Olectra Chart™

2D DLL
Programmer's Guide & Reference Manual

Version 1.1

The Leader in GUI Components

260 King Street East
Toronto, Ontario, Canada M5A 1K3
(416) 594-1026
www.klg.com

June 1996

RefNo: 2DLLPR-CH/11-06/96

Printed in Canada on recycled paper. 

Table of Contents

Preface

Introduction

Olectra Chart Control

Windows contains many different pre-defined controls such as buttons, scrollbars, and list boxes. Conceptually, Olectra Chart adds a new control class to Windows. The chart control displays data graphically in a window and can interact with a user.

The chart control has properties that determine how the chart will look and behave. Writing programs using Olectra Chart is similar to writing any other kind of Windows program; you now have one more control to work with.

Olectra Chart has properties which allow control of:

- Chart type (plot, area, bar, stacking-bar, pie, or combination).
- Header and footer positioning, border style, text, font, and color.
- Data styles: line colors and patterns, fill colors and patterns, line thickness, point style, size, and colors.
- Legend positioning, orientation, border style, anchor, font, and color.
- Chart positioning, border style, color, width, height, and 3D effect.
- Axis labelling using Point-labels, Set-labels, Value-labels, or Time-labels.
- Axis and data minimums and maximums, axis numbering methods, numbering and ticking increments, grid increments, font, origins, axis directions and precisions.
- Data transposition and axis inversion.
- Placement of axes, annotation, and origins.
- Markers and Text Areas.

Olectra Chart also provides several procedures and methods which:

- Allocate and load data structures containing the numbers to be displayed.

- Display new data very quickly in certain circumstances.
- Assist the developer in dealing with user-events.
- Assist the developer with setting and getting indexed properties.

Assumptions

This manual assumes that the reader is proficient with the C language and the Windows API. C concepts such as "an array of **char ***" and "pointer to a structure" must be understood before beginning to program Windows and Olectra Chart applications. An understanding of basic Windows programming concepts such as event-driven programming and programming Windows controls is required before continuing with this manual. See page 3 for information on Windows programming references.

Typographical Conventions Used in This Manual

Bold	■ Chart property and method names. ■ Language-specific keywords, constants, variables, and function names. ■ Commands that you enter at a command prompt.
Italic Text	■ Parameter names and information you specify. ■ New terms as they are introduced, and to emphasize important words. ■ Figure and table titles. ■ The names of other documents referenced in this manual, such as the *Getting Started with Olectra Chart* booklet.
UPPERCASE	■ File and directory names, key names, and key sequences.
`Monospace`	■ Code examples, variables in body text, and error text.
...[XYY2]..	Many Olectra Chart properties used to specify axis information are similar for the X-, Y-, and Y2-axis. The syntax **[XYY2]** used in an property name refers to one of all of the X, Y, or Y2 versions of this property. For example, "**XRT_XNUM**, **XRT_YNUM**, and **XRT_Y2NUM**" is shortened to "**XRT_[XYY2]NUM**".

Overview of Manual

Part I describes how to use Olectra Chart.

Chapter 1, “Getting Started: Developing a Simple Olectra Chart Program”, should be read by all programmers learning Olectra Chart. It provides a quick introduction to Olectra Chart programming.

Chapter 2, “Olectra Chart Basics”, provides basic information that you need to know before starting to develop applications with Olectra Chart.
It gives basic terminology and some programming approaches which apply to many properties.

Chapter 3, “Programming Olectra Chart”, provides programming information for most Olectra Chart properties.

Chapter 4, “Olectra Chart Data”, provides details on how to get data into charts.

Chapter 5, “Programming User Interaction”, provides programming information for handling user interaction.

Chapter 6, “Advanced Programming Topics”, discusses more advanced and less commonly-used aspects of Olectra Chart.

Part II contains detailed technical reference information in a number of appendices.

Appendix A , “Property Reference”, provides a concise reference of all Olectra Chart properties in alphabetical order.

Appendix B , “Procedures and Methods Reference”, lists all Olectra Chart procedures, methods, and convenience routines in alphabetical order.

Appendix C , “Macros”, lists the convenience macros.

Appendix D , “Message Reference”, provides a reference of Olectra Chart’s notification messages.

Appendix E , “Data Types”, lists the Olectra Chart data structures.

Appendix F , “Sample Code”, provides complete listings of one example program discussed in this manual, and briefly describes all of the sample programs included with Olectra Chart.

Related Documents

The following documents are useful references for Windows application development:

- *Programming Windows 3.1* by Charles Petzold, Microsoft Press.
- *Windows 3.1 SDK* and *Windows 3.1 API* online documentation, Microsoft Corporation.
- *Advanced Windows Programming* by Jeffrey Richter, Microsoft Press.
- *Win32 SDK* and *Win32 API* online documentation, Microsoft Corporation.

Part I

Using the Chart

1

Getting Started: Developing a Simple Olectra Chart Program

Introduction ■ *A Basic Plot*
Loading Data From a File ■ *Changing Chart Type*
Adding Header, Footer and Labels ■ *Inverting and Transposing the Chart*
Putting it all Together

1.1 Introduction

This chapter allows you to immediately try out Olectra Chart by compiling and running an example application. This application graphs the 1963 Quarterly Expenses and Revenues for "Michelle's Microchips", a small company a little ahead of its time.

The data to be displayed is shown in the following table:

	Q1	Q2	Q3	Q4
Expenses	150.0	175.0	160.0	170.0
Revenue	125.0	100.0	225.0	300.0

1.2 A Basic Plot

Figure 2 lists our starting point, PLOT1.C. This minimal Windows program creates a dialog window, loads the data from a file, and then sets it to the chart control that plots the data. This program is located in Olectra Chart's \CHART\2D\DEMOS\DLL\SDK\PLOT1 directory.

When PLOT1.C is compiled and run, the window shown in Figure 1 displays.

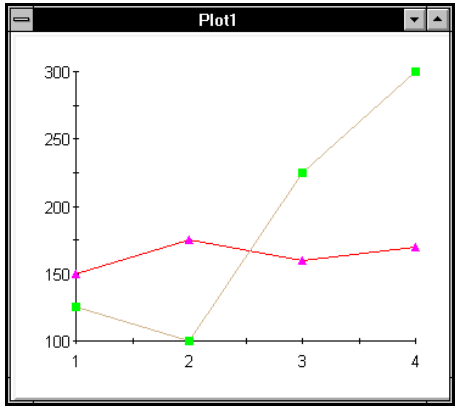


Figure 1 The Plot1 window

Most of the code in *PLOT1.C* should be familiar to Windows programmers. For example, lines 57–70 set the window class information, and lines 72–73 create the main dialog window.

The statements on lines 15 to 17 get the window handle and chart handle. Line 20 uses the chart handle to allocate space for the **XrtData** structure (pointed to by **my_data**) and loads that structure with the data from the *MM63.DAT* file. Line 21 sets the chart's **XRT_DATA** property to **my_data** by calling the **XrtSetValues**() method.

Olectra Chart provides two methods for setting any chart property: **XrtSetValues**() and **XrtSetPropString**(). Correspondingly, two methods retrieve the current value of chart properties: **XrtGetValues**() and **XrtGetPropString**(). See the next chapter for more details on property setting/retrieving.

Olectra Chart supports five different *types* of charts: plots, area charts, bar charts, stacking bar charts, and pie charts. The type is specified with the **XRT_TYPE** property which must be **XRT_TYPE_AREA**, **XRT_TYPE_PLOT**, **XRT_TYPE_BAR**, **XRT_TYPE_STACKING_BAR** or **XRT_TYPE_PIE**. Because **XRT_TYPE_PLOT** is the default, we do not have to specify the **XRT_TYPE** property in PLOT1.C.

```
1 #include <windows.h>
2 #include <olch2d.h>
3 #include "plot1.h"
4
5  long WINAPI
6  WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)
7  {
8   static HWND hwndXrt2D;
9   static HXRT2D hChart;
10  static XrtData *my_data;
11
12  switch (msg) {
13  case WM_INITDIALOG:
14      /* Get graph handle */
15      hwndXrt2D = GetDlgItem(hWnd, IDGRAPH);
16      hChart = XrtCreate();
17      XrtAttachWindow(hChart, hwndXrt2D);
18
19      /* Allocate and load data, set it to graph's Data property */
20      my_data = XrtMakeDataFromFile((LPSTR) "mm63.dat", NULL);
21      XrtSetValues(hChart,XRT_DATA, my_data, NULL);
22      break;
23
24  case WM_CLOSE:
25      XrtDestroyData(my_data, TRUE);
26      DestroyWindow(hWnd);
27      break;
28
29  case WM_DESTROY:
30      PostQuitMessage(0);
31      break;
32
33  case XRTN_PALETTECHANGED:
34      SendMessage(XrtGetWindow(hChart), WM_QUERYNEWPALETTE, 0, 0);
35      break;
36
37  case WM_QUERYNEWPALETTE:
38  case WM_PALETTECHANGED:
39      SendMessage(XrtGetWindow(hChart), msg, wParam, lParam);
40      break;
41
42  default:
43      return FALSE;
44  }
45  return TRUE;
46 }
47
48 int PASCAL
49 WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpszCmdLine, int nCmdShow)
50 {
51  static char szAppName[] = "plot1";
52  HWND    hWnd;
53  MSG     msg;
54  WNDCLASS wc;
55  DLGPROC dlgprc;
56
```

continues on next page...

```
57  if (!hPrevInstance) {
58      wc.style            = CS_HREDRAW | CS_VREDRAW;
59      wc.lpfnWndProc      = WndProc;
60      wc.cbClsExtra       = 0;
61      wc.cbWndExtra       = 0;
62      wc.hInstance        = hInstance;
63      wc.hIcon            = LoadIcon (hInstance, IDI_APPLICATION);
64      wc.hCursor          = LoadCursor (NULL, IDC_ARROW);
65      wc.hbrBackground    = GetStockObject (WHITE_BRUSH);
66      wc.lpszMenuName     = NULL;
67      wc.lpszClassName    = szAppName;
68
69      if (!RegisterClass(&wc)) return FALSE;
70  }
71
72  dlgprc = (DLGPROC) MakeProcInstance((FARPROC)WndProc, hInstance);
73  hWnd = CreateDialog(hInstance, szAppName, 0, dlgprc);
74
75  ShowWindow(hWnd, nCmdShow);
76
77  while (GetMessage(&msg, NULL, 0, 0)) {
78      if (!IsDialogMessage(hWnd, &msg)) {
79          TranslateMessage(&msg);
80          DispatchMessage(&msg);
81      }
82  }
83  return msg.wParam;
84 }
```

Figure 2 PLOT1.C listing

1.3 Loading Data From a File

A common task in any Olectra Chart program is to load the chart data (called a *dataset*) into a format that the chart can use. To allocate and load the data from a file, use the **XrtMakeDataFromFile**() function.[1]

Figure 3 shows the contents of the MM63.DAT file. This file is in a format that **XrtMakeDataFromFile**() understands. Lines beginning with a '#' symbol are treated as comments. The first line tells **XrtMakeDataFromFile**() that the data which follows is in **XRT_DATA_ARRAY** style, and is made up of 2 *sets* of 4 *points*.

1. A more general way is to allocate an **XrtData** structure using the *XrtMakeData()* procedure, and then to populate this structure with data using C code.

```
ARRAY 2 4
# X-values
1.0 2.0 3.0 4.0
# Y-values set 1
150.0 175.0 160.0 170.0
# Y-values set 2
125.0 100.0 225.0 300.0
```

Figure 3 The MM63.DAT file

The MM63.DAT file defines 2 *sets* of 4 *points*. Line 3 in the file defines the X-values (i.e. points) shared by both of the sets (Y-values). In this case, each X-value corresponds to one of the columns of Quarterly Results. Lines 5 and 7 in MM63.DAT define each of the sets of Y-values.

There are two types of chart data: **ARRAY** and **GENERAL**. Use **ARRAY** when the sets of Y-values share common X-values. Use **GENERAL** when the Y-values do not share common X-values, or when all sets do not have the same number of values. **GENERAL** data may only be displayed in a plot or area chart; it cannot be displayed as a bar chart or pie chart.

1.4 Changing Chart Type

A powerful feature of Olectra Chart is the ability to change the chart type independently of any other property.[1] We will change the type of the chart from plot to bar by setting the chart's **XRT_TYPE** property as follows:

```
XrtSetValues(hChart, XRT_TYPE, XRT_TYPE_BAR, NULL);
```

Figure 4 displays the resulting chart.

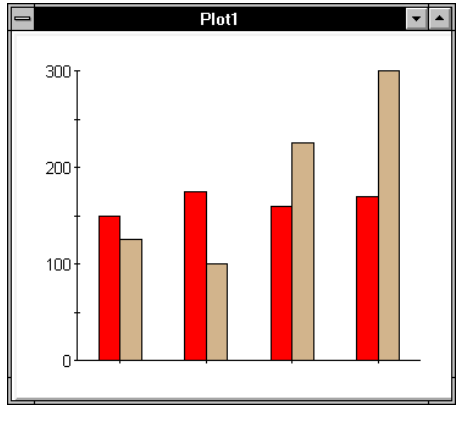


Figure 4 Changing chart type from plot to bar

1. Although there are interdependencies between some properties, most are completely orthogonal. Interdependencies are documented in Appendix A on page 95, "**Property Reference**".

1.5 Adding Header, Footer and Labels

The chart produced by PLOT1.C is not very useful. There is no header, footer, or legend, and the X-axis labelling is not meaningful. Olectra Chart always tries to display a reasonable chart even if very few properties have been specified.[1] We'll change the code to set some other chart properties.

Properties may be changed at any time during the execution of the program with **XrtSetValues**() or **XrtSetPropString**(). You may ask for the current value of any property with **XrtGetValues**() or **XrtGetPropString**().

To add a header, footer, legend, and meaningful X-axis labels, a few more properties will be added to the **XrtSetValues**() call. The following lines specify a header and footer to display above and below the chart:

```
XRT_HEADER_STRINGS, header_strings,
XRT_FOOTER_STRINGS, footer_strings,
```

Both of these properties take a **NULL**-terminated array of strings as their value. These must be defined elsewhere, for example:

```
static char *header_strings[] = { "Michelle's
            Microchips", NULL };
static char *footer_strings[] = { "1963 Quarterly
            Results", NULL };
```

Header, footer and legend strings use the default typeface, Arial. The header is centered above the chart and the footer is centered below it by default. Also, by default, the legend is located to the right of the chart.

All array data used with a chart should be thought of as *n sets* of *m points*. In this case, there are 2 sets of 4 points. Labels for each of the sets and each of the points can be specified as follows:

```
static char *set_labels[] = { "Expenses",
            "Revenue", NULL };
static char *point_labels[] = { "Q1", "Q2",
            "Q3", "Q4", NULL };
```

The chart will use the Set-labels in the legend automatically. To use the Point-labels for the X-axis, we need to set **XRT_XANNOTATION_METHOD** to **XRT_ANNO_POINT_LABELS**. All the changed chart properties are set with the following call:

```
XrtSetValues(hChart,
    XRT_DATA,              my_data,
    XRT_TYPE,              XRT_TYPE_BAR,
    XRT_HEADER_STRINGS,    header_strings,
    XRT_FOOTER_STRINGS,    footer_strings,
    XRT_SET_LABELS,        set_labels,
    XRT_POINT_LABELS,      point_labels,
    XRT_XANNOTATION_METHOD, XRT_ANNO_POINT_LABELS,
    NULL);
```

1. The defaults calculated by Olectra Chart can change quite frequently, for example by changes in the data or in the size of the chart control.

The resulting window is shown below in Figure 5.

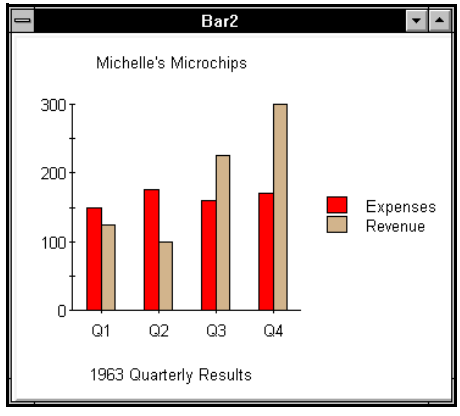


Figure 5 The Bar2 window

1.6 Inverting and Transposing the Chart

Two commonly-used but sometimes confusing chart properties are **XRT_INVERT_ORIENTATION** and **XRT_TRANSPOSE_DATA**.

XRT_INVERT_ORIENTATION inverts the orientation of the *axes*. Most charts display the X-axis horizontally and the Y-axis vertically. It is often appropriate, however, to invert the sense of the X- and Y-axes.

XRT_TRANSPOSE_DATA switches the *meaning of the sets and points* in the dataset. Transposing the dataset defined by the MM63.DAT file (2 sets, each with 4 points) results in 4 sets, each with 2 points.[1] Figure 6 shows the effect of both properties.

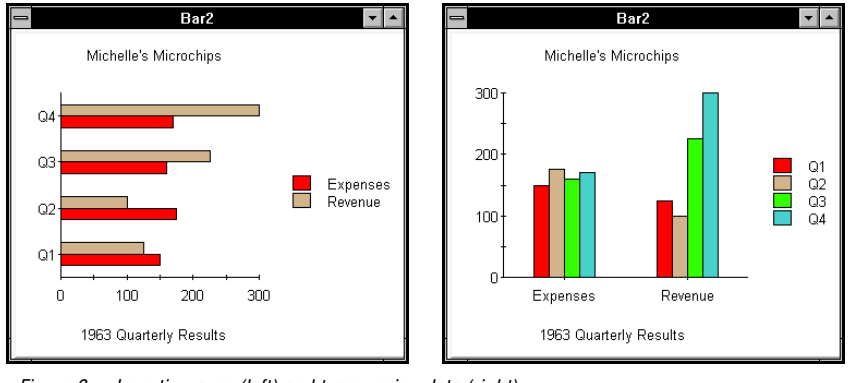


Figure 6 Inverting axes (left) and transposing data (right)

1. Not all datasets are meaningful when transposed.

1.7 Putting it all Together

In PANEL.C, the concepts developed in this chapter are brought together in one program. When PANEL.C is compiled and run, the window in Figure 7 displays. PANEL.C is located in Olectra Chart's \CHART\2D\DEMOS\DLL\SDK\PANEL directory.

The user can change how Michelle's Microchips data is displayed by clicking on one or more of the button controls at the top of the window.

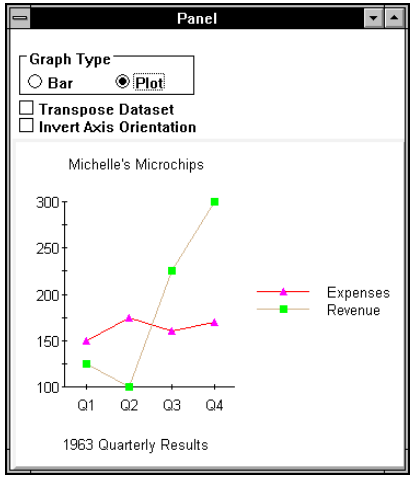


Figure 7 The Panel window

Each button control has code that is invoked when the button is clicked, as shown by the following code for the invert toggle button:

```
case WM_COMMAND:
    switch (wParam) {
    case IDINV:
        /* invert the axes of the chart */
        wCheck = IsDlgButtonChecked(hWnd, IDINV);
        CheckDlgButton(hWnd, IDINV, !wCheck);
        XrtSetValues(hChart,
            XRT_INVERT_ORIENTATION, !wCheck,
            NULL);
        break;
```

The complete PANEL.C program is listed in Appendix F on page 159.

2

Olectra Chart Basics

Terminology ■ *Property Setting and Retrieving*
USE_DEFAULT Properties ■ *Pointer Properties*
String Properties ■ *Font Properties*
Programming with C++ ■ *Distributing Olectra Chart Applications*

Conceptually, Olectra Chart adds a new control to the pre-defined Windows control classes (such as Button, ScrollBar, and ListBox). A Windows programmer manipulates this control similarly to other controls.

2.1 Terminology

Figure 8 shows the major components of the chart control. There are four major areas in every chart: Header, Footer, Legend and Graph. Each area has its own origin, width and height. In Figure 8, each area has been drawn with a plain, 2-pixel-wide border.

2.2 Property Setting and Retrieving

To set the value of chart properties, use either the **XrtSetValues**() or **XrtSetPropString**() procedure. To retrieve the current value of chart properties, use either the **XrtGetValues**() or **XrtGetPropString**() procedure.

SetValues / GetValues

XrtSetValues() and **XrtGetValues**() allow you to set or get any number of properties at once. The following example sets three chart properties to the values contained in the **my_data**, **bg**, and **fg** variables:

```
XrtSetValues(hChart,
    XRT_DATA,               my_data,
    XRT_BACKGROUND_COLOR,   bg,
    XRT_FOREGROUND_COLOR,   fg,
    NULL);
```

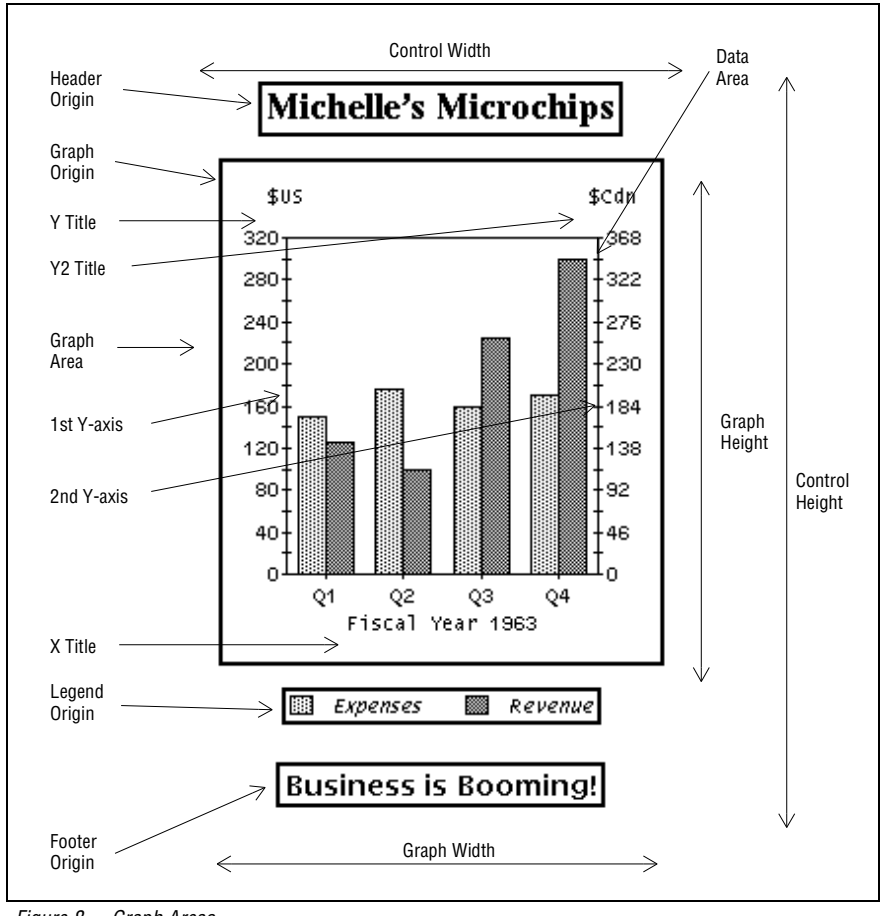


Figure 8 Graph Areas

To retrieve values using **XrtGetValues**(), pass the address of the variable to contain the value retrieved, for example:

```
XrtGetValues(hChart,
    XRT_DATA,               &my_data,
    XRT_BACKGROUND_COLOR,   &bg,
    XRT_FOREGROUND_COLOR,   &fg,
    NULL);
```

Make sure that each variable is of the proper type before calling **XrtGetValues**(). The resource value types are listed at the start of Chapter 3.

SetPropString / GetPropString

XrtSetPropString() and **XrtGetPropString**() allow you to set or get a property value as a string. This is particularly useful for setting fonts, colors, and datastyles. Olectra

Chart converts a string to/from the data type of the property. The following example sets the axis font:

```
XrtSetPropString(hChart,
    XRT_AXIS_FONT,  "Times,10,Bold");
```

To retrieve a value using **XrtGetPropString**(), pass the address of the variable to contain the value retrieved, for example:

```
char *my_font;
XrtGetPropString(hChart,
    XRT_AXIS_FONT,  &my_font);
...
XrtFreePropString(my_font);
```

Use **XrtFreePropString**() to free the string allocated by **XrtGetPropString**() after use.

2.3 USE_DEFAULT Properties

Many Olectra Chart properties have corresponding **USE_DEFAULT** properties, for example, **XRT_YNUM** and **XRT_YNUM_USE_DEFAULT**.

USE_DEFAULT properties are Booleans that determine if Olectra Chart should calculate a default value for the property.

If, for example, **XRT_YNUM_USE_DEFAULT** is **TRUE**, every time a chart is drawn, Olectra Chart will go to considerable effort to determine a reasonable default value. It will consider the size of the graph area, the value of **XRT_YPRECISION**, and possibly other factors. If, on the other hand **XRT_YNUM_USE_DEFAULT** is **FALSE**, Olectra Chart will use the last specified value of **XRT_YNUM** as the **XRT_YNUM** value.

Attempts to set a **USE_DEFAULT** property to **FALSE** will be ignored by Olectra Chart unless the corresponding property has been explicitly set or previously calculated by the control. A side effect of setting any property that has a corresponding **USE_DEFAULT** property is that the **USE_DEFAULT** property will be set to **FALSE**.

The following code will force the value of **XRT_YNUM** to whatever its current value is. It will also have the side effect of setting **XRT_YNUM_USE_DEFAULT** to **FALSE**.

```
double  ynum;
XrtGetValues(hChart, XRT_YNUM, &ynum, NULL);
XrtSetValues(hChart, XRT_YNUM, ynum, NULL);
```

The following code will revert back to the default behavior, which is to have Olectra Chart calculate a default value for **XRT_YNUM** when it draws the chart.

```
XrtSetValues(hChart,
    XRT_YNUM_USE_DEFAULT,       TRUE,
    NULL);
```

2.4 Pointer Properties

Many Olectra Chart properties return pointers to structures or characters when used in an **XrtGetValues**() call. A program should never use this pointer to change any data in memory. Everything the pointer points to should be considered "read-only". The convenience routines **XrtDupDataStyles**() and **XrtDupStrings**() can be used to make copies of some types of data. These copies can be freed with **XrtFreeDataStyles**() and **XrtFreeStrings**().

