

Charting Tools for Windows

Revision 3.0

Windows 95
Windows 98
Windows NT

- Revision 3.0 includes 32-bit DLLs only**
- 32-bit DLLs** - for Visual C++ 5.0 and higher, Borland Delphi 3.0 and higher, and Visual Basic 5.0 and higher.
- Scientific and Business Charting Library for Windows**
- Built-In Dialog Boxes for Editing Chart Characteristics**
- Presentation-Quality Hardcopy Output**
- No Runtime Royalties**
- Many Different Chart Types to Choose From** - line plots, area plots, horizontal and vertical bar graphs (with 3-D bar options), floating bars, scatter plots, group plots, high-low-close plots, error bar plots, pie charts (with 3-D pie options), contour plots, waterfall plots, stock market open-high-low-close plots, candlestick plots and box-whisker plots.
- Scaling and Grid Options** - charts can have linear, logarithmic and semi-log scaling.
- Graph Enhancements** - add labels, legends, text, arrows and geometric drawings to graphs.
- Windows Bitmap and Metafile Support** - bitmaps and metafiles can be added to charts.
- Automatic Axes** - scale, draw x- and y-axes, and label axes with a single function call. Special routines to handle multiple data sets, group data and contour data.
- Import Charts Into Other Windows Applications** - metafiles and bitmaps (DIBs).
- Spline Smoothing and Data Reduction**
- Legends for All Graph Types**
- Built-in Mouse Support** - click on chart objects like data plots, axes, titles, legend windows, and pop-up the associated dialog box. Move graphical objects (axes, legends, text and data) with the mouse
- Zooming and Data Cursors** - data markers, XOR data cursors and super zoom support for multiple x and y-axes.
- Bad Data Point Handling**
- C++, Microsoft Foundation Class (MFC)** - example programs are provided. The *Graphics Class Libraries (GCL)* for MFC, available as an add-on product, integrate these tools with the MFC Document/View architecture. See the *GCL* data sheets for detailed information.
- FFT Functions** - real FFT, complex FFT, power spectrum, magnitude and phase functions.
- Documentation and Demo/Example Programs** - the software includes a comprehensive 450 page manual with many programming examples, and 30 complete demo programs (supplied on disk).
- Source Code Available** - see description of the WIN-BMC-102, WIN-VB-102, and WIN-BD-102 products.



The Charting Tools for Windows will add spectacular scientific and business graphics to your Windows programs.

Quinn-Curtis, Inc. presents the **Charting Tools for Windows**. This tool kit helps you create and incorporate sophisticated business and scientific charts into your own Windows applications. This product includes 32-bit DLLs that can be used to create 32-bit applications for Windows NT, Windows 95 and Windows 98.

A comprehensive library of functions supports a wide variety of chart types including line plots, area plots, horizontal bars, scatter plots, group plots, open-high-low-close plots, contour plots, and pie charts. Once the charts are created they can be output to Windows supported printers at the resolution of the output device, creating presentation quality charts, slides or transparencies. There are no royalties or runtime fees when these tools are used to create a Windows application program.

The main part of this software is provided as a Windows dynamic link library (DLL). The DLL for all versions of the **Charting Tools for Windows** is the same and the source code for the DLL is written in C and compiled using Visual C++.

ORDERING INFORMATION

PART #	DESCRIPTION	PRICE
WIN-BMC-100	Charting Tools - C/C++	\$300
WIN-BMC-101	Charting Tools DLL Source	\$300
WIN-BMC-102	Tools and Source for C/C++	\$600
WIN-BD-100	Charting Tools - Delphi	\$300
WIN-BD-102	Tools and Source for Delphi	\$600
WIN-VB-100	Charting Tools - Visual Basic	\$300
WIN-VB-102	Tools and Source for Visual Basic	\$600

SHIPPING CHARGES

UPS Ground	UPS Blue	UPS Red	DHL	Canada
11	18	28	44	15

Product Descriptions

WIN-BMC-100

DLL

Charting Tools for Windows for C/C++

This product is the base version of the Quinn-Curtis **Charting Tools for Windows** package. It contains a 450 page user manual, the Quinn-Curtis **Charting Tools** 32-bit DLL, C interface files, and demo programs for Visual C++. The source code to the Quinn-Curtis **Charting Tools** DLL is sold separately as the software package WIN-BMC-101.

WIN-BMC-101

Source

Charting Tools for Windows for C/C++

This product contains the commented source code to the Quinn-Curtis **Charting Tools for Windows** DLL. It cannot be used unless the user also has certain support files which are part of the WIN-BMC-100 software package.

WIN-BMC-102

DLL + Source

Charting Tools for Windows for C/C++

Combines the WIN-BMC-100 and the WIN-BMC-101 software.

WIN-VB-100

DLL

Charting Tools for Windows for Visual Basic

The Visual Basic version of WIN-BMC-100. All of the documentation and example programs are specific to Visual Basic. A 2nd DLL (source code provided in C) is used in addition to the main **Charting Tools for Windows** DLL to interface to Visual Basic. 32-bit DLLs are included.

WIN-VB-102

DLL + Source

Charting Tools for Windows for Visual Basic

The Visual Basic version of WIN-BMC-102. The source code of the **Charting Tools for Windows** DLL is written in C.

WIN-BD-100

DLL

Charting Tools for Windows for Delphi

The Borland Delphi version of WIN-BMC-100. All of the documentation and example programs are specific to Borland Delphi. A 32-bit DLL is included.

WIN-BD-102

DLL + Source

Charting Tools for Windows for Delphi

The Borland Delphi version of WIN-BMC-102. The source code of the **Charting Tools for Windows** DLL is written in C.

Introduction to Charting Tools for Windows Terminology

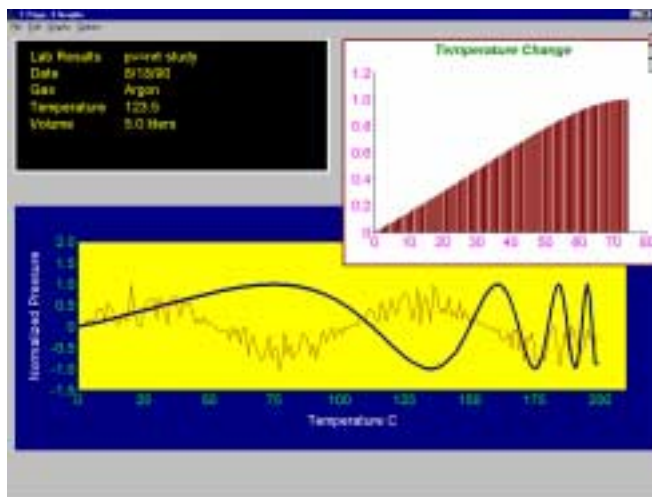
Handle

You will see the term handle referenced throughout these data sheets as well as Windows documentation. Handles are simply numbers used to identify objects of any kind. A handle can be interpreted as a unique name given to an object. After a handle is associated with an object, you can always reference that object by its handle.

Microsoft Windows uses handles to reference windows, menus, dialog boxes, memory blocks, GDI objects, etc. In addition to that, in the Quinn-Curtis **Charting Tools for Windows**, handles are used to reference data sets, graphical objects and pictures consisting of many graphical objects.

Page

Page is an entity based on a window that can contain one or more graphs as child windows. If page windows are created with the recommended default style, as pop-up overlapped windows, they are movable and sizable. They can be printed or copied to the Clipboard. They may have a title bar and a sizable window frame. They can be minimized and maximized. A page window may have a menu bar. A page window can be owned by an application's main window or any other user created window. Up to sixty pages can simultaneously exist in one application.



Multiple graphs can be combined in the same page window.

Graph

Graph is an entity based on a child window belonging to a parent page window. It contains a plotting area and graphical objects, like axes, plotted data and text. Every page can contain up to sixteen graphs. A graph window can be individually printed or copied to the Clipboard. Graphs belonging to the same page can be placed next to each other or superimposed.

Plotting Area

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

Graphical Objects

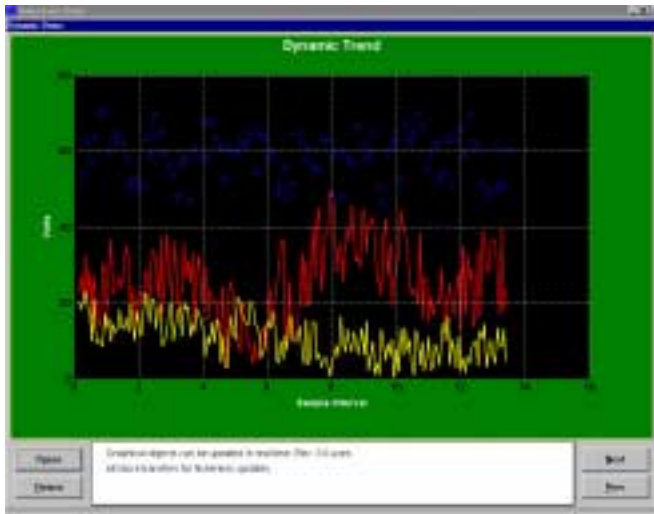
Every graph consists of a number of graphical elements or objects. A graphical object may represent a high level entity, like a pie chart or bar graph, or a drawing element, like a straight line. It is described by a structure which depends on

the object type. Every graphical object is identified by the graph to which it belongs, and a handle. One graph can have up to sixty graphical objects.

