( dataset, group ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>group</i>	The group number within the data set to modify.
<i>value</i>	Long (long). The justification value for bars of the selected <i>dataset</i> and <i>group</i> .

Remarks

It must be one of the following values :

<u>Value</u>	<u>Meaning</u>
LT_VBAR	Vertical two-dimensional bars.
LT_3DVBAR	Vertical three-dimensional bars.
LT_HBAR	Horizontal two-dimensional bars.
LT_3DHBAR	Horizontal three-dimensional bars.

GPlotBarWidth

Bar width in the units of the axis that the bar width is parallel to.

Syntax

```
GPlotBarWidth( dataset, group ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>group</i>	The group number within the data set to modify.
<i>value</i>	Double (double). The bar width in the units of the axis that the bar width is parallel to.

GPlotLineColor

The RGB color of the group line for a given data set and group.

Syntax

```
GPlotLineColor( dataset, group ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>group</i>	The group number within the plot to modify.
<i>value</i>	Long (long). The RGB color for bars of the selected <i>dataset</i> and <i>group</i> .

Remarks

The color is specified using an RGB color value. When displayed on 256 color displays, the RGB color will be converted to the nearest of 20 pure colors. When displayed on 24-bit color displays (True Color) the color is the exact RGB color specified.

GPlotLineStyle

Specifies the line style of the line associated with the selected group line.

Syntax

```
GPlotLineStyle( dataset, group ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>group</i>	The group number within the plot to modify.
<i>value</i>	Long (long). The line style for lines of the selected <i>dataset</i> and <i>group</i> .

Remarks

The styles can have the same values as those for Windows pen styles: PS_SOLID, PS_DASH, PS_DOT, PS_DASHDOT, and PS_DASHDOTDOT. Only the PS_SOLID linestyle is available for lines with a width greater than 1.

GPlotLineThickness

Specifies the width, in screen pixels, of the line associated with the selected group line.

Syntax

```
GPlotLineThickness( dataset, group ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>group</i>	The group number within the plot to modify.
<i>value</i>	Long (long). The width, in screen pixels of lines of the selected <i>dataset</i> and <i>group</i> .

Remarks

When a graph is sent to the printer, the width of a line relative to the graph width remains constant unless the value of this parameter is 0. In this case the width of line in device units is always one pixel, resulting in the thinnest possible line.

GPlotRefAxes

Specifies the reference axes for the selected group plot.

Chapter 5 Properties

Syntax

```
GPlotRefAxes( dataset ) [= value ]
```

Parameter	Description
-----------	-------------

<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long). The reference axes for a group plot. Use AXES1 for the first set of x and y axes (X_AXIS1 and Y_AXIS1) and AXES2 for the second set of x and y axes (X_AXIS2 and Y_AXIS2).

Remarks

If one x-axis and 2 y-axes have been defined, specify AXES2 as the axis value. If X_AXIS2 is not enabled X_AXIS1 will be used with Y_AXIS2.

GPlotScatterShape

Determines the shape of the symbols used in the High-Low-Close group chart type.

Syntax

```
GPlotScatterShape( dataset ) [= value ]
```

Parameter	Description
-----------	-------------

<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long). The value of the scatter plot symbol. Use one of the constants listed under remarks.

Remarks

Use one of the following constants:

MK_X, MK_UPTRIANGLE, MK_DOWNTRIANGLE, MK_BOX, MK_PLUS, MK_ASTERISK, MK_DIAMOND, MK_CIRCLE, MK_DOT.

GPlotScatterSize

Determines the size in points (1/72") of the symbols used in the High-Low-Close group chart type.

Syntax

```
GPlotScatterSize( dataset ) [= value ]
```

Parameter	Description
-----------	-------------

<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long). The size in points of the scatter plot symbol.

GPlotScatterStyle

Specifies if the symbols in a High-Low-Close group plot are filled with color or empty.

Syntax

```
GPlotScatterStyle( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Boolean (BOOL). If TRUE the scatter plot symbol is filled, otherwise it is a hollow symbol.

GPlotType

Determines the group plot graph format. Available group plot types are: stacked lines, stacked bars, 3D deep bars, group (side by side) bars, floating bars, high-low-close and error bars.

Syntax

```
GPlotType( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long). The group plot type. Use one of the group plot constants under Remarks.

Remarks

It can be: GT_STACKEDLINES, GT_STACKEDBARS, GT_GROUPBARS, GT_DEEPBARS, GT_FLOATBARS, GT_HILOCLOSE, and GT_ERRORBARS,

LeftPlotArea

Position of the left side of the plotting area relative to the graph width. Value of 0.0 indicates that the left side of the plotting area coincides with the left side of the graph window.

Syntax

```
LeftPlotArea[= value ]
```

Parameter	Description
<i>value</i>	Double (double). Position of the left side of the plotting area relative to the graph width.

Remarks

Used in conjunction with **TopPlotArea**, **BottomPlotArea** and **RightPlotArea**.

Chapter 5 Properties

MajorTickSize

Length of the major tick marks, in screen pixels.

Syntax

```
MajorTickSize[= value ]
```

Parameter	Description
-----------	-------------

<i>value</i>	Long (long). The length of major tick marks in pixels.
--------------	--

Remarks

This property affects all axes in the graph.

MinorTickSize

New length of the minor tick marks, in screen pixels.

Syntax

```
MinorTickSize[= value ]
```

Parameter	Description
-----------	-------------

<i>value</i>	Long (long). The length of minor tick marks in pixels.
--------------	--

Remarks

This property affects all axes in the graph.

Pie3DLook

Determines if the pie chart “look” is two- or three-dimensional.

Syntax

```
Pie3DLook( dataset ) [= value ]
```

Parameter	Description
-----------	-------------

<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long). It may be one of the two values: PIE_3D or PIE_2D.

PieCenterX

Position of the center of the pie relative to the graph width.

Syntax

```
PieCenterX( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Double (double) A normalized value (0.0 - 1.0) which positions the center of the pie chart, relative to the graph width, within the graph window.

Remarks

Value of 0.5 indicates that the center of the pie is positioned horizontally in the middle of the graph window.

PieCenterY

Position of the center of the pie relative to the graph height.

Syntax

```
PieCenterY( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Double (double) A normalized value (0.0 - 1.0) which positions the center of the pie chart, relative to the graph height, within the graph window.

Remarks

Value of 0.5 indicates that the center of the pie is positioned vertically in the middle of the graph window.

PieDiameter

Pie diameter expressed in graph normalized coordinates.

Syntax

```
PieDiameter( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Double (double) A normalized value (0.0 - 1.0) which sets the diameter of pie chart, relative to the graph width, within the graph window.

Remarks

Value of 0.3 indicates that the pie diameter is 30% the size of the graph window.

PieFontColor

Color of the numeric text labels for the pie chart.

Chapter 5 Properties

Syntax

```
PieFontColor( dataset ) [= value ]
```

Parameter	Description
-----------	-------------

<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long). The RGB color of the axis labels.

Remarks

The color is specified using an RGB color value. When displayed on 256 color displays, the RGB color will be converted to the nearest of 20 pure colors. When displayed on 24-bit color displays (True Color) the color is the exact RGB color specified.

PieFontSize

Size of the numeric text labels for the pie chart, in points (1/72").

Syntax

```
PieFontSize( dataset ) [= value ]
```

Parameter	Description
-----------	-------------

<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long) The size in points (1/72") of the font used for the pie chart text.

PieFontStyle

Font style for the pie chart numeric text.

Syntax

```
PieFontStyle( dataset ) [= value ]
```

Parameter	Description
-----------	-------------

<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long) The style (bold, italics, underline) of the font used for the pie chart text.

Remarks

The font style can be zero, if no special text attributes are required, or a logical OR of the constants TEXT_BOLD, TEXT_ITAL, and TEXT_UNDERLINE.

PieNumericFont

ASCII name of the pie font typeface.

Syntax

```
PieNumericFont( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	String (CString). The name of the font used for the axis labels.

PieNumericStyle

Specifies pie slice numeric labeling. Pie slices can be labeled with actual values, percent values, both or neither.

Syntax

```
PieNumericStyle( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long) The numeric style for labeling of pie slices. See Remarks for a list of available constants.

Remarks

It can be one of the following values:

<u>Value</u>	<u>Meaning</u>
PIE_NUM_NONE	No numeric labeling.
PIE_NUM_PERC	Pie slices are labeled with relative values of each slice in percents.
PIE_NUM_VAL	Pie slices are labeled with their actual numeric values.
PIE_NUM_BOTH	Pie slices are labeled with both relative and actual values.

PieNumberPos

Specifies the placement of the pie slice labels. Labels can be placed inside or outside of the pie slices.

Syntax

```
PieNumberPos( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long) The pie numeric label positioning value. If set to PIE_IN, the labels will be drawn inside pie slices, if set to PIE_OUT, the labels will be drawn outside.

PieSliceBorderColor

Color of the border for the specified pie slice.

Syntax

```
PieSliceBorderColor( dataset, slice ) [= value ]
```

Chapter 5 Properties

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>slice</i>	The pie slice number within the pie chart to modify.
<i>value</i>	Long (long). The RGB color for the border of the of the selected <i>dataset</i> and pie <i>slice</i> .

Remarks

The color is specified using an RGB color value. When displayed on 256 color displays, the RGB color will be converted to the nearest of 20 pure colors. When displayed on 24-bit color displays (True Color) the color is the exact RGB color specified.

PieSliceExplode

Sets the explosion value for a pie slice relative to the radius (diameter/2) of the pie.

Syntax

```
PieSliceExplode( dataset, slice ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>slice</i>	Long (long). The pie <i>slice</i> number within the pie chart to modify.
<i>value</i>	Double (double). The pie slice explosion value, relative to the pie chart radius.

Remarks

For example, if the coefficient for a particular pie slice is set to 0.5, the slice will be exploded out 50% from the center of the pie. If it is set to 0, the slice will not be exploded.

PieSliceFillColor

Color of a pie slice .

Syntax

```
PieSliceFillColor( dataset, slice ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>slice</i>	Long (long). The pie <i>slice</i> number within the pie chart to modify.
<i>value</i>	Long (long). The RGB color for pie slices of the selected <i>dataset</i> and pie <i>slice</i> .