The only exceptions to this rule are the **XRT_DATA** and **XRT_DATA2** properties. These properties should always point to **XrtData** structures which exist in program memory.

Each chart makes its own copy of all data passed through a pointer property (except for **XRT_DATA** and **XRT_DATA2**).

For example, if a program wants to interchange the first header string with the first footer string, the following code could be used:

```
char    **hs, **fs, **myhs, **myfs, *tmp;
XrtGetValues(hChart,/* Get read-only ptrs */
    XRT_HEADER_STRINGS, &hs,
    XRT_FOOTER_STRINGS, &fs,
    NULL);

myhs = XrtDupStrings(hs); /* Make copies */
myfs = XrtDupStrings(fs);

tmp = myfs[0];      /* Interchange 1st */
myfs[0] = myhs[0];
myhs[0] = tmp;

XrtSetValues(hChart,/* Reset them */
    XRT_HEADER_STRINGS, myhs,
    XRT_FOOTER_STRINGS, myfs,
    NULL);

XrtFreeStrings(myhs);/* Free my copies */
XrtFreeStrings(myfs);
```

2.5 String Properties

Several properties take strings or an array of strings as their values, for example, **XRT_XTITLE** and **XRT_POINT_LABELS**.

Be sure that only displayable characters are specified for these properties. Attempts to display strings containing ASCII control characters (such as '**\n**') may lead to unusual behavior.

If your application is using standard library routines such as **fgets**() to obtain its data, it will have to strip the trailing **\r\n** from each string before using the string as a value for an Olectra Chart property.

2.6 Font Properties

If you are using Microsoft Windows 3.1, you should always cast the property value to an **int** when setting font properties. For example:

```
HFONT   hfont;

XrtSetValues(hChart, XRT_HEADER_FONT, (int)hfont, NULL);
```

2.7 Programming with C++

Olectra Chart provides two C++ interfaces to the control:

1. MFC – For use within version 2.0 or greater of MFC, subclassed from the **CWnd** visual object class.
2. OWL – For use within version 2.5 or greater of OWL, subclassed from the **TControl** ObjectWindows class.

To use the MFC classes, include the OCH2DMFC.H header file and OCH2DMFC.CPP source file in your application. Both files are located in Olectra Chart's \INCLUDE directory. The \CHART\2D\DEMOS\DLL\MFC directory contains a simple MFC example program.

To use the OWL classes, include the OCH2DOWL.H header file and OCH2DOWL.CPP source file in your application. Both files are located in Olectra Chart's \INCLUDE directory. The \CHART\2D\DEMOS\DLL\OWL directory contains a simple OWL example program.

The MFC and OWL implementations create the following classes:

MFC Class	OWL Class	Methods
CChart2D	TChart2D	■ A constructor that creates the chart object. ■ A destructor that calls **DestroyWindow**(). ■ A method for each of the C procedures that take an chart control as its first argument. For example, the method for **XrtGetNthPointLabel**() is **GetNthPointLabel**(). ■ A method to set each settable property. For example, the method to set **XRT_TYPE** is **SetType**(). ■ A method to retrieve each property. For example, the method to retrieve **XRT_BACKGROUND_COLOR** is **GetBackgroundColor**().

MFC Class	OWL Class	Methods
CChart2DData	TChart2DData	■ A constructor that calls one of **XrtMakeData**(), **XrtMakeDataFromFile**(), or **XrtDataCopy**(). ■ A destructor that calls **XrtDestroyData**(). ■ A method for most of the C procedures that take an **XrtData** structure as its first argument. For example, the method for **XrtArrDataAppendPts**() is **ArrDataAppendPts**(). Methods do not exist for **XrtDataExtract**() or **XrtDataConcat**(). ■ Inline methods for **XrtData** macros. ■ Overloaded = operator to perform **XrtDataCopy**(). ■ Casting support for (**XrtData ***).
CChart2DTextArea	TChart2DTextArea	■ A constructor that calls **XrtTextCreate**(). ■ A destructor that calls **XrtTextDestroy**(). ■ A method for each of the C procedures that take an **XrtTextHandle** structure as its second argument. For example, the method for **XrtTextUpdate**() is **TextUpdate**().

2.8 Distributing Olectra Chart Applications

You can freely distribute any applications that you create with Olectra Chart. An Olectra Chart application needs its dynamic link library present on the system it is run on.

Distributing 32-bit Applications

When distributing 32-bit applications, you may only distribute the OLCH2D32.DLL dynamic link library.

Distributing 16-bit Applications

When distributing 16-bit applications, you may only distribute the OLCH2D16.DLL dynamic link library.

3

Programming Olectra Chart

Property Summary ■ *Axis Labelling and Annotation*
Axis Controls ■ *Positioning Graph Areas*
3D Effect ■ *Header, Footer and Legend Display*
Data Styles ■ *Special Bar Chart Properties*
Special Pie Chart Properties ■ *Foreground and Background Colors*
Markers ■ *Area Borders*
Double Buffering ■ *Output and Printing*

3.1 Property Summary

This section summarizes all of the Olectra Chart properties. It is not necessary to remember all the properties in order to program Olectra Chart effectively. For most charts, many properties may be left with their default settings.

The use of most properties is described in this chapter. Appendix A on page 95 contains a reference of all Olectra Chart properties.

Property	Type
XRT_DATA_AREA_BACKGROUND_COLOR	COLORREF
XRT_GRAPH_BACKGROUND_COLOR	COLORREF
XRT_GRAPH_BORDER	XrtBorder (enum)
XRT_GRAPH_BORDER_WIDTH	int
XRT_GRAPH_DEPTH	int
XRT_GRAPH_FOREGROUND_COLOR	COLORREF
XRT_GRAPH_HEIGHT	int
XRT_GRAPH_HEIGHT_USE_DEFAULT	BOOL
XRT_GRAPH_INCLINATION	int
XRT_GRAPH_MARGIN_BOTTOM	int
XRT_GRAPH_MARGIN_BOTTOM_USE_DEFAULT	BOOL
XRT_GRAPH_MARGIN_LEFT	int
XRT_GRAPH_MARGIN_LEFT_USE_DEFAULT	BOOL
XRT_GRAPH_MARGIN_RIGHT	int
XRT_GRAPH_MARGIN_RIGHT_USE_DEFAULT	BOOL
XRT_GRAPH_MARGIN_TOP	int
XRT_GRAPH_MARGIN_TOP_USE_DEFAULT	BOOL
XRT_GRAPH_ROTATION	int
XRT_GRAPH_WIDTH	int
XRT_GRAPH_WIDTH_USE_DEFAULT	BOOL
XRT_GRAPH_[XY]	int
XRT_GRAPH_[XY]_USE_DEFAULT	BOOL
XRT_TYPE	XrtType (enum)
XRT_TYPE2	XrtType (enum)

Figure 9 Chart Properties

Property	Type
XRT_DATA	XrtData *
XRT_DATA2	XrtData *
XRT_FRONT_DATASET	int
XRT_TRANSPOSE_DATA	BOOL
XRT_[XYY2]MAX	double
XRT_[XYY2]MAX_USE_DEFAULT	BOOL
XRT_[XYY2]MIN	double
XRT_[XYY2]MIN_USE_DEFAULT	BOOL

Figure 10 Data Properties

Property	Type
XRT_POINT_LABELS	char * *
XRT_POINT_LABELS2	char * *
XRT_SET_LABELS	char * *
XRT_SET_LABELS2	char * *
XRT_TIME_BASE	time_t
XRT_TIME_FORMAT	char *
XRT_TIME_FORMAT_USE_DEFAULT	BOOL
XRT_TIME_UNIT	XrtTimeUnit (enum)
XRT_[XYY2]ANNOTATION_METHOD	XrtAnnoMethod (enum)
XRT_[XYY2]LABELS	XrtValueLabel * *

Figure 11 Labelling Properties

Property	Type
XRT_DATA_STYLES	XrtDataStyle * *
XRT_DATA_STYLES2	XrtDataStyle * *
XRT_DATA_STYLES_USE_DEFAULT	BOOL
XRT_DATA_STYLES2_USE_DEFAULT	BOOL

Figure 12 Data Style Properties

Property	Type
XRT_MARKER_DATASET	int
XRT_MARKER_DATA_STYLE	XrtDataStyle *
XRT_MARKER_DATA_STYLE_USE_DEFAULT	BOOL
XRT_[XY]MARKER	double
XRT_XMARKER_SET	int
XRT_XMARKER_POINT	int
XRT_[XY]MARKER_SHOW	BOOL

Figure 13 Marker Properties

Property	Type
XRT_AXIS_BOUNDING_BOX	BOOL
XRT_AXIS_FONT	HFONT
XRT_INVERT_ORIENTATION	BOOL
XRT_[XY]ANNO_PLACEMENT	XrtAnnoPlacement (enum)
XRT_[XYY2]ANNOTATION_ROTATION	XrtRotate (enum)
XRT_[XYY2]AXIS_MAX	double
XRT_[XYY2]AXIS_MAX_USE_DEFAULT	BOOL
XRT_[XYY2]AXIS_MIN	double
XRT_[XYY2]AXIS_MIN_USE_DEFAULT	BOOL
XRT_[XYY2]AXIS_SHOW	BOOL
XRT_[XYY2]AXIS_LOGARITHMIC	BOOL
XRT_[XYY2]AXIS_REVERSED	BOOL
XRT_[XY]GRID	double
XRT_[XY]GRID_USE_DEFAULT	BOOL
XRT_[XY]GRID_DATA_STYLE	XrtDataStyle *
XRT_[XY]GRID_DATA_STYLE_USE_DEFAULT	BOOL
XRT_[XYY2]NUM	double
XRT_[XYY2]NUM_USE_DEFAULT	BOOL
XRT_[XYY2]NUM_METHOD	XrtNumMethod (enum)
XRT_[XY]ORIGIN	double
XRT_[XY]ORIGIN_USE_DEFAULT	BOOL
XRT_[XY]ORIGIN_PLACEMENT	XrtOriginPlacement (enum)
XRT_[XYY2]PRECISION	int
XRT_[XYY2]PRECISION_USE_DEFAULT	BOOL
XRT_[XYY2]TITLE	char *
XRT_[XYY2]TITLE_ROTATION	XrtRotate (enum)
XRT_[XYY2]TICK	double
XRT_[XYY2]TICK_USE_DEFAULT	BOOL
XRT_YAXIS_CONST	double
XRT_YAXIS_MULT	double

Figure 14 Axis Properties

Property	Type
XRT_BAR_CLUSTER_OVERLAP	int
XRT_BAR_CLUSTER_WIDTH	int

Figure 15 Bar Properties

Property	Type
XRT_OTHER_DATA_STYLE	XrtDataStyle *
XRT_OTHER_DATA_STYLE_USE_DEFAULT	BOOL
XRT_OTHER_LABEL	char *
XRT_PIE_MIN_SLICES	int
XRT_PIE_ORDER	XrtPieOrder (enum)
XRT_PIE_THRESHOLD_METHOD	XrtPieThresholdMethod (enum)
XRT_PIE_THRESHOLD_VALUE	double

Figure 16 Pie Properties

Property	Type
XRT_HEADER_ADJUST	XrtAdjust (enum)
XRT_HEADER_BACKGROUND_COLOR	COLORREF
XRT_HEADER_BORDER	XrtBorder (enum)
XRT_HEADER_BORDER_WIDTH	int
XRT_HEADER_FOREGROUND_COLOR	COLORREF
XRT_HEADER_STRINGS	char * *
XRT_HEADER_FONT	HFONT
XRT_HEADER_HEIGHT	int
XRT_HEADER_WIDTH	int
XRT_HEADER_[XY]	int
XRT_HEADER_[XY]_USE_DEFAULT	BOOL
XRT_FOOTER_ADJUST	XrtAdjust (enum)
XRT_FOOTER_BACKGROUND_COLOR	COLORREF
XRT_FOOTER_BORDER	XrtBorder (enum)
XRT_FOOTER_BORDER_WIDTH	int
XRT_FOOTER_FOREGROUND_COLOR	COLORREF
XRT_FOOTER_STRINGS	char * *
XRT_FOOTER_FONT	HFONT
XRT_FOOTER_HEIGHT	int
XRT_FOOTER_WIDTH	int
XRT_FOOTER_[XY]	int
XRT_FOOTER_[XY]_USE_DEFAULT	BOOL

Figure 17 Header and Footer Properties

Property	Type
XRT_LEGEND_ANCHOR	XrtAnchor (enum)
XRT_LEGEND_BACKGROUND_COLOR	COLORREF
XRT_LEGEND_BORDER	XrtBorder (enum)
XRT_LEGEND_BORDER_WIDTH	int
XRT_LEGEND_FONT	HFONT
XRT_LEGEND_FOREGROUND_COLOR	COLORREF
XRT_LEGEND_HEIGHT	int
XRT_LEGEND_ORIENTATION	XrtAlign (enum)
XRT_LEGEND_SHOW	BOOL
XRT_LEGEND_WIDTH	int
XRT_LEGEND_[XY]	int
XRT_LEGEND_[XY]_USE_DEFAULT	BOOL

Figure 18 Legend Properties

Property	Type
XRT_BACKGROUND_COLOR	COLORREF
XRT_FOREGROUND_COLOR	COLORREF
XRT_BORDER_WIDTH	int
XRT_DEBUG	BOOL
XRT_DOUBLE_BUFFER	BOOL
XRT_HEIGHT	int
XRT_NAME	char *
XRT_REPAINT	BOOL
XRT_BORDER	XrtBorder (enum)
XRT_WIDTH	int

Figure 19 Other Properties

3.2 Axis Labelling and Annotation

A variety of properties combine to determine how the axes are annotated, and what text appears beside each element in the legend. To understand how and why Olectra Chart operates the way it does, it is important to distinguish between the two types of X-axis used in graphing[1]: *continuous* and *discrete*.

Continuous X-axis

A chart's X-axis is *continuous* when spacing variations between points along the axis are important. Charts displaying general (as opposed to array) data *always* have a

1. Although pie charts don't have axes, they behave like discrete X-axis charts.

continuous X-axis. Plots and area charts have a continuous X-axis only if the data they display is not transposed (**XRT_TRANSPOSE_DATA** is **FALSE**) and the axis is not annotated with Point-labels. Bar and stacking bar charts *never* have a continuous X-axis.

When the X-axis is continuous, it can be annotated by any method except Point-labels and the legend always displays Set-labels.

Discrete X-axis

A chart's X-axis is *discrete* when spacing variations between points along the axis are not important. Bar and stacking bar charts *always* have a discrete X-axis. Plots and area charts have a discrete X-axis if they are transposed (**XRT_TRANSPOSE_DATA** is **TRUE**) or are annotated with Point-labels. Charts displaying general (as opposed to array) data *never* have a discrete X-axis.

When the X-axis is discrete, it can only be annotated with Point-labels (or Set-labels if transposed) and the legend displays Set-labels (or Point-labels if transposed).

Continuous Y-axis

The Y- and Y2-axes are always continuous.

3.2.1 Annotation Method

X-Axis

There are four ways to annotate the X-axis. The method used is determined by the value of **XRT_XANNOTATION_METHOD**. It can be one of the following:

XRT_ANNO_VALUES (default)	Olectra Chart chooses appropriate X-axis annotation based on the data. Data may be in array or general format. The X-axis is continuous.
XRT_ANNO_POINT_LABELS	Olectra Chart evenly spaces the points across the X-axis and annotates them with any supplied Point-labels. The X-axis is discrete. The data must be in array format.
XRT_ANNO_VALUE_LABELS	Olectra Chart annotates the axis using the value-string pairs supplied as Value-labels. The X-axis is continuous. Data may be in array or general format.
XRT_ANNO_TIME_LABELS	Olectra Chart interprets the X-values as time units, and annotates the X-axis with Time-labels. The X-axis is continuous. Data may be in array or general format.

The following sections describe how to use each type of annotation. For each annotation method, Olectra Chart makes use of some axis properties and ignores

others. The following table shows the relationship between each annotation method and the axis properties, for the X-axis.

X-Axis Classification		Axis Properties Used																		
Annotation Method	**Axis Type**	XRT_YANNO_PLACEMENT	XRT_XAXIS_MAX	XRT_XAXIS_MAX_USE_DEFAULT	XRT_XAXIS_MIN	XRT_XAXIS_MIN_USE_DEFAULT	XRT_XAXIS_LOGARITHMIC	XRT_XAXIS_REVERSED	XRT_XGRID	XRT_XGRID_USE_DEFAULT	XRT_XNUM	XRT_XNUM_METHOD	XRT_XNUM_USE_DEFAULT	XRT_XORIGIN	XRT_XORIGIN_PLACEMENT	XRT_XORIGIN_USE_DEFAULT	XRT_XPRECISION	XRT_XPRECISION_USE_DEFAULT	XRT_XTICK	XRT_XTICK_USE_DEFAULT
Values	Continuous	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Point-labels	Discrete									✔										
Value-labels	Continuous	✔	✔	✔	✔	✔	✔	✔	✔	✔				✔	✔	✔			✔	✔
Time-labels	Continuous	✔	✔	✔	✔	✔		✔	✔	✔				✔	✔	✔			✔	✔

Y- and Y2-Axes

There are two ways to annotate the Y- and Y2-axis. The method used is determined by **XRT_[YY2]ANNOTATION_METHOD**. It can be one of the following:

XRT_ANNO_VALUES (default) — Olectra Chart chooses appropriate axis annotation from the dataset.

XRT_ANNO_VALUE_LABELS — Olectra Chart annotates the axis using the value-string pairs supplied as Value-labels. Data may be in array or general format.

3.2.2 Point-labels and Set-labels

Point-labels and Set-labels label the rows and columns of **XRT_DATA_ARRAY** data. If the data is **XRT_DATA_GENERAL**, Point-labels are ignored, and Set-labels are used to label the various *lines* of data.

Consider the POP.DAT file listed in Figure 20. It is in a form suitable for loading with **XrtMakeDataFromFile**(). It defines a 3 by 5 array of values which are the historical and projected population of 3 cities. Since **XrtMakeDataFromFile**() requires X-values to be present, "fake" X-values have been entered in the POP.DAT file.

```
ARRAY 3 5

# X Samples (One for the beginning of each decade 1960 to 2000)
1 2 3 4 5
# Mexico
5 8 14 19 24
# Tokyo
11 15 18 21 22
# New York
14 16 15.5 15.2 15.4
```

Figure 20 The POP.DAT File

Using Olectra Chart terminology, there are "3 *sets* of 5 *points*". The Set-labels are the city names, and the Point-labels are the year names. When this file is used to create an **XrtData** structure called **pop**, the following code can be used to define the labels (the result is shown in Figure 21):

```
static char *sl[] = { "Mexico City", "Tokyo",
                      "New York", NULL };
static char *pl[] = { "60", "70", "80", "90",
                      "2000", NULL };
XrtSetValues(hChart,
    XRT_XANNOTATION_METHOD, XRT_ANNO_POINT_LABELS,
    XRT_DATA,       pop,
    XRT_SET_LABELS  sl,
    XRT_POINT_LABELS,pl,
    NULL);
```

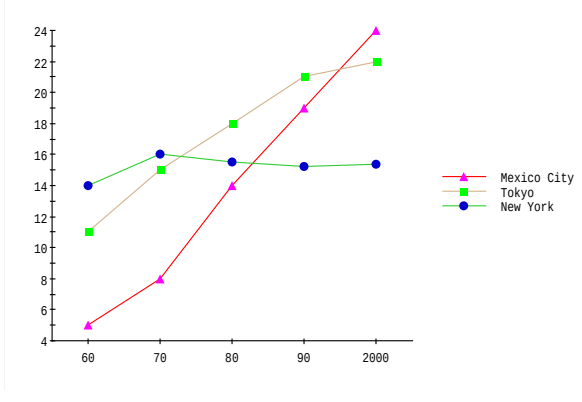


Figure 21 Point-label X-axis Annotation and Set-label Legend Text

As an alternative to using the **XRT_SET_LABELS** and **XRT_POINT_LABELS** properties, individual labels can be accessed with the *SetNth* and *GetNth* methods, which are documented in Appendix B on page 117. For example:

```
char buffer[100];
/* Note that index is zero-based */
sprintf(buffer, "Third Set-label is %s",
    XrtGetNthSetLabel(hChart, 2) );
MessageBox(hWnd, buffer, "Information",  MB_OK);

/* Make 3rd Point-label "eighty" */
XrtSetNthPointLabel(hChart, 2, "eighty");
```

It is sometimes useful to *transpose* the sets and points. When **XRT_TRANSPOSE_DATA** is **TRUE**, Olectra Chart will consider the rows and columns to be switched, and the Point-labels and Set-labels to be switched.

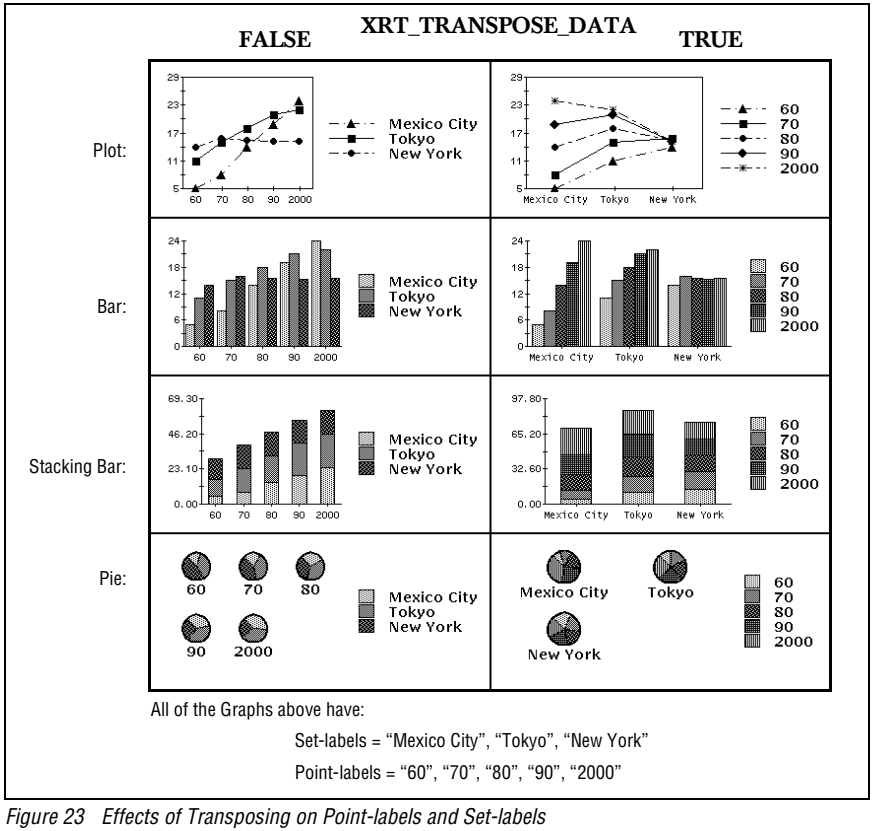
If the data for a chart is in **XRT_DATA_GENERAL** format, it does not make sense to define Point-labels or to transpose the data. In this case, Olectra Chart will ignore **XRT_POINT_LABELS** and **XRT_TRANSPOSE_DATA** when displaying charts.

Figure 22 and Figure 23 illustrate how Point-labels and Set-labels interact with data transposition.

When Graph Type is	and XRT_TRANSPOSE_DATA is	then Legend annotation comes from:	and X-Axis annotation comes from:
Bar/Stack Bar	FALSE	Set-labels	Point-labels
Bar/Stack Bar	TRUE	Point-labels	Set-labels
Pie	FALSE	Set-labels	Point-labels[1]
Pie	TRUE	Point-labels	Set-labels[2]
Plot/Area	FALSE	Set-labels	(depends)[3]
Plot/Area	TRUE	Point-labels	Set-labels

1. Point-labels are used to annotate each pie.
2. Set-labels are used to annotate each pie.
3. This could be a continuous graph. X annotation depends on **XRT_XANNOTATION_METHOD**.

Figure 22 Use of Point-labels/Set-labels on different chart types



All of the Graphs above have:

Set-labels = “Mexico City”, “Tokyo”, “New York”

Point-labels = “60”, “70”, “80”, “90”, “2000”

Figure 23 Effects of Transposing on Point-labels and Set-labels

3.2.3 Value-labels

When **XRT_[XYY2]ANNOTATION_METHOD** is **XRT_ANNO_VALUE_LABELS**, Olectra Chart uses text strings you specify to label the axis. Use “Value-labels” to label significant axis values. Each Value-label is defined by an **XrtValueLabel** structure:

```
typedef struct {
    double  value;
    char    *string;
} XrtValueLabel;
```

value defines the X-, Y-, or Y2-value at which to display the label, and *string* is the text to display. An entire array of Value-labels can be programmed for an axis using **XRT_[XYY2]LABELS**. Alternatively, the **XrtSetValueLabel**() and **XrtGetValueLabel**() procedures can be used to access individual Value-labels.

Value-labels should be used whenever it is important to honor the data's X-or Y-values, and custom axis annotation is required. For example, to put a custom Value-label of "foobar" on the X-axis at X=7.43, the following code could be used:

```
XrtValueLabelmylabel;
my_label.value = 7.43;
my_label.string = "foobar";
XrtSetValueLabel(hChart, XRT_AXIS_X, &mylabel);
```

The **XrtGetValueLabel**() method takes a pointer to an **XrtValueLabel** structure, and returns a pointer to another **XrtValueLabel** structure. The returned structure is filled in with the value-string pair of the Value-label that is closest to the value in the structure that was passed in. If there are no Value-labels, it returns **NULL**. The following example finds the closest Value-label to X=5.0:

```
XrtValueLabel my_label, *closest_label;
char buffer[100];
my_label.value = 5.0;
closest_label = XrtGetValueLabel(hChart,
            XRT_AXIS_X, &my_label);
if (closest_label == NULL)
    MessageBox(hWnd, "No X-axis Value-labels
        exist!", "Error", MB_OK);
else {
    sprintf(buffer, "Closest is at %f, text is %s",
        closest_label->value,
        closest_label->string);
    MessageBox(hWnd, buffer, "Information", MB_OK);
}
```

To clear all Value-labels for an axis, set **XRT_[XYY2]LABELS** to **NULL**.

To delete a particular Value-label, set its *string* to **NULL**. Beware of floating-point round-off errors when deleting Value-labels. The best way to delete a Value-label is to first get it, set the string to **NULL**, and then set it again. The following example deletes the label closest to X=5.0:

```
XrtValueLabellabel, *label_ptr;

label.value = 5.0;
label_ptr = XrtGetValueLabel(hChart,
            XRT_AXIS_X, &label);
if (label_ptr == NULL)
    MessageBox(hWnd, "No labels to delete!",
        "Error", MB_ICONEXCLAMATION | MB_OK);
else {
    label = *label_ptr;
    label.string = NULL;
    XrtSetValueLabel(hChart, XRT_AXIS_X, &label);
}
```

Common uses for Value-labels include annotating particular dates on the X-axis and replacing Y-values with more meaningful text. Figure 24 shows an example of Value-labels on both axes.