Coordinate Systems

Dimensions and positions of windows and graphical objects can be expressed in different units based on the chosen coordinate system. Three different coordinate systems are supported - Page Normalized, Graph Normalized, and Physical. They all are device independent.

A programmer can specify the position of a graph inside a page window using Page Normalized coordinate system. In this system the upper-left corner of a page window has coordinates (0.0, 0.0) and the lower-right corner has coordinates (1.0, 1.0).



Data in displayed in a Charting Tools graph can be updated in real-time (though not as nearly as fast as with our **Real-Time Graphics Tools**).

Positions of graphical objects in a graph can be specified using Graph Normalized or Physical coordinate system. 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 allows you to position a graphical object at the location specified by a pair of (X, Y) data values based on a given pair of axes.

Adding Charts to Your Program

Creating a Page Window

Graphics pages are created with the function **WGCreatePage**. It can be issued anywhere in the application program's main window procedure, or in any other user window procedure. Parameters passed to **WGCreatePage** specify the page window parameters.

Two different types of image scaling can be chosen when a page is created - fixed and proportionate. If the fixed mode is chosen, the sizes of the graphs and their graphical objects will always remain the same regardless of the page window size. For example, if the window is reduced in size, the image is clipped. In proportionate mode the sizes of all the graphs and their graphical objects will be always proportionate to the page window size. The aspect ratio will not be preserved.

Creating a Graph Window

To draw a graph in the graph window the user has to write a graph building procedure containing a sequence of calls to graphical object drawing functions. The graph building procedure will be called from the graph window procedure which is invisible to the user. The syntax of this user written procedure must be as in the following example:

```
// C code
void FAR DrawPlGl (PGRAPH_DEF pGrDesc, HDC hdc);

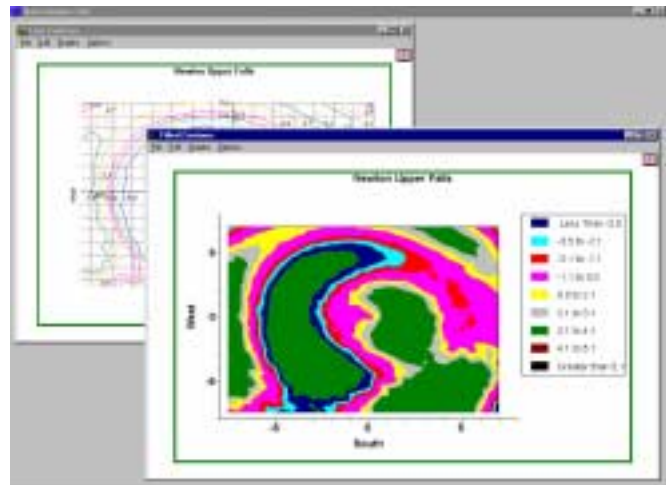
` Visual Basic code
Sub DrawPlGl (pGrDesc As Long, hdc As Integer)
```

The first Quinn-Curtis function called from the graph building procedure should normally be **WGSetPlotArea**. (It is not required only if the chart has no axes, for example, it only contains a pie chart.) It determines the size, position, and the background color of the plotting area. If any axes are going to be part of the graph, a call to **WGScalePlotArea** followed by axes drawing functions, or **WGAutoAxes** must be issued. Then follows a sequence of calls to drawing functions building the graph. No actual drawing takes place at this time. What really happens - the graphical objects are created, and all the information passed to these functions is saved in the graphical objects' descriptors. After the building function finishes its job, the graph window is forced to update, and that is the time when actual drawing takes place.

Elements of a Graph

Axes

A great deal of effort has gone into making the axis drawing and scaling as painless as possible. The X- and Y-intercepts of the axes are adjustable, along with tick mark spacing. Individual axes can be either linear or logarithmic and a combination of two can produce linear, logarithmic or semi-log graphs. Multiple sets of X- and Y-axes, each with different scaling and X- and Y-intercepts, can reside in a single graph. The easiest way of building a graph involves using the **WGAutoAxes** function. It will analyze a set of (X,Y) data and automatically choose the appropriate axes scaling. **WGAutoAxes** will also make sure that the tick mark spacing is in round increments, making the graph easy to read. You



Rev. 3.0 adds line and area fill contour plotting to the **Charting Tools**.

If more control over axes is required, the graph can be drawn by directly choosing the X- and Y-intercepts (**WGSetXYIntercepts**), scaling the plotting area to the exact ranges (**WGScalePlotArea**), drawing the X- and Y-axes with the specified number of tick marks (**WGDrawXAxis**, **WGDrawYAxis**) and labeling the major tick marks (**WGLabelAxis**). Axes can be labeled with numeric values in decimal, engineering and scientific formats, or with text strings.



The new Rev. 3.0 stock market Open-High-Low-Close chart, combined with an Area Fill chart, is an excellent way of displaying stock market data. Rev. 3.0 also adds automatic date calculations for axis labeling.

Plotting Data

There are nine basic data chart types. These are bar graphs, floating bar graphs, grouped bar graphs, high-low-close plots, line plots, line plots with markers, error bars, pie charts and scatter plots. One or more basic data plotting functions can be used in the same graph window. Most of the basic data plotting functions have additional modes that create variations on the basic chart types.

Line Plot - This is a basic plot type in which adjacent data points in a graph are connected with straight lines. It has a 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 also controllable.

Scatter Plot - Every 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. The size, color and fill of the symbol 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. 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 - The 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), on top of one another (cumulative "stacked" format), or behind one another at 45 degrees ("deep" 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. It 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.

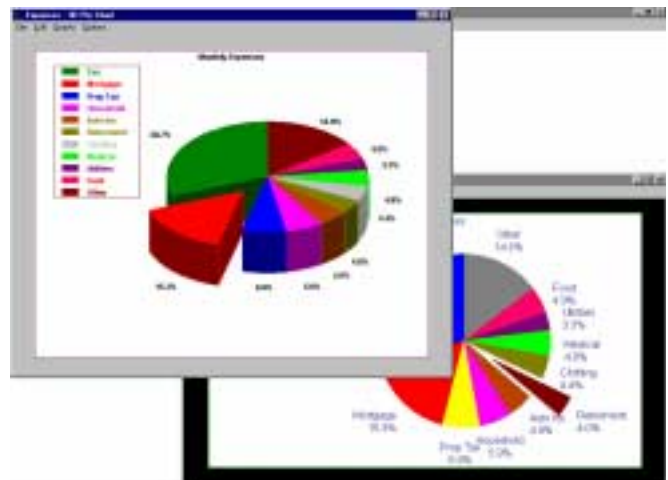
Open-High-Low-Close - Two types of Open-High-Low-Close charts are supported. The first is the standard "stock market" method of presenting data which displays the high and low value as a vertical line, the open and close values as horizontal lines intersecting the vertical line. The other type uses scatter plot symbols to represent the open and close values.

Error Bars - The error 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. They can be two- or three-dimensional. Colors, fill patterns, text, exploding of pie sections are all under the users control.

Contour Chart - Contour charts will display 3D data in standard 2D chart format. Line contour charts and filled contour charts are supported. Evenly spaced xyz grids and randomly spaced xyz data is supported.

Candlestick Chart - This function creates a "Candlestick" type plot for the specified group data set. Every item of the

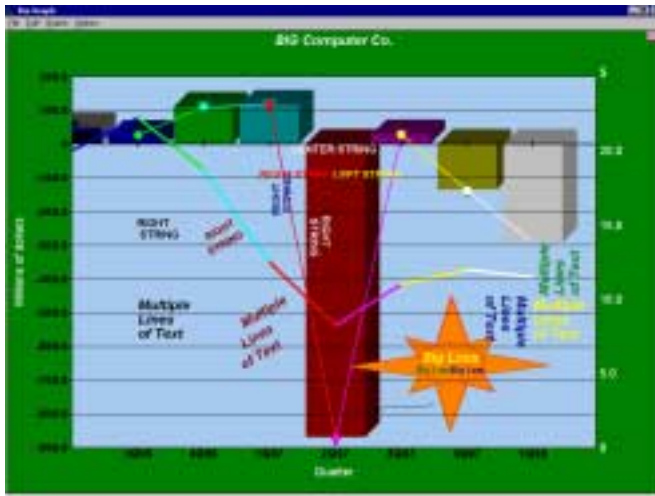


Pie Charts have 2-D/3-D, color, fill type, font and explode options.

plot is a group of two horizontal lines representing “High” and “Low” values which are connected with a vertical line and a box representing the “Open” and “Close” values. If the “Open” value is greater than the “Close” value for a particular group the box is filled, otherwise it is unfilled

Box Whisker Chart - Creates a “Box and Whisker” type plot for the specified group data set. A box is placed around the midpoint (i.e., mean or median) representing a selected range (percent, min-max, constant, standard deviation, or standard error) and whiskers outside of the box representing a selected range (percent, min-max, constant, standard deviation, standard error, or outliers). Outliers and extreme values can be represented as markers.