PieSliceHatch

Hatch style for a pie slice.

Syntax

```
PieSliceHatch( dataset, slice ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>slice</i>	Long (long). The pie <i>slice</i> number within the pie chart to modify.
<i>value</i>	Long (long). The hatch style for pie slices of the selected <i>dataset</i> and pie <i>slice</i> .

Remarks

Possible values are the same as Windows hatch styles: HS_HORIZONTAL, HS_VERTICAL, S_FDIAGONAL, HS_BDIAGONAL, HS_CROSS, and HS_DIAGCROSS. A value of -1 means that no hatching is used for the particular pie slice.

PieSliceLabel

Text string to be used as a label for a pie slice.

Syntax

```
PieSliceLabel( dataset, slice ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>slice</i>	Long (long). The pie <i>slice</i> number within the pie chart to modify.
<i>value</i>	String (CString). The text string used to label selected <i>dataset</i> and pie <i>slice</i> .

PlotBackgroundColor

Background color code of the plotting area. The plotting area is the area bounded by the axes.

Syntax

```
PlotBackgroundColor[= value ]
```

Parameter	Description
<i>value</i>	Long (long). The RGB color of the plotting area.

PrintBorder

If this flag is TRUE, a graph or a page is printed with the rectangular border around it. Default value of this parameter is FALSE.

Syntax

```
PrintBorder[= value ]
```

Parameter	Description
<i>value</i>	Boolean (BOOL). If TRUE then print border, if FALSE do not print border.

Chapter 5 Properties

PrintBottom

Bottom edge of the print rectangle, relative to the full size of the printed page. Used when the **PrintStyle** is PRT_POSITIONONPAGE.

Syntax

```
PrintBottom[= value ]
```

Parameter	Description
<i>value</i>	Double (double). A value in the range 0.0 to 1.0 corresponding to the position of the bottom edge of the printing area relative to the height of the printed page.

Remarks

Used in conjunction with **PrintLeft**, **PrintRight** and **PrintTop**.

PrintGraphBackground

If this flag is TRUE, the background color of a graph is printed, otherwise it is ignored. Default value of this parameter is FALSE.

Syntax

```
PrintGraphBackground[= value ]
```

Parameter	Description
<i>value</i>	Boolean (BOOL). If TRUE print graph background, if FALSE do not print graph background.

PrintLeft

Left edge of the print rectangle, relative to the full size of the printed page. Used when the **PrintStyle** is PRT_EXACT and PRT_POSITIONONPAGE.

.

Syntax

```
PrintLeft[= value ]
```

Parameter	Description
<i>value</i>	Double (double). A value in the range 0.0 to 1.0 corresponding to the position of the left edge of the printing area relative to the width of the printed page.

Remarks

Used in conjunction with **PrintBottom**, **PrintRight** and **PrintTop**.

PrintMaintainAspectRatio

If this flag is TRUE, the aspect ratio is maintained when printing with PRT_FULL or PRT_PROP styles. Otherwise the page window is mapped to the full printed page, and the aspect ratio may change.

Syntax

```
PrintMaintainAspectRatio[= value ]
```

Parameter	Description
<i>value</i>	Boolean (BOOL). If TRUE the aspect ratio is maintained, if FALSE, do not maintain aspect ratio.

Remarks

Maintaining the aspect ratio is the only way to print circles and text using their true proportions. Otherwise text may appear elongated and circles will be ellipses.

PrintPlotBackground

If this flag is TRUE, the background color of a graph's plotting area is printed, otherwise it is ignored. Default value of this parameter is FALSE.

Syntax

```
PrintPlotBackground[= value ]
```

Parameter	Description
<i>value</i>	Boolean (BOOL). If TRUE, print plot background, if FALSE do not print plot background.

PrintRight

Right edge of the print rectangle, relative to the full size of the printed page. Used when the **PrintStyle** is set to PRT_POSITIONONPAGE.

.

Syntax

```
PrintRight[= value ]
```

Parameter	Description
<i>value</i>	Double (double). A value in the range 0.0 to 1.0 corresponding to the position of the right edge of the printing area relative to the width of the printed page.

Remarks

Used in conjunction with **PrintBottom**, **PrintLeft** and **PrintTop**.

Chapter 5 Properties

PrintStyle

Specifies the manner in which a page or a graph is printed. The graph can be printed to the full page, or it can be printed to the exact coordinates specified.

Syntax

```
PrintStyle[= value ]
```

Parameter	Description
<i>value</i>	Double (double). Specifies the style in which a page or graph is printed. See Remarks for available options.

Remarks

It can be one of the following:

<u>Value</u>	<u>Meaning</u>
PRT_FULL	Print the graph with the maximum size that a paper sheet allows.
PRT_PROP	Same as PRT_FULL in this implementation.
PRT_EXACT	Maintain the original size of the graph. Only this option guarantees that text and symbols will be printed with their specified size. If any other option is selected, text and symbols are scaled so that their size relative to other objects remains constant. The upper left hand corner of the graph can be positioned using the PrintLeft and PrintTop properties.
PRT_POSITIONONPAGE	Position the graph on the printed page using the rectangle defined by the four properties PrintLeft , PrintTop , PrintRight and PrintBottom .

Default value of this parameter is PRT_FULL.

PrintTop

Top edge of the print rectangle, relative to the full size of the printed page. Used when the **PrintStyle** is set to PRT_EXACT and PRT_POSITIONONPAGE.

Syntax

```
PrintTop[= value ]
```

Parameter	Description
<i>value</i>	Double (double). A value in the range 0.0 to 1.0 corresponding to the position of the top edge of the printing area relative to the height of the printed page.

Remarks

Used in conjunction with **PrintLeft**, **PrintRight** and **PrintBottom**.

RightPlotArea

Position of the right side of the plotting area relative to the graph width. Value of 1.0 indicates that the right side of the plotting area coincides with the right side of the graph window.

Syntax

```
RightPlotArea[= value ]
```

Parameter **Description**

<i>value</i>	Double (double). Position of the right side of the plotting area relative to the graph width.
--------------	---

Remarks

Used in conjunction with **TopPlotArea**, **BottomPlotArea** and **LeftPlotArea**.

SDataEnable

Enable a data set and the associated plot.

Syntax

```
SDataEnable( dataset ) [= value ]
```

Parameter **Description**

<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Boolean (BOOL). TRUE and the plot associated with <i>dataset</i> is enabled, FALSE and the plot is disabled.

Remarks

If a data set is not enabled, it is not selectable in the *Simple Plots* and *Group Plots* property pages.

SDataName

A text string used to identify a data set.

Syntax

```
SDataName( dataset ) [= value ]
```

Parameter **Description**

<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	String (CString). A string used to identify a dataset.

Remarks

The data set name will be displayed next to the data set index in the *Simple Plots and Group Plots* property pages.

SDataNumGroups

The number of groups in a group data set. If a data set is of the *Group* type (GROUP_DATA_TYPE), it can have up to 16 sets of y-values for each set of x-values. The number of columns of y-values is equal to the number of groups. Simple data sets (SIMPLE_XY_DATA_TYPE) can have only one group.

Chapter 5 Properties

Syntax

```
SDataNumGroups( dataset ) [= value ]
```

Parameter	Description
-----------	-------------

<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long). The number of groups in the <i>dataset</i> . Limited to a range of 1-16.

Remarks

1. If **SDataType** is SIMPLE_XY_DATA_TYPE it can have only one group. If the data is to be used with a *Simple* plot type, do not define more than one group.
2. If **SDataType** is PIECHART_DATA_TYPE it can have only one group. The y values for the Pie Chart are not used or displayed in the *Data* property page grid.
3. If **SDataType** is GROUP_DATA_TYPE from 1 to 16 groups can be specified.

See also: **SDataNumPlotPoints** and **SDataType**.

SDataNumPlotPoints

The number of values (rows) for one column of a data set.

Syntax

```
SDataNumPlotPoints( dataset ) [= value ]
```

Parameter	Description
-----------	-------------

<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long). The number of rows in the <i>dataset</i> .

Remarks

There is not practical limit on the number of plot points (rows) in a data set. It is practical to view up to 10,000 rows using the grid control in the *Data* property page. The initialization of the grid control slows down proportional to the number of values displayed.

SDataType

The data set type. There are three data set types: *Simple XY*, *Group* and *Pie Chart*.

Syntax

```
SDataType( dataset ) [= value ]
```

Parameter	Description
-----------	-------------

<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long). The data set type. It should be one of the data set types described under Remarks.

Remarks

It can be one of the following values: SIMPLE_XY_DATA_TYPE, GROUP_DATA_TYPE, or PIECHART_DATA_TYPE. The SIMPLE_XY_DATA_TYPE consists of two columns of data, the

first column contains the x-values for each point, and the second column contains the y-values for each point. The GROUP_DATA_TYPE consists of two or more columns of data, the first column contains the x-values for each point and the other columns contain the values for each group in the data set. The PIECHART_DATA_TYPE consists of one column and it contains the values for a pie chart.

See also: **SDataNumPlotPoints** and **SDataNumGroups**.

SLegendBackgroundColor

Legend rectangle background color.

Syntax

`SLegendBackgroundColor[= value]`

Parameter	Description
<i>value</i>	Long (long). The RGB color of the legend background area.

SLegendBorderColor

Color of the legend rectangle border.

Syntax

`SLegendBorderColor[= value]`

Parameter	Description
<i>value</i>	Long (long). The RGB color of the legend border.

Remarks

The color is specified using an RGB color value. When displayed on 256 color displays, the RGB color will be converted to the nearest of 20 pure colors. When displayed on 24-bit color displays (True Color) the color is the exact RGB color specified.