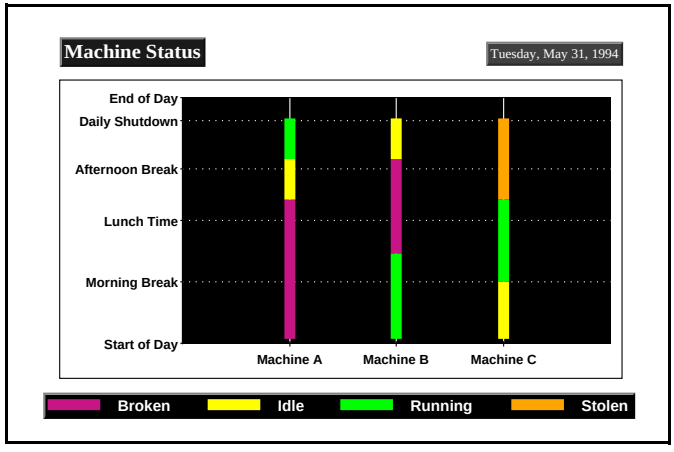


Figure 24 Using Value-labels to Annotate axes

3.2.4 Time-axis Labels

Time-axis Overview

Olectra Chart provides special support for an X-axis that represents the time dimension. Properties that are used to specify and control a time-axis are:

XRT_TIME_BASE	Specifies the moment in time your X-values begin measuring.
XRT_TIME_UNIT	Specifies the time units the X-values are measured in.
XRT_TIME_FORMAT	Specifies the formatting style used for annotating the time-axis.
XRT_TIME_FORMAT_USE_DEFAULT	Specifies that the time format should be calculated dynamically.
XRT_XANNOTATION_METHOD	Specifies the method used for annotating the X-axis. Set this to **XRT_ANNO_TIME_LABELS** for Time-labels.

Time-axis Criteria

The time-axis is ideal for applications that graph something measured in seconds, minutes, hours, days, weeks, months or years. Several criteria must be met before you can use a time-axis:

- The X-axis must be continuous (that is, the chart must be either a plot or area chart, and **XRT_TRANSPOSE_DATA** must be **FALSE**).

- The time range represented by the axis must be greater than the beginning of the year 1970 and less than the year 2038.
- The annotation resolution desired must not be less than one second.

If your needs do not meet these criteria, you will have to use another method for annotating the X-axis, such as Value-labels.

Time Units

When you set **XRT_XANNOTATION_METHOD** to **XRT_ANNO_TIME_LABELS** on a continuous X-axis chart, Olectra Chart will interpret the X-values in your data using time units. Seconds are the default time unit, so if your X-values range from a low of 20 to a high of 55, Olectra Chart will draw an X-axis that spans 35 seconds.[1]

You can set a different time unit using the **XRT_TIME_UNIT** property. Valid values are: **XRT_TMUNIT_SECONDS**, **XRT_TMUNIT_MINUTES**, **XRT_TMUNIT_HOURS**, **XRT_TMUNIT_DAYS**, **XRT_TMUNIT_WEEKS**, **XRT_TMUNIT_MONTHS** and **XRT_TMUNIT_YEARS**.

Time Base

The data's X-values will be measured relative to the time base. For example, if the X-values are days in 1993, (9 means Jan. 10, 31 is Feb. 1 etc.) you will need to set the time base to the beginning of the 1993 year. The **XRT_TIME_BASE** defines the time from which your X-values begin measuring. This property is a *long* that dynamically defaults to Jan. 1, 1970 in your time zone. It represents the number of seconds since Jan. 1, 1970 GMT. Since your data is quite likely to begin at a different time, you will almost always have to set this property when using a time-axis.

The **XrtMakeTime**() procedure is useful for converting a normal date and time into the number of seconds since Jan. 1, 1970. For example, if the base time for your data is April 19, 1993, 08:30:05, you could use the following code to set the time base:

```
long tval;
struct tm ltime;

/* Set time base to April 19, 1993, 08:30:05 */
tval = XrtMakeTime(93, 4-1, 19, 8, 30, 5);
XrtSetValues(hChart, XRT_TIME_BASE, tval, NULL);
```

Once you have set **XRT_TIME_UNIT** and **XRT_TIME_BASE**, you can use the **XrtTimeToValue**() procedure to convert a time stored as a **time_t** to its equivalent position on the Time-axis. The **XrtValueToTime**() procedure converts a Time-axis position to its equivalent **time_t** value.

Time Format

Olectra Chart uses the value of the string specified by **XRT_TIME_FORMAT** and your system's ANSI C standard function **strftime**() for formatting the time-axis labels. Since **strftime**() does vary slightly from system to system, you should read your

1. This assumes that you haven't fixed the value of **XRT_XAXIS_MIN**, **XRT_XAXIS_MAX**, **XRT_XMIN** or **XRT_XMAX**.

system's documentation to understand the details of how this string is interpreted. A summary is provided with the **XRT_TIME_FORMAT** description in Appendix A.

To let Olectra Chart calculate an appropriate time format by default, set **XRT_TIME_FORMAT_USE_DEFAULT** to **TRUE**.

The following table shows some examples of how the time 06:55:50 on Monday April 12, 1993 could be formatted:

Time Format String	Example
%Y	1993
%b %y	Apr 93
%b	Apr
%b %d	Apr 12
%a %d	Mon 12
%a	Mon
%a %H:00	Mon 06:00
%H:%M	06:55
%H:%M:%S	06:55:50

To display tick marks at the time annotation, set **XRT_XTICK_USE_DEFAULT** to **TRUE**.

Time-axis Example

The TIME.C program, located in Olectra Chart's \CHART\2D\DEMOS\DLL\SDK\TIME directory, reads in the file \CHART\2D\SAMPLES\TIME.DAT. This data file has 12 X-values ranging from 0 to 11, representing the 12 months of the year 1992. When TIME.C is compiled and run, the window below appears:

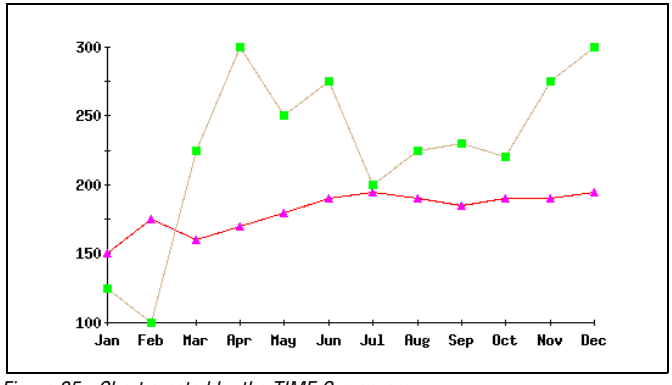


Figure 25 Chart created by the TIME.C program

```
ARRAY 2 12
# x values, for jan to dec 1992.
0 1 2 3 4 5 6 7 8 9 10 11
# y values line 1
150 175 160 170 180 190 195 190 185 190 190 195
# y values line 2
125 100 225 300 250 275 200 225 230 220 275 300
```

Figure 26 The TIME.DAT data file

3.3 Axis Controls

This section describes how to program the chart axes, including numbering, titling, and visual attributes. This section concentrates on the X- and Y-axes. Information on using a second Y-axis (Y2) is found in section 6.1 on page 83.

3.3.1 Axis Numbering

Numbering Method

XRT_[XYY2]NUM_METHOD specifies the method used to number the axis when annotation method is **XRT_ANNO_VALUES**. Valid values are:

XRT_NUM_ROUND (default)	Axis numbering is rounded to values whose most-significant digit ends in 1, 2, or 5. It uses the value of **XRT_[XYY2]PRECISION** to format the annotation.
XRT_NUM_PRECISION	Axis numbering is calculated using the value of **XRT_[XYY2]PRECISION**, as well as **XRT_[XYY2]NUM** and **XRT_[XYY2]TICK**.

When **XRT_NUM_ROUND**, Olectra Chart numbers the axes by analyzing the data to be charted, determining numbering increment, ticking increment, and appropriate numbering precision. Then it rounds the numbering and draws the axis. This method usually provides the most pleasing numbering.

When **XRT_NUM_PRECISION**, numbering is done the same as described above, except the numbering is not rounded.

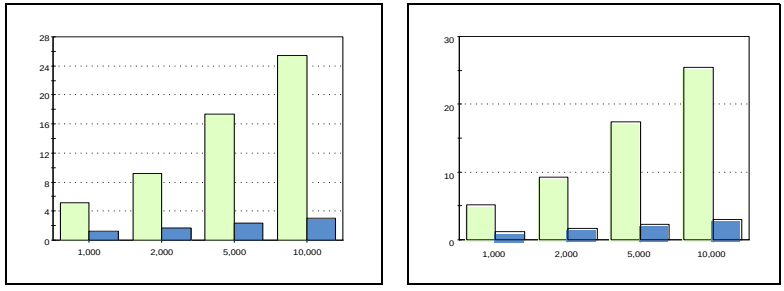


Figure 27 "Precision" and "Rounded" Axis Numbering

Precision

XRT_[XYY2]PRECISION serves two purposes, depending on the axis numbering method used:

- If **XRT_[XYY2]NUM_METHOD** is **XRT_NUM_ROUND**, it specifies only the formatting of the axis numbering.
- If **XRT_[XYY2]NUM_METHOD** is **XRT_NUM_PRECISION**, it in large part controls the axis numbering.

Positive values of **XRT_[XYY2]PRECISION** indicate the number of places after the decimal place. Negative values indicate the (positive) number of zeros to use before the decimal place.

For example, if **XRT_XPRECISION** is -3 and **XRT_YPRECISION** is 2, the X-axis numbering will be multiples of 1000 and the Y-axis numbering will be shown to 2 decimal places.

When **XRT_[XYY2]PRECISION_USE_DEFAULT** is **TRUE**, Olectra Chart determines appropriate precision at run-time.

Numbering Increment

The **XRT_[XYY2]NUM** property is used to specify the interval between annotations along the axis. For example, if **XRT_YNUM** is 3.2, Olectra Chart will annotate every 3.2 units along the Y-axis, starting at the origin.

Tick Increment

XRT_[XYY2]TICK is similar to **XRT_[XYY2]NUM** except that ticks are drawn instead of increment-labels. Numbering increments do not have to be multiples of tick increments. The following table shows how X-axis tick and grid increments

interact with the various X-axis annotation methods (described in section 3.2 on page 26).

X-axis Annotation Method	When USE_DEFAULT is TRUE	When USE_DEFAULT is FALSE, and Grid/Tick = 0.0	When USE_DEFAULT is FALSE, and Grid/Tick = 0.0
Point-labels	ticks and grid-lines display at labels	ticks display at labels, no grid-lines	ticks and grid-lines display at labels
Value-labels or Time-labels	ticks and grid-lines display at labels	no ticks no grid-lines	tick and grid values are honored

Be careful when explicitly setting the numbering, ticking and grid increment properties. If they are set too small, charts or axes may be illegible. To allow Olectra Chart to determine appropriate numbering and tick increments, set the **USE_DEFAULT** properties to **TRUE**.

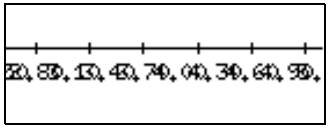


Figure 28 Axis with Numbering Increment Set too Small

Logarithmic Axes

To convert from a linear to a $\log_{10}$ axis, set **XRT_[XYY2]AXIS_LOGARITHMIC** to **TRUE**. Since logarithmic axes are only capable of plotting values greater than zero, any data values less than or equal to zero are treated as holes (that is, as missing data values). It is not possible to set the numbering increment, ticking increment, or precision for a logarithmic axis. Axis or data minimum/maximum or origin cannot be set less than or equal to zero.

Logarithmic X-axes are subject to the following additional rules:

- The X-axis must be continuous.
- The X-axis annotation method cannot be **XRT_ANNO_TIME_LABELS**.

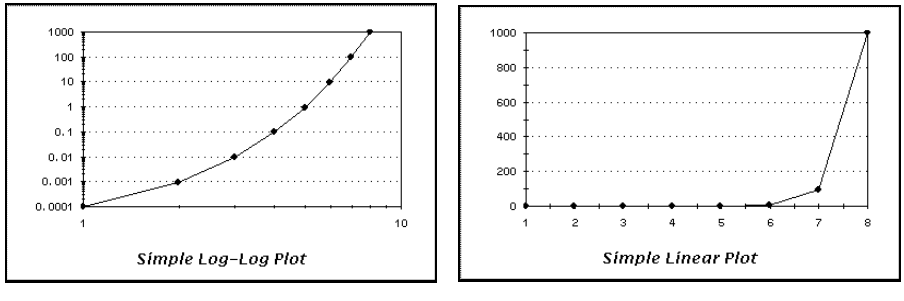


Figure 29 Linear and Logarithmic Axes

3.3.2 Axis and Data Bounds

By default, Olectra Chart displays all data in the **XrtData** structure. It determines the extent of the axes (**XRT_[XYY2]AXIS_MIN** and **XRT_[XYY2]AXIS_MAX**) based on the minimum and maximum data values **XRT_[XYY2]MIN** and **XRT_[XYY2]MAX**), the origin, and the numbering increment. You can frame the chart data by either specifying new bounds for the axes or new bounds for the data. Generally, framing the axis gives better results than framing the data, due to the way Olectra Chart calculates axis extents.

Axis Min and Max

XRT_[XYY2]AXIS_MIN and **XRT_[XYY2]AXIS_MAX** specify the minimum and maximum axis values to be displayed. Use these properties to frame a chart display precisely at particular *axis* values. For example, the following code "zooms" in on the center of a chart by a factor of 2:

```
double      xmax, xmin, ymax, ymin;
double      xmax2, xmin2, ymax2, ymin2;
XrtGetValues(hChart,
    XRT_XAXIS_MIN, &xmin,
    XRT_XAXIS_MAX, &xmax,
    XRT_YAXIS_MIN, &ymin,
    XRT_YAXIS_MAX, &ymax,
    NULL);
xmin2 = xmin + (xmax - xmin) / 4.0;
xmax2 = xmax - (xmax - xmin) / 4.0;
ymin2 = ymin + (ymax - ymin) / 4.0;
ymax2 = ymax - (ymax - ymin) / 4.0;
XrtSetValues(hChart,
    XRT_YAXIS_MIN, xmin2,
    XRT_XAXIS_MAX, xmax2,
    XRT_YAXIS_MIN, ymin2,
    XRT_YAXIS_MAX, ymax2,
    NULL);
```

Olectra Chart will override any invalid hard-coded axis bounds by setting the corresponding **USE_DEFAULT** property to **TRUE**. For example, if **XRT_YAXIS_MAX** is -20 (and **XRT_YAXIS_MAX_USE_DEFAULT** is **FALSE**) and **XRT_YAXIS_LOGARITHMIC** is set to **TRUE**, **XRT_YAXIS_MAX_USE_DEFAULT** will be set to **TRUE** because logarithmic axes can only display values greater than zero.

Data Min and Max

XRT_[XYY2]MAX and **XRT_[XYY2]MIN** specify the minimum and maximum data values to be displayed. Use these properties when you want to frame a chart display at particular *data* values.

3.3.3 Origin

Origin Coordinates

You can specify the coordinates of the origin with **XRT_[XY]ORIGIN**. For example, to force the Y-axis to cross the X-axis at X=10, set **XRT_XORIGIN** to 10.0.

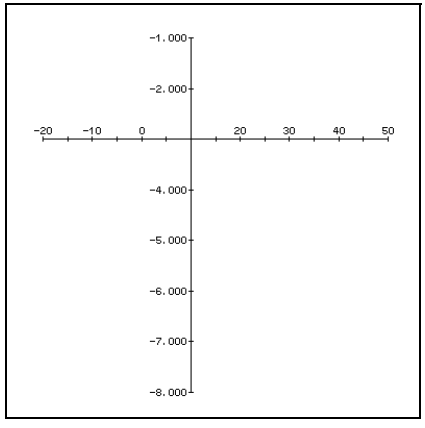


Figure 30 Axes with Origin (10, -3), X Precision is -1, Y Precision is 3

Origin Placement

You can set the placement of the axis origins on plot, area, and bar charts with **XRT_[XY]ORIGIN_PLACEMENT**. Valid values are:

XRT_ORIGIN_AUTO (default)	For plot/area charts, the origin is placed at the minimum axis value or at zero if the dataset contains positive and negative values. For bar charts, the origin is placed at zero.
XRT_ORIGIN_ZERO	Origin is placed at 0.
XRT_ORIGIN_MIN	Origin is placed at the minimum axis value.
XRT_ORIGIN_MAX	Origin is placed at the maximum axis value.

This property is ignored when **XRT_[XY]ORIGIN_USE_DEFAULT** is **FALSE**; setting an origin explicitly overrides this property.

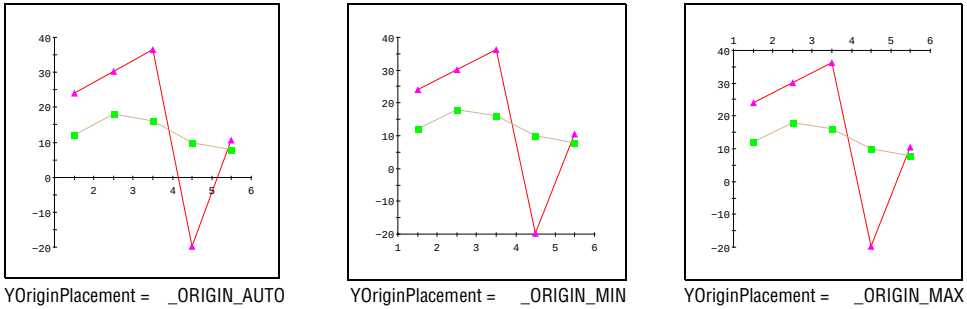


Figure 31 Effect of Origin Placements

3.3.4 Annotation/Title Rotation and Placement

Title Rotation

The title on the vertical axes (that is, the Y- and Y2-axis unless **XRT_INVERT_ORIENTATION** is **TRUE**) can be rotated 90 degrees counter-clockwise by setting **XRT_[XYY2]TITLE_ROTATION** to **XRT_ROTATE_90**.[1]

You can also rotate titles 270 degrees counter-clockwise by setting the rotation property to **XRT_ROTATE_270**. 270-degree rotation usually looks best on the Y2-axis. To return to horizontal titles, set the rotation property to **XRT_ROTATE_NONE**.

Titles on the horizontal axis (that is, the X-axis unless **XRT_INVERT_ORIENTATION** is **TRUE**) cannot be rotated.

Annotation Rotation

Axis annotation can be rotated using **XRT_[XYY2]ANNOTATION_ROTATION**.

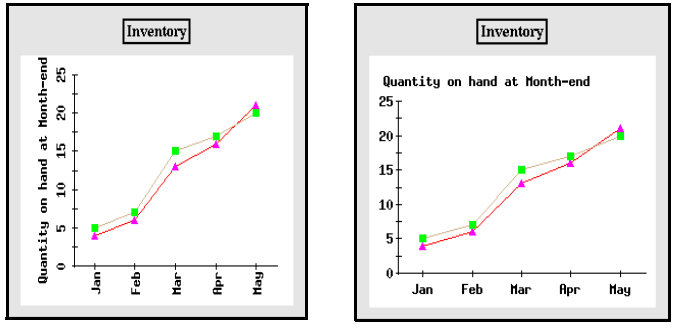


Figure 32 Rotating Annotation and Title by 90 Degrees

Annotation/Title Placement

You can control the placement of axis annotation and titles with **XRT_[XY]ANNO_PLACEMENT**. You can fix each axis' annotation/title at the origin, the axis minimum, or the axis maximum. Valid values are:

XRT_ANNO_AUTO (default)	Annotation is placed automatically. For plot and area charts, annotation is placed at the origin. For bar and stacking-bar charts, annotation is placed at the end of the axis closest to the origin.
XRT_ANNO_ORIGIN	Annotation is placed at the origin.
XRT_ANNO_MIN	Annotation is placed at the minimum axis value.

1. The axis font (specified by **XRT_AXIS_FONT**) must be scalable (e.g. TrueType) when rotating the annotation/title. Olectra Chart cannot rotate raster or bitmap fonts.

XRT_ANNO_MAX Annotation is placed at the maximum axis value.

XRT_YANNO_PLACEMENT is ignored on dual-axis charts and discrete X-axis charts.

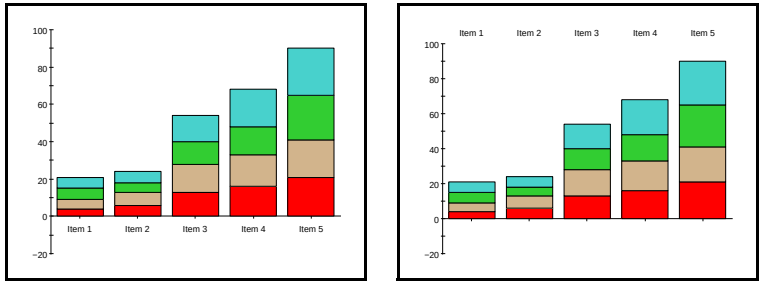


Figure 33 Placing X-Axis Annotation at Y-Origin and Y-Maximum

3.3.5 Other Axis Controls

Ignored Properties

All axis properties are ignored when **XRT_TYPE** is **XRT_TYPE_PIE**. Most X-axis properties are ignored on charts with a discrete X-axis (see section 3.2 on page 26 for a definition). The only axis properties not ignored are **XRT_XTITLE**, **XRT_XAXIS_SHOW**, **XRT_XGRID_DATA_STYLE** and **XRT_XGRID_DATA_STYLE_USE_DEFAULT**.

Title

The **XRT_[XYY2]TITLE** property may be used to specify a title for each axis.

Orientation

When **XRT_INVERT_ORIENTATION** is **FALSE**, all the X-axis properties apply to the horizontal axis, and the Y-axis properties apply to the vertical axis. When **XRT_INVERT_ORIENTATION** is **TRUE**, the X-axis is vertical, and the Y-axis is horizontal.

Margins

You can adjust the space between the axes and the edge of the graph area with the margin properties (**XRT_GRAPH_MARGIN_TOP**, **XRT_GRAPH_MARGIN_BOTTOM**, **XRT_GRAPH_MARGIN_LEFT** and **XRT_GRAPH_MARGIN_RIGHT**). Each property specifies the number of pixels between one edge of the graph area and its axis. By default, Olectra Chart calculates margins automatically (the **USE_DEFAULT** properties are **TRUE**), setting enough space to display axis annotation and titles.

Margins can be used to:

- Line up the axes of several chart controls; for example, use **XRT_GRAPH_MARGIN_LEFT** to line up several charts along the Y-axis.

- Scale a chart; for example, “zoom in” by setting all the Margin properties to negative values, “zoom out” by setting them to positive values.
- Translate a chart; for example, increase **XRT_GRAPH_MARGIN_TOP** and decrease **XRT_GRAPH_MARGIN_BOTTOM** by the same amount to shift the chart down in the graph area.

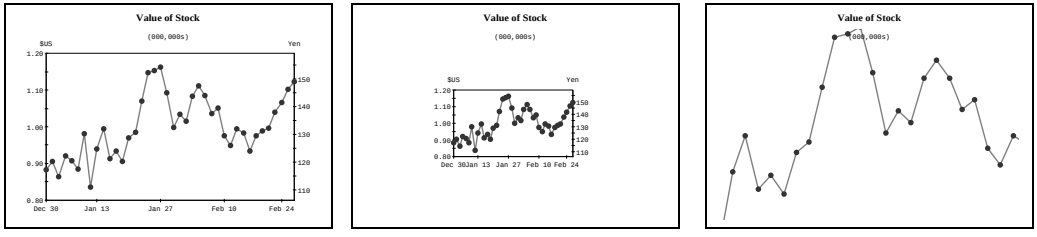


Figure 34 Using Margins to scale a chart within the graph area

Grid-lines

XRT_[XY]GRID specifies the spacing between grid-lines. By default, these properties are 0.0, resulting in no grid-lines at all. If **XRT_[XY]GRID_USE_DEFAULT** is **TRUE**, **XRT_[XY]GRID** will track **XRT_[XY]NUM**. On discrete X-axis charts, X grid-lines will appear at each X-axis label. The style of the grid is determined by **XRT_[XY]GRID_DATA_STYLE** and **XRT_[XY]GRID_DATA_STYLE_USE_DEFAULT**. See the table on page 38 for information on how X-axis grid-lines interact with the various X-axis Annotation Methods (which are discussed in section 3.2).

Axes Font

By default, the font used for axes numbering and titles is 12-point Arial. You may change the font to any font available on your system using the **XRT_AXIS_FONT** property. Changing the axes font is similar to changing the header, footer or legend font, as discussed in section 3.6 on page 46.

Reversed Axis

You can change the direction of the X-, Y-, and Y2-axes by setting **XRT_[XYY2]AXIS_REVERSED** to **TRUE**.

X-axis–Reversal is only allowed if the X-axis is continuous. A similar effect can be achieved on charts with a discrete X-axis by reversing the order of the Point-labels and the data.

Y- and Y2-axis–Once a relationship has been specified between the Y- and Y2- axis, you can only specify the direction of the Y-axis. If the relationship multiplier is negative, the direction of the Y2-axis will be the opposite of the Y-axis.

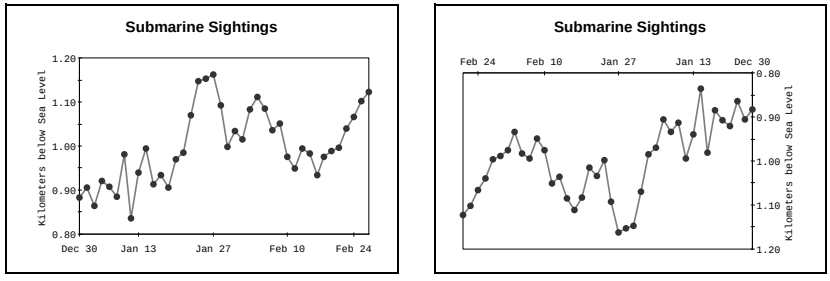


Figure 35 Normal and Reversed X- and Y-axes

Axis Show

XRT_[XYY2]AXIS_SHOW specifies whether Olectra Chart should draw the axis at all. If **FALSE**, the axis will not be drawn.