Waterfall Plot - A group plot types displays data as a waterfall plot where each group is stacked behind the other along a diagonal.



Rev. 3.0 adds multi-line text and the ability to assign individual colors to bars and line segments of charts objects.

Titles and Labels

Each graph can have a main title that is centered at the top of the graph. Every axis can have a title that is positioned parallel to it. Besides, every axis can have numerical or text labels positioned next to major tick marks. It is also possible to add to a graph other text strings. They can be positioned and oriented exactly as desired.

Fonts and Text Attributes

Fonts used for all kinds of text in the graphs can be selected using **WGSetTextParams** and **WGSetTextByName** functions. The former uses font family for font selection. Font families describe the look of a font in a general way. They are intended for specifying fonts when it is not known if the exact typeface desired is available. The **WGSetTextByName** function specifies the exact typeface of the font. The font and its parameters set by one of these functions remain in effect for all the subsequent text drawing functions like **WGText**, **WGTitleAxis**, **WGLabels**, etc., until they are set again. Font sizes are specified in points.

Special Features

Exchange of Images

Graphic images created with these tools can be exported to other applications via the Clipboard (functions **WGCOPYPage** and **WGCOPYGraph**) or Windows Metafiles (**WGSavePageMeta**). Images copied to the Clipboard can be pasted to documents in other applications. Images saved as metafiles can be imported into an application directly. Images can also be retrieved as Windows bitmaps (**WGGetGraphBitmapHandle**), or saved to disk as a Windows DIBs (device independent bitmaps) (**WGSavePageDIB**).

Printing Graphs

A page with all its graphs, or a single selected graph, can be printed on any printer or other hardcopy device supported by Windows. The driver for the printer must be installed and selected from the Windows Control Panel prior to an attempt to print. New features include sending the graph to any portion of the printed page and setting the printer orientation through code.

Mouse Interaction

If any mouse related event occurs while a graph window has the input focus, the control is transferred to the function **WGGraphMouseEvent**. In the C and Delphi versions of the software the **WGGraphMouseEvent** function is found in the file **HOOK.C** (or **HOOK.PAS**). In the Visual Basic version of the software this function is found in **VBHOOK.DLL**. The source code to **VBHOOK.DLL** is supplied in C and it can only be modified using the Visual C++ compiler. By default this function processes the left mouse button click and double click events. In the first case, if the mouse cursor is positioned over a graphical object at the time of the click, the object is visually highlighted. In case of the double click the **WGEDitObject** function is called. It calls the function **EditObject**, which also belongs to **HOOK.C** (**HOOK.PAS** for Delphi, **VBHOOK.DLL** for Visual Basic). This function causes the default dialog box for editing the object to appear.



Edit chart data using the Data Table window.

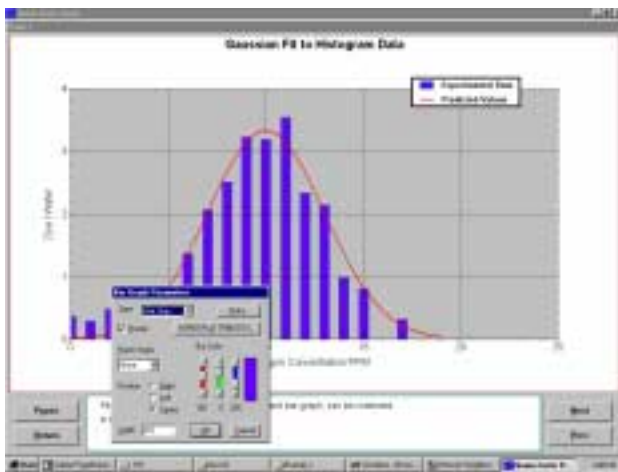
Mouse movement can be tracked with functions **WGGetMousePos** and **WGGetMousePosNorm**. They return current mouse cursor coordinates.

Drag Graph Objects with the Mouse - The mouse can be used to drag and move/resize axes, legends and text, along with other graphical objects such as arrows, rectangles and lines.

Drag Data Points with the Mouse - The mouse can be used to click and drag actual plot data values to new positions in the graph.

Super Zoom - The Super Zoom function will handle simultaneous zooming of up to ten x and y axes, each with unique scaling values. Rescaled axes will always display at the same normalized position in the graph. The initial pre-zoom state for up to 10 pairs of xy-axes in a single graph can be saved and subsequently restored at any time.

XOR Data Cursors - New data cursor routines XOR with the screen for flickerless updates. The new data cursor routines are used with the existing data cursor routines, which are actually data markers. Horizontal only, vertical only, horizontal and vertical, cross hairs and box data cursors are sup-



Built-in dialog boxes can be used to edit important chart characteristics such as colors, fill styles and borders.

ported. Data cursors can be clipped to the plotting area of the graph, or to the entire graph area.

Programmable Mouse Events - The mouse events for a graph window can be vectored into the main application program. The default handling of mouse events, such as a double click that brings up a dialog box for editing a graphical object, can be changed under program control.

Get Nearest Point Functions - A **WGGetNearestPointNormalized** function looks up the nearest point in a dataset using normalized rather than physical coordinates. This is useful if the x and y axes are not homogeneous, i.e. do not have the same scaling. Another function, **WGNearestPoint**, will only search for the nearest point within a specified subset of the plotting objects within the target graph.

Add New Objects to an Existing Graph - Graphical objects can now be added to an existing graph at any time. Add new plot objects, text annotations, arrows, etc. to a graph after it has already been created.

Auto Axes Routines - Every axis scaling and auto-axes routine ever needed by any of our customers has been added to the software to handle virtually every axis scaling situation. Included are auto-axes functions for multiple data sets in a single graph. auto-axis a single axis while specifying the exact range for the other axis, auto-axes stacked group plots, auto-axes contour data, auto-axes existing graphs.

Editing Graphs

The **Charting Tools for Windows** provide a number of dialog boxes for editing graphs. By default, double clicking the left mouse button, when the mouse cursor is over an object, will bring up an appropriate dialog box. Dialog boxes are available for editing graph parameters and most of the graphical objects.

Graph Windows - edit the size, position and background color.

Plotting Area - edit the color and position of the plotting area within a graph window.

Plot Attributes - edit the line, bar, pie wedge and symbol attributes for the basic data plotting objects.

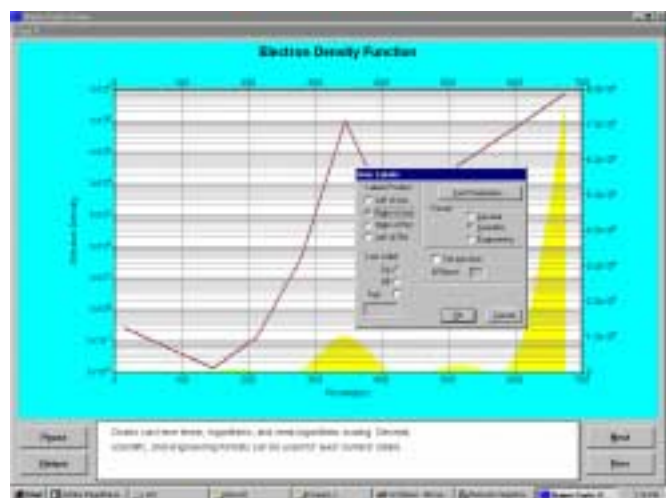
Data - edit the X- and Y-data which make up the plot portion of a graph. This is not really a Windows dialog box, rather a special editing window.

Axes - edit the physical coordinate scaling, X- and Y-intercepts, major and minor tick mark spacing, color, and thickness of the selected axis.

Axes Labels - edit the size, font, color, and format of numeric labels.

Titles - edit the text, size, font and color of graph and axes titles.

Legends - edit the color, position, size, font attributes and contents of a legend window.



Decimal, scientific and engineering formats are available for axes numeric labels.

Error Handling

Various types of errors can occur when *Charting Tools for Windows* functions are called. For example, one or more arguments passed to a function may have illegal values, or system resources may not be available. Most of the functions in this package give an indication if they were successful. An error code is assigned to every error. It can be obtained by calling function **WGGetLastError**.

Miscellaneous Compiler Support Features

Floating Point Number Data Types

Type *realtype* is used for all floating point numbers. By default, it is defined to be equivalent to *double* but can be changed to *float*. The Visual Basic version of the software only supports the 8-byte double format.

Memory Models

C++ Support

The *Charting Tools for Windows* have been written using standard ANSI C syntax, and are compatible with C and C++ programs. The Tools are also compatible with the Microsoft Foundation Class library. The *Graphics Class Libraries (GCL)* for MFC, available as an add-on product, integrate these Tools with the MFC Document/View architecture. See the *GCL* data sheets for detailed information.

Charting Tools for Windows Function Summary

This section lists all the functions in their respective functional categories and provides their brief descriptions.

Graph Management Functions

WGClosePage - Destroys the specified page and its window.

WGCreateGraph - Creates and initializes a new graph in the specified page, creates the graph window.

WGCreatePage - Creates and initializes a new page, creates



Floating-Bar Graphs are useful in Project and Time-Line charts.

the page window.