SLegendBorderThickness

Line width of the legend rectangle border, in screen pixels.

Syntax

`SLegendBorderThickness[= value]`

Parameter	Description
<i>value</i>	Long (long). The line thickness of the legend border.

Chapter 5 Properties

SLegendBottom

Position of the bottom of the legends' bounding rectangle relative to the graph height. Value of 1.0 indicates that the bottom of the legend rectangle coincides with the bottom of the graph window.

Syntax

```
SLegendBottom[= value ]
```

Parameter	Description
<i>value</i>	Double (double). Position of the bottom of the legend rectangle relative to the graph height.

Remarks

Used in conjunction with **SLegendTop**, **SLegendLeft** and **SLegendRight**.

SLegendEnable

Enable legends for the current graph.

Syntax

```
SLegendEnable[= value ]
```

Parameter	Description
<i>value</i>	Boolean (BOOL). TRUE and legends are enable, FALSE and legends are disabled.

SLegendFont

ASCII name of the font typeface.

Syntax

```
SLegendFont( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	String (CString). The name of the font used for the legend text.

SLegendFontColor

Color of the text in a legend.

Syntax

```
SLegendFontColor( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long). The RGB color of the legend text.

Remarks

The color is specified using an RGB color value. When displayed on 256 color displays, the RGB color will be converted to the nearest of 20 pure colors. When displayed on 24-bit color displays (True Color) the color is the exact RGB color specified.

SLegendFontSize

Font size in points (1/72").

Syntax

```
SLegendFontSize( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long) The size in points (1/72") of the font used for the legend text.

SLegendFontStyle

Font style for the legend text.

Syntax

```
SLegendFontStyle( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long) The style (bold, italics, underline) of the font used for the legend text.

Remarks

The font style can be zero, if no special text attributes are required, or a logical OR of the constants TEXT_BOLD, TEXT_ITAL, and TEXT_UNDERLINE.

SLegendLeft

Position of the left side of the legends' bounding rectangle relative to the graph width. Value of 0.0 indicates that the left side of the legend rectangle coincides with the left side of the graph window.

Syntax

```
SLegendLeft [= value ]
```

Chapter 5 Properties

Parameter	Description
<i>value</i>	Double (double). Position of the left edge of the legend rectangle relative to the graph width.

Remarks

Used in conjunction with **SLegendTop**, **SLegendBottom** and **SLegendRight**.

SLegendOrientation

Flag specifying the legend strings layout inside the bounding rectangle.

Syntax

```
SLegendOrientation[= value ]
```

Parameter	Description
<i>value</i>	Long (long). Orientation of the legends within the legend rectangle. Use the orientation constants listed under Remarks.

Remarks

The **SLegendOrientation** property has one of the following values:

Value	Meaning
OR_VERT	Each legend string is displayed under the previous one.
OR_HORZ	Each legend string is displayed on the same line to the right of the previous one. If the width of the bounding rectangle is not sufficient to accommodate all the legends, they will be arranged on more than one line.

SLegendRight

Position of the right side of the legends' bounding rectangle relative to the graph width. Value of 1.0 indicates that the right side of the legend rectangle coincides with the right side of the graph window.

Syntax

```
SLegendRight[= value ]
```

Parameter	Description
<i>value</i>	Double (double). Position of the right edge of the legend rectangle relative to the graph width.

Remarks

Used in conjunction with **SLegendTop**, **SLegendBottom** and **SLegendLeft**.

SLegendStrings

A string holding the text substrings used to label the legend symbols.

Syntax

```
SLegendStrings [= value ]
```

Parameter **Description**

<i>value</i>	String (CString). A string which contains the substrings use to label the legend symbols.
--------------	---

Remarks

Value holds n strings, where n is the number of legends in the legend rectangle. Individual substrings within the **SLegendStrings** string are delimited using the CR (ASCII 13) character.

SLegendTop

Position of the top of the legends' bounding rectangle relative to the graph height. Value of 0.0 indicates that the top of the legend rectangle coincides with the top of the graph window.

Syntax

```
SLegendTop[= value ]
```

Parameter **Description**

<i>value</i>	Double (double). Position of the top of the legend rectangle relative to the graph height.
--------------	--

Remarks

Used in conjunction with **SLegendRight**, **SLegendBottom** and **SLegendLeft**.

SLegendType

A legend can either be a simple legend (SIMPLE_LEGEND), or a group legend (GROUP_LEGEND).

Syntax

```
SLegendType[= value ]
```

Parameter **Description**

<i>value</i>	Long (long). A value, either SIMPLE_LEGEND or GROUP_LEGEND specifying the legend type.
--------------	--

SPlotAreaFill

If this flag is TRUE, the area between line plot and the x-axis is filled with color.

Syntax

```
SPlotAreaFill( dataset ) [= value ]
```

Chapter 5 Properties

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Boolean (BOOL) If this flag is TRUE, the area between line plot and the x-axis is filled with color.

Remarks

The color is specified using an RGB color value. When displayed on 256 color displays, the RGB color will be converted to the nearest of 20 pure colors. When displayed on 24-bit color displays (True Color) the color is the exact RGB color specified.

SPlotBarColor

The RGB color of the bar plot for a given data set.

Syntax

```
SPlotBarColor( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long). The RGB color for bar plot of the selected <i>dataset</i> .

Remarks

SPlotBarHatch

Specifies the hatch style codes for bars of the selected plot.

Syntax

```
SPlotBarHatch( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long). The hatch style codes for bars of the selected <i>dataset</i> .

Remarks

Possible values are the same as Windows hatch styles: HS_HORIZONTAL, HS_VERTICAL, S_FDIAGONAL, HS_BDIAGONAL, HS_CROSS, and HS_DIAGCROSS. A value of -1 means that no hatching is used.

SPlotBarJustify

Controls placement of bars. Vertical bars can be left-, center-, or right-justified, horizontal bars can be top-, center-, or bottom-justified, relative to the actual values of the independent variable.

Syntax

```
SPlotBarJustify( dataset ) [= value ]
```

Parameter Description

dataset Long (long). The *dataset* index number (0 to 31) of the plot to modify.
value Long (long). The justification value for bars of the selected *dataset*.

Remarks

The possible values of this parameter are constants POS_LEFT, POS_RIGHT, POS_MIDDLE, POS_ABOVE, POS_BELOW.

SPlotBarType

Controls the direction and type of the bars.

Syntax

```
SPlotBarType( dataset ) [= value ]
```

Parameter Description

dataset Long (long). The *dataset* index number (0 to 31) of the plot to modify.
value Long (long). The justification value for bars of the selected *dataset*.

Remarks

It must be one of the following values :

Value	Meaning
LT_VBAR	Vertical two-dimensional bars.
LT_3DVBAR	Vertical three-dimensional bars.
LT_HBAR	Horizontal two-dimensional bars.
LT_3DHBAR	Horizontal three-dimensional bars.

SPlotBarWidth

Bar width in the units of the axis the bar width is parallel to.

Syntax

```
SPlotBarWidth( dataset ) [= value ]
```

Parameter Description

dataset Long (long). The *dataset* index number (0 to 31) of the plot to modify.
value Double (double). The bar width in the units of the axis the bar width is parallel to.

SPlotLineColor

The RGB color of the line plot for a given data set.

Chapter 5 Properties

Syntax

```
SPlotLineColor( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long). The RGB color for line plot of the selected <i>dataset</i> .

Remarks

The color is specified using an RGB color value. When displayed on 256 color displays, the RGB color will be converted to the nearest of 20 pure colors. When displayed on 24-bit color displays (True Color) the color is the exact RGB color specified.

SPlotLineStyle

Specifies the line style of the line associated with the selected line.

Syntax

```
SPlotLineStyle( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long). The line style for lines of the selected <i>dataset</i> .

Remarks

The styles can have the same values as those for Windows pen styles: PS_SOLID, PS_DASH, PS_DOT, PS_DASHDOT, and PS_DASHDOTDOT. Only the PS_SOLID linestyle is available for lines with a width greater than 1.

SPlotLineThickness

Specifies the width, in screen pixels, of the line associated with the selected data set.

Syntax

```
SPlotLineThickness( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long). The width, in screen pixels of lines of the selected <i>dataset</i> .

Remarks

When a graph is sent to the printer, the width of a line relative to the graph width remains constant unless the value of this parameter is 0. In this case the width of line in device units is always one pixel, resulting in the thinnest possible line.

SPlotRefAxes

Specifies the reference axes for the selected plot.

Syntax

```
SPlotRefAxes( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long). The reference axes for a simple plot. Use AXES1 for the first set of x and y axes (X_AXIS1 and Y_AXIS1) and AXES2 for the second set of x and y axes (X_AXIS2 and Y_AXIS2).

Remarks

If one x-axis and 2 y-axes have been defined, specify AXES2 as the axis value. If X_AXIS2 is not enabled X_AXIS1 will be used with Y_AXIS2.

SPlotScatterColor

Specifies the color of the symbols in the selected scatter plot.

Syntax

```
SPlotScatterColor( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long). The RGB color for scatter plot symbols of the selected <i>dataset</i> .

Remarks

The color is specified using an RGB color value. When displayed on 256 color displays, the RGB color will be converted to the nearest of 20 pure colors. When displayed on 24-bit color displays (True Color) the color is the exact RGB color specified.

SPlotScatterDrop

If TRUE, a line is dropped from each marker to the X-axis.

Syntax

```
SPlotScatterDrop( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Boolean (BOOL). If TRUE, a line is dropped from each marker to the X-axis.

Chapter 5 Properties

SPlotScatterLine

If TRUE, all of the scatter plot points are connected with a contiguous line.

Syntax

```
SPlotScatterLine( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Boolean (BOOL). If TRUE, all of the scatter plot points are connected with a contiguous line.

SPlotScatterShape

Determines the shape of symbols used in the scatter plot chart type.

Syntax

```
SPlotScatterShape( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long). The value of the scatter plot symbol. Use one of the constants listed under remarks.