Bounding Box

To cause a box to be drawn which surrounds the axes, set **XRT_AXIS_BOUNDING_BOX** to **TRUE**.

3.4 Positioning Graph Areas

Each of the four areas (header, footer, legend and graph) will by default be positioned by Olectra Chart at run-time. The default positioning for each area depends on a large number of factors, including:

- The control's current width and height.
- The size of the legend, header and footer areas. These, in turn, depend on the text and fonts being used.
- The value of the **XRT_LEGEND_ANCHOR** property.
- The positioning of areas which have been explicitly positioned by the user program.

Olectra Chart's default positioning algorithms will size and position the header, footer and legend areas first. The graph area will be sized and positioned to fit into the largest remaining rectangular area.

A program can determine and adjust area positioning through the use of the positioning properties.

XRT_HEADER_X	XRT_HEADER_Y	(XRT_HEADER_WIDTH)	(XRT_HEADER_HEIGHT)
XRT_FOOTER_X	XRT_FOOTER_Y	(XRT_FOOTER_WIDTH)	(XRT_FOOTER_HEIGHT)
XRT_LEGEND_X	XRT_LEGEND_Y	(XRT_LEGEND_WIDTH)	(XRT_LEGEND_HEIGHT)
XRT_GRAPH_X	XRT_GRAPH_Y	XRT_GRAPH_WIDTH	XRT_GRAPH_HEIGHT

(Properties in Parentheses are Read-Only)

Figure 36 Area Positioning Properties

When used in an **XrtGetValues**() call, positioning properties will return the values used the last time the chart was displayed. These properties do not have meaningful values until they are either explicitly set, or a chart has been displayed at least once.

A program should not explicitly set any of the positioning properties unless it is prepared to recalculate them when the control's size changes. See section 5.8 on page 80 for more information on handling window resizing.

In some situations, it may be worthwhile to explicitly set some or all of the positioning properties. For example, if the program will be displaying data that changes in real-time, the default positioning may change slightly with each redisplay. These small positioning changes could distract the user. In this situation, the program should use one of the following strategies:

- Hardcode all the positioning properties. Window resizing should not be allowed.

or

- Let Olectra Chart calculate default positioning for all areas when the chart first displays, and for the first display after any window resize. After the first display, explicitly set the positioning properties to the values calculated by Olectra Chart.

3.5 3D Effect

Data in bar, stacking-bar and pie charts can be displayed with a three-dimensional appearance by setting the **XRT_GRAPH_DEPTH**, **XRT_GRAPH_INCLINATION** and **XRT_GRAPH_ROTATION** properties. These properties have no effect on plots, area charts and combination charts.

Chart depth is the apparent depth as a percentage of chart width. Inclination is the eye's position above the X-axis, measured in degrees. Rotation is the number of degrees the eye is positioned to the right of the Y-axis. Rotation has no effect on pie charts.

37 3D Effect on Bar Charts with Depth equal to 30

38 3D Effect on Pie Charts with Depth equal to 30

3.6 Header, Footer and Legend Display

Unless a program has explicitly positioned areas, Olectra Chart will always attempt to display the header at the top, and the footer at the bottom of the window. The default legend position depends on the value of the **XRT_LEGEND_ANCHOR** property. See section 3.4 on page 44 for details on overriding the default area positioning.

Header & Footer Text

The header and footer areas can both contain multiple lines of text. The text will be aligned within the area, depending on the value of the **XRT_HEADER_ADJUST** or **XRT_FOOTER_ADJUST** property. The values **XRT_ADJUST_LEFT**, **XRT_ADJUST_RIGHT** and **XRT_ADJUST_CENTER** cause the text to be left-justified, right-justified or centered. **XRT_ADJUST_CENTER** is the default.

The text for the header and footer areas is specified using the **XRT_HEADER_STRINGS** and **XRT_FOOTER_STRINGS** properties. Both of these properties have **NULL**-terminated arrays of strings as their values.

The code below will set two header lines and left-adjust them:

```
static char *hs[] = { "Experiment A Results",
            "Last 60 Days", NULL };
XrtSetValues(hChart,
    XRT_HEADER_STRINGS, hs,
    XRT_HEADER_ADJUST,  XRT_ADJUST_LEFT,
    NULL);
```

Legend

The text displayed in the legend corresponds to either the Set-labels or the Point-labels. If the data is transposed, the Point-labels are used. Otherwise the Set-labels are used. For more information on Point-labels and Set-labels, see section 3.2.2 on page 28.

By default, Olectra Chart will attempt to list the legend contents vertically, and position the legend to the right of the graph area.

The legend layout is controlled through the **XRT_LEGEND_ORIENTATION** property. It may be either **XRT_ALIGN_VERTICAL** or **XRT_ALIGN_HORIZONTAL**. If the legend is too large to fit in one row or column, Olectra Chart will attempt to layout the legend in several rows or columns.

The default legend positioning relative to the graph area is controlled with the **XRT_LEGEND_ANCHOR** property. Valid values correspond to the eight points of the compass: **XRT_ANCHOR_NORTH**, **XRT_ANCHOR_SOUTH**, **XRT_ANCHOR_EAST**, **XRT_ANCHOR_WEST**, **XRT_ANCHOR_NORTHWEST**, **XRT_ANCHOR_NORTHEAST**, **XRT_ANCHOR_SOUTHEAST**, and **XRT_ANCHOR_SOUTHWEST**.

Font Specification

A font may be specified for each of the header, footer and legend areas and also for the axes annotation. Olectra Chart can use any font available on the system at runtime.

Use the **CreateFont**() or **CreateFontIndirect**() Windows API call to create an **HFONT** structure for use with Olectra Chart's font properties. The Windows API **EnumFontFamilies**() function determines which fonts are available on the system. Consult your Windows programming documentation for further details on finding and setting fonts.

Another way to set a font property is to use the **XrtSetPropString**() function. This allows you to avoid creating and destroying an **HFONT** or **LOGFONT** structure and set a font using a simple string, such as "Arial,24,Italic".

The following example uses both methods to set font properties:

```
/* Use CreateFont() to set Header font */
HFONT   hFont;
hFont = CreateFont(24, 0, 0, 0, 0,  /* Set only Size & */
    0, 0, 0, 0, 0, 0, 0, 0, "MS Serif");    /* Typeface */
XrtSetValues(hChart, XRT_HEADER_FONT, hFont, NULL);
DeleteObject(hFont);

/* Use XrtSetPropString() to set Legend font */
XrtSetPropString(hChart, XRT_LEGEND_FONT,
            "Times,12,bold");
```

For information on setting font properties in the Windows 3.1 environment, refer to section 2.6 on page 19.

3.7 Data Styles

How a data value looks when it is displayed (i.e. color, line pattern, fill pattern, point style, line thickness etc.) depends on the data style that has been defined for that data value.

For example, the values in the third set of data will be rendered on screen using the third data style. If the data is transposed, the values of the third point in each set of data will be rendered using the third data style.

Default Data Styles

If **XRT_DATA_STYLES_USE_DEFAULT** is **TRUE**, the chart will use its default data styles. Even after setting explicit data styles, a program may again use default data styles by setting **XRT_DATA_STYLES_USE_DEFAULT** to **TRUE**.

The **XrtDataStyle** structure is declared in the OLCH2DCM.H header file:

```
typedef struct {
    XrtLinePattern  lpat;   /* line pattern */
    XrtFillPattern  fpat;   /* fill pattern */
    COLORREF        color;  /* color */
    int             width;  /* line width */
    XrtPoint        point;  /* point style */
    COLORREF        pcolor; /* point color */
    int             psize;  /* point size */
} XrtDataStyle;
```

The **XrtDataStyle** data structure contains all the information about how a set of data will be represented graphically. The fields are broken down as follows:

lpat The line pattern used for plots. Must be one of the **XRT_LPAT_** constants listed in Figure 39.

fpat The fill pattern used in area charts and bar and pie charts. Must be one of the **XRT_FPAT_** constants listed in Figure 40.

color The color used when drawing lines in plots and for fills in area charts and bar and pie charts. This is a valid Windows color value. See section 3.10 on page 55 for details of color specification.

width The line width used for plots. Must be greater than or equal to 1. Line width must be 1 when using a dashed or dotted line pattern on ***Windows 3.x*** and ***Windows 95***.

point The point style used for plots. Must be one of the **XRT_POINT_** constants listed in Figure 41.

pcolor The point color used for points in plots. This is a valid Windows color value. See section 3.10 on page 55 for details of color specification.

psize The size of points that appear in plots. Must be greater than or equal to 0. A size of 0 will result in no point being drawn. A point size is a relative measure. Do not assume that a point size of 12 means that the point's glyph will be exactly 12 pixels from top to bottom.

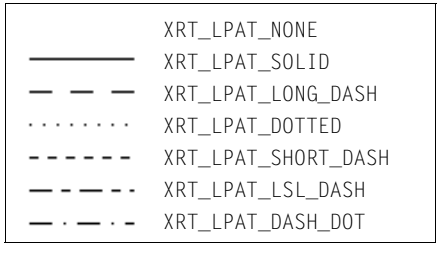


Figure 39 Line Patterns

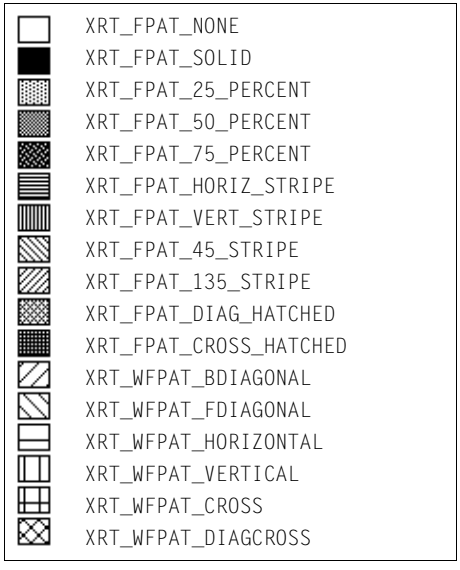


Figure 40 Fill Patterns

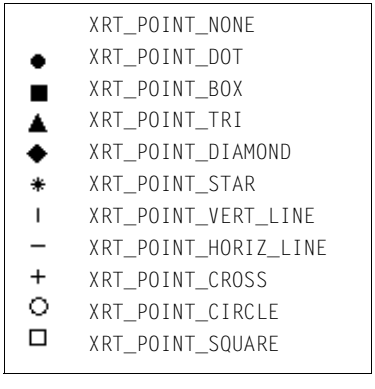


Figure 41 Point Styles

Explicitly Setting Data Styles

It is usually satisfactory to use Olectra Chart's default data styles. If a program explicitly sets any data styles, it is responsible for ensuring that the data style colors are different from the graph area and data area background colors.

A side effect of setting any data style element explicitly is that the **XRT_DATA_STYLES_USE_DEFAULT** property will be set to **FALSE**. Olectra Chart will use the default data styles for any sets of data without an explicit data style. The most straightforward way to set data styles in a program is shown in Figure 42.

```
static XrtDataStyle color_styles[] = {
/* 0 */ { XRT_LPAT_SOLID, XRT_FPAT_SOLID, RGB(255,0,0), 1,  XRT_POINT_TRI,
        RGB(255,0,255), XRT_DEFAULT_POINT_SIZE },
/* 1 */ { XRT_LPAT_SOLID, XRT_FPAT_SOLID, RGB(255,192,128), 1,  XRT_POINT_BOX,
        RGB(128,255,128), XRT_DEFAULT_POINT_SIZE },
/* 2 */ { XRT_LPAT_SOLID, XRT_FPAT_SOLID, RGB(0,255,0), 1,  XRT_POINT_DOT,
        RGB(128,128,255), XRT_DEFAULT_POINT_SIZE },
/* 3 */ { XRT_LPAT_SOLID, XRT_FPAT_SOLID, RGB(128,255,223), 1,
        XRT_POINT_DIAMOND, RGB(255,0,0), XRT_DEFAULT_POINT_SIZE },
/* 4 */ { XRT_LPAT_SOLID, XRT_FPAT_SOLID,    RGB(0,0,255), 1,   XRT_POINT_STAR,
        RGB(255,0,0), XRT_DEFAULT_POINT_SIZE },
/* 5 */ { XRT_LPAT_SOLID, XRT_FPAT_SOLID, RGB(255,0,255), 1,   XRT_POINT_TRI,
        RGB(128,128,255), XRT_DEFAULT_POINT_SIZE },
/* 6 */ { XRT_LPAT_SOLID, XRT_FPAT_SOLID, RGB(192,255,0), 1,   XRT_POINT_BOX,
        RGB(255,0,255), XRT_DEFAULT_POINT_SIZE },
/* 7 */ { XRT_LPAT_SOLID, XRT_FPAT_SOLID, RGB(128,128,255), 1,   XRT_POINT_DOT,
        RGB(128,255,255), XRT_DEFAULT_POINT_SIZE },
};

static XrtDataStyle *my_dstyles_ptr[] = {
     &color_styles[0],
     &color_styles[1],
     &color_styles[2],
     &color_styles[3],
     &color_styles[4],
     &color_styles[5],
     &color_styles[6],
     &color_styles[7],
     NULL
};

...
       HWND hwndXrt2D;
       HXRT2D hChart;
...
       hChart = XrtCreateWindow("", 5,5,300,200,hWnd, hInstance);
       hwndXrt2D = XrtGetWindow(hChart);
...
       XrtSetValues(hChart, XRT_DATA_STYLES, my_dstyles_ptr, NULL);
```

Figure 42 Setting Data Styles

Accessing Data Styles

Data styles are accessed through the **XRT_DATA_STYLES** property and/or through the **XrtSetNthDataStyle**() and **XrtGetNthDataStyle**() methods.

XRT_DATA_STYLES can be used to get or set the entire array of data styles. When used with **XrtGetValues**(), the returned array pointer should be considered read-only. The **XrtDupDataStyles**() convenience routine may be used to duplicate the

returned data styles array pointer. For example, to set all line widths to 5-pixels thick, the following code could be used:

```
int              i;
XrtDataStyle     **ds, **myds;
XrtGetValues(hChart, XRT_DATA_STYLES, &ds, NULL);
myds = XrtDupDataStyles(ds);
for(i=0; myds[i]; i++){
    myds[i]->width = 5;
}
XrtSetValues(hChart,
    XRT_DATA_STYLES, myds,
    NULL);
XrtFreeDataStyles(myds);
```

A program can get a pointer to a particular **XrtDataStyle** structure using **XrtGetNthDataStyle**(). For example, if **hChart** is a plot of the population data in Figure 20, the following code will print out the line thickness being used for the set of Tokyo data, and double it:

```
XrtDataStyle    *dstyle, my_dstyle;
char            buffer[50];
dstyle = XrtGetNthDataStyle(hChart, 1);
sprintf(buffer, "line width for 2nd set of data:
            %d", dstyle->width);
MessageBox(hWnd, buffer, "Information", MB_OK);
my_dstyle = * dstyle;
my_dstyle.width = my_dstyle.width * 2;
XrtSetNthDataStyle(hChart, 1, &my_dstyle);
```

3.8 Special Bar Chart Properties

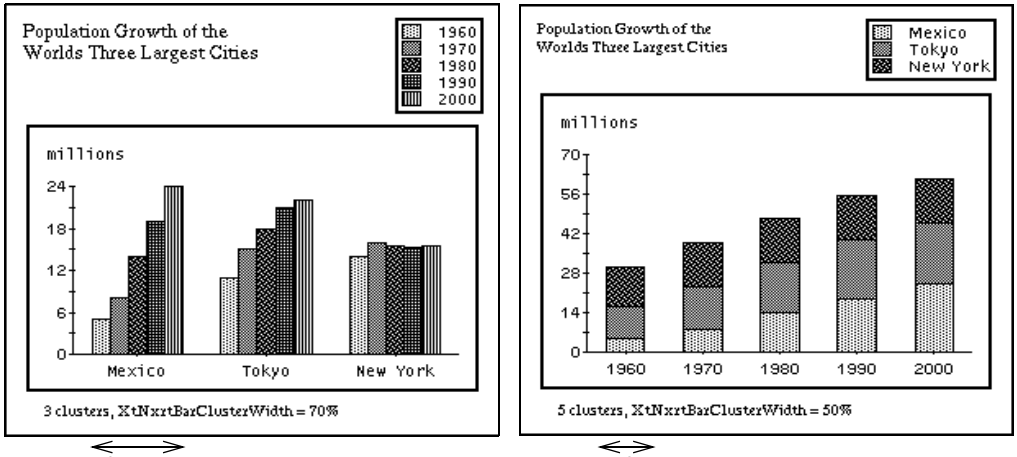
Bar charts have two special properties that control exactly how the bars are sized and spaced. Bar charts have their bars grouped into *clusters*. When the data is not transposed, there is a cluster for each point. When the data is transposed, there is a cluster for each set.

Cluster Width

In non-stacking bar charts, bars are grouped into clusters. To specify how much of the available cluster space should be occupied by the bars, use the **XRT_BAR_CLUSTER_WIDTH** property. By default, **XRT_BAR_CLUSTER_WIDTH** is 50, which means that the bars will occupy 50% of the available cluster space.

Cluster Overlap

The spacing between bars in the same cluster is controlled using the **XRT_BAR_CLUSTER_OVERLAP** property. Cluster overlap is specified as a percentage of the bar's width.



Maximum Cluster Width is 1/3 of total available.

Maximum Cluster Width is 1/5 of total available.

Figure 43 Cluster Width in Bar Charts and Stacking Bar Charts

Stacking Bar Charts

Data will be displayed as a stacking bar chart when **XRT_TYPE** is set to **XRT_TYPE_STACKING_BAR**. In stacking bar charts, all data values less than zero are treated as zero. Since stacking bar charts have only one bar per cluster, **XRT_BAR_CLUSTER_OVERLAP** has no effect.

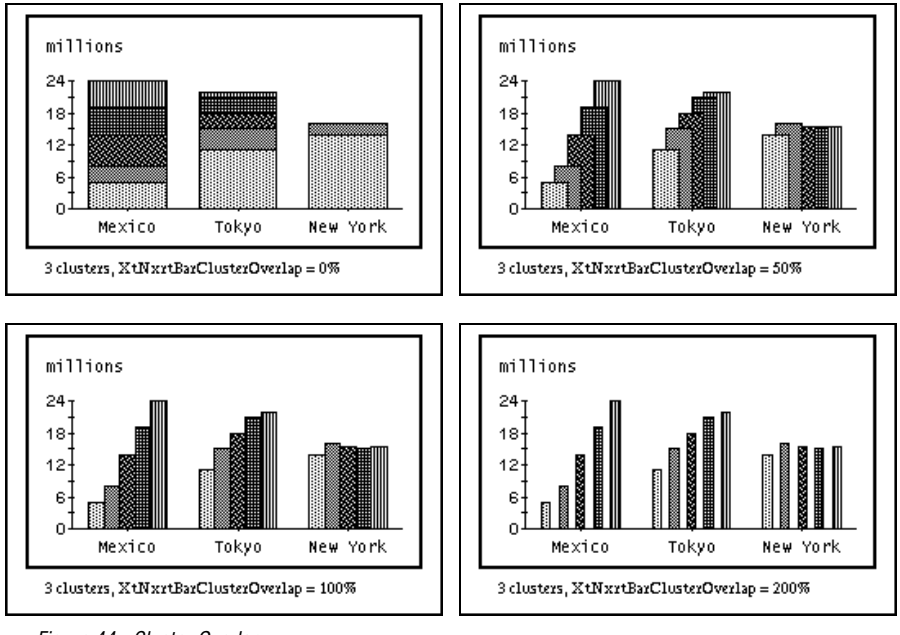


Figure 44 Cluster Overlap

3.9 Special Pie Chart Properties

Pie charts are significantly different from plots and bar charts since they do not have the notion of a 2-dimensional grid which is common to plots and bar charts. Pie charts also introduce the concept of an *other* category, into which all values below a specified threshold are grouped.

Olectra Chart will draw a pie chart comparing each point across all sets. To begin, the sum of all of the first points is calculated (then all of the second points etc.). Then the percentage of the sum is calculated for each point. It is suggested that data structures with large numbers of points not be displayed as pie charts. Negative data values are treated as zero when displayed in a pie chart.

To compare the relative sizes of each of the points within a set, make **XRT_TRANSPOSE_DATA TRUE**. This will also cause the Point-labels to appear in the legend.

Because of the unique nature of pie charts, Olectra Chart defines a few properties which pertain only to charts when **XRT_TYPE** is **XRT_TYPE_PIE**.

Other Slice Threshold

There are two factors which contribute to threshold interpretation. *Threshold method* and *threshold value* determine which values will be displayed with their own slice, and which will contribute to the *other* slice.

If **XRT_PIE_THRESHOLD_METHOD** is **XRT_PIE_SLICE_CUTOFF**, Olectra Chart will examine the value of **XRT_PIE_THRESHOLD_VALUE**. All data values whose percentage of the total is less than the *threshold value* are grouped into the *other* slice.

If **XRT_PIE_THRESHOLD_METHOD** is **XRT_PIE_PERCENTILE**, as many of the smallest data values as necessary are grouped into the *other* slice so that the *other* slice is less than or equal to the *threshold value* percent of the total.

The **XRT_PIE_THRESHOLD_VALUE** defines a floating-point value between 0.0 and 100.0. To disable creation of the *other* slice, set **XRT_PIE_THRESHOLD_VALUE** to 0.0.

An example of various threshold methods is shown in Figure 45.

Other Slice Label

The label used to annotate the *other* slice is determined by **XRT_OTHER_LABEL**. By default, the *other* slice is labelled “Other”. This property can be used to specify a more appropriate label.

Other Slice Data Style

XRT_OTHER_DATA_STYLE specifies the data style to be used for displaying the *other* pie slice. See section 3.7 on page 48 for more information on the **XrtDataStyle** structure. By default, Olectra Chart renders the *other* slice on pie charts in a solid fill pattern of yellow. An application can go back to using the default data style for the *other* slice by setting **XRT_OTHER_DATA_STYLE_USE_DEFAULT** to **TRUE**.

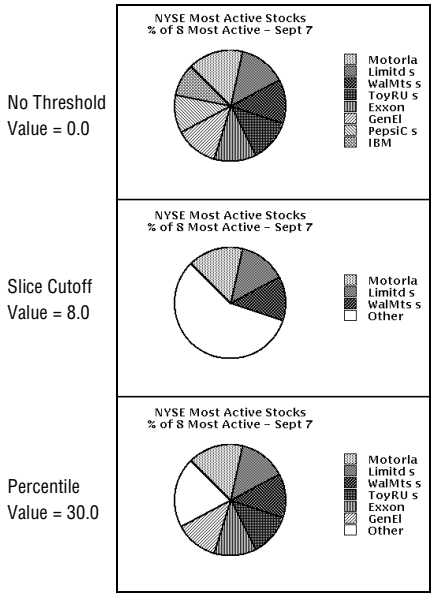


Figure 45 Threshold Methods

Pie Ordering

The order of pie slices around the pie is determined by **XRT_PIE_ORDER**. It must be one of **XRT_PIEORDER_ASCENDING**, **XRT_PIEORDER_DESCENDING** or **XRT_PIEORDER_DATA_ORDER**. If it is **XRT_PIEORDER_DATA_ORDER**, the slices not contributing to *other* are displayed in the same order that they appear in the data. The *other* slice is always the last slice in any ordering.

Minimum Slices

Finally, **XRT_PIE_MIN_SLICES** specifies a minimum number of slices to display. This property has precedence over the threshold method and value. If, for example, **XRT_PIE_MIN_SLICES** is set to 5, Olectra Chart will always try to display 5 slices before grouping any data into the *other* slice.

All-Zero Pies

If all of a pie's data values are zero (or negative), no pie will be drawn and it will not be possible to set a marker on the pie. Also, the **XrtPick**() procedure will indicate the correct pie index (as if the pie had been drawn), but the slice index will always be zero. **XrtPick**() will always return a non-zero distance value for all-zero pies.

3.10 Foreground and Background Colors

Olectra Chart supports the specification of colors for the window background and foreground, as well as for the lines, fill patterns and points that represent data in the

chart itself. Olectra Chart will choose default colors for the application, so simple applications need not concern themselves with color specification.

Background Colors

The window background color is specified through the **XRT_BACKGROUND_COLOR** property. It is white by default.

Each of the header, footer, legend and graph areas also have a background color which is **XRT_DEFAULT_COLOR** (transparent) by default. For example, the property to change the legend background color is **XRT_LEGEND_BACKGROUND_COLOR**. The data area of the chart also has its own background color, specified by **XRT_DATA_AREA_BACKGROUND_COLOR**.

Foreground Colors

The window foreground color is black by default, and is specified with the **XRT_FOREGROUND_COLOR** property. It will be used as the foreground color for each of the header, footer, legend and graph areas, unless a different foreground color is specified for one or more of these areas. For example, to specify a different header foreground color, use the **XRT_HEADER_FOREGROUND_COLOR** property.

Data Colors

Data styles specify the colors used for graphed data (among other things). For information on customizing data styles, see section 3.7 on page 48.

Specifying Colors

All color properties take a valid Windows color reference as their value. Use **XrtSetValues**() to set a property to a color reference created with the **RGB** macro.

Alternately, use **XrtSetPropString**() to set a property to a named color string, as shown by the following example:

```
XrtSetPropString(hChart,
    XRT_BACKGROUND_COLOR,"skyblue");
```

Olectra Chart recognizes over 200 colors (ranging from “Aquamarine” to “YellowGreen”). A list of recognized colors can be found in the file OC_COLOR.H, located in Olectra Chart’s \INCLUDE directory.

Palette Handling

Because the charts created by Olectra Chart look best when rendered using solid colors, Olectra Chart automatically adds new solid colors to the Windows palette when creating the chart or changing color properties. This saves the programmer the step of allocating a new color in the palette before setting a color property. If the palette is full, the color is set to the nearest palette color or by dithering the closest palette colors, depending on the macro or function you used to specify the color.

Once a chart is created, an application should not update or change the Windows palette directly. Changes to any of the chart’s colors should be made by updating the appropriate chart properties (such as **XRT_BACKGROUND_COLOR** and **XRT_DATA_STYLES**).

The Windows color palette is a shared resource. In some situations, the number of colors in use by all the applications being displayed is more than the number of palette entries available. In this case, the colors in some windows will "flash" to inappropriate colors as the user uses different applications.

Palette Notification Message

To ensure proper color palette handling, your application needs to handle the **XRTN_PALETTECHANGED** notification message, as well as **WM_QUERYNEWPALETTE** and **WM_PALETTECHANGED**. The **XRTN_PALETTECHANGED** message is sent to a chart's parent window after the chart control has changed its color palette. Refer to Appendix D for more information on this message.