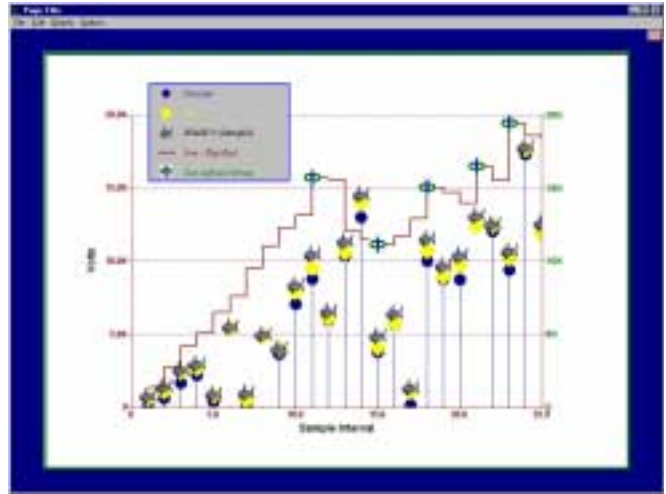
WGGetPageDesc - Returns the pointer to the page descriptor if the page window handle is known.

WGPostAddGraphObject - . The **WGPreAddGraphObject** and **WGPostAddGraphObject** function will enable you to add graphical objects to a graph at any time.

WGPreAddGraphObject - The **WGPreAddGraphObject** and **WGPostAddGraphObject** function will enable you to add graphical objects to a graph at any time.

WGScalePlotArea - Scales the plotting area of the graph window for a physical coordinate system.

WGSetPlotArea - Controls the relative placement of the graph plotting area within the graph window and sets the color of the plotting area.



Independent graphics objects, either bitmaps or metafiles, can be used in a chart.

Plotting Functions

Plotting functions can be seen as the graphical building blocks which are assembled on-screen to produce the complete graph.

To reduce the amount of work required to build a good looking graph, it is possible to automate some operations using **WGAutoAxes**. In this case the position and size of axes, labels and data are calculated internally.

The data for graphing can be read from an ASCII file, Windows clipboard, created by numerical calculations, or entered by the user.

Axes

WGAutoAxes - Analyses the data set and automatically scales the plotting area, selects intercepts, draws a pair of axes and labels them.

WGAutoAxesContourData - This function is an auto axes function for group data sets containing contour data.

WGAutoAxesGroupData - This function is an auto axes function for group data sets.

WGAutoAxesDataSets - This function is an auto axes function for multiple data sets.

WGAutoOneAxisDataSets - This function will carry out an auto axis in one dimension only.

WGAxes - This function takes the values *xMin*, *yMin*, *xMax* and *yMax* and scales the graph for these values. The graph can be scaled to rounded values, or to the exact values passed in .

WGDrawGrid - Draws grid lines.

WGDrawXAxis - Draws a horizontal axis with tick marks.

WGDrawYAxis - Draws a vertical axis with tick marks.

WGLabelAxisDate - This function writes a sequence of numeric labels at the tick marks of the specified axis.

WGReAutoAxes -This function is used outside of the graph building function to rescale a graph for new data.

WGReAutoAxesContourData - This function is used outside of the graph building function to rescale a graph for new data.

WGReAutoAxesDataSets - This function is used outside of the graph building function to rescale a graph for new data.

WGReAutoAxesGroupData - This function is used outside of the graph building function to rescale a graph for new group data.

WGReAutoOneAxisDataSets - This function is used outside of the graph building function to rescale a graph for new data.

WGReAutoAxes - This function is used outside of the graph building function to rescale a graph for new data.

WGRestoreAxesState - This function restores the state of the axes saved with the function **WGSaveAxesState**.

WGRoundAxis3 - This function adjusts given minimum and maximum values to round numbers.

WGSaveAxesState - This function saves the state of up to 10 pairs of x and y-axes. The axes can be restored by calling **WGResoreAxesState**.

WGSetAxesType - Controls the scaling (logarithmic or linear) used for x- and y-axes.

WGSetXYIntercepts - Sets the y-intercept for the x-axis and the x-intercept for the y-axis.

Plotting Data

WGBargraph - Draws a bar graph.

WGBoxWhisker - This function creates a “Box and Whisker” type plot for the specified group data set.

WGCandlestick - This function creates a “Candlestick” type plot for the specified group data set.

WGContourPlot - This function plots a *group* data set containing 3D data as a either a line or filled contour plot.

WGErrorBars - Draws an “Error bar” type plot.

WGFloatingBars - Draws floating bar graph.

WGGroupBargraph - Draws bar graphs in a group format.

WGGroupLines - This function draws the specified group data set as a non-cumulative group of stacked line plots.

WGHighLowClose - Draws a “High-Low-Close” type plot.

WGLineMarkerPlot - Draws a combination of line and scatter plots.

WGLinePlot - Draws a line plot.

WGLinePlotEx - This function draws a line plot for the given data set using the previously set line style, color and width.

WGOpenHighLowClose -This function creates a “Open-High-Low Close” type plot for the specified group data set.

WGPieChart - Plots a two- or three-dimensional pie chart.

WGScatterPlot - Draws a scatter plot.

WGScatterPlotEx - This function draws a scatter plot for the given data set. It varies from **WGScatterPlot** in that it can use a bitmap or a metafile as the marker shape.

WGStackedLines - Draws cumulative stacked line plot.

WGStockOpenHighLowClose - This function creates a “Open-High-Low-Close” type plot, in the style used in stock market report, for the specified group data set.

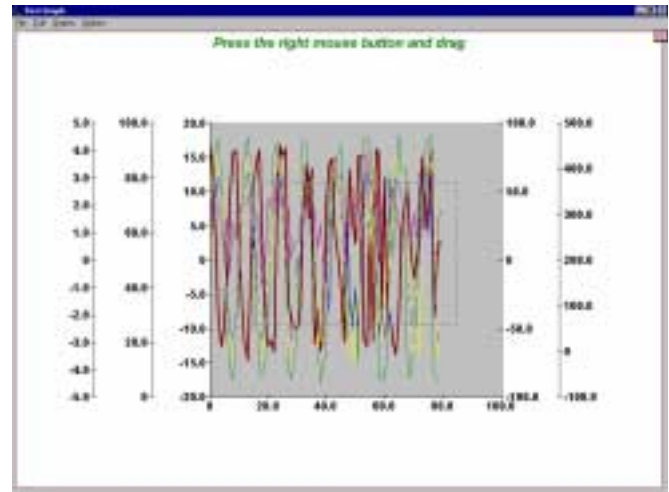
WGWaterfallPlot - This function draws the specified group data set as a waterfall plot using offsets for the x and y directions.

Text

WGAutoLegend - Draws string legends.

WGAutoLegendGroup - Draws string legends.

WGFlexLegend - This function draws a legend window with text strings and symbols that are user-defined and not related to any plotting object in a graph.



The Rev. 3.0 super zoom function will zoom up to ten sets of x and y-axes.

WGLabelAxis - Labels an axis with numbers.

WGLabelAxisDayWk - Labels an axis with names of days of the week.

WGLabelAxisMonth - Labels an axis with names of the months.

WGLabelAxisStrings - Labels an axis with text strings.

WGLabelPie - Adds text string labels to each slice of a pie chart.

WGLegend - Draws string legends.

WGLegendPtr - Same as **WGLegend**, but uses pointers instead of memory handles.

WGLegendGroup - Draws string legends.

WGMultiText - This function is identical to **WGText** but takes a delimited string as the *lpString* parameter.

WGMultiTextNorm -This function is identical to **WGTextNorm** but takes a delimited string as the *lpString* parameter.

WGGroupLegendPtr - Same as **WGGroupLegend**, but uses pointers instead of memory handles.

WGSetTextByName - Selects a specified font and sets text

parameters for all the subsequent text drawing.

WGSetTextParams - Sets text parameters for all the subsequent text drawing.

WGText - Writes text anywhere in the graph window, using physical coordinates for text position.

WGTextNorm - Writes text anywhere in the graph window, using normalized graph coordinates for text position.

WGTitleAxis - Writes an alphanumeric title for the specified axis.

WGTitleGraph - Writes an alphanumeric graph title horizontally at the top of the graph window.

Geometric Drawing Functions

Drawing functions draw simple objects like straight lines, arcs, rectangles, etc.

WGArc - Draws an arc in the specified graph window using physical coordinates.

WGArcNorm - Draws an arc in the specified graph window using graph normalized coordinates.

WGArrow - Draws a straight line with arrows attached to its one or both ends in the specified graph window using physical coordinates.

WGArrowNorm - Draws a straight line with arrows attached to its one or both ends in the specified graph window using graph normalized coordinates.

WGEllipse - Draws an ellipse in the specified graph window using physical coordinates.

WGEllipseNorm - Draws an ellipse in the specified graph window using graph normalized coordinates.

WGLine - Draws a straight line in the specified graph window using physical coordinates.

WGLineNorm - Draws a straight line in the specified graph window using graph normalized coordinates.

WGPie - Draws a pie-shaped wedge in the specified graph window using physical coordinates.

WGPieNorm - Draws a pie-shaped wedge in the specified graph window using graph normalized coordinates.

WGPolygon - Draws a polygon in the specified graph window using physical coordinates.