Remarks

Use one of the following constants:

MK_X, MK_UPTRIANGLE, MK_DOWNTRIANGLE, MK_BOX, MK_PLUS, MK_ASTERISK, MK_DIAMOND, MK_CIRCLE, MK_DOT.

SPlotScatterSize

Determines the size of in points (1/72") of the symbols used in the scatter plot chart type.

Syntax

```
SPlotScatterSize( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long). The size in points of the scatter plot symbol.

SPlotScatterStyle

Specifies if the symbols in a scatter plot are drawn as filled with color or empty.

Syntax

```
SPlotScatterStyle( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Boolean (BOOL). If FALSE the scatter plot symbol is filled, otherwise it is a hollow symbol.

SPlotSpline

Turns on and off spline smoothing for the associated line plot. Spline smoothing has no effect on data sets larger than 600 values.

Syntax

```
SPlotSpline( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Boolean (BOOL). If TRUE the spline smoothing is used, otherwise spline smoothing is off.

Remarks

Spline smoothing has no effect on data sets larger than 600 values.

SPlotType

Sets the plot type for a simple XY data set. Simple plot types are line plots, bar graphs and scatter plots.

Syntax

```
SPlotType( dataset ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>value</i>	Long (long). The value of the simple chart type. Use one of the constants listed under remarks.

Remarks

It can be: ST_LINEPLOT, ST_BARGRAPH or ST_SCATTERPLOT.

TitleColor

The color of the graph titles.

Chapter 5 Properties

Syntax

```
TitleColor( title ) [= value ]
```

Parameter	Description
-----------	-------------

<i>title</i>	Long (long). The <i>title</i> index number (TITLE1, TITLE2 and FOOTER) of the title to modify. Use the title constants listed under Remarks.
<i>value</i>	Long (long). The RGB color for the color of the selected titles.

Remarks

The color is specified using an RGB color value. When displayed on 256 color displays, the RGB color will be converted to the nearest of 20 pure colors. When displayed on 24-bit color displays (True Color) the color is the exact RGB color specified.

TitleFont

The ASCII name of the font typeface for the graph titles.

Syntax

```
TitleFont( title ) [= value ]
```

Parameter	Description
-----------	-------------

<i>title</i>	Long (long). The <i>title</i> index number (TITLE1, TITLE2 and FOOTER) of the title to modify. Use the title constants listed under Remarks.
<i>value</i>	String (CString). The name of the font used for the title text.

TitleFontSize

The font size in points (1/72") of the graph titles.

Syntax

```
TitleFontSize( title ) [= value ]
```

Parameter	Description
-----------	-------------

<i>title</i>	Long (long). The <i>title</i> index number (TITLE1, TITLE2 and FOOTER) of the title to modify. Use the title constants listed under Remarks.
<i>value</i>	String (CString). The name of the font used for the title text.

TitleFontStyle

The text font style of the graph titles.

Syntax

```
TitleFontStyle( title ) [= value ]
```

Parameter	Description
<i>title</i>	Long (long). The <i>title</i> index number (TITLE1, TITLE2 and FOOTER) of the title to modify. Use the title constants listed under Remarks.
<i>value</i>	Long (long) The style (bold, italics, underline) of the font used for the selected title.

Remarks

The font style can be zero, if no special text attributes are required, or a logical OR of the constants TEXT_BOLD, TEXT_ITAL, and TEXT_UNDERLINE.

TitleString

Text associated with the selected title.

Syntax

```
TitleString( title ) [= value ]
```

Parameter	Description
<i>title</i>	Long (long). The <i>title</i> index number (TITLE1, TITLE2 and FOOTER) of the title to modify. Use the title constants listed under Remarks.
<i>value</i>	String (CString). The title string.

TopPlotArea

Position of the top of the plotting area relative to the graph height. Value of 0.0 indicates that the top of the plotting area coincides with the top of the graph window.

Syntax

```
TopPlotArea[= value ]
```

Parameter	Description
<i>value</i>	Double (double). Position of the top of the plotting area relative to the graph height.

Remarks

Used in conjunction with PlotArea, BottomPlotArea and LeftPlotArea.

WindowBackgroundColor

Background color of the entire graph area.

Syntax

```
WindowBackgroundColor[= value ]
```

Chapter 5 Properties

Parameter	Description
<i>value</i>	Long (long). The RGB color of the graph area.

WindowBorderColor

Border color for the border around the graph area.

Syntax

```
WindowBorderColor[= value ]
```

Parameter	Description
<i>value</i>	Long (long). The RGB color of the graph window.

Remarks

The color is specified using an RGB color value. When displayed on 256 color displays, the RGB color will be converted to the nearest of 20 pure colors. When displayed on 24-bit color displays (True Color) the color is the exact RGB color specified.

WindowBorderStyle

Border style for the border around the graph area.

Syntax

```
WindowBorderStyle[= value ]
```

Parameter	Description
<i>value</i>	Long (long). The border style of the graph window. Use the border style constants listed under Remarks.

Remarks

Value	Meaning
RC_FLAT	No 3D effect.
RC_HIGH	Raised graph.
RC_DEEP	Deep lowered graph.

WindowBorderThickness

Thickness of the lines used in drawing the border around the graph area.

Syntax

```
WindowBorderThickness[= value ]
```

Parameter	Description
<i>value</i>	Long (long). The in thickness in pixels of the graph window border.

XDataValues

Set and retrieve individual x-data values for a data set.

Syntax

```
XDataValues( dataset, index ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>index</i>	The <i>index</i> of a value in the x data array.
<i>value</i>	Double (double). The value stored at <i>index</i> in the x data array.

YDataValues

Set and retrieve individual y-data values for a data set. For simple XY data sets *group* is always 0.

Syntax

```
YDataValues( dataset, index , group ) [= value ]
```

Parameter	Description
<i>dataset</i>	Long (long). The <i>dataset</i> index number (0 to 31) of the plot to modify.
<i>index</i>	The <i>index</i> of a value in the x data array.
<i>group</i>	The group number of a value in the y data array. For simple XY data sets <i>group</i> is always 0.
<i>value</i>	Double (double). The value stored at <i>index</i> in the y data array.

Chapter 6 - Charting Tools ActiveX Method Reference

CopyToClipboard

This method copies the contents of the current graph to the Windows clipboard where it can then be pasted into other documents.

Syntax

```
Function CopyToClipboard As Boolean;
```

```
BOOL CopyToClipboard ()
```

Return Value

The return value specifies the outcome of the method. It is TRUE if the method is successful, FALSE otherwise.

DefineDataSet

Defines a data set based on two one-dimensional arrays - those of independent variable values X and dependent variable values Y. Data is copied from the supplied pointers and is stored local to the control.

Syntax

```
Sub DefineDataSet(dataset As Long,  
                 sName As String,  
                 pXData As Double,  
                 pYData As Double,  
                 nNumDataPoints As Long)
```

```
void DefineDataSet(long dataset,  
                  LPCTSTR sName,  
                  double FAR* pXData,  
                  double FAR* pYData,  
                  long nNumPoints)
```

Parameter	Description
<i>dataset</i>	A number in the range of 0 to 31 designating the data set being defined.
<i>sName</i>	A string declaring a unique name for the data set.
<i>pXData</i>	Pointer to an array of X-values.
<i>pYData</i>	Pointer to an array of Y-values.
<i>nNumPoints</i>	Number of (X, Y) pairs in the data set. Arrays <i>pXData</i> and <i>pYData</i> must both have length not less than <i>nNumPoints</i> .

Remarks

Up to 32 data sets can be defined, each with as many points as you have memory for.

DefineGroupDataSet

Defines a data set based on a one-dimensional array of independent variable's values X and a two-dimensional array of dependent variable's values Y. This kind of a data set can be used for creating group type plots, that is those showing multiple values of Y for every single value of X. The second dimension of the array Y is equal to the number of the group members.

Syntax

```
Sub DefineGroupDataSet(dataset As Long,  
                      sName As String,  
                      pXData As Double,  
                      pYData As Double,  
                      nNumDataPoints As Long,  
                      nNumGroups As Long)
```

```
void DefineGroupDataSet(long dataset,  
                        LPCTSTR sName,  
                        double FAR* pXData,  
                        double FAR* pYData,  
                        long nNumPoints,  
                        long nNumGroups)
```

Parameter	Description
<i>dataset</i>	A number in the range of 0 to 31 designating the data set being defined.
<i>sName</i>	A unique name for the data set.
<i>pXData</i>	Pointer to an array of x-values.
<i>pYData</i>	Pointer to an array of y-values.
<i>nNumPoints</i>	Number of data points in the data set.
<i>nNumGroups</i>	Number of y variables in the group. It is interpreted as a number of columns in a two-dimensional array of y-values. The value of this parameter cannot exceed 16.

Remarks

Certain graph types require data sets with specific group sizes. For example, the High-Low-Close plot type requires a data set defined with *nNumGroups* equal to three, Floating bars and error bars require *nNumGroups* equal to two.

FFTComplexFFT

This method computes direct and inverse fast Fourier transform of a set of complex data values. It uses a modified Cooley-Tukey algorithm.

Syntax

```
Sub FFTComplexFFT(pRealData As Double,  
                 pImData As Double,  
                 nNumDataPoints As Long,  
                 bInverse As Boolean)
```

```
void FFTComplexFFT(double FAR* pRealData,  
                  double FAR* pImData,
```