The PLOT1 example program in Chapter 1 shows how to handle these messages for single-chart applications. See the STOCK example in Olectra Chart's \CHART\2D\DEMOS\DLL\SDK\STOCK directory and the MSDN articles "Palette Awareness" and "The Palette Manager: How & Why" for help with multiple-chart applications.

3.11 Markers

Markers provide an application with a way to identify a particular data element on the screen, or a particular value in a plot. There are many potential uses for markers, but the most likely one is to provide feedback to the application user when a particular part of a chart is selected with the mouse.

Plots, area charts, bar charts and stacking bar charts support the notion of both an X- and Y-marker. On plots, area charts and bar charts, the X-marker is a line drawn across the graph area perpendicular to the X-axis. On stacking bar charts, the X-marker is a short line drawn through one of the bar blocks. The Y-marker is a line drawn across the graph area perpendicular to the Y-axis. The thickness, color and style of line is defined by **XRT_MARKER_DATA_STYLE**. By default, it is a black, 1-pixel wide, dashed line.

Pie charts support the notion of just one marker: a line drawn from the center of the pie and extending outward through the center of the selected slice.

For combination charts (see section 6.2 on page 84), specify the marker dataset using **XRT_MARKER_DATASET**.

Marker Performance

Updating the marker positioning properties occurs very quickly. This is because Olectra Chart does not repaint the entire chart window to reposition the markers. Marker performance is improved by setting **XRT_DOUBLE_BUFFER** to **TRUE**. Generally though, chart performance is improved by turning double-buffering off.

X-marker Positioning

The position of the X-marker is defined by setting both the **XRT_XMARKER_SET** and **XRT_XMARKER_POINT**, *or* the **XRT_XMARKER** properties.

XRT_XMARKER_SET and **XRT_XMARKER_POINT** *must* be used when positioning the X-marker on discrete X-axis charts, and *may* be used for continuous X-axis charts. (See section 3.2 on page 26 for a definition of *discrete* and *continuous* axis.) **XRT_XMARKER_SET** and **XRT_XMARKER_POINT** are indices into the **XrtData** structure that together define the data point through which the X-marker will be drawn.

On continuous X-axis charts, the X-marker position may also be specified through the use of the **XRT_XMARKER** floating-point property. In this case, the **XRT_XMARKER** value is the X-axis value through which the marker will be drawn.

Y-marker Positioning

The Y-marker is positioned through the **XRT_YMARKER** floating-point property. It specifies the value on the Y-axis through which the Y-marker will be drawn. This property is ignored when the chart is a pie.

Neither X- nor Y-markers will appear unless the appropriate **XRT_[XY]MARKER_SHOW** property is set to **TRUE**.

3.12 Area Borders

Each of the 4 graph areas (header, footer, legend and graph) may be enhanced with a border. There is a Border and a Border Width property for each of these areas (for example, **XRT_GRAPH_BORDER** and **XRT_GRAPH_BORDER_WIDTH**). In addition, you can specify a border for the entire control.

The Border properties may be set to any of: **XRT_BORDER_NONE**, **XRT_BORDER_3D_OUT**, **XRT_BORDER_3D_IN**, **XRT_BORDER_ETCHED_IN**, **XRT_BORDER_ETCHED_OUT**, **XRT_BORDER_SHADOW** or **XRT_BORDER_PLAIN**. The width of the border (in pixels) is controlled with the corresponding Border Width property. The width must be between 0 and 20.

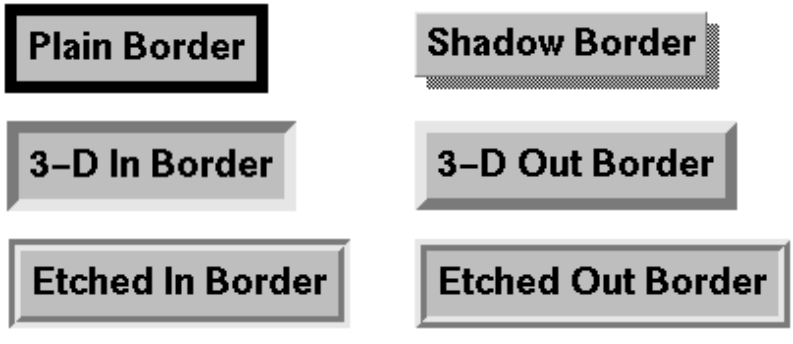


Figure 46 Border Types

3.13 Double Buffering

Double buffering is a graphics technique which will reduce the amount of flashing perceived by a user when a chart changes.

When **XRT_DOUBLE_BUFFER** is **TRUE**, every time Olectra Chart changes a chart it will:

- allocate (if necessary) and clear an off-screen bitmap.
- render the complete chart to the off-screen bitmap.
- copy the off-screen bitmap to the screen.

When **XRT_DOUBLE_BUFFER** is **FALSE**, every time Olectra Chart changes a chart it will clear the screen image (possibly causing a visual flash) and then render the complete image to the visible window (possibly allowing the user to see the chart being drawn piece by piece).

By default, **XRT_DOUBLE_BUFFER** is **TRUE**. However, setting it to **FALSE** can improve the graphing performance and reduce memory requirements.

3.14 Output and Printing

Many applications need to provide the user with a way to get a hardcopy of a chart. Olectra Chart provides several procedures which may be used to output chart representations to files or printers. These procedures work correctly even when the chart window is obscured by other windows.

Output/Printing Procedures

The procedures listed below are fully documented in Appendix B on page 117:

XrtDrawToClipboard()	Outputs a chart image to the Windows clipboard using the graphics format you specify.
XrtDrawToDC()	Outputs a chart image to any device context (DC) at the scale and graphics format you specify.
XrtDrawToFile()	Outputs a chart image to a file using the graphics format you specify.
XrtPrint()	Outputs a chart image to a printer, using the Windows Print dialog box.

Printing Charts

To print a chart using the standard Windows Print dialog box, call **XrtPrint**(). This procedure allows you to specify the image format and the size/position as described above; its only difference is that it displays the Print dialog box.

XrtDrawToDC()

Use the **XrtDrawToDC**() procedure for more complex chart output. For instance, to print several charts on one page, call **XrtDrawToDC**() for each chart, using the *top*, *left*, *width* and *height* arguments to specify each chart's different size/position. Figure 47 shows using **XrtDrawToDC**() for a printing function that prints a chart to the default printer without using the Windows Print dialog box.

```
BOOL
printGraph(HXRT2D hChart)
{
    PRINTDLGpd;
    DOCINFO di;
    HDC     hdc;
    TEXTMETRIC tm;
    RECT    rect;

    memset ((void *) &pd, 0, sizeof(pd));
    pd.lStructSize = sizeof(pd);
    pd.hwndOwner   = NULL;
    pd.Flags       = PD_RETURNDC | PD_RETURNDEFAULT;
    pd.hInstance   = NULL;
    PrintDlg(&pd);
    hdc = pd.hDC;

    if (!hdc) {
        return (FALSE);
    }

    di.cbSize = sizeof(di);
    di.lpszDocName = "My Graph";
    di.lpszOutput = NULL;

    StartDoc(hdc, &di);
    StartPage(hdc);

    GetTextMetrics(hdc, &tm);

    /* Define graph size and output graph */
    rect.left = tm.tmAveCharWidth * 2;
    rect.top = (tm.tmHeight + tm.tmExternalLeading) * (3);
    rect.right = tm.tmAveCharWidth * 42;
    rect.bottom = (tm.tmHeight + tm.tmExternalLeading) * (25);

    XrtDrawToDC(hChart, hdc, XRT_DRAW_METAFILE, XRT_DRAWSCALE_NONE,
     rect.left, rect.top, rect.right-rect.left, rect.bottom-rect.top)

    EndPage(hdc);
    EndDoc(hdc);
    DeleteDC(hdc);
    return (TRUE);
}
```

Figure 47 Non-interactive chart printing procedure

4

Olectra Chart Data

Getting Data into Charts ■ *The XrtData Structure*
Changing the Data ■ *Example Using Static Data*

4.1 Getting Data into Charts

Data to be displayed can originate from diverse sources: files, databases, real-time data feeds, or even unrelated processes running on the machine.

Perhaps the most difficult task facing an Olectra Chart developer is retrieving the data to be displayed, and inserting it into an **XrtData** structure for displaying. This section discusses the **XrtData** structure in detail and offers several examples of allocating and loading the **XrtData** structure.

Data from a File

If the data exists in a file (and it does not need to be changed or updated in real time) there are two options to consider. The first option is to massage the data so that the file conforms to a syntax understood by **XrtMakeDataFromFile**(). This procedure will allocate the **XrtData** structure and load it with data from a named file. **XrtMakeDataFromFile**() is documented in Appendix B on page 117.

Another approach is to allocate an **XrtData** structure using **XrtMakeData**(), and populate it with data by reading the data file (perhaps using **fgets**() or **fscanf**()). When the structure is loaded, the chart can be created and the **XRT_DATA** property set to point to the **XrtData** structure.

Fast Update Procedures

For some types of real-time applications, particularly when creating a strip-chart that has data continually being added to it, or a scatter plot which is receiving new points, the Fast Update procedures will be helpful. See section 6.5 on page 90 for more information.

Responsibility for XrtData

It is the application's responsibility to allocate and destroy all required **XrtData** structures. Olectra Chart does not make a copy of the data, instead it references data in the application's memory which is pointed to by **XRT_DATA**.

If Olectra Chart has to repaint the chart window, it will use the data pointed to by **XRT_DATA**.

4.2 The XrtData Structure

When Olectra Chart begins to draw a chart, it uses the data values located in the **XrtData** structure pointed to by the **XRT_DATA** property.

The **XrtData** structure is defined in the OLCH2DCM.H header file. (It is also listed in Appendix E.) The **XrtData** structure itself is defined as a union of **XrtArray** and **XrtGeneral**:

```
typedef union {
    XrtArray    a;
    XrtGeneral  g;
} XrtData;
```

When **XrtMakeData**() and **XrtMakeDataFromFile**() allocate **XrtData** structures, they initialize the **type** field to **XRT_DATA_ARRAY** or **XRT_DATA_GENERAL**. They also initialize the **hole** field to **XRT_HUGE_VAL**. The type and hole field are common to both *a* and *g* elements.

Holes in the Data

Whenever Olectra Chart sees a Y-value in the dataset equal to the value in the **hole** field, it considers that data value to be missing, that is, a hole. By default, **hole** is **XRT_HUGE_VAL**. You can set the hole value to the value in your dataset that represents missing data.

To initialize a particular value for **hole** when using **XrtMakeDataFromFile**(), insert "**HOLE <value>**" as the second line in the data file.

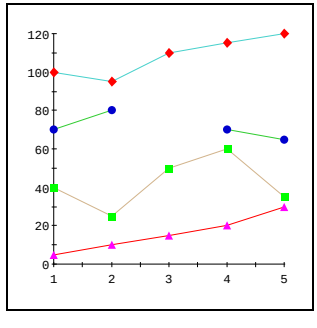


Figure 48 Example of a Data Hole in a Plot

Array Data

The **XrtArray** structure contains a **type** field (always **XRT_DATA_ARRAY**), a **hole** value field, integers defining the number of sets and number of points, and an **XrtArrayData** structure:

```
typedef struct {
    XrtDataType  type;/*=XRT_DATA_ARRAY*/
    double       hole;
    int          nsets;
    int          npoints;
    XrtArrayData data;
} XrtArray;
```

The **XrtArrayData** structure contains a pointer to an array of X-values, and a pointer to an array of pointers to Y-values:

```
typedef struct {
    double      *xp;
    double      **yp;
} XrtArrayData;
```

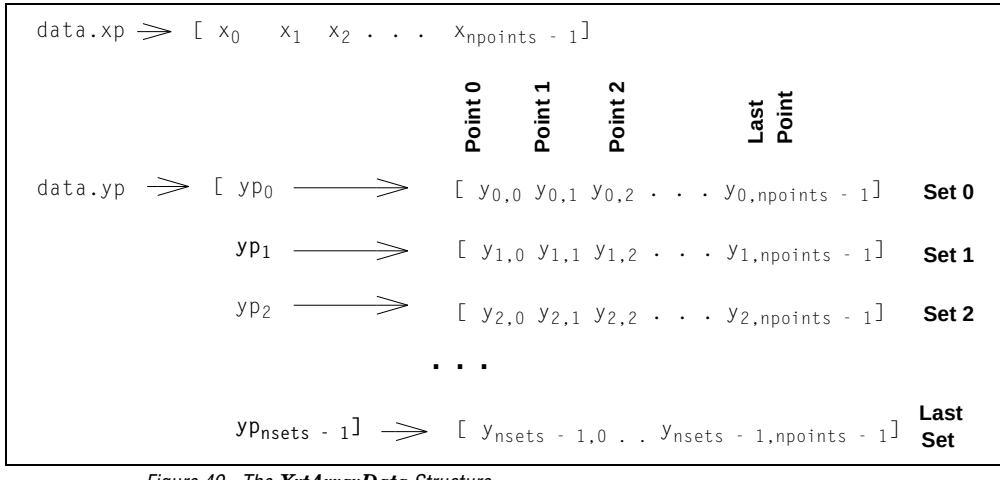


*Figure 49 The **XrtArrayData** Structure*

General Data

The **XrtGeneral** structure contains a **type** field (always **XRT_DATA_GENERAL**), a **hole** value field, an integer defining the number of sets, and a pointer to an array of **XrtGeneralData** structures:

```
typedef struct {
    XrtDataType     type; /*=XRT_DATA_GENERAL*/
    double          hole;
    int             nsets;
    XrtGeneralData  *data;
} XrtGeneral;
```

The **XrtGeneralData** structure contains an integer defining how many points are in the set, followed by a pointer to an array of X-values and a pointer to an array of Y-values:

```
typedef struct {
    int     npoints,
    double  *xp;
    double  *yp;
} XrtGeneralData;
```

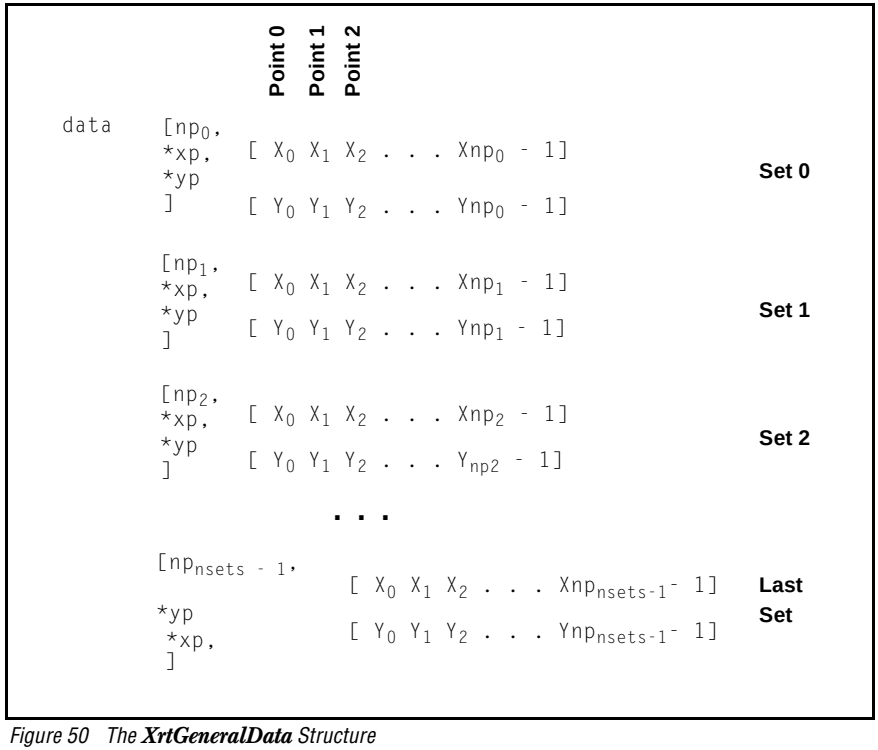


*Figure 50 The **XrtGeneralData** Structure*

Convenience Macros

There are a number of convenience macros in OLCH2DCM.H (and listed in Appendix C on page 143) that facilitate accessing pieces of the **XrtData** structure.

Macros for accessing array structures are prefixed with **arr_**. General structure macros are prefixed **gen_**. For example, **arr_xel(j)** accesses the *jth* X-value in an **XrtArrayData** structure located below an **XrtData** structure.

The following code fragment uses some of the convenience macros to read the values in both an **XRT_DATA_ARRAY** and **XRT_DATA_GENERAL XrtData** structure and place them in a text buffer:

```
static int
pr_arr(XrtData *d, char *buffer)
{
/* Reads elements of XRT_DATA_ARRAY type XrtData Structure into a buffer */
    int i, j;
    int numChars = 0;
    numChars = sprintf(buffer, "Printing array data with %d sets,
                %d points\n", d->arr_nsets, d->arr_npoints);
    numChars += sprintf(buffer + numChars, "X-Values:\n");
    for (j = 0; j < d->arr_npoints; j++)
        numChars += sprintf(buffer + numChars, "%f\t", d->arr_xel(j));
    for (i = 0; i < d->arr_nsets; i++) {
        numChars += sprintf(buffer + numChars, "Y %d Values:\n", i);
        for (j = 0; j < d->arr_npoints; j++)
            numChars += sprintf(buffer + numChars, "%f\t",
                    d->arr_yel(i,j));
        numChars += sprintf(buffer + numChars, "\n");
    }
    return numChars;
}

static int
gen_arr(XrtData *d, char *buffer)
{
/* Read elements of XRT_DATA_GENERAL type XrtData structure into a buffer */
    int i, j;
    int numChars = 0;
    numChars = sprintf(buffer, "Printing general data with %d sets.\n",
            d->gen_nsets);
    for (i = 0; i < d->gen_nsets; i++) {
        numChars += sprintf(buffer + numChars, "Sets %d Values:\n", i);
        for (j = 0; j < d->gen_npoints(i); j++)
            numChars += sprintf(buffer + numChars, "(%f, %f)\t",
                d->gen_xel(i,j), d->gen_yel(i,j));
        numChars += sprintf(buffer + numChars, "\n");
    }
    return numChars;
}

static int
pr_data(XrtData *d, char *buffer)
{
    if (XrtGetDataType(d) == XRT_DATA_ARRAY) /* is array */
        return pr_arr(d, buffer);
    else                /* is general */
        return pr_gen(d, buffer);
}
```

Figure 51 XrtData structure Output Routines

Convenience Procedures

There are a number of **XrtData** structure manipulation convenience procedures included with Olectra Chart. These procedures are fully described in Appendix B on page 117.

XrtDataSort()	Sorts the points in each set by increasing X-value.
XrtDataCopy()	Creates a copy of an **XrtData** structure.
XrtGenDataAppendPt()	Appends an (x,y) value to the end of a set of data in a General **XrtData** structure.
XrtArrDataAppendPts()	Appends a column of points to an Array **XrtData** structure.
XrtGenDataRemovePt()	Removes a point from a General **XrtData** structure.
XrtArrDataRemovePts()	Removes a point from each set in an Array data structure.
XrtDataRemoveSet()	Removes a set from an **XrtData** structure.
XrtDataExtractSet()	Creates a new **XrtData** structure loaded with one set of data from an existing **XrtData** structure.
XrtDataConcat()	Creates a new **XrtData** structure by concatenating two existing **XrtData** structures.

4.3 Changing the Data

Olectra Chart makes it easy to change data while keeping chart characteristics unchanged. Programs can create a large number of **XrtData** structures and then switch among them by setting the **XRT_DATA** property to point to the current data.

The code in Figure 52 sets the **XRT_DATA** property to point to the appropriate data and updates the header text.

```
static char *dec_header[] = { "DEC", NULL, };
case WM_COMMAND:
   switch (wParam) {
   case ID_DEC:
       CheckRadioButton(hWnd, ID_DEC, ID_IBM, ID_DEC);
       XrtSetValues(hChart,
           XRT_DATA, dec0990,
           XRT_HEADER_STRINGS, dec_header,
           NULL);
       break;
```

Figure 52 Changing Datasets in a message handler

The strategy for changing data in real-time is just as simple. First, create a new **XrtData** structure and populate it with the new data (or update the structure that is currently being displayed). Second, tell the chart to use the new data by setting the **XRT_DATA** property.

4.4 Example Using Static Data

If the data to be displayed is known at compile time, it can simply be defined in static variables and then copied into an **XrtData** structure for displaying.

The example below defines the values to be displayed in static C variables **xdata[]** and **ydata[]**. The code simply allocates an **XrtData** structure of the correct size, and copies the static values into the structure.

```
#define NSETS 3
#define NPOINTS 5
static double xdata[NPOINTS] = { 1,2,3,4,5,}; /* Array data placeholders */
static double ydata[NSETS][NPOINTS] = {
    {5, 8, 14, 19, 24 },/* mexico */
    {11, 15, 18, 21, 22 },/* tokyo */
    {14, 16, 15.5, 15.2, 15.4} /* new york */
};
static char *sl[] = { "Mexico", "Tokyo", "New York", NULL, };
static char *pl[] = { "60", "70", "80", "90", "2000", NULL, };
...
static HXRT2D hChart;
static XrtData *pop;
int i, j;
...
switch (msg) {
case WM_CREATE:
    // Create graph control
    hChart = XrtCreateWindow(“”, 5,5,500,300,hWnd, hInstance);
    // Get data
    pop = XrtMakeData(XRT_DATA_ARRAY, NSETS, NPOINTS, TRUE);
    for (i=0; i < NPOINTS; i++) {
        pop->arr_xel(i) = xdata[i];
        for (j=0; j < NSETS; j++) {
            pop->arr_yel(j, i) = ydata[j][i];
        }
    }
    XrtSetValues(hChart,
                XRT_DATA,   pop,
                XRT_SET_LABELS,   sl,
                XRT_POINT_LABELS,pl,
                XRT_XANNOTATION_METHOD,XRT_ANNO_POINT_LABELS,
                NULL);
    break;
...
```

Figure 53 Loading Data Structure from Static C Variables

5

Programming User Interaction

This chapter describes the user-interaction features of Olectra Chart–how a user can interact with the chart and how an application can control interaction.

5.1 Default User Interaction

By default, the user can scale, translate, and zoom into all types of charts. 3-D[1] bar, stacking bar, and pie charts can also be rotated by the user. An application can also define action maps which manipulate a chart programmatically. Figure 54 shows the user interactions enabled by Olectra Chart's default action maps. Note that if you have a three-button mouse, holding down the middle mouse button is equivalent to simultaneously holding down the left and right mouse buttons.

5.2 Overview of Action Maps and Messages

Olectra Chart's default action maps define user events that cause some interactive action within the control. You can customize user interaction through the following mechanisms:

- Action maps–An application can change or remove the default actions.

1. 3D charts must have a depth (**XRT_GRAPH_DEPTH**) greater than zero to allow interactive rotation.

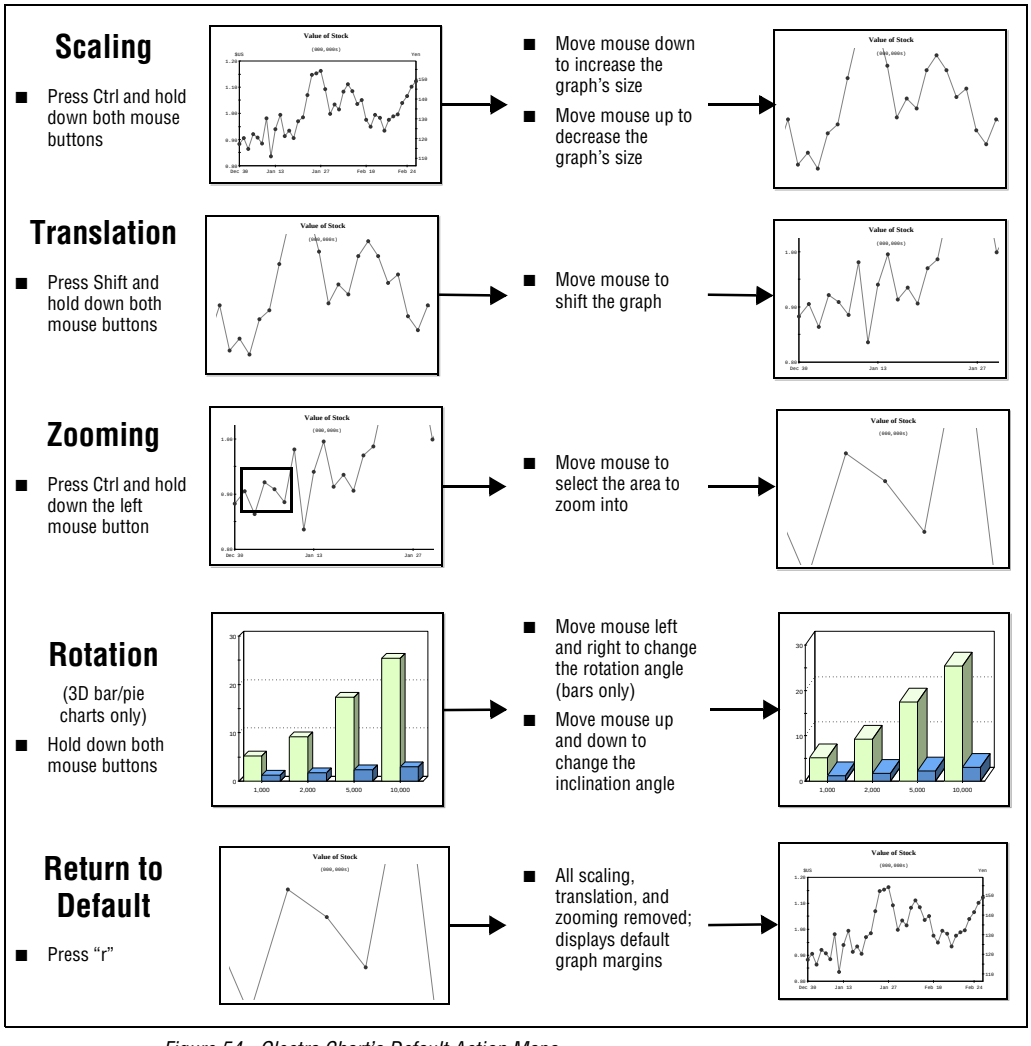


Figure 54 Olectra Chart's Default Action Maps

- Messages–An application can be notified as a user interacts with the control by defining message handler procedures that are called before, during, and after user interaction. A message handler procedure can affect each interaction in such ways as disallowing or constraining it. See the section on each interaction for details on using its callback.