WGPolygonNorm - Draws a polygon in the specified graph window using graph normalized coordinates.

WGPolyline - This function draws a polyline in the specified graph window.

WGPolylineNorm - This function draws a polyline in the specified graph window.

WGRectangle - Draws a rectangle in the specified graph window using physical coordinates.

WGRectangleNorm - Draws a rectangle in the specified graph window using graph normalized coordinates.

WGSetDataCursor - Creates a data cursor.

Data Set Management Functions

WGCompressData - Creates a new, compressed data set based on the specified data set.

WGCreateEvenGridDataSetPtr - This function creates a group data set, to be used in a contour plot, from an array of z-values sampled on an evenly spaced grid of x and y-values.

WGDefineDataSet - Defines data set based on two data arrays X and Y.

WGDefineDataSetPtr - Same as **WGDefineDataSet**, but uses pointers instead of memory handles.

WGDefineEvenGridDataSetPtr - This function creates a group data set, to be used in a contour plot. The data must be evenly spaced on an x and y grid.

WGDefineGroupDataSet - Defines data set for GROUP type graph based on one-dimensional array X and two-dimensional array Y.

WGDefineGroupDataSetPtr - Same as **WGDefineGroupDataSet**, but uses pointers instead of memory handles.

WGFreeDataSet - Deletes data set, frees memory occupied with data arrays.

WGGetDataSet - Returns the handle of the data set used by the specified object.

WGGetDataSetArrays - Returns memory handles to independent and dependent variable arrays belonging to the specified data set

WGGetDataSetArraysPtr - Same as **WGGetDataSetArrays**, but uses pointers instead of memory handles.

WGGetMinMaxX - Determines the maximum and minimum values of an independent variable of the specified data set.

WGGetMinMaxY - Determines the maximal and minimal values of dependent variable of the specified data set.

WGGetNearestPoint - Searches a graph for the data point nearest to the point with the specified physical coordinates.

WGGetNearestPointNormalized - This function searches all the static plotting objects in the specified graph and finds the data point nearest to the point using normalized coordinates.

WGGetNearestPointPlotObjs - This function searches *the specified* static plotting objects in the specified graph and finds the data point nearest to the point using normalized or physical coordinates.

WGIIsDataPointGood - Checks if the specified point is marked as invalid.

WGLoadASCIIDataSet - Loads data arrays from an ASCII file and creates a data set based on these arrays.

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

WGPasteDataSet - Import an ASCII data table from the Windows clipboard.

WGReconnectDataSet - Disconnects the current data set from the specified graphical object and connects the new one to it.

WGSaveASCIIDataSet - Saves a data set to an ASCII file.

WGSetDataFormat - Sets format for displaying data in the data table window.

WGShowData - Displays a data table window that allows the user to modify data values belonging to the specified graphical object.

WGSortDataX - Sorts arrays belonging to the specified data set in ascending or descending order of X independent variable.

WGSortDataY - Sorts arrays belonging to the specified data set in ascending or descending order of dependent variable.

Image Exchange and Import Functions

WGBitmap - Loads a Windows bitmap and displays it in the specified graph window using physical coordinates.

WGBitmapNorm - Loads a Windows bitmap and displays it in the specified graph window using graph normalized coordinates.

WGConvertPageToMeta - Converts the contents of the specified page to a memory metafile.

WGCopyGraph - Copies the specified graph to the clipboard.

WGCopyPage - Copies the specified page with all its graphs to the clipboard.

WGDeleteMetafile - Deletes the specified metafile from memory.

WGFreeBitmap - Free a bitmap created by the **WGGetPageBitmapHandle** or **WGGetGraphBitmapHandle** functions.

WGFreeDIB - Free a DIB created by the **WGGetPageDIBHandle** or **WGGetGraphDIBHandle** functions.

WGGetMetaFile - Loads the specified metafile, creates its descriptor, and returns the pointer to the descriptor.

WGGetPageBitmapHandle - This function allocates a bitmap (device dependent bitmap), copies a page to the bitmap and returns the handle of the bitmap.

WGGetPageDIBHandle - This function allocates a DIB (device independent bitmap), copies a page to the DIB and returns the handle to the DIB.

WGMetafile - Loads a Windows metafile and displays it in the specified graph window using physical coordinates.

WGMetafileNorm - Loads a Windows metafile and displays it in the specified graph window using graph normalized coordinates.

WGPlayMetaFile - Plays the contents of the specified metafile in the specified window.

WGSaveGraphDIB - This function saves a graph window as a DIB (device independent bitmap) file.

WGSavePageDIB - This function saves a page as a DIB (device independent bitmap) file. **WGSavePageMeta** - Saves the specified page with all its graphs as a Windows metafile.

User Interface Support Functions

This group of functions allows users to change and get the status of many parameters of pages, graphs, and graphical objects. They can be used to modify graphs after they have been created, and for building menus and dialog boxes.

Data Cursors and Zooming

WGGetZoom - Returns physical coordinates of the zoomed area.

WGGetObjMove - Gets the bounding rectangle of the last object moved with the function **WGStartObjectMove**.

WGGetXCursor - Gets the ending coordinates of the last XOR cursor operation with the function **WGStartXCursor**.

WGMoveDataCursor - Changes position and state of a data cursor.

WGSetDataCursor - Creates a data cursor.

WGSetMouseEventAction - This function sets predefined and user defined actions in response to mouse events.

WGStartObjMove - This function attempts to grab and move the currently selected object.

WGStartSuperZoom - This function starts automatic zooming using the super zooming function. Unlike regular zooming, super zooming can handle multiple x and y axes.

WGStartZoom - Zooms in on the selected area of a graph.

WGStartXCursor - Start an XOR cursor.

WGSetXCursorParams - Get the line style and size of the XOR cursor.

Graph and Page

WGBlowupGraph - Blows up the graph so that it fills the whole page, or redraws it in its original size.

WGChangeGraphBackgnd - Changes the specified graph's window background color.

WGChangePlotBackgnd - Changes the specified graph's plotting area color.

WGChangePlotPos - Changes the size and position of the specified graph's plotting area.

WGDeleteGraph - Destroys the specified graph.

WGEditGraphDlg - Brings up a standard dialog box for editing the selected graph of the specified page.

WGGetGraphBackgnd - Gets the specified graph window background color.

WGGetGraphDesc - Returns the pointer to the graph descriptor if the graph window handle is known.

WGGetGraphWindow - Returns the specified graph's window handle.

WGGetPageDesc - Returns the pointer to the page descriptor if the page window handle is known.

WGGetPageWindow - Returns the window handle of the specified page.

WGGetPlotBackgnd - Gets the graph plotting area color.

WGGetPlotPos - Gets the specified graph's plotting area size and position.

WGGetSelectedGraph - Returns the pointer to the descriptor of the currently selected graph in the specified page.

WGRedrawGraph - Forces a graph window update.

WGSelectGraph - Marks the specified graph as selected.

WGSetGraphBorder - Draws or removes the graph border.

WGToggleHigh - Turns on or off selected graph highlighting for the specified page.

WGToggleSelectors - Displays or hides graph selector buttons.

Graphical Objects

WGChangeAreaColor - Changes the RGB color of the specified object's solid area.

WGChangeLineColor - Changes the color of the specified line based graphic object.

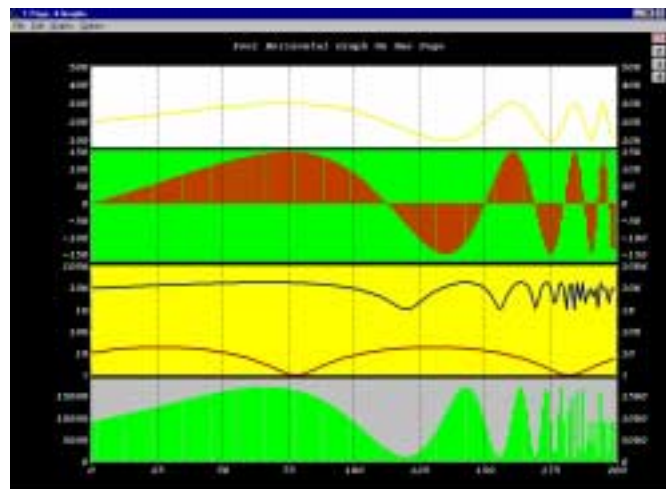
WGChangeObjectCoord - Changes physical coordinates of the specified object.

WGChangeObjectCoordNorm - Changes graph normalized coordinates of the specified object.

WGDeleteObject - Permanently deletes the specified object.

WGDuplicateObject - Creates a copy of the specified object and returns the handle for the new object.

WGGetAreaColor - Gets the RGB color of the specified object's solid area.



Axes with different scaling can be stacked on top of one another.

WGGetLineColor - Returns the color of the specified line based graph object.

WGGetLineStyle - Returns the line style of the specified object.

WGGetObjectCoord - Gets physical coordinates of the specified object's bounding rectangle.

WGGetObjectCoordNorm - Gets graph normalized coordinates of the specified object's bounding rectangle.

WGGetObjectType - Returns the type code of the specified object.

WGGetObjSize - Returns the size of the specified object's descriptor.

WGGetObjTypeSize - Retrieves the descriptor size of the graphic objects of the specified type.