```
long nNumPoints,
BOOL bInverse)
```

Parameter	Description
<i>pRealData</i>	Pointer to the array of real components of data. On completion of the direct FFT it contains Fourier real coefficients - <i>pRealData[0]</i> holds the DC component, <i>pRealData[1]</i> holds harmonic 1, <i>pRealData[2]</i> holds harmonic 2, etc.
<i>pImData</i>	Pointer to the array of imaginary components of data. On completion of the direct FFT it contains Fourier imaginary coefficients.
<i>nNumPoints</i>	Number of data points, must be a power of 2.
<i>bInverse</i>	Transform direction flag. If it is FALSE, direct Fourier transform, otherwise inverse transform is computed.

Remarks

1. The number of data points must be a power of 2 - i.e. 32, 64, 128, etc.
2. The arrays *pRealData* and *pImData* are used for both input and output to conserve memory. If you want to preserve original data, copy it into other arrays prior to calling this method.
3. The calculated coefficients are relative to the number of data points in the sampled data set.
4. Use the methods **FFTMagnitude**, **FFTPhase**, and **FFTFrequency** to calculate the normalized magnitude, phase angle, and frequency associated with a given harmonic index.

FFTDSPWindow

Windowing functions are frequently used in digital signal processing. One of the common applications is to reduce the undesirable effects related to spectral leakage. Spectral leakage is the result of discontinuities in the sampled waveform. Since the waveform is sampled for a finite length of time, discontinuities occur at the end points of the waveform. Windows are weighting functions applied to the raw data to reduce the spectral leakage associated with the finite observation interval. A window function typically multiplies the data in the sample interval by a function which has a value of one at its center and tapers to zero at both end points.

Syntax

```
Sub FFTDSPWindow(pRealData As Double,
                 pImData As Double,
                 nNumDataPoints As Long,
                 nWindowType As Long)
```

```
void FFTDSPWindow(double FAR* pRealData,
                  double FAR* pImData,
                  long nNumPoints,
                  long nWindowType)
```

Parameter	Description
<i>pRealData</i>	Pointer to an array of real components of data. On completion of this method it contains the windowed values.
<i>pImData</i>	Pointer to an array of imaginary components of data. On completion of this method it contains the windowed values.
<i>nNumPoints</i>	Number of data points, must be a power of 2.
<i>nWindowType</i>	Specifies the window function to apply to sampled data. It can be one of the following six constants:

<u>Constant</u>	<u>Window</u>
-----------------	---------------

Chapter 6 - Methods

DSPWIN_RECTANG	Rectangular window (no window)
DSPWIN_PARZEN	Parzen window
DSPWIN_HANNING	Hanning window
DSPWIN_WELCH	Welch window
DSPWIN_HAMMING	Hamming window
DSPWIN_EXACTB	Exact Blackman window

FFTFrequency

This method calculates the frequency associated with a given harmonic index and sampling frequency.

Syntax

```
Function FFTFrequency(nNumDataPoints As Long,  
                     rSampleFreq As Double,  
                     index As Long) As Double
```

```
double FFTFrequency(long nNumPoints,  
                   double rSampleFreq,  
                   long index)
```

Parameter	Description
<i>nNumPoints</i>	Number of data points in the sampled signal.
<i>rSampleFreq</i>	Sample frequency, in Hz, of the sampled waveform.
<i>index</i>	Harmonic index in the range of 0 to <i>nNumPoints</i> /2.

Return Value

The method returns the harmonic frequency value.

FFTMagnitude

This method calculates the FFT magnitude associated with a given harmonic index. The method uses both positive and negative frequencies in the calculation.

Syntax

```
Function FFTMagnitude(pRealData As Double,  
                    pImData As Double,  
                    nNumDataPoints As Long,  
                    index As Long) As Double
```

```
double FFTMagnitude(double FAR* pRealData,  
                   double FAR* pImData,  
                   long nNumPoints,  
                   long index)
```

Parameter	Description
<i>pRealData</i>	Pointer to an array of double with size of <i>nNumPoints</i> . It is the array of real coefficients resulting from a call to the direct FFTComplexFFT or FFTRealFFT method.

<i>pImData</i>	Pointer to an array of double with size of <i>nNumPoints</i> . It is the array of imaginary coefficients resulting from a call to the direct FFTComplexFFT or FFTRealFFT method.
<i>nNumPoints</i>	Number of data points in the original sampled signal.
<i>index</i>	Harmonic index in the range 0 to <i>nNumPoints</i> / 2.

Return Value

The method returns the normalized magnitude value.

FFTPhase

This method calculates the FFT phase associated with a given harmonic index.

Syntax

```
Function FFTPhase(pRealData As Double,
                  pImData As Double,
                  nNumDataPoints As Long,
                  index As Long) As Double
```

```
double FFTPhase(double FAR* pRealData,
                 double FAR* pImData,
                 long nNumPoints,
                 long index)
```

Parameter	Description
<i>pRealData</i>	Pointer to an array of double with size of <i>nNumPoints</i> . It is the array of real coefficients resulting from a call to the direct FFTComplexFFT or FFTRealFFT method.
<i>pImData</i>	Pointer to an array of double with size of <i>nNumPoints</i> . It is the array of imaginary coefficients resulting from a call to the direct FFTComplexFFT or FFTRealFFT method.
<i>nNumPoints</i>	Number of data points in the original sampled signal.
<i>index</i>	Harmonic index in the range 0 to <i>nNumPoints</i> / 2.

Return Value

The method returns the phase value.

FFTPowerSpectrum

This method calculates the power spectrum periodogram of a sampled data set. The power spectral density of each frequency bin is returned, along with the exact frequency in Hz for each frequency bin.

Method:

If *N* data points of a waveform *w(t)* are collected at equal intervals, then the FFT of the waveform can be denoted as *W(t)*. The periodogram estimate of the power spectrum, *P(k)* is defined as:

=

= $k = 1, 2, \dots, (N/2 - 1)$

Chapter 6 - Methods

=

where k is defined for zero and positive frequencies.

Syntax

```
Sub FFTPowerSpectrum(pRealData As Double,  
                    pImData As Double,  
                    nNumDataPoints As Long,  
                    rInterval As Double)
```

```
void FFTPowerSpectrum(double FAR* pRealData,  
                    double FAR* pImData,  
                    long nNumPoints,  
                    double rInterval)
```

Parameter	Description
<i>pRealData</i>	Pointer to the array of real components of data. On completion of this method it contains the power spectrum of original data. Valid results are in the array elements from 0 to $nNumPoints / 2 - 1$.
<i>pImData</i>	Pointer to the array of imaginary components of data. If data is real, this array should be initialized to all zeros. On completion of FFTPowerSpectrum it contains the frequency values of power spectrum harmonics. Valid results are in the array elements from 0 to $nNumPoints / 2 - 1$.
<i>nNumPoints</i>	Number of data points, must be a power of 2.
<i>rInterval</i>	Time between samples.

Remarks

1. The number of data points must be a power of 2 - i.e. 32, 64, 128, etc.
2. The arrays *pRealData* and *pImData* are used for both input and output to conserve memory. If you want to preserve original data, copy it into other arrays prior to calling this method.

FFTRealFFT

This method computes direct and inverse fast Fourier transforms of a set of real data values. This routine is almost twice as fast as the **FFTComplexFFT** method because it only deals with the real part of a time domain signal.

Syntax

```
Sub FFTRealFFT(pRealData As Double,  
              pImData As Double,  
              nNumDataPoints As Long,  
              bInverse As Boolean)
```

```
void FFTRealFFT(double FAR* pRealData,  
                double FAR* pImData,  
                long nNumPoints,  
                BOOL bInverse)
```

Parameter	Description
<i>pRealData</i>	Pointer to the array of real components of data. On completion of the direct FFT it contains Fourier real coefficients - <i>pRealData[0]</i> holds the DC component, <i>pRealData[1]</i> holds harmonic 1, <i>pRealData[2]</i> holds harmonic 2, etc.
<i>pImData</i>	Pointer to the array of imaginary components of data. On completion of the direct FFT it contains Fourier imaginary coefficients.
<i>nNumPoints</i>	Number of data points, must be a power of 2.
<i>bInverse</i>	Transform direction flag. If it is FALSE, direct Fourier transform, otherwise inverse transform is computed.

Remarks

1. The number of data points must be a power of 2 - i.e. 32, 64, 128, etc.
2. The array *pRealData* is used for both input and output to conserve memory. If you want to preserve original data, copy it into another array prior to calling this method.
3. Use the methods **FFTMagnitude**, **FFTPhase** and **FFTFrequency** to calculate the magnitude, phase angle and frequency associated with a given harmonic index.

GetDatasetMaxX

Retrieve the maximum value in the x array of a data set.

Syntax

```
Function GetDatasetMaxX(dataset As Long) As Double
```

```
double GetDatasetMaxX(long dataset)
```

Parameter	Description
<i>dataset</i>	The <i>dataset</i> index number (0 to 31).

Return Value

The method returns the maximum x value in the data set.

GetDatasetMaxY

Retrieve the maximum value in the y array of a data set.

Syntax

```
Function GetDatasetMaxY(dataset As Long) As Double
```

```
double GetDatasetMaxY(long dataset)
```

Parameter	Description
<i>dataset</i>	The <i>dataset</i> index number (0 to 31).

Return Value

The method returns the maximum y value in the data set.

Chapter 6 - Methods

GetDatasetMinX

Retrieve the minimum value in the x array of a data set.

Syntax

```
Function GetDatasetMinX(dataset As Long) As Double
```

```
double GetDatasetMinX(long dataset)
```

Parameter	Description
<i>dataset</i>	The <i>dataset</i> index number (0 to 31).

Return Value

The method returns the minimum x value in the data set.

GetMousePos

Retrieve the current x and y position of the mouse in the coordinates of the scaled axes.

Syntax

```
Function GetMousePos(nAxes As Long, rXPos As Double, rYPos As Double) As Boolean
```

```
BOOL GetMousePos (long nAxes, double *rXPos, double *rYPos)
```

Parameter	Description
<i>nAxes</i>	Enter 0 for first axes, enter 1 for second axes.
<i>rXPos</i>	Returns the x position of the mouse in the units of the associated x-axis.
<i>rYPos</i>	Returns the y position of the mouse in the units of the associated y-axis.

Return Value

The method returns TRUE if the function is successful.

GetMousePosNorm

Retrieve the current x and y position of the mouse in normalized (0-1) coordinates.

Syntax