Three Interaction Stages

Olectra Chart's default user interaction passes through three stages:

- Starting user interaction

- Updating user interaction, and
- Ending user interaction

An interaction must pass through these stages in sequence, and an application can be notified by messages during each stage. Each stage is described in the following sections.

5.3 Starting User Interaction

The **XRTN_MODIFY_START** message is passed to the window's message handler to notify the application that a user interaction is about to begin. Figure 55 illustrates this.

Figure 55 XRT_ACTION_MODIFY_START Action

Disabling All User Interaction

You can use the **XRTN_MODIFY_START** message to disable any user interaction regardless of the action maps installed. The window's message handler is passed the following message:

```
XRTN_MODIFY_START:
hWnd = (HWND) wParam;
mcb = (XrtModifyCallbackStruct *) lParam;

typedef struct {
    BOOL doit;
} XrtModifyCallbackStruct;
```

Set the *doit* parameter of this structure to **FALSE** (when the action is **XRT_ACTION_MODIFY_START**) to disable all user interactions. When *doit* is **FALSE**, all "update" actions are disabled until the next XRTN_MODIFY_START message is passed.

5.4 Updating User Interaction

One of several action routines is called to notify the application that a user interaction is about to be updated. None of these routines updates the chart unless the interaction has successfully passed through the **ModifyStart**() action.

One of several messages is passed to notify the application that a user interaction is about to be updated. No action can update the chart unless the interaction has successfully passed through the **XRT_ACTION_MODIFY_START** action.

5.4.1 Scaling, Translation, and Zooming

The **XRT_ACTION_SCALE** action updates interactive scaling of the chart. The **XRT_ACTION_TRANSLATE** action updates interactive translation of the chart. The **XRT_ACTION_ZOOM_END** action zooms into the chart at the area defined by the "zoom rubberband" (defined by the **XRT_ACTION_ZOOM_UPDATE** action). These routines all alter the Margin properties[1]. Figure 56 illustrates these action routines .

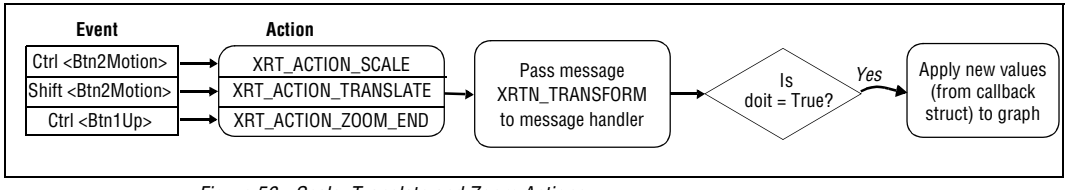


Figure 56 Scale, Translate and Zoom Actions

Controlling Interaction

You can use the **XRTN_TRANSFORM** message to control scaling, translation, or zooming. The following message is passed to the window's message handler:

```
XRTN_TRANSFORM:
hWnd = (HWND) wParam;
tcb = (XrtTransformCallbackStruct *) lParam;

   typedef struct {
       BOOL        reset;/* Read-only */
       int         left_margin;
       int         right_margin;
       int         top_margin;
       int         bottom_margin;
       BOOL        doit;
   } XrtTransformCallbackStruct;
```

A Transform action can change the *left_margin*, *right_margin*, *top_margin*, *bottom_margin*, and *doit* parameters, which are then applied to the chart. For example, to constrain scaling, examine and change the margin parameters.

Resetting Interactions

The **XRT_ACTION_RESET** action restores all Margin properties to their default values, setting their corresponding **USE_DEFAULT** properties to **TRUE**.

The **XRTN_TRANSFORM** message is sent to the window's message handler. An application can deny a reset by setting *doit* to **FALSE**.

1. The Margin properties are specified by **XRT_GRAPH_MARGIN_BOTTOM**, **XRT_GRAPH_MARGIN_TOP**, **XRT_GRAPH_MARGIN_LEFT**, and **XRT_GRAPH_MARGIN_RIGHT**.

5.4.2 Rotation

The **XRT_ACTION_ROTATE** action updates interactive rotation of the chart. Only 3-D bar, stacking bar and pie charts with **XRT_GRAPH_DEPTH** greater than zero can be interactively rotated. This routine alters the **XRT_GRAPH_ROTATION** and **XRT_GRAPH_INCLINATION** properties. Figure 57 illustrates this action routine.

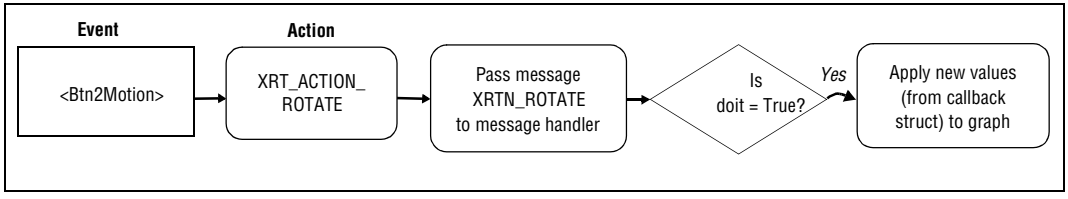


Figure 57 XRT_ACTION_ROTATE Action

Controlling Rotation

You can use the **XRT_ACTION_ROTATE** action to control rotation. The following message is sent to the window's message handler:

```
XRTN_ROTATE:
hWnd = (HWND) wParam;
tcb = (XrtRotateCallbackStruct *) lParam;

   typedef struct {
       int         rotation;
       int         inclination;
       BOOL        doit;
   } XrtRotateCallbackStruct;
```

A Rotate action can change the *rotation*, *inclination*, and *doit* parameters.

5.5 Ending User Interaction

The **XRT_ACTION_MODIFY_END** action notifies the application that a user interaction has finished. The message **XRTN_MODIFY_END** is sent:

```
XRTN_MODIFY_END:
hWnd = (HWND) wParam;
```

Note that no structure is passed.

5.6 Programming Actions

All Olectra Chart actions are customizable: you can determine which Microsoft Windows message should call a particular action, and decide on the appropriate steps to perform in each case.

Only mouse messages and the **WM_KEYDOWN** and **WM_KEYUP** messages are recognized.

5.6.1 Changing the Action Maps

An action map is a mapping of a particular Windows message to a predefined action. Each action map consists of three parts: the Windows message, any modifier flags, and the keycode (only if **WM_KEYDOWN** or **WM_KEYUP**).

The following messages are recognized:

WM_LBUTTONDBLCLK	double-click left mouse button
WM_MBUTTONDBLCLK	double-click both mouse buttons
WM_RBUTTONDBLCLK	double-click right mouse button
WM_LBUTTONDOWN	press left mouse button
WM_MBUTTONDOWN	press both mouse buttons
WM_RBUTTONDOWN	press right mouse button
WM_LBUTTONUP	release left mouse button
WM_MBUTTONUP	release both mouse buttons
WM_RBUTTONUP	release right mouse button
WM_MOUSEMOVE	move mouse
WM_KEYDOWN	press key
WM_KEYUP	release key

Note that if you have a three-button mouse, holding down the middle mouse button is equivalent to simultaneously holding down the left and right mouse buttons.

Modifier Flags

The following modifier flags are recognized:

MK_LBUTTON	left mouse button
MK_MBUTTON	both mouse buttons
MK_RBUTTON	right mouse button
MK_ALT	Alt key
MK_SHIFT	Shift key
MK_CONTROL	Ctrl key

All actions are normalized to match the event sent by Microsoft Windows. For example, **MK_LBUTTON** is added to the modifier flags if a **WM_LBUTTONDOWN** message is sent.

Recognized Keycodes

Any valid **VK_** value is treated as a recognized keycode. Note the following, however:

- All alphabetic characters are forced to upper case.
- **MK_SHIFT** must appear in the modifier if capitals are desired.

- The CapsLock key toggles the meaning of the **MK_SHIFT** modifier.

Determining Action Mappings

To determine which action is mapped to a particular Microsoft Windows message, use the **XrtGetAction**() function. For example, the following code determines which action is mapped to the left mouse button down message:

```
XrtAction action;

action = XrtGetAction(hXrt2D, WM_LBUTTONDOWN, 0, 0);
```

Any unmapped action returns **XRT_ACTION_NONE**.

To return the entire list of action maps, call **XrtGetActionList**(). The pointer returned points to read-only memory.

Programming Action Mappings

To program an action mapping, call **XrtSetAction**(). For example, the following code removes all previously defined actions:

```
XrtActionItem *item, *next;

item = XrtGetActionList(hChart);
for (; item; item = next) {
    next = item->next;
    XrtSetAction(hChart, item->msg, item->modifier,
        item->keycode, XRT_ACTION_NONE);
}
```

Setting an action mapping to **XRT_ACTION_NONE** removes the action.

The following example uses the left mouse button plus the Alt key for rotation instead of both mouse buttons:

```
/* remove all use of both mouse buttons */
XrtSetAction(hChart, WM_MBUTTONDOWN, 0, 0, XRT_ACTION_NONE);
XrtSetAction(hChart, WM_MOUSEMOVE, MK_MBUTTON, 0,
    XRT_ACTION_NONE);
XrtSetAction(hChart, WM_MBUTTONUP, 0, 0, XRT_ACTION_NONE);

/* reprogram for Alt+Left */
XrtSetAction(hChart, WM_LBUTTONDOWN, MK_ALT, 0,
    XRT_ACTION_MODIFY_START);
XrtSetAction(hChart, WM_MOUSEMOVE, MK_LBUTTON|MK_ALT, 0,
    XRT_ACTION_ROTATE);
XrtSetAction(hChart, WM_LBUTTONUP, MK_ALT, 0,
    XRT_ACTION_MODIFY_END);
```

5.6.2 Disabling and Disallowing Interactions

The easiest way to disallow interactions with charts is to catch the **XRTN_MODIFY_START** message, and set the *doit* element in the passed structure to **FALSE**. Another approach is to remove all action mappings, as shown in the previous section.

To remove individual interactions, use **XrtSetAction**() to set the desired interaction to **XRT_ACTION_NONE**.

5.6.3 Calling Actions Directly

To call a chart action directly, use **XrtCallAction**(). This function expects four arguments:

- The chart handle
- The action to be called
- The X- and Y-coordinates of the window location at which the action is to be called

When an action is invoked, the window coordinates specified by **XrtCallAction**() must be contained within the graph area.

5.7 Interacting with Chart Data

Microsoft Windows notifies applications about user interaction with controls by passing messages to the application in its message loop. An application built using Olectra Chart can add a message handling procedure to react to events that happen over a chart control. (For details on message handling, consult your Microsoft Windows programming documentation.)

Some messages include the pixel coordinates of the event within the **MSG** structure. Olectra Chart provides procedures for mapping the pixel coordinates of a message to:

- set and point indices of the data displayed closest to the event coordinates.
- chart coordinates.

XrtPick()

XrtPick() provides an easy mechanism for the programmer to discover "what the user is pointing at" when he or she clicks the mouse on a chart.

The **XrtPick**() procedure takes a chart handle, dataset specification, (x,y) pixel coordinates, a pointer to an **XrtPickResult** structure and a focus hint as arguments. It fills in the **XrtPickResult** structure with information about the data values closest to the specified pixel coordinates on the specified chart.

```
XrtRegion
XrtPick(
    HXRT2D          graph,
    XrtDsGroup      ds_group,
    int             pix_x,
    int             pix_y,
    XrtPickResult   *pick,
    XrtFocus        focus
    )
```

ds_group indicates which dataset the pick results will apply to. In combination charts it can be set to **XRT_DATASET1** or **XRT_DATASET2** to restrict the pick results to one of the datasets. If set to **XRT_DATASET1 | XRT_DATASET2**, the closest point

from either of the datasets will be returned. When the chart is not a combination chart, set this to **XRT_DATASET1**.

focus can be **XRT_XFOCUS** or **XRT_YFOCUS** to constrain the search for the closest point in the direction of the X- or Y-axis. To search for the closest point in cartesian space, set *focus* to **XRT_XFOCUS | XRT_YFOCUS**.

XrtPick() returns one of the values described below:

XRT_RGN_NOWHERE The given pixel coordinates are not close enough to anything to be picked. In this case the **XrtPickResult** structure is not filled in.

XRT_RGN_IN_GRAPH The given pixel coordinates are close to one of the displayed data values in the chart. In this case, the **set** and **point** values in the **XrtPickResult** structure identify the closest data. **distance** indicates how close the picked data is to the pixel coordinates.

XRT_RGN_IN_LEGEND The given pixel coordinates are close to one of the elements in the legend area. In this case, if sets are displayed in the legend, **set** will indicate which set is closest (**point** will be **XRT_RGN_IN_LEGEND**). Otherwise, points are displayed in the legend, so **point** is the closest point and **set** is **XRT_RGN_IN_LEGEND**. **distance** indicates how close the picked data is to the pixel coordinates.

XRT_RGN_IN_HEADER The given pixel coordinates are not in the chart, but are in the header area. The **XrtPickResult** structure is not filled in.

XRT_RGN_IN_FOOTER The given pixel coordinates are not in the chart, but are in the footer area. The **XrtPickResult** structure is not filled in.

The **XrtPickResult** structure is defined as:

```
typedef struct {
    int pix_x, pix_y;
    int dataset;
    int set, point;
    int distance;
} XrtPickResult;
```

The fields are broken down as follows:

pix_x The x pixel coordinate passed to **XrtPick**().

pix_y The y pixel coordinate passed to **XrtPick**().

dataset	Either **XRT_DATASET1** or **XRT_DATASET2** to indicate which dataset the results apply to.
set	The set index of the data element displayed closest to the pixel coordinates. This can be **XRT_RGN_NOWHERE** if no data is displayed
point	The point index of the data element is displayed closest to the pixel coordinates. This can be **XRT_RGN_NOWHERE** if no data is displayed.
distance	The screen distance (in pixels) between the given pixel coordinates and the on-screen display of the data indexed by **set** and **point**, as determined by focus.

Things are slightly more complex when the picked element corresponds to the *other* slice of a pie. If the pie's data is transposed, then **set** indicates the selected set, and **point** is **XRT_OTHER_SLICE**. Otherwise **set** is **XRT_OTHER_SLICE** and **point** indicates the selected point.

The code in Figure 58 determines the chart data closest to the point clicked and puts information about it in the chart's header.

XrtUnpick()

The **XrtUnpick**() procedure is the opposite of **XrtPick**(). It determines the pixel coordinate of a data point and set. It is provided for advanced applications that want to draw on top of a chart. See Appendix B on page 117 for more details.

```
case WM_LBUTTONDOWN:
    POINT pnt;
    XrtPickResult p;
    int rc;
    char *nhs[2], buffer[100];

    pnt = MAKEPOINT(lParam);
    rc = XrtPick(hChart, XRT_DATASET1, pnt.x, pnt.y, &p,
                 XRT_XFOCUS | XRT_YFOCUS);
    if (rc == XRT_RGN_IN_GRAPH) {
        XrtGetValues(hChart, XRT_DATA, &tmp_data, NULL);
        sprintf(buffer, "Set %d, Point %d -- Y-value: %.2f",
                p.set, p.point,
                tmp_data->arr_yel(p.set, p.point));
        nhs[0] = buffer;
        nhs[1] = NULL;
        XrtSetValues(hChart, XRT_HEADER_STRINGS, nhs, NULL);
    }
    break;
```

Figure 58 Using XrtPick() when user clicks on graph area

XrtMap()

The **XrtMap**() procedure maps from pixel coordinates to chart coordinates (i.e. it maps from pixel space to chart space). It takes a chart handle, an axis specification, (x,y) pixel coordinates and a pointer to a map structure and writes the pixel coordinates, axis specification and the floating-point chart coordinates into the map structure.

Calling **XrtMap**() with **yaxis** set to 1 maps to the chart coordinates defined by the first Y-axis. Setting **yaxis** to 2 maps to the chart coordinates defined by the second Y-axis.

```
XrtRegion
XrtMap(
    HXRT2D          graph,
    int             yaxis,
    int             pix_x,
    int             pix_y,
    XrtMapResult    *map
    )
```

XrtMap() returns **XRT_RGN_NOWHERE**, **XRT_RGN_IN_FOOTER**, **XRT_RGN_IN_HEADER** or **XRT_RGN_IN_GRAPH**.

The **XrtMapResult** structure is defined as follows:

```
typedef struct {
    int pix_x, pix_y;
    int yaxis;
    double x, y;
} XrtMapResult;
```

The fields are broken down as follows:

pix_x The x coordinate passed into **XrtMap**().

pix_y The y coordinate passed into **XrtMap**().

yaxis The axis specification passed into **XrtMap**().

x The mapped chart coordinate X-value. A mapped X-value is not defined for discrete X-axis charts. In this case, **x** is **XRT_HUGE_VAL**.

y The mapped chart coordinate Y-value. If the chart is a pie chart, **y** is **XRT_HUGE_VAL**.

The code in Figure 59 determines the chart coordinates of a left mouse down and a left mouse up event. It then zooms the plot in on the rectangular area specified. It assumes the existence of **fmin**() and **fmax**(), which return the floating-point minimum or maximum of their arguments.

```
static BOOL bButtonDown = FALSE;
POINT   pnt;
XrtMapResult m;
static double x1, y1;
int     rc;
...
case WM_LBUTTONDOWN:
    pnt = MAKEPOINT(lParam);
    rc = XrtMap(hChart, XRT_DATASET1, pnt.x, pnt.y, &m);
    if (rc == XRT_RGN_IN_GRAPH) {
        x1 = m.x;
        y1 = m.y;
        SetCapture(hWnd);
        bButtonDown = TRUE;
    }
    break;
case WM_LBUTTONUP:
    if (!bButtonDown) break;
    ReleaseCapture();
    bButtonDown = FALSE;
    pnt = MAKEPOINT(lParam);
    rc = XrtMap(hChart, 1, pnt.x, pnt.y, &m);
    if (rc == XRT_RGN_IN_GRAPH) {
        XrtSetValues(hChart,
                XRT_XAXIS_MIN, fmin(x1, m.x),
                XRT_XAXIS_MAX, fmax(x1, m.x),
                XRT_YAXIS_MIN, fmin(y1, m.y),
                XRT_YAXIS_MAX, fmax(y1, m.y),
                NULL);
    }
    break;
```

Figure 59 Using XrtMap() to zoom in on a chart

XrtUnmap()

The **XrtUnmap**() procedure is the opposite of **XrtMap**(). It maps from chart coordinates to pixel coordinates. It is provided for advanced applications that want to draw on top of a chart. See Appendix B on page 117 for more details.

5.8 Window Resizing

Most applications should allow the user to resize a window containing a chart control, and have the chart adjust to the new window size.

To resize the chart when the user resizes the window containing it, change the chart control's size in a **WM_SIZE** message handler, for example:

```
case WM_SIZE:
{
    int width = LOWORD(lParam);
    int height = HIWORD(lParam);

    SetWindowPos(hwndXrt2D, NULL, 0, 0, width, height,
            SWP_NOMOVE | SWP_NOZORDER);
    break;
}
```

Repaint & Resize Messages

An application can find out when the chart control is repainted or resized by checking for the **XRTN_REPAINTED** or **XRTN_RESIZED** notification messages. These messages contain information in the **lParam** parameter. See Appendix D on page 145 for complete details. The resize message is sent after the chart is resized. The repaint message is sent after the chart is redrawn.

An application can use these messages to draw onto the chart image using Windows API functions. Another use is to adjust the control properties, depending on the size of the chart control.

The following message handler uses **XRTN_RESIZED** to remove the legend from the chart when it gets too small:

```
case XRTN_RESIZED:
    cb = (XrtCallbackStruct *) lParam;
    if (cb->width <= 300) {
        XrtSetValues(hChart,
            XRT_LEGEND_SHOW, FALSE,
            NULL);
    } else {
        XrtSetValues(hChart,
            XRT_LEGEND_SHOW, TRUE,
            NULL);
    }
    break;
```

6

Advanced Programming Topics

Adding a Second Y-axis ■ *Combination Charts*
Adding Text Areas ■ *Batching Property Updates*
Fast Update Procedures

6.1 Adding a Second Y-axis

The left-most Y-axis is called the *first Y-axis* or *primary Y-axis* in Olectra Chart. It is possible to add a second Y-axis (Y2) to the right-hand side of the chart. If the chart is inverted, the first Y-axis is at the bottom, and the second Y-axis is at the top. If the X-axis is reversed the position of the Y-axes is switched.

There are two ways to create a second Y-axis on a chart. The simplest way is to specify a relationship between the first and second Y-axis. A second way is to create a combination chart by attaching a second dataset. Combination charts are discussed in section 6.2 on page 84.

Specifying a Relationship

A relationship between the Y-axes is specified using the **XRT_YAXIS_MULT** and **XRT_YAXIS_CONST** properties. Olectra Chart will create the second Y-axis from the first Y-axis using the equation:

```
Y2 = (Y1 * XRT_YAXIS_MULT) + XRT_YAXIS_CONST
```

For example, if a chart is plotting temperatures that have been measured in degrees Celsius, the following code would create a second Y-axis for Fahrenheit:

```
XrtSetValues(hChart,
    XRT_YAXIS_CONST, 32.0,
    XRT_YAXIS_MULT,  9.0/5.0,
    XRT_Y2PRECISION,0,
    XRT_Y2TITLE, "Fahrenheit",
    NULL);
```

To eliminate the second Y-axis, disable the relationship by setting **XRT_YAXIS_MULT** to 0.0.

XRT_Y2AXIS_LOGARITHMIC is ignored if a relationship is specified between the Y-axes.

Precision Issues

When **XRT_Y2PRECISION_USE_DEFAULT** is **TRUE**, Olectra Chart handles all Y2-axis precision issues. However, if you are explicitly setting Y2-axis precision, it is important to allow enough digits. Setting the precision too small may cause unexpected behavior, such as the disappearance of Y2 annotation. In general, the Y2 precision should be at least as many digits as the maximum of:

- the number of digits of Y1 precision plus the implied precision of **XRT_YAXIS_MULT**.
- the implied precision of **XRT_YAXIS_CONST**.

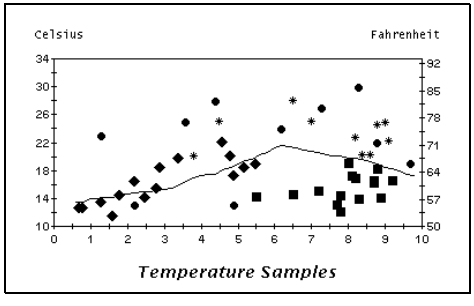


Figure 60 Second Y-axis Showing a Linear Relationship

6.2 Combination Charts

Combination charts have two datasets displayed in the same graph area. Plot, area, bar and stacking bar charts can all be used in combination charts.

A combination chart has a second dataset attached to it using the **XRT_DATA2** property. The second chart type is set using **XRT_TYPE2** (defaults to **XRT_TYPE_BAR**). By default, the first chart will display in front (i.e. closer to the viewer). This can be switched by setting **XRT_FRONT_DATASET** to **XRT_DATASET2**. Markers can also be used with combination charts. The **XRT_MARKER_DATASET** property determines which dataset is used for marker positioning.

Special Rules

There are several special rules that apply just to combination charts:

- If either of the combined charts has a discrete X-axis, the combination chart is also discrete. (See section 3.2 on page 26 for a definition of *discrete* X-axis.)
- Since both charts share the same X-axis, discrete combination chart datasets must have the same number of points. (If **XRT_TRANSPOSE_DATA** is **TRUE** they must have the same number of sets.)

- When the combination chart is discrete, the first chart's Point-labels are used to annotate the X-axis. (If **XRT_TRANSPOSE_DATA** is **TRUE** the first chart's Set-labels are used.)
- If either chart is a pie chart, the second chart will not be displayed.

Second Y-axis

The minimum, maximum, numbering increment, ticking increment and precision of the second chart is programmable using the Y2 versions of the axis properties. The second Y-axis can be logarithmic if no axis relationship has been specified.

Second Data Styles

Data styles for the second chart are programmed using the **XRT_DATA_STYLES2** and **XRT_DATA_STYLES2_USE_DEFAULT** properties.

Legend

The legend in a combination chart with **XRT_TRANSPOSE_DATA** set to **FALSE** is a concatenation of each chart's Set-labels. If the data is transposed, then each chart's Point-labels are concatenated for the legend. Point-labels and Set-labels for the second chart are programmed using **XRT_POINT_LABELS2** and **XRT_SET_LABELS2**. **NULL** labels or zero-length strings are not included in the legend.

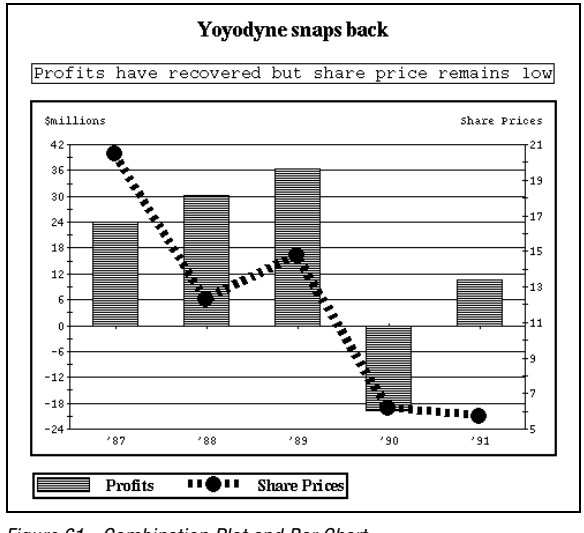


Figure 61 Combination Plot and Bar Chart

6.3 Adding Text Areas

A text area is an independent rectangular region which can be attached to the chart in one of four ways:

- at a particular pixel.
- at a data (x, y) value in the graph area (either dataset).
- to a (set, point) in the graph area (either dataset).
- to a (set, point, Y-value) in the graph area (either dataset).

Text areas support the following elements:

- any valid Windows font.
- background and foreground colors.
- strings (the content).
- text adjustment (Left, Center, Right).
- border type and border width.
- anchor position (relative to the attach position).
- offset along anchor direction from attachment position.
- option of drawing a connecting line to attachment position.

Any number of text areas may be attached to the chart. These areas can be dynamically created and destroyed, attached and detached, queried and updated. Text areas are drawn on top of other chart elements; Olectra Chart does not rearrange other chart elements to make room for the text areas.