WGGetSelObject - Returns the handle of the currently selected object.

WGObjUpdate - Copies the contents of the duplicate object previously created by **WGDuplicateObject** to the original object. The duplicate object is destroyed.

WGRedrawObject - Immediately redraws the specified object.

WGSelectObject - Marks the specified graphic object as selected.

WGShowObject - Hides or shows the graphical object.

Axes

WGChangeAxisIntercept - Changes the intercept value for the specified axis in an existing graph.

WGChangeAxisRange - Changes the upper and lower limits for the specified axis in an existing graph.

WGChangeAxisScale - Changes the specified axis scaling from linear to logarithmic or vice versa.

WGChangeAxisTicks - Changes the number and position of the tick marks of the specified axis.

WGChangeGrid - Modifies the style, color and width of the grid lines.

WGGetAxisDir - Determines if the specified axis is vertical or horizontal.

WGGetAxisRanges - Returns the maximal, minimal, and intercept values for the specified axis.

WGGetAxisScale - Determines if the specified axis scaling is linear or logarithmic.

WGGetAxisTicks - Returns axis tick mark parameters.

WGGetGrids - Returns parameters describing grid lines based on the specified axis.

WGRoundAxis - Calculates rounded minimum and maximum values and the distance between adjacent major tick marks. Optionally, updates the parameters of the specified axis and rescales the graph.

WGSetAxisIntercepts - This function sets the intercept of an axis with respect to the axis perpendicular to it. It also sets the starting point, or origin of the axis labels.

WGSetTickSize - Sets major and minor tick mark sizes, in screen pixels, for all the axes belonging to graphs of the specified page.

WGToggleGrid - Turns the grid lines based on the specified axis on or off.

WGToggleGridsOrder - Changes the order of drawing axes and grids.

Bar Graphs

WGChangeBarJust - Changes bar or bar group position relative to the coordinate of the independent variable.

WGChangeBarWidth - Changes the bar width expressed in

physical units of independent variable for the specified bar graph.

WGChangeHatchStyle - Changes the hatch style for the specified bar graph, member of a group bar graph, or sector of a pie chart.

WGGetBarJust - Returns the bar justification code describing the bar or bar group position relative to the coordinate of the independent variable.

WGGetBarWidth - Returns the bar width for the specified bar graph expressed in physical units of independent variable.

WGGetHatchStyle - Returns the hatch style of the specified bar graph, member of a group bar graph, or sector of a pie chart.

Line, Scatter, and Other Plots

WGChangeMarkerColor - Changes the color of markers used in scattered plots, error bars, High-low-close plots.

WGChangeMarkerType - Changes the shape, style and size of the markers used in the specified scatter plot, error bars, or high-low-close plot.

WGChangeObjFill - Turns on or off the flag that determines if the area under the curve (for line plots) or between curves (stacked line plot) is filled.

WGChangePlotType - Changes the plot type.

WGGetFillFlag - Returns the status of the flag that determines if the area under the curve (for line plots) or between curves (stacked line plot) is filled.

WGGetMarkerColor - Gets the color of markers used in the specified scattered plot, error bars, or high-low-close plot.

WGGetMarkerType - Returns the shape, style and size of the markers used in the specified scattered plot, error bars, or high-low-close plot.

WGGetPlotType - This function returns the plot type of the specified graphical object.

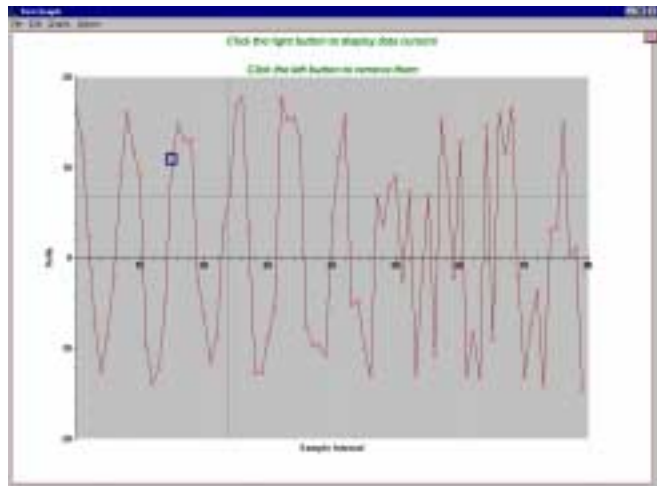
WGGetSplineFlag - Returns the status of the spline interpolation flag for the specified line plot.

WGToggleSpline - Turns on or off spline interpolation for the specified line plot.

Labels

WGChangeLabelsFormat - Changes the format in which the specified labels are displayed.

WGChangeLabelsPos - Changes the specified axis labels



XOR data cursors and data markers can be used to select and mark data points.

position relative to the axis and plotting area.

WGChangeLastLabel - Changes the way the last axis label is drawn.

WGGetLabelsFormat - Returns the number indicating in what format the specified labels are displayed.

WGGetLabelsPos - Returns the number indicating how the specified axis labels are positioned relative to the axis and plotting area.

WGGetLabelsPrec - Returns the number of digits after the decimal point in the specified numerical axis labels.

WGGetLastLabelFlag - Returns the flag indicating how the last label is displayed.

WGGetLastLabelText - Returns the pointer to the text string used as the last label.

Pie Charts

WGChangePieExpl - Changes the relative explosion coefficient for the specified pie chart slice.

WGChangePieTextPos - Changes the specified pie chart labels placement.

WGChangePieValue - Changes the input value corresponding to the specified slice of a pie chart.

WGGetPieExpl - Returns the relative explosion coefficient for the specified pie chart slice.

WGGetPieTextPos - Returns the indicator of the specified pie chart labels placement.

WGGetPieValue - Returns the value corresponding to the specified pie chart slice.

Arrows

WGChangeArrow - This function changes length and width of the specified arrow.

WGChangeArrowFill - Changes the way the specified arrow is drawn from filled to empty or vice versa.

WGGetArrowFill - Returns the flag indicating if the specified arrow is drawn as filled or empty.

WGGetArrowLength - Returns the length of the specified arrow, in points.

WGGetArrowPos - Returns the number indicating if the specified arrow graphic object has an arrow attached to its beginning, end, or both.

WGGetArrowWidth - Returns the width of the specified arrow, in points.

Text

WGChangeFont - Modifies the text parameters of the specified graphical object.

WGChangeString - Changes the contents of the text string associated with the specified object.

WGChangeTextColor - Changes the text color of the specified text, labels, legends, or pie chart object.

WGGetFontItal - Returns the flag indicating if the font used by the specified object is italic.

WGGetFontName - Returns the typeface name of the font used by the specified object.

WGGetFontOrient - Returns the orientation of the font used by the specified object.

WGGetFontSize - Returns the size of the font used by the specified object, in points.

WGGetFontUnder - Returns the flag indicating if the font used by the specified object is underlined.

WGGetFontWeight - Returns the weight of the font used by the specified object.

WGGetString - Returns the pointer to the text string associated with the specified graphic object.

WGGetTextColor - Returns the text color code for the specified object.

WGGetTextSizeNorm - Returns relative width and height of a line of text in normalized graph coordinates.

Printing Functions

WGPrinterSetup - Brings up the dialog box that allows the user to change any configuration settings for the default printer.

WGPrintGraph - Prints the specified graph.

WGPrintOptionsDlg - Brings up a dialog box which allows to change print options interactively.

WGPrintPage - Prints the specified page with all its graphs.

WGPrintPageEx - Prints the specified page to the specified printer rectangle.

SetPrintOptions - Sets printing options for functions

WGPrintGraph and **WGPrintPage**.

WGSetPrintOrient - Set the printer orientation programmatically to output a printed page or graph in either portrait or landscape mode.

Coordinate Conversion Functions

WGDevicePixelsToLogicalUnits

WGDevicePixelsToNormalizedUnits

WGDevicePixelsToPhysicalUnits - These three functions convert Device (or pixel) coordinates to Logical, Normalized and Physical coordinates.

WGLogicalUnitsToDevicePixels

WGLogicalUnitsToPhysicalUnits

WGLogicalUnitsToNormalizedUnits - These three functions convert Logical coordinates to Device, Normalized and Physical coordinates.

WGNormalizedUnitsToPhysicalUnits

WGNormalizedUnitsToLogicalUnits

WGNormalizedUnitsToDevicePixels - These three functions convert Normalized coordinates to Device, Logical and Physical coordinates.

WGPhysicalUnitsToDevicePixels

WGPhysicalUnitsToLogicalUnits

WGPhysicalUnitsToNormalizedUnits - These three functions convert Physical coordinates to Device, Logical and Normalized coordinates.

Miscellaneous Functions

WGCopyMem - Makes a copy of the specified global memory block.

WGCreateShadeColor - Returns the RGB color which is darker than a specified color and is appropriate for shading.

WGENableErrorMessage - Enables or disables reporting lower priority errors in a message box.

WGGetLastError - Returns the last error code.

WGGetMousePos - Determines the mouse cursor physical coordinates.

WGGetMousePosNorm - Determines the mouse cursor graph normalized coordinates.

WGGetPureColor - Returns the integer color code which best matches a specified RGB color.