```
Function GetMousePosNorm(rXPos As Double, rYPos As Double) As Boolean
```

```
BOOL GetMousePosNorm (double *rXPos, double *rYPos)
```

Parameter	Description
<i>rXPos</i>	Returns the x position of the mouse in normalized coordinates.
<i>rYPos</i>	Returns the y position of the mouse in normalized coordinates.

Return Value

The method returns TRUE if the function is successful.

GetDatasetMinY

Retrieve the minimum value in the y array of a data set.

Syntax

```
Function GetDatasetMinY(dataset As Long) As Double
```

```
double GetDatasetMinY(long dataset)
```

Parameter	Description
-----------	-------------

<i>dataset</i>	The <i>dataset</i> index number (0 to 31).
----------------	--

Return Value

The method returns the minimum y value in the data set.

GetNearestPoint

This method searches all the plotting objects in the specified graph and finds the data point nearest to the point with given physical coordinates. It returns X and Y values of the data point, the index of the data set to which the point belongs, and the sequential number of the point in the data set.

Syntax

```
Function GetNearestPoint(rX As Double,
                        rY As Double,
                        nMode As Long,
                        prX As Double,
                        prY As Double,
                        pnDatasetNum As Long,
                        pnIndexNum As Long) As Boolean
```

```
BOOL GetNearestPoint(double rX,
                     double rY,
                     long nMode,
                     double FAR* prX,
                     double FAR* prY,
                     long FAR* pnDatasetNum,
                     long pnIndexNum)
```

Parameter	Description
-----------	-------------

<i>rX</i>	Physical X-coordinate of the specified point.
-----------	---

<i>rY</i>	Physical Y-coordinate of the specified point.
-----------	---

<i>nMode</i>	Specifies the search criterion. It must be one of the following values :
--------------	--

Value	Meaning
-------	---------

FNP_X	The method looks for the data point with X value closest to <i>rX</i> .
-------	---

FNP_Y	The method looks for the data point with Y value closest to <i>rY</i> .
-------	---

FNP_DIST	The method looks for the data point with the shortest geometrical distance from the point (<i>rX</i> , <i>rY</i>).
----------	--

<i>prX</i>	Returns the physical X-coordinate of the nearest data point.
------------	--

<i>prY</i>	Returns the physical Y-coordinate of the nearest data point.
------------	--

<i>pnDatasetNum</i>	Returns the index to the data set containing the nearest data point.
---------------------	--

<i>pnIndexNum</i>	Returns the index within the data set of the nearest point.
-------------------	---

Chapter 6 - Methods

Return Value

The return value specifies the outcome of the method. It is TRUE if the method is successful, FALSE otherwise.

Remarks

The method **GetNearestPoint** is typically used to find the actual data point nearest to the current mouse cursor position.

GraphZoomReset

Call this function to force the graph to return to the scaling range established using calls to **EventZoomResetMaxX**, **EventZoomResetMinX**, **EventZoomResetMaxY** and **EventZoomResetMinY**.

Syntax

```
Sub GraphZoomReset  
  
void GraphZoomReset ()
```

LoadASCIIDataSet

This method loads data arrays from an ASCII file and creates a data set based on these arrays.

Syntax

```
Function LoadASCIIDataSet(sFilename As String,  
                           sDatasetName As String,  
                           nDataSetType As Long,  
                           nDataSetNum As Long) As Boolean  
  
BOOL LoadASCIIDataSet(LPCTSTR sFilename,  
                      LPCTSTR sDatasetName,  
                      long nDataSetType,  
                      long nDataSetNum)
```

+Parameter	Description
------------	-------------

<i>sFilename</i>	ASCII file name. It can specify the full path or name only, in which case this method searches for a matching file according to the same rules as the Windows OpenFile function.
<i>sDatasetName</i>	Name identifying the new data set.
<i>nDataSetType</i>	
<i>nDataSetNum</i>	The file is loaded into the data set with the index number <i>nDataSetNum</i> .

Return Value.

The return value specifies the outcome of the method. It is TRUE if the method is successful, FALSE otherwise.

Remarks

LoadASCIIDataSet analyzes the ASCII file *sFilename*, determines the number of rows and columns in this file, allocates data arrays of appropriate dimensions. Then it reads the contents of the file to these arrays and creates the data set based on these arrays. This method imports ASCII files with numbers separated by any character that is not a digit, '+', '-', 'e', 'E', or '.'. The file can contain numbers stored in decimal or scientific format. The rows of the file must be terminated with carriage returns.

MarkDataPoint

Marks a data point of a specified static data set as valid or invalid.

Syntax

```
Function MarkDataPoint(dataset As Long,
                      index As Long,
                      state As Boolean) As Boolean
```

BOOL MarkDataPoint(long *dataset*, long *index*, BOOL *state*)

Parameter	Description
<i>dataset</i>	The <i>dataset</i> index number (0 to 31).
<i>index</i>	Sequential number of the data point, zero based.
<i>state</i>	If this flag is set to TRUE, the data point is marked as invalid and is skipped when the data set is plotted using line plot, scatter plot, bar graph, error bars, or HLC plot.

Return Value

The return value specifies the outcome of the method. It is TRUE if the method is successful, FALSE otherwise.

Remarks

When a static data set is plotted as line plot, bar graph, scattered plot, error bars, or high-low-close graph, the points marked as invalid are omitted. Do not use spline interpolation if you want the line plot to omit invalid data points.

MoveDataCursor

Moves a data cursor to the specified location.

Syntax

```
Sub MoveDataCursor (index As Long,
                   rX As Double,
                   rY As Double,
                   bVisible As Boolean) As Boolean
```

```
void MoveDataCursor(long index, double rX,
                   double rY, BOOL bVisible)
```

Parameter	Description
<i>index</i>	The index number of the data cursor (0 to 31).

Chapter 6 - Methods

rX, rY The new location in physical coordinates of the data cursor.
bVisible If this flag is set to TRUE, the data point is marked as invalid and is skipped when the data set is plotted using line plot, scatter plot, bar graph, error bars, or HLC plot.

Remarks

Use this function in conjunction with **SetDataCursor**.

PrinterSetup

This method invokes the printers setup dialog box.

Syntax

```
Sub PrinterSetup  
  
void PrinterSetup()
```

PrintGraph

This method prints the graph using the printer properties.

Syntax

```
Sub PrintGraph  
  
void PrintGraph()
```

ReconnectDataSet

This method disconnects the current data set from the specified graphical object of type GO_PLOT or GO_GROUP and connects the new one to it.

Syntax

```
Sub ReconnectDataSet(dataset As Long,  
                    sName As String,  
                    pXData As Double,  
                    pYData As Double,  
                    nNumDataPoints As Long)  
  
void ReconnectDataSet(long dataset,  
                    LPCTSTR sName,  
                    double FAR* pXData,  
                    double FAR* pYData,  
                    long nNumPoints)
```

Parameter	Description
<i>dataset</i>	A number in the range of 0 to 31 designating the data set being modified with new data.
<i>SName</i>	A new name for the data set.

pXData Pointer to a new array of double X-values.
pYData Pointer to a new array of double Y-values.
nNumPoints Number of (X, Y) pairs in the data set. Arrays *pXData* and *pYData* must both have length not less than *nNumPoints*.

Remarks

This method allows the same graphical object to plot different data sets, possibly of different length. It should be called before the old data set is destroyed.

SaveASCIIDataSet

This method exports the specified data set to an ASCII file.

Syntax

Function SaveASCIIDataSet(*sFilename* As String,
nDataSetNum As Long,
sFileFormatString As String) As Boolean

```
BOOL SaveASCIIDataSet(LPCTSTR sFileName,
                      long nDataSetNum,
                      LPCTSTR sFormatString)
```

Parameter	Description
<i>sFileName</i>	ASCII file name.
<i>nDataSetNum</i>	The dataset index number (0 to 31).
<i>SFormatString</i>	ASCII string containing conversion format specification in the same form as the C printf function, for example, "%9.4F".

Return Value

The return value specifies the outcome of the method. It is TRUE if the method is successful, FALSE otherwise.

Remarks

SaveASCIIDataSet creates an ASCII file and writes to it the ASCII representation of all the numbers in the data set. Each record in the file consists of a value of the independent variable followed by one or more values of dependent variables. Numbers are separated with spaces. Records are terminated with the "carriage return, line feed" sequence. If the file *sFileName* already exists, it is truncated and its previous content is lost.

SavePageMeta

This method saves the specified page with all its graphs as a Windows metafile.

Syntax

Function SavePageMeta(*mftype* As Long,
filename As String) As Boolean

```
BOOL SavePageMeta(long mftype, LPCTSTR filename)
```

Chapter 6 - Methods

Parameter	Description								
<i>mftype</i>	Specifies the metafile type. It can be one of the following: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>MF_WIN</td><td>standard Windows metafile</td></tr><tr><td>MF_PL</td><td>"placeable" metafile</td></tr><tr><td>MF_ENH</td><td>new enhanced metafile</td></tr></tbody></table>	Value	Meaning	MF_WIN	standard Windows metafile	MF_PL	"placeable" metafile	MF_ENH	new enhanced metafile
Value	Meaning								
MF_WIN	standard Windows metafile								
MF_PL	"placeable" metafile								
MF_ENH	new enhanced metafile								
<i>filename</i>	File name. If this parameter is NULL, the dialog box allowing selection of the file name is called.								

Return Value

The return value specifies the outcome of the method. It is TRUE if the method is successful, FALSE otherwise.

Remarks

1. Graphs saved as metafiles can be imported by various applications. For example, most of the illustrations in this manual have been created using Quinn-Curtis **Charting Tools for Windows**, saved as metafiles, and imported by the MS Word word processor.
2. Placeable metafile is a standard Windows metafile that has an additional 22-byte header. The header contains information about the aspect ratio and original size of the metafile, permitting applications to display the metafile in its intended form. Some popular applications can import only placeable metafiles.
3. The Enhanced metafile format has been introduced in Win32 API. It is not supported for 16-bit programs and it is not implemented in Win32s API.

SerializeLoadFile

This method loads a graph from a previously saved serialization file..

Syntax