XrtTextDesc

This structure is the basis of all attached text, and completely describes a text area's state:

```
typedef struct { /* Attached text structure */
    XrtTextPosition position;
    char **         strings;
    XrtAnchor       anchor;
    int             offset;
    int             connected;
    XrtAdjust       adjust;
    COLORREF        fore_color;
    COLORREF        back_color;
    XrtBorder       border;
    int             border_width;
    HFONT           font;
    XrtRectangle    coords; /* Read-only */
} XrtTextDesc;
```

The fields are broken down as follows:

position Specifies the position of attachment. This is described later in this section.

strings A pointer to a **NULL**-terminated list of strings displayed in the text area.

anchor	Indicates how to position the text area relative to the text attachment position. In addition to compass positions (**XRT_ANCHOR_NORTH**, **XRT_ANCHOR_SOUTHEAST**, etc.), two additional anchors are allowed**.** If **XRT_ANCHOR_HOME**, the text area will be centered over the attachment position. If **XRT_ANCHOR_BEST**, Olectra Chart will determine the best location to attach the text.
offset	Specifies a distance in the anchor direction to offset the final text area position.
connected	If **TRUE**, a line is drawn from the text area to the attachment position.
adjust	Specifies how to adjust the strings within the text area. Must be **XRT_ADJUST_LEFT**, **XRT_ADJUST_RIGHT** or **XRT_ADJUST_CENTER**.
fore_color	Specifies the name of the color to use for the foreground of the text area. If **XRT_DEFAULT_COLOR**, it uses the control foreground color.
back_color	Specifies the name of the color to use for the background of the text area. If **XRT_DEFAULT_COLOR**, the text area is transparent.
border	Specifies the type of border to use around the text area.
border_width	Specifies the thickness of the border, in pixels. Must be between 0 and 20.
font	Specifies the font to use in this text area. If **NULL**, the axes font is used.
coords	Specifies the pixel coordinates of the text area's bounding box, filled in by the **XrtTextDetail**() procedure. This is a read-only member—you cannot use it to change the size of the text area.

Text Area Performance

It is highly recommended that you batch around the creation or manipulation of multiple text-areas. See section 6.4 on page 90 for information on batching.

Managing Text Areas

To create and attach a text area, fill in an **XrtTextDesc** structure and pass it to **XrtTextCreate**():

```
XrtTextHandle
XrtTextCreate(
    HXRT2D          hChart,
    XrtTextDesc     *textd
    )
```

XrtTextCreate() returns a handle to the text area. This can be used in subsequent calls to text area management procedures. To detach or reattach the text area using **XrtTextDetach**() and **XrtTextAttach**():

```
void
XrtTextDetach(
    HXRT2D          hChart,
    XrtTextHandle   handle
    )

void
XrtTextAttach(
    HXRT2D          hChart,
    XrtTextHandle   handle
    )
```

You can avoid managing the text handles currently attached to the chart with **XrtGetTextHandles**(). This procedure returns the number of text areas defined, and allocates and fills in a **NULL**-terminated list of text handles (free this list after use by calling **XrtFreeTextHandles**()). To get all current text area handles, call **XrtGetTextHandles**():

```
int
XrtGetTextHandles(
    HXRT2D          hChart,
    XrtTextHandle   **list  /* Returned */
    )
```

To modify the text area in any way, including moving it, modify the fields in the structure and call **XrtTextUpdate**():

```
void
XrtTextUpdate(
    HXRT2D          hChart,
    XrtTextHandle   handle,
    XrtTextDesc     *textd
    )
```

The **XrtTextDetail**() procedure is used to get back a filled-in **XrtTextDesc** given its handle. It returns 1 if **handle** is a valid text area and 0 otherwise. In addition to all of the programmer-specified fields, **XrtTextDetail**() also fills in the **coords** section of the **XrtTextDesc** structure. The **XrtRectangle** structure contains **x**, **y**, **width** and **height** fields.

```
int
XrtTextDetail(
    HXRT2D          hChart,
    XrtTextHandle   handle,
    XrtTextDesc     *textd
    )
```

Finally, to permanently remove a text area, use **XrtTextDestroy**():

```
void
XrtTextDestroy(
    HXRT2D          hChart,
    XrtTextHandle   handle
    )
```

Positioning Text Areas

The **XrtTextPosition** field in the **XrtTextDesc** structure specifies the manner and location used to attach the text area. It is a C union of four structures, each of which contains the fields necessary to describe the attachment. Note how the first field of each structure specifies the type of attachment:

```
typedef union {
    struct {
        XrtAttachType type;
        int     x, y;
    } pixel;
    struct {
        XrtAttachType type;
        int     dataset;
        double  x, y;
    } value;
    struct {
        XrtAttachType type;
        int     dataset;
        int     set, point;
    } data;
    struct {
        XrtAttachType type;
        int     dataset;
        int     set, point;
        double  y;
    } data_value;
} XrtTextPosition;
```

The attachment type is one of the following:

```
typedef enum {
    XRT_TEXT_ATTACH_PIXEL,
    XRT_TEXT_ATTACH_VALUE,
    XRT_TEXT_ATTACH_DATA,
    XRT_TEXT_ATTACH_DATA_VALUE,
} XrtAttachType;
```

Positioning using Pixel

XRT_TEXT_ATTACH_PIXEL is supported on all chart types at all times. In this case, the text area will be attached at the specified pixel location. The origin is at the top left corner of the chart window for the purposes of specifying pixel coordinates.

Positioning using Value

XRT_TEXT_ATTACH_VALUE is only supported when the X-axis is continuous. Any part of the text area that falls outside of the graph area is clipped.

Positioning using Data

XRT_TEXT_ATTACH_DATA is supported on all chart types. **dataset** should always be set to 1, unless text is being positioned on the second dataset in a combination chart. **set** and **point** are indices into the data set. The exact location of the text depends on the chart type:

Plot, Area	Position is the pixel at which the specified data is plotted.

Bar	If the bar's value is above the origin, the position is at the middle top of the front face of the bar. If the value is below the origin, the position will be at the middle bottom of the front face.
Stacking Bar	Position is in the center of the front face of the specified portion of the stacking bar.
Pie	Position is at the center of the arc of the specified slice. Use **XRT_OTHER_SLICE** to index the pie's *other* slice.

Any part of the text area that falls outside of the graph area is clipped.

Positioning using Data-Value

XRT_TEXT_ATTACH_DATA_VALUE is supported on all chart types except pies. In this case, the **set** and **point** indices are used to determine the x component of the positioning, and **value** is used for the Y component. This is most useful for discrete X-axis charts. Any part of the text area that falls outside of the graph area is clipped.

6.4 Batching Property Updates

Normally, property changes take effect immediately after the property is set. If you prefer to make several changes to the control's properties before causing a repaint, set **XRT_REPAINT** to **FALSE**. All property changes will be batched until **XRT_REPAINT** is set to **TRUE**.

It is recommended that you batch around the creation or updating of multiple text areas.

6.5 Fast Update Procedures

Normally, when adding new data to a chart, you simply update the chart's **XrtData** structure(s) and reset the **XRT_DATA** and/or **XRT_DATA2** properties. Whenever this happens, Olectra Chart examines the entire data and repaints the entire graph area.

The Fast Update procedures provide an alternative and potentially much faster way to add new data to existing charts. The fast update procedures will only be faster if the new data falls within the chart's current axes bounds. If any of the new data falls outside these bounds, the entire chart will be repainted.[1]

1. Fast Update procedures trigger **XRTN_REPAINTED** events only when new data falls outside the axes bounds.

The Fast Update procedures are fully documented in Appendix B on page 117. The following is a brief description of the procedures:

XrtArrDataFastUpdate() This procedure charts new points that have been added to each set in an array dataset.

XrtGenDataFastUpdate() This procedure charts new points that have been added to one set in a general dataset.

XrtArrCheckAxisBounds() This procedure examines new points that have been added to each set in an array dataset, and determines which (if any) axes bounds would be exceeded if these new points were displayed.

XrtGenCheckAxisBounds() This procedure examines new points that have been added to one set in a general dataset, and determines which (if any) axes bounds would be exceeded if these new points were displayed.

XrtArrDataShiftPts() This procedure copies existing points (in all sets) to the left. Use when axis bounds have been exceeded. Shifted-out points are set to the hole value.

XrtGenDataShiftPts() This procedure copies existing points (in one set) to the left. Use when axis bounds have been exceeded. Shifted-out points are set to the hole value.

To use the Fast Update procedures you should:

- Add the new points to the chart's existing **XrtData** structure referenced by the **XRT_DATA** or **XRT_DATA2** property.
- Optionally call the appropriate CheckAxisBounds procedure if you want to know in advance if the update will be fast or regular speed.
- Call the appropriate Fast Update procedure to perform the update.

Any non-solid line patterns (like dashed or dotted) are not guaranteed to be continued correctly using Fast Update. You should use solid lines if this is important to your application.

The following code sample shows how a scrolling strip chart is implemented using the Fast Update procedures:

```
01  if (XrtArrDataAppendPts(data, new_xvalue, new_yvector)) {
02  bound = XrtArrCheckAxisBounds(hChart, 1, 1);   /* new points fit in axes? */
03  if ((bound & XRT_GTX) == 0) {         /* less than xmax? */
04      XrtArrDataFastUpdate(hChart, 1, 1);     /* Fast Update */
05  } else {
06      new_numpoints = data->arr_npoints - increment;
07      /* shift x values */
08      XrtArrDataShiftPts(data, 0, increment, new_numpoints);
09      data->npoints = new_numpoints;
10      xmin = data->arr_xel(0);
11      xmax = data->arr_xel(new_numpoints) + increment * x_increment;
12      XrtSetValues(hChart,      /* repaint with new axis */
13              XRT_DATA, data,
14              XRT_XMIN, xmin,
15              XRT_XMAX, xmax,
16              NULL);
17  }
18  }
```

Figure 62 Scrolling Strip Chart Code

On line 1, new data points are added to the chart's **XrtData** structure (which must be the current value of the chart's **XRT_DATA** property).

Line 2 checks to see if the new points fit within the current axes bounds. If they do, the Fast update procedure is called (line 4). Otherwise, the entire dataset is shifted downward (lines 6-11), and then graphed (line 12-16).

Part II

Reference Appendices

A

Property Reference

Control Synopsis
Properties

This appendix lists all of the Olectra Chart properties in alphabetical order. Listed after the property name are its data type and default value.

A.1 Control Synopsis

Include File: \INCLUDE\OLCH2D.H

Class Name: "OlectraChart2D"

A.2 Properties

XRT_AXIS_BOUNDING_BOX **BOOL** **FALSE**

When **TRUE**, Olectra Chart will draw a box inside the graph area bounding the chart's data area, that is, the axes, all displayed data, and the grid-lines.

XRT_AXIS_FONT **HFONT** **Arial, 12 pt**

Specifies the font to be used for the axes numbering, annotation and titles. If the Arial TrueType font is not available on the system, the System font is used. If you are using Microsoft Windows 3.1, cast the value to an **int** when setting this property.

XRT_BACKGROUND_COLOR **COLORREF** **RGB(255,255,255)**

Specifies the window background color. The value is a Windows color reference. See section 3.10 on page 55 for details of color specification. This color will be inherited as the default background color for the chart, data area, header, footer and legend.

XRT_BAR_CLUSTER_OVERLAP **int** **100**

Specifies the overlap between bars in a cluster, as a percentage of the bar width. A setting of 0 will cause the bars in the cluster to completely overlap each other. A setting of 100 will cause the bars to be fitted next to each other. The value must be between 0 and 200. The actual bar width is determined at run-time by Olectra Chart and depends on a number of factors, including the current window size. Since stacking bars have only one bar in each cluster, this property is meaningless when **XRT_TYPE** is **XRT_TYPE_STACKING_BAR**.

XRT_BAR_CLUSTER_WIDTH **int** **50**

Specifies the amount of space to be used for the bars in a cluster, as a percentage of the total amount of space assigned to the cluster. The value must be between 0 and 100. Setting the value to 100 will cause the bars to abut the bars in the next cluster.

XRT_BORDER **XrtBorder (enum)** **XRT_BORDER_NONE**

Specifies the style of border used to enclose the chart. Valid styles are **XRT_BORDER_NONE**, **XRT_BORDER_3D_IN**, **XRT_BORDER_3D_OUT**, **XRT_BORDER_ETCHED_IN**, **XRT_BORDER_ETCHED_OUT**, **XRT_BORDER_SHADOW**, and **XRT_BORDER_PLAIN**.

XRT_BORDER_WIDTH **int** **0**

Specifies the width of the 3D border around the chart control, in pixels. Must be between 0 and 20. When set to 0 (the default), this border is not visible.

XRT_DATA **XrtData *** **none**
XRT_DATA2 **XrtData *** **none**

Specifies the data to be displayed. The **XrtData** structure should be allocated using the procedure **XrtMakeData**() or **XrtMakeDataFromFile**(). **XRT_DATA2** is used to specify the data for the second chart in a combination chart. The Data properties, unlike all other Olectra Chart properties, are *not* copied by the control. It is the application's responsibility to allocate and deallocate these data structures.

XRT_DATA_AREA_BACKGROUND_COLOR **COLORREF** **XRT_DEFAULT_COLOR**

Specifies the background color of the chart's data area. When **XRT_DEFAULT_COLOR**, the data area background is transparent. The value is a Windows color reference. See section 3.10 on page 55 for details of color specification. This property is ignored in pie charts.

XRT_DATA_STYLES **XrtDataStyle **** **dynamic**
XRT_DATA_STYLES2 **XrtDataStyle **** **dynamic**

The **XrtDataStyle** structure defines a line pattern, fill pattern, color, line width, point style, point size and point color for each set of data. Olectra Chart will only create default **XrtData** styles as required at drawing time throughout the life of the chart object.

Use **XRT_DATA_STYLES** to determine what data styles are currently being used, or to define the data styles to be used. The list of styles is terminated with a **NULL** pointer.

When used with **XrtGetValues**(), a pointer to an array of **XrtDataStyle** structures is returned. The length of the array is the larger of: the most sets of data graphed so far in the life of the chart object, and the highest index of any data style specified so far by the program.

XRT_DATA_STYLES2 is used to access the second chart's data styles in a combination chart.

XRT_DATA_STYLES_USE_DEFAULT **BOOL** **TRUE**
XRT_DATA_STYLES2_USE_DEFAULT **BOOL** **TRUE**

When **TRUE**, Olectra Chart's default data styles values are used for the chart. When explicit data styles are provided, Olectra Chart sets this property to **FALSE**.

XRT_DATA_STYLES2_USE_DEFAULT applies to the second chart's data styles in a combination chart.

XRT_DEBUG **BOOL** **FALSE**

Specifies whether to send warning messages to a debug window. When set to **FALSE**, only property conversion errors are output.

XRT_DOUBLE_BUFFER **BOOL** **TRUE**

When **TRUE**, chart updates are first rendered into an off-screen bitmap, then copied to the display area. This reduces flashing as similar charts are displayed on the screen. Setting to **FALSE** will cause charts to be rendered directly to the display and will result in less memory used.

XRT_FOOTER_ADJUST **XrtAdjust (enum)** **XRT_ADJUST_CENTER**

Specifies how multiple footer text lines should be adjusted within the footer area. Must be one of **XRT_ADJUST_CENTER**, **XRT_ADJUST_LEFT** or **XRT_ADJUST_RIGHT**.

XRT_FOOTER_BACKGROUND_COLOR **COLORREF** **XRT_DEFAULT_COLOR**

Specifies the footer area background color. When **XRT_DEFAULT_COLOR**, the footer background is transparent. The value is a Windows color reference. See section 3.10 on page 55 for details of color specification.

XRT_FOOTER_BORDER **XrtBorder (enum)** **XRT_BORDER_NONE**

Specifies the style of border used to identify the footer area. Valid styles are **XRT_BORDER_NONE**, **XRT_BORDER_3D_IN**, **XRT_BORDER_3D_OUT**, **XRT_BORDER_ETCHED_IN**, **XRT_BORDER_ETCHED_OUT**, **XRT_BORDER_SHADOW**, and **XRT_BORDER_PLAIN**.

XRT_FOOTER_BORDER_WIDTH **int** **2**

Specifies the width of the footer area border in pixels. Must be between 0 and 20.

XRT_FOOTER_FONT **HFONT** **Arial, 12 pt**

Specifies the font to use for the footer strings. If the Arial TrueType font is not available on the system, the System font is used. If you are using Microsoft Windows 3.1, cast the value to an **int** when setting this property.

XRT_FOOTER_FOREGROUND_COLOR **COLORREF** **XRT_DEFAULT_COLOR**

Specifies the footer area foreground color. When **XRT_DEFAULT_COLOR**, the window foreground color is used. The value is a Windows color reference. See section 3.10 on page 55 for details of color specification.

XRT_FOOTER_STRINGS **char **** **none**

Specifies the text to be displayed in the footer area.

XRT_FOOTER_HEIGHT **int** **dynamic**
XRT_FOOTER_WIDTH **int** **dynamic**

Contains the width and height of the footer area in pixels. *NOTE:* This property cannot be set.

XRT_FOOTER_X **int** **centered under chart**
XRT_FOOTER_Y **int** **centered under chart**

Specifies the (X,Y) coordinates of the top left corner of the footer area. A side-effect of setting these properties is that the corresponding **USE_DEFAULT** property is set to **FALSE**.

XRT_FOOTER_X_USE_DEFAULT **BOOL** **TRUE**
XRT_FOOTER_Y_USE_DEFAULT **BOOL** **TRUE**

When **TRUE**, **XRT_FOOTER_X** and **XRT_FOOTER_Y** values are determined by Olectra Chart at render-time. When explicit **XRT_FOOTER_X** and **XRT_FOOTER_Y** values are provided, Olectra Chart sets these Booleans to **FALSE**. You cannot set these properties to **FALSE** unless you have previously provided **XRT_FOOTER_X** or **XRT_FOOTER_Y** values.

XRT_FOREGROUND_COLOR **COLORREF** **RGB(0,0,0)**

Specifies the window foreground color. The value is a Windows color reference. See section 3.10 on page 55 for details of color specification. This color will be inherited as the default foreground color for the header, footer, and legend text, as well as the chart axes.

XRT_FRONT_DATASET **int** **XRT_DATASET1**

Specifies which dataset is displayed in front (i.e. closer to the viewer) in a combination chart. Valid values are **XRT_DATASET1** and **XRT_DATASET2**.

XRT_GRAPH_BACKGROUND_COLOR **COLORREF** **XRT_DEFAULT_COLOR**

Specifies the graph area background color. When **XRT_DEFAULT_COLOR**, the chart is transparent. The value is a Windows color reference. See section 3.10 on page 55 for details of color specification.

XRT_GRAPH_BORDER **XrtBorder (enum)** **XRT_BORDER_NONE**

Specifies the style of border used to identify the graph area. Valid styles are **XRT_BORDER_NONE**, **XRT_BORDER_3D_IN**, **XRT_BORDER_3D_OUT**, **XRT_BORDER_ETCHED_IN**, **XRT_BORDER_ETCHED_OUT**, **XRT_BORDER_SHADOW**, and **XRT_BORDER_PLAIN**.

XRT_GRAPH_BORDER_WIDTH **int** **2**

Specifies the width of the graph area border in pixels. Must be between 0 and 20.

XRT_GRAPH_DEPTH **int** **0**

Specifies the apparent chart depth as a percentage of chart width when a 3D effect is desired. Chart depth, and either inclination or rotation must be non-zero for a 3D effect to appear. Must be between 0 and 500. This has no effect on plots, area charts and combination charts.

XRT_GRAPH_FOREGROUND_COLOR COLORREF XRT_DEFAULT_COLOR

Specifies the graph area foreground color. When **XRT_DEFAULT_COLOR**, the window foreground color is used. The value is a Windows color reference. See section 3.10 on page 55 for details of color specification.

XRT_GRAPH_INCLINATION int 0

Specifies the apparent degree of inclination of the eye-point above the X-axis when a 3D effect is desired. Chart depth, and either inclination or rotation must be non-zero for a 3D effect to appear. Must be between 0 and 45. This has no effect on plots, area charts and combination charts.

XRT_GRAPH_MARGIN_BOTTOM int dynamic

Specifies the pixel offset (with respect to the bottom of the graph area) that the data area should be drawn. If **XRT_GRAPH_MARGIN_BOTTOM_USE_DEFAULT** is **TRUE**, Olectra Chart calculates the value of this property based on the space required for the annotation and titles. A side-effect of setting this property is that the corresponding **USE_DEFAULT** property is set to **FALSE**. This property must be between **XRT_MIN_MARGIN** and **XRT_MAX_MARGIN**.

XRT_GRAPH_MARGIN_BOTTOM_USE_DEFAULT BOOL TRUE

When **TRUE**, **XRT_GRAPH_MARGIN_BOTTOM** is determined by Olectra Chart at render-time. When you set **XRT_GRAPH_MARGIN_BOTTOM**, this property is set to **FALSE**. You cannot set this property to **FALSE** unless a value for **XRT_GRAPH_MARGIN_BOTTOM** has been provided or calculated.

XRT_GRAPH_MARGIN_LEFT int dynamic

Specifies the pixel offset (with respect to the left side of the graph area) that the data area should be drawn. If **XRT_GRAPH_MARGIN_LEFT_USE_DEFAULT** is **TRUE**, Olectra Chart calculates the value of this property based on the space required for the annotation and titles. A side-effect of setting this property is that the corresponding **USE_DEFAULT** is set to **FALSE**.

XRT_GRAPH_MARGIN_LEFT_USE_DEFAULT BOOL TRUE

When **TRUE**, **XRT_GRAPH_MARGIN_LEFT** is determined by Olectra Chart at render-time. When you set **XRT_GRAPH_MARGIN_LEFT**, this property is set to **FALSE**. You cannot set this property to **FALSE** unless a value for **XRT_GRAPH_MARGIN_LEFT** has been provided or calculated.

XRT_GRAPH_MARGIN_RIGHT int dynamic

Specifies the pixel offset (with respect to the right side of the graph area) that the data area should be drawn. If **XRT_GRAPH_MARGIN_RIGHT_USE_DEFAULT** is **TRUE**, Olectra Chart calculates the value of this property based on the space required for the annotation and titles. A side-effect of setting this property is that the corresponding **USE_DEFAULT** is set to **FALSE**.

XRT_GRAPH_MARGIN_RIGHT_USE_DEFAULT BOOL TRUE

When **TRUE**, **XRT_GRAPH_MARGIN_RIGHT** is determined by Olectra Chart at render-time. When you set **XRT_GRAPH_MARGIN_RIGHT**, this property is set to **FALSE**. You

cannot set this property to **FALSE** unless a value for **XRT_GRAPH_MARGIN_RIGHT** has been provided or calculated.

XRT_GRAPH_MARGIN_TOP **int** **dynamic**

Specifies the pixel offset (with respect to the top of the graph area) that the data area should be drawn. If **XRT_GRAPH_MARGIN_TOP_USE_DEFAULT** is **TRUE**, Olectra Chart calculates the value of this property based on the space required for the annotation and titles. A side-effect of setting this property is that the corresponding **USE_DEFAULT** is set to **FALSE**.

XRT_GRAPH_MARGIN_TOP_USE_DEFAULT **BOOL** **TRUE**

When **TRUE**, **XRT_GRAPH_MARGIN_TOP** is determined by Olectra Chart at render-time. When you set **XRT_GRAPH_MARGIN_TOP**, this property is set to **FALSE**. You cannot set this property to **FALSE** unless a value for **XRT_GRAPH_MARGIN_TOP** has been provided or calculated.

XRT_GRAPH_ROTATION **int** **0**

Specifies the apparent degree of rotation of the eye-point to the right of the Y-axis when a 3D effect is desired. Chart depth, and either inclination or rotation must be non-zero for a 3D effect to appear. Must be between 0 and 45. Chart rotation has no effect on pie charts, plots, area charts and combination charts.

XRT_GRAPH_HEIGHT **int** **dynamic**
XRT_GRAPH_WIDTH **int** **dynamic**

Specifies the width and height of the graph area in pixels. A side-effect of setting these properties is that the corresponding **USE_DEFAULT** property is set to **FALSE**.

The graph area is the area surrounding the axis and annotation, but does not include the header, footer and legend. To adjust the size of the whole chart control use a Windows API function like **SetWindowPos**().

XRT_GRAPH_WIDTH_USE_DEFAULT **BOOL** **TRUE**
XRT_GRAPH_HEIGHT_USE_DEFAULT **BOOL** **TRUE**

When **TRUE**, **XRT_GRAPH_WIDTH** and **XRT_GRAPH_HEIGHT** values are determined by Olectra Chart at render-time. When explicit **XRT_GRAPH_WIDTH** and **XRT_GRAPH_HEIGHT** values are provided, Olectra Chart sets these Booleans to **FALSE**. You cannot set these properties to **FALSE** unless you have previously provided **XRT_GRAPH_WIDTH** or **XRT_GRAPH_HEIGHT** values.

XRT_GRAPH_X **int** **dynamic**
XRT_GRAPH_Y **int** **dynamic**

Specifies the (X,Y) coordinates of the top left corner of the graph area. A side-effect of setting these properties is that the corresponding **USE_DEFAULT** property is set to **FALSE**.

XRT_GRAPH_X_USE_DEFAULT **BOOL** **TRUE**
XRT_GRAPH_Y_USE_DEFAULT **BOOL** **TRUE**

When **TRUE**, **XRT_GRAPH_X** and **XRT_GRAPH_Y** values are determined by Olectra Chart at render-time. When explicit **XRT_GRAPH_X** and **XRT_GRAPH_Y** values are provided,

Olectra Chart sets these Booleans to **FALSE**. You cannot set these properties to **FALSE** unless you have previously provided **XRT_GRAPH_X** or **XRT_GRAPH_Y** values.

XRT_HEADER_ADJUST **XrtAdjust (enum)** **XRT_ADJUST_CENTER**

Specifies how multiple header text lines should be adjusted within the header area. Must be one of **XRT_ADJUST_CENTER**, **XRT_ADJUST_LEFT** or **XRT_ADJUST_RIGHT**.

XRT_HEADER_BACKGROUND_COLOR **COLORREF** **XRT_DEFAULT_COLOR**

Specifies the header area background color. When **XRT_DEFAULT_COLOR**, the header is transparent. The value is a Windows color reference. See section 3.10 on page 55 for details of color specification.

XRT_HEADER_BORDER **XrtBorder (enum)** **XRT_BORDER_NONE**

Specifies the style of border used to identify the header area. Valid styles are **XRT_BORDER_NONE**, **XRT_BORDER_3D_IN**, **XRT_BORDER_3D_OUT**, **XRT_BORDER_ETCHED_IN**, **XRT_BORDER_ETCHED_OUT**, **XRT_BORDER_SHADOW**, and **XRT_BORDER_PLAIN**.

XRT_HEADER_BORDER_WIDTH **int** **2**

Specifies the width of the header area border in pixels. Must be between 0 and 20.