WGGetRGBColor - Returns the RGB color value for the

given integer color code.

WGIInsideRect - Determines if the specified point lies within a given rectangle.

WGMenuCheck - Changes the state of a specified pop-up menu item in the menu belonging to the specified page.

WGSetLineStyle - Sets pen attributes for subsequent line drawing.

WGSetGraphBitBltMode - Turns on/off the bit block transfer mode for graphs.

WGOKMsgBox - Formats and displays a series of characters and values in a message box.

WGSetOKCursor - Sets the cursor and focus to the OK button in a dialog box.

WGGetObjUserID - This function reads the user ID stored in the structure of all graphical objects.

WGGetUserIDObj - This function returns a handle to the graphical object associated with the specified user ID.

FFT Functions

WGComplexFFT - Computes the direct and inverse fast Fourier transform of a set of complex values using a modified Cooley-Tukey algorithm.

WGFFTFrequency - Calculates the frequency associated with a given harmonic index and sampling frequency.

WGFFTMagnitude - Calculates the FFT magnitude associated with a given harmonic index.

WGFFTPhase - Calculates the FFT phase associated with a given harmonic index.

WGPowerSpectrum - Calculates the power spectrum periodogram of a sampled data set.

WGRealFFT - Computes the direct and inverse fast Fourier transforms of an array of real data values.

WGDSPWindow - Applies a specified windowing function to a data array.

Prerequisites

This product makes it much easier for you to write Windows charting applications. Nevertheless, it is not designed for programming novices. Quinn-Curtis cannot teach you how to program using the Microsoft Windows environment. So, we expect successful users of this product should have as a minimum:

- An excellent background in the programming language you have purchased the product for: C, Delphi or Visual Basic.
- At least 3 months experience with the compiler you plan to use (Visual C++, Borland Delphi or Visual Basic). C users must be able to create projects or .MAK files for building programs involving many different source, object, resource and library files. Visual Basic users must be able to create Visual Basic projects.
- A good background in programming simple Windows applications (we believe that reading Charles Petzold's book "Programming Windows" is a must). You must be able to write, compile, link and run simple Windows programs of your own creation before you can use our Windows programming tools.

Our technical support personnel will expect this level of programming experience should technical support for this product be required.

Example Programs

The following example programs are supplied on disk.

	Demo Name	Description
1.	EASYGRAF	Simplest demo program
2.	DEMO1P3G	1 Page Window, 3 Graphs
3.	DEMO3P1G	3 Page Windows, 1 Graph/Window
4.	METADEMO	Using metafile functions
5.	MDIDEMO	Multiple Document Interface example
6.	LINEMARK	Line plot with markers
7.	HLCDEMO	High-Low-Close example
8.	ASCDEMO	Plot 3-D bars from an ASCII file
9.	ERRBARD	Error bar plots
10.	FFTDEMO	FFT example with graphs.
11.	BARDEMO	Bar graph example.
12.	PIECHART	Pie chart example.
13.	GRPBARD	Group bar graph example.
14.	AXSCALE	Axes scaling example.
15.	BIGDEMO	Large demo combining many chart types.
16.	FLODEMO	Floating bar graph example.
17.	MONKYDEM	Bitmaps used in chart example.
18.	DYNDEMO	Dynamic object update example.
19.	CURSDM	Data cursor example.
20.	AUTOAXES	Auto axes using multiple datasets.
21.	COMPDATA	Compress a large data set.
22.	CONTOURD	Contour plot example.
23.	D1P4GAX	Four horizontal graphs in a page.
24.	GROUPLN	Group line example.
25.	MOVEDATA	Moving data points with the mouse.
26.	MULTAXES	Four y-axes in a single graph.
27.	MULTIPLT	Multi-line text and multiple colors for plot objects.
28.	NEWDATA	Replace old data with new data.
29.	PANGRAPH	Pan a graph using a scrollbar.
30.	SCATSTEP	Scatter plots with bitmaps and metafiles.
31.	STOCKHLC	Stock market Open-High-Low-Close.
32.	SUPRZOOM	Super zoom example with 5 y-axes.
33.	UIOBJ	Move data, axes, strings, legends and bitmaps with the mouse.
34.	WATERFAL	Example of a waterfall plot.
35.	WCTDEMO3	The Charting Tools demo program from our www site.
36.	VCPPDEMO.CPP	MS C Foundation Classes example (C only).
37.	OWL2DEMO.CPP	Borland Object Windows example (C only).

C Programming Example

This program demonstrates how easy it is to add charting capability to a Windows program. Most of the work is done by the DrawP1G1 function. A single, simple graph is created in a parent page window. Once it has been created, it can be moved and resized in typical Windows fashion.

A number of dialog boxes are automatically made available for editing graph objects such as:

- line plot characteristics
- X-axis characteristics

Y-axis characteristics
 X-axis label characteristics
 Y-axis label characteristics
 plotting area positioning and background colors
 titles (X-axis, Y-axis and window)

A dialog box is selected by double clicking on the object you wish to edit.

A menu also appears at the top of the graph ("PageMenu" in the resource file) which can save the graph as a Windows Metafile (suitable for importing into spreadsheets and word processors), set up a printer, then send the graph to a printer.

Function calls in bold are calls to *Charting Tools for Windows* routines.

```
// *****
//
// Description: C example building one graph with
//             Charting Tools for Windows
//
// *****

#include <windows.h>
#include <stdlib.h>
#include <math.h>
#include "qcwin.h"

#define NUMP1 100

HINSTANCE hInst; // global instance handle
GLOBALHANDLE hX1,hY1; // global handles to data

char szAppName[] = "Demo"; // Name passed to CreateWindow.

// Prototypes of forward referenced functions
LRESULT CALLBACK _export MainWndProc (HWND hwnd, UINT message,
                                     WPARAM wParam, LPARAM lParam);

BOOL InitApplication(HINSTANCE);
BOOL InitInstance(HINSTANCE, int);
void FAR StartGraphs1 (PPAGE_DEF pPageDesc);
void FAR DrawP1G1 (PGRAPH_DEF pGrDesc, HDC hdc);
realtyp randreal(void);

/*****
FUNCTION: WinMain (HINSTANCE, HINSTANCE, LPSTR, int)

PURPOSE: calls initialization function,
         processes message loop
*****/
int PASCAL WinMain (HINSTANCE hInstance,
                  HINSTANCE hPrevInstance,
                  LPSTR lpCmdLine, int nCmdShow)
{
    MSG msg;

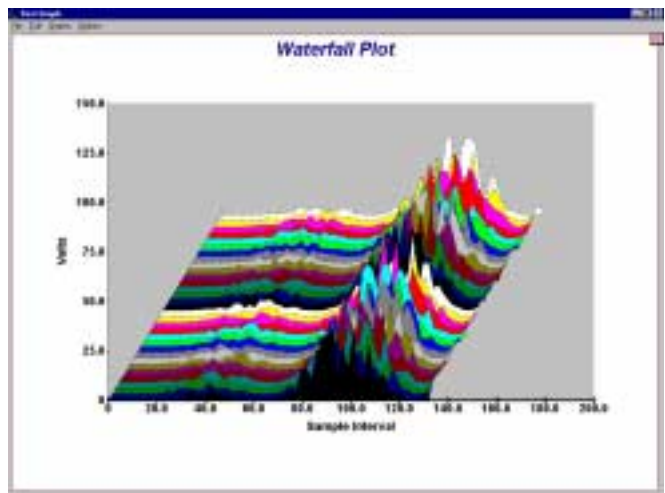
    if (!hPrevInstance)
        if (!InitApplication(hInstance))
            return (FALSE);

    if (!InitInstance(hInstance, nCmdShow))
        return (FALSE);

    // Message loop
    while (GetMessage(&msg, NULL, NULL, NULL))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    };
    return (msg.wParam);
}

/*****
FUNCTION: InitApplication (HANDLE)
PURPOSE: Initializes window data and
         registers window classes
*****/
BOOL InitApplication (HINSTANCE hInstance)
{
    WNDCLASS wc;

    // main window class
    wc.style = CS_HREDRAW | CS_VREDRAW;
    wc.lpfWndProc = MainWndProc;
    wc.cbClsExtra = 0;
```