```
Function SerializeLoadFile (sFilename As String) As Boolean
```

```
BOOL SerializeLoadFile(LPCTSTR sFilename)
```

Parameter	Description
<i>sFilename</i>	File name.

Return Value

The return value specifies the outcome of the method. It is TRUE if the method is successful, FALSE otherwise.

Remarks

Use this method in conjunction with the **SerializeSaveFile** method.

SerializeSaveFile

This method saves the state of the graph to the specified serialization file..

Syntax

Function SerializeSaveFile (sFilename As String) As Boolean

```
BOOL SerializeSaveFile(LPCTSTR sFilename)
```

Parameter	Description
<i>sFilename</i>	File name.

Return Value

The return value specifies the outcome of the method. It is TRUE if the method is successful, FALSE otherwise.

Remarks

Use this method in conjunction with the **SerializeLoadFile** method.

SetDataCursor

Creates a data cursor to highlight a data point.

Syntax

```
Function SetDataCursor(index As Long,
                       type As Long,
                       axis As Long,
                       size As Long,
                       width As Long,
                       style As Long,
                       color As Long) As Long
```

```
long SetDataCursor(long index, long type,
                   long axis, long size,
                   long width, long style,
                   long color)
```

Parameter	Description										
<i>index</i>	The data cursor index. Data cursor numbering has the range 0 to 31.										
<i>type</i>	Data cursor type. It can be one of the following: <table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>DC_VLINE</td> <td>Vertical line.</td> </tr> <tr> <td>DC_HLINE</td> <td>Horizontal line.</td> </tr> <tr> <td>DC_CROSS</td> <td>Cross.</td> </tr> <tr> <td>DC_BOX</td> <td>Box.</td> </tr> </tbody> </table>	Value	Meaning	DC_VLINE	Vertical line.	DC_HLINE	Horizontal line.	DC_CROSS	Cross.	DC_BOX	Box.
Value	Meaning										
DC_VLINE	Vertical line.										
DC_HLINE	Horizontal line.										
DC_CROSS	Cross.										
DC_BOX	Box.										
<i>axis</i>	Enter 0 and the data cursor is tied to X_AXIS1 and Y_AXIS1, enter 1 and the data cursor is tied to X_AXIS2 and Y_AXIS2.										
<i>size</i>	For cross and box cursors this parameter specifies the size of the cursor in points (1/72").										
<i>width</i>	Cursor line width, in screen pixels.										
<i>style</i>	Specifies the pen style.										
<i>color</i>	Color for the cursor lines.										

Chapter 6 - Methods

Remarks

1. Data cursor object is not visible when it is created. To make it visible and change its position use the function **MoveDataCursor**.
2. You can create more than one data cursor in any graph.

SortDataX

This method sorts data arrays belonging to the specified data set in ascending or descending order of independent variable.

Syntax

```
Function SortDataX(dataset As Long,  
                  direction As Boolean) As Boolean
```

```
BOOL SortDataX(long dataset, BOOL direction)
```

Parameter	Description
<i>dataset</i>	The <i>dataset</i> index number (0 to 31).
<i>direction</i>	Specifies sorting direction. If TRUE, sorting is done in ascending order.

SortDataY

This method sorts data arrays belonging to the specified data set in ascending or descending order of dependent variable. It cannot be used with group type data sets containing more than one dependent variable.

Syntax

```
Function SortDataY(dataset As Long,  
                  bDirection As Boolean) As Boolean
```

```
BOOL SortDataY(long dataset, BOOL bDirection)
```

Parameter	Description
<i>dataset</i>	The <i>dataset</i> index number (0 to 31).
<i>direction</i>	Specifies sorting direction. If TRUE, sorting is done in ascending order.

UpdateGraph

Force an update of the current graph.

Syntax

```
Sub UpdateGraph
```

```
void UpdateGraph ()
```


Chapter 7 - FFTs

A large class of numerical analysis tools involves the use of the Fourier analysis. Applications for the Fourier transform include speech, image, radar, and general signal processing.

Implementations of Fourier transform using sampled data on computers are generally referred to as DFT (Discrete Fourier Transform). The equation below computes the DFT, $X(k)$, of the input signal, $x(n)$.

$$X(k) = \sum_{n=0}^{N-1} x(n) * W_N^{kn}$$

where

$$W_N = e^{-j2\pi / N}$$

A complex summation of N complex multiplications is required for each of N samples. This adds up to N^2 complex multiplications and N^2 complex additions to compute an N point DFT. A 1024 point DFT calculated using the equation above would require 4 million floating point multiplications and 4 million floating point additions. In the early 1960's researchers (notably Cooley and Tukey) noticed patterns in the DFT calculation that, when exploited properly, could be used to reduce the number of complex multiplications to $N * \text{Log}_2 N$. The number of floating point multiplications in a 1024 point DFT is reduced by 99%, to 40,000. The entire class of Fast DFT algorithms is known as FFT (Fast Fourier Transform).

FFT Harmonics

The underlying assumption of an FFT is that an arbitrary, sampled signal can be recreated as a summation of other periodic waveforms. The FFT functions will calculate the periodic waveforms that sum up to make the original signal. The FFT functions give you enough information to calculate the frequency, magnitude and phase shift of each harmonic component. The frequency of the harmonic components in the final answer depends on the sample rate used when sampling the original, raw waveform. The FFT can only return frequencies that are simple integer fractions of the original sampling frequency (sample frequency = SF). The range of frequencies will range from 0 (DC level or average value of the signal) to SF/2 (or the Nyquist frequency for the sample rate of SF). For example, assume that 16 samples of a periodic signal are taken at a rate of 480 Hz. The FFT algorithm will calculate the degree to which the following frequencies contribute to the signal makeup.

Harmonic	Frequency
0	(0/16) * 480 = 0 Hz = DC level
1	(1/16) * 480 = 30 Hz
2	(2/16) * 480 = 60 Hz
3	(3/16) * 480 = 90 Hz
4	(4/16) * 480 = 120 Hz
5	(5/16) * 480 = 150 Hz
6	(6/16) * 480 = 180 Hz
7	(7/16) * 480 = 210 Hz
8	(8/16) * 480 = 240 Hz

Chapter 7 - FFTs

If you were interested in the degree to which 60 Hz (AC line frequency) noise was affecting your signal, then you could concentrate on harmonic #2 which in the example above is at 60 Hz. The general equation for calculating the frequency for a given harmonic index is:

Formula 1

$$F = (H/N) * SF$$

where F = frequency of harmonic H,
H = harmonic index (range 0 to N/2),
N = number of sampled data points,
SF = sample frequency.

Use the method **FFTFrequency** to calculate the frequency for a given FFT harmonic.

Assume that you are sampling waveform at 50KHz and you want to resolve individual frequency components down as low as 8 Hz. How many data points do you need to sample? The formula becomes:

$$8 \text{ Hz} = (1 / N) * 50\text{KHz}, \quad \text{or} \\ N = (1/8) * 50\text{KHz} = 6250$$

The FFT algorithms in this package require a number of data points that is a power of 2. You should round the result to the next power of 2 above the calculated value, or 8192. The first harmonic of this sampled waveform becomes $(1/8196) * 50\text{KHz}$, or 6.1 Hz, that is as close as you can get to the original 8 Hz requirement without changing the sample rate.

The FFT routines use the original waveform as input. This is passed to the function as an array (vector or pointer) of floating point numbers. The output of the FFT function is one complex number ($a + b i$) for each harmonic component in the original waveform. The output is passed back as two arrays one of which holds real parts of the complex numbers and another - imaginary parts.

The input to the FFT can be an array of real or complex numbers ($a + b i$). Accordingly, a real (**FFTRealFFT**) or complex (**FFTComplexFFT**) FFT algorithm should be used. FFT can use a considerable amount of memory, so in order to conserve space the results of an FFT calculation are returned in the same arrays as the original waveform. Calculate an N point complex FFT, and the primary results are returned in the first N/2 elements of the real and imaginary arrays. These elements hold the harmonic values for the positive frequencies that make up the waveform. The last half of the arrays hold the harmonic values for the negative frequencies that make up the waveform. The negative frequencies are symmetrical to the positive frequencies about the folding frequency of the harmonic (N/2). The folding frequency represented by the harmonic N/2 is also the Nyquist sample frequency for the original signal. Continuing a previous example, assume that 16 points of a signal are sampled at a rate of 480 Hz. The sampled waveform is contained in the array *w*.

w =
(6.0, 1.85, -2.59, 0.765, 4.0, -0.765, -5.41, -1.85, -2.0, -1.85, -5.41, -0.765, 4.0, 0.765, -2.59, 1.85)

The FFT function returns a complex number for each positive and negative harmonic in the waveform. Assume that the real parts are returned in the vector *va* and the imaginary parts in vector *vb*. The complex number representing the value for harmonic #3 would be $va[3] + vb[3] i$. The complex number for the highest harmonic in a 16 point FFT would be $va[8] + vb[8]i$.

Many questions arise regarding interpretation of the FFT results. The absolute values of a FFT are proportional to the number of data points used in the calculation. Some FFT routines normalize the results of the FFT by dividing the calculated values by N or N/2, while the majority do not. We do not normalize the results because this would make our algorithms different from the majority of popular FFT algorithms described in the literature. This can be confusing to people who have never calculated an FFT before. If you are looking for the relative magnitude of one harmonic versus the other you can ignore the normalization factor. If you need to recover the exact value of a given harmonic, normalize the results of the FFT by N/2 if you intend to use the positive harmonics, and N if you plan to use both positive and negative harmonics. The frequency, magnitude and phase shift for each harmonic index are calculated according to the formulas below. It is assumed that results of the FFT have been returned in vector's *va* (real parts) and *vb* (imaginary parts).

Frequency See Formula 1 above.

Use the method **FFTFrequency** to calculate the frequency for a given FFT harmonic.

FFT Normalized Magnitude

The normalized magnitude takes the square root of the sum of the squares of the real and imaginary parts of the FFT results. It sums the positive and negative frequency halves before the squaring takes place.

Formula 2

$$\text{mag}[0] = \sqrt{va[0]^2 + vb[0]^2} / N$$

(DC component handled separately)

For $k = 1$ to $N/2$

$$\text{mag}[k] = \sqrt{(va[k] + va[N-k])^2 + (vb[k] + vb[N-k])^2} / N$$

The function **FFTMagnitude** can be used to calculate the magnitude for a given FFT harmonic.

FFT Phase Shift

Formula 3

For $k = 0$ to $N/2$

$$\text{phase}[k] = \arctan(vb[k] / va[k])$$

Use the method **FFTPhase** to calculate the phase angle for a given FFT harmonic.

Chapter 7 - FFTs

Typical FFT Results

Element	Sampled Data w[k]	FFT Result va[k]	Frequency	Magnitude	Description
0	6.0	0.0	0.0	0.0	DC component
1	1.85	16	30.0	2.0	
2	-2.59	0.0	60.0	0.0	
3	0.765	0.0	90.0	0.0	
4	4.0	32	120.0	4.0	Positive
5	-0.765	0.0	150.0	0.0	Frequencies
6	-5.41	0.0	180.0	0.0	
7	-1.85	0.0	210.0	0.0	
8	-2.0	0.0	240.0	0.0	Center, Nyquist, or Folding Frequency
9	-1.85	0.0	-210.0		
10	-5.41	0.0	-180.0		
11	-0.765	0.0	-150.0		
12	4.0	32	-120.0		Negative
13	0.765	0.0	-90.0		Frequencies
14	-2.59	0.0	-60.0		
15	1.85	16	-30.0		

Note that there are two frequencies that are not "mirrored" about the center frequency. Element 0, representing harmonic 0, or the DC level, is not mirrored. Element 8 (the N/2 element), the center frequency is also not mirrored. Since the magnitudes take into account the negative frequencies, magnitudes for frequencies higher than the N/2 harmonics are not meaningful. From the table above we see that the signal has significant harmonic components at 30 and 120 Hz.

FFT References

If you want more information about FFT fundamentals, please refer to the following books.

1. Harris, F. J. (1978), "On the Use of Windows for Harmonic Analysis with the Fast Fourier Transform", *Proceedings of the IEEE*, Volume 66, No. 1, January 1978.
2. Rabiner L. R., Gold, B. (1974), *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ.
3. E. Oran Brigham, (1988), *The Fast Fourier Transform and It's Applications*, Prentice-Hall, Englewood Cliffs, NJ.

Index

AxisAutoAxis, 19, 21, 29, 42, 53
AxisColor, 30, 43
AxisEnable, 29, 43
AxisGridColor, 31, 43
AxisGridEnable, 31, 44
AxisGridLineStyle, 31, 44
AxisGridLineWidth, 31, 44
AxisGridType, 31, 45
AxisIntercept, 29, 45
AxisInterceptTrack, 29, 45, 46
AxisLabelColor, 30, 46
AxisLabelDecs, 30, 46
AxisLabelFont, 30, 47
AxisLabelFontSize, 19, 21, 30, 47
AxisLabelFontStyle, 30, 47
AxisLabelPos, 30, 48
AxisLabelsEnable, 30, 48
AxisLabelStrings, 31, 48, 49
AxisLabelStringsEnable, 30, 49
AxisLabelStringsStart, 31, 49
AxisLineWidth, 30, 50
AxisMajorTickInterval, 29, 42, 43, 50, 52, 53
AxisMax, 29, 42, 43, 50, 51, 53
AxisMin, 29, 42, 43, 50, 51, 53
AxisMinorTicks, 29, 42, 43, 50, 51
AxisNumericStyle, 30, 52
AxisScaleMode, 29, 52
AxisTickStyle, 53
AxisTitleColor, 31, 53
AxisTitleFont, 31, 54
AxisTitleFontSize, 31, 54
AxisTitleFontStyle, 31, 54
AxisTitlePos, 54, 55
AxisTitleString, 19, 21, 31, 55
Bar Graph, 14
BottomPlotArea, 27, 55, 65, 75, 89
Coordinate Systems, 12
CopyToClipboard, 93
Data Sets, 12
DefineDataSet, 12, 19, 21, 93
DefineGroupDataSet, 12, 94
Error Bars, 16
EventDataCursorType, 56
EventRetrieveDataValue, 56
EventShowPropPage, 56
EventZoomResetMaxX, 58
EventZoomResetMaxY, 58
EventZoomResetMinX, 58
EventZoomResetMinY, 59
EventZoomXRoundMode, 59
EventZoomYRoundMode, 60
FFTComplexFFT, 94, 96, 97, 98
FFTDSPWindow, 95
FFTFrequency, 95, 96, 99
FFTMagnitude, 95, 96, 99
FFTPhase, 95, 97, 99
FFTPowerSpectrum, 97, 98
FFTRealFFT, 96, 97, 98
Floating Bars, 14
GetDatasetMaxX, 99
GetDatasetMaxY, 99
GetDatasetMinX, 100
GetDatasetMinY, 101
GetMousePos, 100
GetMousePos, 100
GetMousePosNorm, 100
GetMousePosNorm, 100
GetNearestPoint, 101, 102
GPlotAreaFill, 35, 60
GPlotBarHatch, 35, 61
GPlotBarJustify, 35, 61
GPlotBarType, 35, 62
GPlotBarWidth, 35, 62
GPlotLineColor, 35, 62
GPlotLineStyle, 35, 63
GPlotLineThickness, 35, 63
GPlotRefAxes, 34, 63, 64
GPlotScatterShape, 35, 64
GPlotScatterSize, 35, 64
GPlotScatterStyle, 35, 65
GPlotType, 34, 65
GraphZoomReset, 102
Grouped Bar Graphs, 15
High-Low-Close, 16
LeftPlotArea, 27, 56, 65, 75, 89
Line Marker Plot, 13
Line Plot, 13
LoadASCIIDataSet, 102, 103
MajorTickSize, 66
MarkDataPoint, 103
MinorTickSize, 66
MoveDataCursor, 103
MoveDataCursor, 103
MoveDataCursor, 103
Pie Chart, 16
Pie3DLook, 66
PieCenterX, 36, 66
PieCenterY, 36, 67
PieDiameter, 36, 67
PieFontColor, 36, 67, 68
PieFontSize, 36, 68
PieFontStyle, 36, 68
PieNumberPos, 36, 69
PieNumericFont, 36, 68

Index

PieNumericStyle, 36, 69
PieSliceBorderColor, 36, 69
PieSliceExplode, 36, 70
PieSliceFillColor, 36, 70
PieSliceHatch, 36, 70
PieSliceLabel, 36, 71
PlotBackgroundColor, 27, 71
Plotting Area, 12
PrintBorder, 38, 39, 71
PrintBottom, 39, 72, 73, 74
PrinterSetup, 104
PrintGraph, 104
PrintGraphBackground, 38, 39, 72
PrintLeft, 38, 72, 73, 74
PrintMaintainAspectRatio, 38, 39, 73
PrintPlotBackground, 38, 39, 73
PrintRight, 39, 72, 73, 74
PrintStyle, 38, 39, 40, 72, 73, 74
PrintTop, 38, 72, 73, 74
ReconnectDataSet, 104
RightPlotArea, 27, 56, 57, 65, 74, 75
SaveASCIIDataSet, 105
SaveMetafile, 38
SavePageMeta, 104, 105
Scatter Plot, 13
SDataEnable, 19, 21, 22, 32, 75
SDataName, 32, 75
SDataNumGroups, 32, 75, 76, 77
SDataNumPlotPoints, 32, 76, 77
SDataType, 19, 21, 32, 76
SerializeLoadFile, 28, 106, 107
SerializeSaveFile, 28, 106, 107
SetDataCursor, 107
SLegendBackgroundColor, 37, 77
SLegendBorderColor, 77
SLegendBorderThickness, 37, 77
SLegendBottom, 37, 78, 80, 81
SLegendEnable, 19, 22, 37, 78
SLegendFontColor, 37, 78
SLegendFontSize, 37, 79
SLegendFontStyle, 37, 79
SLegendLeft, 37, 78, 79, 80, 81
SLegendOrientation, 37, 80
SLegendRight, 37, 78, 80, 81
SLegendStrings, 19, 37, 80, 81
SLegendTop, 37, 78, 80, 81
SLegendType, 37, 81
SortDataX, 108
SortDataY, 108
SPlotAreaFill, 33, 81
SPlotBarColor, 19, 21, 33, 82
SPlotBarHatch, 33, 82
SPlotBarJustify, 33, 82, 83
SPlotBarType, 33, 83
SPlotBarWidth, 34, 83
SPlotLineColor, 19, 21, 33, 83, 84
SPlotLineStyle, 33, 84
SPlotLineThickness, 33, 84
SPlotRefAxes, 33, 85
SPlotScatterColor, 34, 85
SPlotScatterDrop, 34, 85
SPlotScatterLine, 34, 86
SPlotScatterShape, 34, 86
SPlotScatterSize, 34, 86
SPlotScatterStyle, 34, 86, 87
SPlotSpline, 34, 87
SPlotType, 19, 21, 22, 33, 87
Stacked Line Plot, 15
TitleColor, 28, 87, 88
TitleFontSize, 28, 88
TitleFontStyle, 28, 88
TitleString, 19, 22, 28, 89
TopPlotArea, 27, 56, 65, 75, 89
UpdateGraph, 108
UpdateGraph, 108
WindowBackgroundColor, 27, 41, 89
WindowBorderColor, 28, 90
WindowBorderStyle, 28, 90
WindowBorderThickness, 28, 90
XDataValues, 32, 91
YDataValues, 32, 91