The new Rev. 3.0 waterfall plot type is used to plot group data where there are multiple y-values for every x-value.

```
    wc.cbWndExtra = 0;
    wc.hInstance = hInstance;
    wc.hIcon = LoadIcon(hInstance, "LOGO");
    wc.hCursor = LoadCursor(NULL, IDC_ARROW);
    wc.hbrBackground =
        (HBRUSH) GetStockObject( LTGRAY_BRUSH);
    wc.lpszMenuName = NULL;
    wc.lpszClassName = szAppName;
    return (RegisterClass(&wc));
}

/*****
FUNCTION: InitInstance(HANDLE, int)
PURPOSE: Saves instance handle and creates main window
*****/
BOOL InitInstance (HINSTANCE hInstance, int nCmdShow)
{
    HWND hwnd;

    // save current instance MUST BE HERE
    hInst = hInstance;

    hwnd = CreateWindow
    (
        szAppName, // Create main window
        "Quinn-Curtis Charting Tools for Windows", // Window title
        WS_OVERLAPPEDWINDOW | WS_CLIPCHILDREN,
        CW_USEDEFAULT, CW_USEDEFAULT,
        CW_USEDEFAULT, CW_USEDEFAULT,
        NULL, NULL, hInstance, NULL
    );

    if (!hwnd) return (FALSE);
    ShowWindow(hwnd, nCmdShow);
    UpdateWindow(hwnd);
    return (TRUE);
}

/*****
Main window procedure
*****/
LRESULT CALLBACK _export MainWndProc (HWND hwnd, UINT message,
                                     WPARAM wParam, LPARAM lParam)
{
    switch (message)
    {
        case WM_RBUTTONDOWN:
            // When the mouse right button is pressed the
            // page is created in the current window
            WGCreatePage("PAGE1", // page ID string
                        hwnd, // handle to the parent window
                        hInst, // application instance handle
                        "First Graph", // Window title string
                        StartGraphs1, // pointer to graph creation function
                        "PageMenu", // Name of page window menu in resource file
                        C_LIGHTGRAY, // window background color
                        MM_PROPORT, // window sizing mode
                        0L, // window style - default
                        PAGE_CLIENT, // window initial size and position option
                        0, 0, 0, 0); // initial window size and position
            // if used (not used here)
        case WM_PAINT: // paint main window
            {
                PAINTSTRUCT ps;
```

```

RECT rect;
HDC hdc = BeginPaint(hwnd, &ps);
// function calls to repaint main window go here
GetClientRect(hwnd, &rect);
SetTextAlign(hdc, TA_BASELINE | TA_CENTER);
TextOut(hdc, rect.right / 2, rect.bottom / 2,
"Press Right Mouse Button!", 26);
EndPaint(hwnd, &ps);
}

return 0;

case WM_DESTROY:
GlobalFree(hX1); // free global data handles
GlobalFree(hY1);
WGCleanup(FALSE); // clean up charting tools memory
PostQuitMessage(0);
return 0;
}

return (DefWindowProc(hwnd, message, wParam, lParam));
}

/*****
Routine StartGraphs1 is called by the Quinn-Curtis
Charting Tools for Windows when a page is created.
It must be filled by the user, normally with
functions WGCleanup that initialize individual graphs.
*****/
void FAR StartGraphs1(PPAGE_DEF pPageDesc)
{
int i;
realtype z;
static BOOL fInit = TRUE;
LPREAL lpX1, lpY1;

// create simulation data for plot
/*-----*/
if (fInit) // do not initialize data twice
{
// get handles for global data
hX1 = GlobalAlloc(GHND, sizeof(realtype) * NUMP1);
hY1 = GlobalAlloc(GHND, sizeof(realtype) * NUMP1);
// get pointers to data
lpX1 = (LPREAL) GlobalLock(hX1);
lpY1 = (LPREAL) GlobalLock(hY1);
// create x and y data to be plotted
for (i = 0; i < NUMP1; i++)
{
z = (realtype) i;
lpX1[i] = z;
lpY1[i] = 15.0 *
cos(M_PI * z / (4.0 + 0.3 * randreal())) +
3.0 * randreal();
}
fInit = FALSE;
}
}
/*-----*/

// Initialize graph
WGCleanup(pPageDesc,
DrawP1G1, // points to function which builds graph
0.002, 0.002, // window relative position
0.998, 0.998, // inside parent page window
C_WHITE, // white background
C_BLACK, // black border
1); // border width in pixels
}

/*****
Builds the graph using Q-C Charting Tools Calls
*****/
void FAR DrawP1G1 (PGRAPH_DEF pGrDesc, HDC hdc)
{
HANDLE hAxisX, hAxisY; // axes handles
HANDLE hDataSet; // data set handle

// define a dataset
hDataSet = WGDefineDataSet ("60 Cycle Noise",
hX1, hY1, NUMP1);

// define the plotting area of the graph
WGSetPlotArea(pGrDesc, hdc,
0.15, 0.15, 0.9, 0.80,
C_LIGHTGRAY);
// axes to be drawn in solid, black, 1 pixels thick
WGSetLineStyle(pGrDesc, hdc, PS_SOLID, 1, C_BLACK);

// set current font to Arial, 12 points, bold
WGSetTextByName (C_BLACK, "Arial", 12, TEXT_BOLD);

// analyze the data set and automatically scale the
// plot area, draw and label the axes
WGAutoAxes(pGrDesc, hdc, hDataSet, AS_ROUNDCLOSE,
INTF_ZERO, &hAxisX, &hAxisY, NULL, NULL);

// set line style of actual plot to RED, dotted
WGSetLineStyle(pGrDesc, hdc, PS_DOT, 0, C_RED);

```

```

// plot the data with spline interpolation on
WGLinePlot (pGrDesc, hdc, hDataSet, FALSE, TRUE);

// Write axes titles
WGTitleAxis (pGrDesc, hdc, hAxisX, POS_BELOW,
"Sample Interval");
WGTitleAxis (pGrDesc, hdc, hAxisY, POS_LEFT, "Volts");

// set current font to Arial, 16 points, bold, italic
WGSetTextByName (C_GREEN, "Arial", 16,
TEXT_BOLD | TEXT_ITAL);
// Write graph title
WGTitleGraph (pGrDesc, hdc, "Input Waveform");
}

/*****
random real number generator in the range of 0.0 to 1.0
*****/
realtype randreal(void)
{
return (realtype)rand()/(realtype)RAND_MAX;
}

```

Visual Basic Programming Example

```

' *****
'
' Description: Visual Basic 16-bit example building one*
' graph with the Charting Tools for Windows *
' *****

Option Explicit
Const NUMP1 = 80

Dim fInit As Integer
Dim X1() As Double
Dim Y1() As Double

Sub DrawP1G1 (pGrDesc As Long, hdc As Integer)
Dim hObj As Integer, success As Integer
Dim hAxisX As Integer, hAxisY As Integer
Dim hDataSet As Integer

' define a dataset
hDataSet = WGDefineDataSetVB("60 Cycle Noise",
X1(0), Y1(0), NUMP1)

' define the plotting area of the graph
Call WGSetPlotArea(pGrDesc, hdc, .15, .15, .9, .8, C_LIGHTGRAY)
' axes to be drawn in solid, black, 1 pixel thick
Call WGSetLineStyle(pGrDesc, hdc, PS_SOLID, 1, C_BLACK)
' set current font to Arial, 12 points, bold
Call WGSetTextByName(C_BLACK, "Arial", 10, TEXT_BOLD)
' analyze the data set and automatically scale the
' plot area, draw and label the axes
success = WGAutoAxes(pGrDesc, hdc, hDataSet, AS_ROUNDCLOSE,
INTF_ZERO, hAxisX, hAxisY, False, False)

' set line style of actual plot to RED
Call WGSetLineStyle(pGrDesc, hdc, PS_SOLID, 0, C_RED)
' plot the data with spline interpolation on
hObj = WGLinePlot(pGrDesc, hdc, hDataSet, False, True)
' write axes titles
hObj = WGTitleAxis(pGrDesc, hdc, hAxisX, POS_BELOW,
"Sample Interval")
hObj = WGTitleAxis(pGrDesc, hdc, hAxisY, POS_LEFT, "Volts")
' set current font to Arial, 16 points, bold, italic
Call WGSetTextByName(C_GREEN, "Arial", 16,
TEXT_BOLD + TEXT_ITAL)

' write graph title
hObj = WGTitleGraph(pGrDesc, hdc, "Input Waveform")
End Sub

Sub Form_Load ()
Dim i As Integer
Dim pPageDesc As Long

For i = 0 To 4150
TopDesc(i) = 0
Next
pPageDesc = WGCleanup("Page1", EasyGrafForm.hWnd,
"Easy Graph", "PageMenu", C_LIGHTGRAY,
MM_PROPORT, 0, PAGE_CLIENT,
0, 0, 0, TopDesc(0))

Call StartGraphs1(pPageDesc)
Call WGUUpdatePage(pPageDesc)
End Sub

Sub Form_Unload()
Call WGCleanup(TopDesc(0), True)
End Sub

Sub StartGraphs1 (pPageDesc As Long)
ReDim X1(NUMP1 - 1) As Double

```

```

ReDim Y1(NUMP1 - 1) As Double

Dim hdc As Integer, As Integer
Dim pGrDesc As Long

If fInit = True Then
    Randomize
    fInit = False
    For i = 0 To NUMP1 - 1
        X1(i) = i
        Y1(i) = 15 * Cos(M_PI * i / (4 + .3 * Rnd) + 3 * Rnd)
    Next
End If
pGrDesc = WGCreateGraph(pPageDesc, .005, .005, .99, .99,
    C_WHITE, C_RED, 1, TopDesc(0), hdc)
Call DrawP1G1(pGrDesc, hdc)
End Sub

```

System Requirements

The ***Charting Tools for Windows*** run on the IBM AT family of computers. Compiling a program using these tools requires:

- A computer configured for one of the following operating systems: Windows NT, Windows 95, or Windows 98.
- The developer must have a complete installation of either the Visual C++, Visual Basic, or Borland Delphi compiler.
- 32-bit application programs will run on systems capable of running Windows NT, Windows 95 and Window 98.