XRT_HEADER_FONT **HFONT** **Arial, 12 pt**

Specifies the font to use for the header strings. If the Arial TrueType font is not available on the system, the System font is used. If you are using Microsoft Windows 3.1, cast the value to an **int** when setting this property.

XRT_HEADER_FOREGROUND_COLOR **COLORREF** **XRT_DEFAULT_COLOR**

Specifies the header area foreground color. When **XRT_DEFAULT_COLOR**, the window foreground color is used. The value is a Windows color reference. See section 3.10 on page 55 for details of color specification.

XRT_HEADER_STRINGS **char **** **none**

Specifies the text to be displayed in the header area.

XRT_HEADER_HEIGHT **int** **dynamic**
XRT_HEADER_WIDTH **int** **dynamic**

Contains the height and width of the header area in pixels. *NOTE:* This property cannot be set.

XRT_HEADER_X **int** **centered above chart**
XRT_HEADER_Y **int** **centered above chart**

Specifies the (X,Y) coordinates of the top left corner of the header area. A side-effect of setting these properties is that the corresponding **USE_DEFAULT** property is set to **FALSE**.

XRT_HEADER_X_USE_DEFAULT **BOOL** **TRUE**
XRT_HEADER_Y_USE_DEFAULT **BOOL** **TRUE**

When **TRUE**, **XRT_HEADER_X** and **XRT_HEADER_Y** values are determined by Olectra Chart at render-time. When explicit **XRT_HEADER_X** and **XRT_HEADER_Y** values are provided, Olectra Chart sets these Booleans to **FALSE**. You cannot set these properties to **FALSE** unless you have previously provided **XRT_HEADER_X** or **XRT_HEADER_Y** values.

XRT_HEIGHT **int** **size of created window**

Specifies the height of the control window.

XRT_INVERT_ORIENTATION **BOOL** **FALSE**

Inverts the normal interpretation of X and Y. When **TRUE**, causes X-values to be graphed against the vertical axis, and Y-values to be graphed against the horizontal axis. Bar and stacking bar charts will render horizontally. This property has no effect on pie charts.

XRT_LEGEND_ANCHOR **XrtAnchor (enum)** **XRT_ANCHOR_EAST**

Specifies where to anchor the legend to the window. Valid values are: **XRT_ANCHOR_NORTH**, **XRT_ANCHOR_SOUTH**, **XRT_ANCHOR_EAST**, **XRT_ANCHOR_WEST**, **XRT_ANCHOR_NORTHWEST**, **XRT_ANCHOR_NORTHEAST**, **XRT_ANCHOR_SOUTHWEST**, and **XRT_ANCHOR_SOUTHEAST**.

XRT_LEGEND_BACKGROUND_COLOR **COLORREF** **XRT_DEFAULT_COLOR**

Specifies the legend area background color. When **XRT_DEFAULT_COLOR**, the legend background is transparent. The value is a Windows color reference. See section 3.10 on page 55 for details of color specification.

XRT_LEGEND_BORDER **XrtBorder (enum)** **XRT_BORDER_NONE**

Specifies the style of border used to identify the legend area. Valid styles are **XRT_BORDER_NONE**, **XRT_BORDER_3D_IN**, **XRT_BORDER_3D_OUT**, **XRT_BORDER_ETCHED_IN**, **XRT_BORDER_ETCHED_OUT**, **XRT_BORDER_SHADOW**, and **XRT_BORDER_PLAIN**.

XRT_LEGEND_BORDER_WIDTH **int** **2**

Specifies the width of the legend area border in pixels. Must be between 0 and 20.

XRT_LEGEND_FONT **HFONT** **Arial, 12 pt**

Specifies the font to be used in the legend. If the Arial TrueType font is not available on the system, the System font is used. If you are using Microsoft Windows 3.1, cast the value to an **int** when setting this property.

XRT_LEGEND_FOREGROUND_COLOR **COLORREF** **XRT_DEFAULT_COLOR**

Specifies the legend area foreground color. When **XRT_DEFAULT_COLOR**, the window foreground color is used. The value is a Windows color reference. See section 3.10 on page 55 for details of color specification.

XRT_LEGEND_ORIENTATION **XrtAlign (enum)** **XRT_ALIGN_VERTICAL**

Specifies the orientation of the legend. Valid values are **XRT_ALIGN_HORIZONTAL** and **XRT_ALIGN_VERTICAL**.

XRT_LEGEND_SHOW **BOOL** **TRUE**

Determines if the legend will be displayed.

XRT_LEGEND_HEIGHT **int** **dynamic**
XRT_LEGEND_WIDTH **int** **dynamic**

Contains the height and width of the legend area in pixels. *NOTE:* This property cannot be set.

XRT_LEGEND_X **int** **depends on anchor**
XRT_LEGEND_Y **int** **depends on anchor**

Specifies the (X,Y) coordinates of the top left corner of the legend area. A side-effect of setting these properties is that the corresponding **USE_DEFAULT** property is set to **FALSE**.

XRT_LEGEND_X_USE_DEFAULT **BOOL** **TRUE**
XRT_LEGEND_Y_USE_DEFAULT **BOOL** **TRUE**

When **TRUE**, **XRT_LEGEND_X** and **XRT_LEGEND_Y** values are determined by Olectra Chart at render-time. When explicit **XRT_LEGEND_X** and **XRT_LEGEND_Y** values are provided, Olectra Chart sets these Booleans to **FALSE**. You cannot set these properties to **FALSE** unless you have previously provided **XRT_LEGEND_X** or **XRT_LEGEND_Y** values.

XRT_MARKER_DATASET **int** **XRT_DATASET1**

Specifies which dataset the markers should be positioned through in combination charts. Change **XRT_MARKER_DATASET** to **XRT_DATASET2** to position through the second dataset using **XRT_XMARKER_POINT**, **XRT_XMARKER_SET** and **XRT_[XY]MARKER**.

XRT_MARKER_DATA_STYLE **XrtDataStyle *** **black, dashed**

Specifies the **XrtDataStyle** to be used when rendering the X- and Y-markers. In particular, line style, color and width elements of the **XrtDataStyle** structure are used. If never specified, Olectra Chart will use a black, 1-pixel wide dashed line for the markers.

XRT_MARKER_DATA_STYLE_USE_DEFAULT **BOOL** **TRUE**

When **TRUE**, the default marker data style will be used. When explicit data styles values are provided, Olectra Chart sets this property to **FALSE**.

XRT_NAME **char *** **dynamic**

Specifies the name of a particular chart instance. By default, Olectra Chart generates a unique name for each chart created.

XRT_OTHER_DATA_STYLE **XrtDataStyle *** **yellow, solid**

Specifies the **XrtDataStyle** to be used when rendering the *other* slice on pie charts. In particular, color and fill pattern elements of the **XrtDataStyle** structure are used.

XRT_OTHER_DATA_STYLE_USE_DEFAULT **BOOL** **TRUE**

When **TRUE**, the *other* data style is determined by Olectra Chart at run-time. It is guaranteed to be unique if **XRT_DATA_STYLES_USE_DEFAULT** is **TRUE**. When an explicit other data style is provided, Olectra Chart sets this Boolean to **FALSE**.

XRT_OTHER_LABEL **char *** **"Other"**

Specifies the label of the *other* slice in a pie chart. By default, the *other* slice is labelled "Other". Attempting to set this property to an invalid value (such as **NULL**, or a zero-length string) results in Olectra Chart using the string "Other".

XRT_PIE_MIN_SLICES **int** **1**

Specifies the minimum number of slices (including the *other* slice) to attempt to display in pie charts.

XRT_PIE_ORDER **XrtPieOrder (enum)** **XRT_PIEORDER_DATA_ORDER**

Specifies the order that pie slices should be displayed in. Must be one of **XRT_PIEORDER_DESCENDING**, **XRT_PIEORDER_ASCENDING** or **XRT_PIEORDER_DATA_ORDER**.

XRT_PIE_THRESHOLD_METHOD **XrtPieThresholdMethod** **XRT_PIE_SLICE_CUTOFF**

Specifies the threshold method used to determine which point values will be grouped into the *other* slice in pie charts. Must be either **XRT_PIE_SLICE_CUTOFF** or **XRT_PIE_PERCENTILE**.

If **XRT_PIE_SLICE_CUTOFF**, all point values with a percentage less than the value of **XRT_PIE_THRESHOLD_VALUE** will be grouped into the *other* slice.

If **XRT_PIE_PERCENTILE**, as many of the smallest points as necessary are grouped together into the *other* slice, providing that the *other* slice is less than or equal to the **XRT_PIE_THRESHOLD_VALUE**. For example, if you want the *other* slice to be no more than 5% of the total, set **XRT_PIE_THRESHOLD_METHOD** to **XRT_PIE_PERCENTILE**, and set **XRT_PIE_THRESHOLD_VALUE** to 5.0.

Note that **XRT_PIE_MIN_SLICES** has precedence over **XRT_PIE_THRESHOLD_METHOD**.

XRT_PIE_THRESHOLD_VALUE **double** **0.0**

Specifies the threshold value associated with **XRT_PIE_THRESHOLD_METHOD**. Must be between 0.0 and 100.0. Set to 0.0 to disable *other* slice creation.

XRT_POINT_LABELS **char **** **NULL**
XRT_POINT_LABELS2 **char **** **NULL**

Specifies the string to render against the associated point value in discrete X-axis charts. When Point-labels are being used, Olectra Chart will:

- ignore the data X-values and X-axis Value-labels;
- space the points evenly across the X-axis; and

- ignore the **XRT_XAXIS_MIN**, **XRT_XAXIS_MAX**, **XRT_XMIN**, **XRT_XMAX**, **XRT_XNUM_METHOD**, **XRT_XNUM**, **XRT_XORIGIN**, **XRT_XPRECISION** and **XRT_XTICK** properties.

Setting this property will have no effect unless **XRT_XANNOTATION_METHOD** is set to **XRT_ANNO_POINT_LABELS**. To clear all Point-labels, set this property to **NULL**.

XRT_POINT_LABELS2 is used to access the second chart's Point-labels in a combination chart. These Point-labels are used in the legend when the chart is transposed.

XRT_REPAINT **BOOL** **TRUE**

When this property is **TRUE**, any changes made to the control are rendered immediately. To batch changes to the control, set this property to **FALSE**, make the changes, and then set this property to **TRUE** again.

XRT_SET_LABELS **char **** **NULL**
XRT_SET_LABELS2 **char **** **NULL**

Specifies the string to render against the associated set value. To clear all Set-labels, set **XRT_SET_LABELS** to **NULL**.

XRT_SET_LABELS2 is used to access the second chart's Set-labels in a combination chart. These Set-labels are used in the legend when the chart is not transposed.

XRT_TIME_BASE **time_t** **dynamic**

Specifies the offset of time data since Jan. 1, 1970 GMT. The default value is Jan. 1, 1970 in your time zone. For example, if your time zone is EST, this property defaults to 18000 (since EST is 5 hours west of Greenwich). This is used whenever **XRT_XANNOTATION_METHOD** is **XRT_ANNO_TIME_LABELS**.

The **XrtMakeTime**() method is helpful when calculating suitable values for this property.

XRT_TIME_FORMAT **char *** **dynamic**

Specifies the format string for Time-axis labels. This property is only used when **XRT_XANNOTATION_METHOD** is **XRT_ANNO_TIME_LABELS**. Olectra Chart uses the string specified by this property with your system's ANSI C standard function **strftime**(). Please check your system documentation for complete details on the formatting options. The following is a summary of ANSI formatting options:

%%	same as the "percent" character (%)
%a	abbreviated weekday name
%A	full weekday name
%b	abbreviated month name
%B	full month name
%c	appropriate date and time representation
%d	day of month (01 - 31)
%H	hour (00 - 23)
%I	hour (01 - 12)
%j	day number of year (001 - 366)

%m	month number (01 - 12)
%M	minute (00 - 59)
%p	equivalent of either AM or PM
%S	seconds (00 - 61), allows for leap seconds
%U	week number of year (00 - 53), Sunday is the first day of week 1
%w	weekday number (0 - 6), Sunday = 0
%W	week number of year (00 - 53), Monday is the first day of week 1
%x	appropriate date representation
%X	appropriate time representation
%y	year within century (00 - 99)
%Y	year as ccyy (for example 1986)
%Z	time zone name or no characters if no time zone exists

When **NULL**, Olectra Chart will generate a default format based on the range of data and the value of **XRT_TIME_UNIT**.

XRT_TIME_FORMAT_USE_DEFAULT **BOOL** **TRUE**

When **TRUE**, Olectra Chart will calculate a default time format dynamically. When an explicit time format is provided, Olectra Chart sets this **BOOL** to **FALSE**.

XRT_TIME_UNIT **XrtTimeUnit (enum)** **XRT_TMUNIT_SECONDS**

Use this property to tell Olectra Chart how to interpret the X-values in data attached to the chart when **XRT_XANNOTATION_METHOD** is **XRT_ANNO_TIME_LABELS**. Valid values are **XRT_TMUNIT_SECONDS**, **XRT_TMUNIT_MINUTES**, **XRT_TMUNIT_HOURS**, **XRT_TMUNIT_DAYS**, **XRT_TMUNIT_WEEKS**, **XRT_TMUNIT_MONTHS** and **XRT_TMUNIT_YEARS**.

For example, if data with X-values ranging from 0 to 4 has been attached to the chart, and if **XRT_TIME_UNIT** is set to **XRT_TMUNIT_WEEKS**, Olectra Chart will annotate the X-axis with the four week period of time, beginning at the time specified by **XRT_TIME_BASE**.

XRT_TRANSPOSE_DATA **BOOL** **FALSE**

For **XRT_DATA_ARRAY** data, this property transposes sets with points. This property is not used by Olectra Chart when rendering **XRT_DATA_GENERAL** data.

XRT_TYPE **XrtType (enum)** **XRT_TYPE_PLOT**
XRT_TYPE2 **XrtType (enum)** **XRT_TYPE_BAR**

Determines the chart type. **XRT_TYPE2** determines the chart type for the second chart in a combination. Valid values are **XRT_TYPE_AREA**, **XRT_TYPE_BAR**, **XRT_TYPE_STACKING_BAR**, **XRT_TYPE_PLOT**, and **XRT_TYPE_PIE**.

XRT_WIDTH **int** **size of created window**

Specifies the height of the control window.

XRT_XANNO_PLACEMENT	**XrtAnnoPlacement (enum)**	**XRT_ANNO_AUTO**
XRT_YANNO_PLACEMENT	**XrtAnnoPlacement (enum)**	**XRT_ANNO_AUTO**

Specifies the placement of the axis annotation and title with respect to the perpendicular axis. When **XRT_ANNO_AUTO**, the annotation and title are placed at the origin (or at the end of the axis closest to the origin for bar charts). **XRT_YANNO_PLACEMENT** is ignored when the X-axis is discrete, or on dual-axis charts.

Other values are: **XRT_ANNO_ORIGIN**, which places the annotation/title along the origin; **XRT_ANNO_MIN**, which places annotation/title at the axis minimum; and **XRT_ANNO_MAX**, which places annotation/title at the axis maximum.

XRT_XANNOTATION_METHOD	**XrtAnnoMethod (enum)**	**XRT_ANNO_VALUES**
XRT_YANNOTATION_METHOD	**XrtAnnoMethod (enum)**	**XRT_ANNO_VALUES**
XRT_Y2ANNOTATION_METHOD	**XrtAnnoMethod (enum)**	**XRT_ANNO_VALUES**

Specifies the method used to annotate the X-, Y- and Y2-axes. For the X-axis, any of the four annotation methods can be specified; these are **XRT_ANNO_VALUES**, **XRT_ANNO_POINT_LABELS**, **XRT_ANNO_VALUE_LABELS**, **XRT_ANNO_TIME_LABELS**. For the Y- and Y2-axes, only are **XRT_ANNO_VALUES** and **XRT_ANNO_VALUE_LABELS** are valid; attempting to set to an invalid value results in **XRT_ANNO_VALUES** being used.

When the annotation method is set to **XRT_ANNO_VALUES**, Olectra Chart annotates the axis at appropriate intervals along the axis using the chart's X- or Y-values.

When the annotation method is set to **XRT_ANNO_VALUE_LABELS**, the value of **XRT_[XYY2]LABELS** is used to annotate the axis.

When **XRT_XANNOTATION_METHOD** is set to **XRT_ANNO_POINT_LABELS**, Point-labels are used to annotate the X-axis. Note that Point-labels can only be used when the X-axis is discrete. Therefore, **XRT_XANNOTATION_METHOD** only applies to plots and area charts when **XRT_TRANSPOSE_DATA** is **FALSE**.

When **XRT_XANNOTATION_METHOD** is set to **XRT_ANNO_TIME_LABELS**, Time-labels are used to annotate the X-axis.

XRT_XANNOTATION_ROTATION	**XrtRotate (enum)**	**XRT_ROTATE_NONE**
XRT_YANNOTATION_ROTATION	**XrtRotate (enum)**	**XRT_ROTATE_NONE**
XRT_Y2ANNOTATION_ROTATION	**XrtRotate (enum)**	**XRT_ROTATE_NONE**

Use this property to rotate axis annotation text. When set to **XRT_ROTATE_90** or **XRT_ROTATE_270**, axis annotations are rotated either 90 or 270 degrees counterclockwise.

Rotation is not allowed if the axis font (specified by **XRT_AXIS_FONT**) is a raster font. Using rotated text can slow down the real-time performance of the chart control dramatically, and is therefore not recommended for charts that may need to be updated many times per second.

XRT_XAXIS_LOGARITHMIC	**BOOL**	**FALSE**
XRT_YAXIS_LOGARITHMIC	**BOOL**	**FALSE**
XRT_Y2AXIS_LOGARITHMIC	**BOOL**	**FALSE**

When **TRUE**, Olectra Chart charts the data against a logarithmic axis. All data values less than or equal to zero will be displayed as holes (missing data values). It is not possible to set **XRT_[XYY2]TICK**, **XRT_[XYY2]NUM** or **XRT_[XYY2]PRECISION** for a logarithmic axis. Since logarithmic axes can only chart values greater than zero, setting

XRT_[XYY2]AXIS_MIN, **XRT_[XYY2]AXIS_MAX**, **XRT_[XYY2]MIN**, **XRT_[XYY2]MAX** or **XRT_[XY]ORIGIN** less than or equal to zero forces its corresponding **USE_DEFAULT** property to **TRUE**.

XRT_XAXIS_LOGARITHMIC is ignored when the X-axis is discrete or the X-axis annotation method is **XRT_ANNO_TIME_LABELS**. **XRT_[XYY2]AXIS_LOGARITHMIC** is ignored on pie charts.

XRT_XAXIS_MAX	**double**	**dynamic**
XRT_YAXIS_MAX	**double**	**dynamic**
XRT_Y2AXIS_MAX	**double**	**dynamic**

Specifies the maximum value of the axis. Chart data is clipped to the smaller of **XRT_[XYY2]AXIS_MAX** and **XRT_[XYY2]MAX**. A side-effect of setting this property is that the corresponding **USE_DEFAULT** property is set **FALSE**.

Unless set explicitly, the value of this property is derived from the values of **XRT_[XYY2]MAX**, **XRT_[XY]ORIGIN**, **XRT_[XYY2]NUM** and **XRT_[XYY2]NUM_METHOD**, and falls on an integral multiple of **XRT_[XYY2]NUM**.

NOTE: When its corresponding **USE_DEFAULT** property is **FALSE**, any change in the axis properties that makes **XRT_[XYY2]AXIS_MAX** invalid causes Olectra Chart to set the **USE_DEFAULT** property to **TRUE**. This forces the axis bounds to be recalculated, which deals with all invalid cases.

XRT_[XYY2]AXIS_MAX is ignored in pie charts. **XRT_XAXIS_MAX** is ignored when the X-axis is discrete. If a relationship exists between the first and second Y-axis, the default for **XRT_Y2AXIS_MAX** is calculated from **XRT_YAXIS_MAX**.

XRT_XAXIS_MAX_USE_DEFAULT	**BOOL**	**TRUE**
XRT_YAXIS_MAX_USE_DEFAULT	**BOOL**	**TRUE**
XRT_Y2AXIS_MAX_USE_DEFAULT	**BOOL**	**TRUE**

When **TRUE**, Olectra Chart will calculate the maximum axis value. When you explicitly set **XRT_[XYY2]AXIS_MAX**, this property is set to **FALSE**. You cannot set this property to **FALSE** unless a value for **XRT_[XYY2]AXIS_MAX** has been provided or calculated.

NOTE: When **FALSE**, any change in the axis properties that makes **XRT_[XYY2]AXIS_MAX** invalid causes Olectra Chart to set the **USE_DEFAULT** property to **TRUE**, forcing the axis bounds to be recalculated.

XRT_XAXIS_MIN	**double**	**dynamic**
XRT_YAXIS_MIN	**double**	**dynamic**
XRT_Y2AXIS_MIN	**double**	**dynamic**

Specifies the minimum value of the axis. Chart data is clipped to the larger of **XRT_[XYY2]AXIS_MIN** and **XRT_[XYY2]MIN**. A side-effect of setting this property is that the corresponding **USE_DEFAULT** property is set to **FALSE**.

Unless set explicitly, the value of this property is derived from the values of **XRT_[XYY2]MIN**, **XRT_[XY]ORIGIN**, **XRT_[XYY2]NUM** and **XRT_[XYY2]NUM_METHOD** properties, and falls on an integral multiple of **XRT_[XYY2]NUM**.

NOTE: When its corresponding **USE_DEFAULT** property is **FALSE**, any change in the axis properties that makes **XRT_[XYY2]AXIS_MIN** invalid causes Olectra Chart to set the

USE_DEFAULT property to **TRUE**. This forces the axis bounds to be recalculated, which deals with all invalid cases.

XRT_[XYY2]AXIS_MIN is ignored in pie charts. **XRT_XAXIS_MIN** is ignored when the X-axis is discrete. If a relationship exists between the first and second Y-axis, the default for **XRT_Y2AXIS_MIN** is calculated from **XRT_YAXIS_MIN**.

XRT_XAXIS_MIN_USE_DEFAULT	**BOOL**	**TRUE**
XRT_YAXIS_MIN_USE_DEFAULT	**BOOL**	**TRUE**
XRT_Y2AXIS_MIN_USE_DEFAULT	**BOOL**	**TRUE**

When **TRUE**, Olectra Chart will calculate the minimum axis value. When you explicitly set **XRT_[XYY2]AXIS_MIN**, this property is set to **FALSE**. You cannot set this property to **FALSE** unless a value for **XRT_[XYY2]AXIS_MIN** has been provided or calculated.

NOTE: When **FALSE**, any change in the axis properties that makes **XRT_[XYY2]AXIS_MIN** invalid causes Olectra Chart to set the **USE_DEFAULT** property to **TRUE**, forcing the axis bounds to be recalculated.

XRT_XAXIS_REVERSED	**BOOL**	**FALSE**
XRT_YAXIS_REVERSED	**BOOL**	**FALSE**
XRT_Y2AXIS_REVERSED	**BOOL**	**FALSE**

When **TRUE**, Olectra Chart will draw and annotate the X-, Y-, or Y2-axis in the reverse direction. This is commonly used to cause the X-axis annotation to increase from right to left or the Y-axis annotation to increase downward. The X-axis may be reversed only if it is continuous.

XRT_Y2AXIS_REVERSED is ignored when **XRT_YAXIS_MULT** is non-zero. If a positive multiplier is specified, the second Y-axis will have the same direction as the first Y-axis. If a negative multiplier is specified, the second Y-axis will have the opposite direction to the first.

XRT_XAXIS_SHOW	**BOOL**	**TRUE**
XRT_YAXIS_SHOW	**BOOL**	**TRUE**
XRT_Y2AXIS_SHOW	**BOOL**	**FALSE**

Determines if the axis should be displayed.

All Y2-axis properties are ignored unless a second dataset has been attached (using **XRT_DATA2**) or **XRT_YAXIS_MULT** is non-zero.

XRT_XGRID	**double**	**0.0**
XRT_YGRID	**double**	**0.0**

Specifies the X- or Y-axis increment between grid-lines, starting from zero. A side-effect of setting these properties is that the corresponding **USE_DEFAULT** property is set to **FALSE**.

On charts with a discrete X-axis, the X grid-lines will be drawn through each label.

No grid-lines are displayed when the value is 0.0.

XRT_XGRID_DATA_STYLE	**XrtDataStyle ***	**dotted line**
XRT_YGRID_DATA_STYLE	**XrtDataStyle ***	**dotted line**

Specifies the **XrtDataStyle** to be used when rendering the X and Y grid-lines. In particular, line style, color and width elements of the **XrtDataStyle** structure are used. If not specified, Olectra Chart will use a 1-pixel wide dotted line in the chart foreground color.

A side-effect of setting these properties is that the corresponding **USE_DEFAULT** property is set to **FALSE**.

XRT_XGRID_DATA_STYLE_USE_DEFAULT	**BOOL**	**TRUE**
XRT_YGRID_DATA_STYLE_USE_DEFAULT	**BOOL**	**TRUE**

When **TRUE**, the default grid data style will be used. When explicit data styles are provided, Olectra Chart sets this Boolean to **FALSE**.

XRT_XGRID_USE_DEFAULT	**BOOL**	**FALSE**
XRT_YGRID_USE_DEFAULT	**BOOL**	**FALSE**

When **TRUE**, **XRT_XGRID** and **XRT_YGRID** values are determined by Olectra Chart at render-time. When explicit **XRT_XGRID** and **XRT_YGRID** values are provided, Olectra Chart sets these Booleans to **FALSE**. You cannot set these properties to **FALSE** unless you have previously provided **XRT_XGRID** or **XRT_YGRID** values.

When **XRT_[XY]GRID_USE_DEFAULT** is **TRUE**, the default **XRT_[XY]GRID** is the value of **XRT_[XY]NUM**.

XRT_XLABELS	**XrtValueLabel ****	**NULL**
XRT_YLABELS	**XrtValueLabel ****	**NULL**
XRT_Y2LABELS	**XrtValueLabel ****	**NULL**

These properties supply the labels for the X-, Y-, and Y2-axis when the value of **XRT_[XYY2]ANNOTATION_METHOD** is **XRT_ANNO_VALUE_LABELS**. The property value is a **NULL**-terminated array of pointers to **XrtValueLabel** structures. Each structure contains a Value-label pair.

XRT_XLABELS is only valid when the X-axis is continuous.

When used in an **XrtGetValues**() call, a **NULL** pointer is returned if there are no Value-labels defined. The returned Value-labels are sorted by their X- or Y-value.

XRT_XMARKER	**double**	**undefined**
XRT_YMARKER	**double**	**undefined**

Specifies position of the markers in chart coordinates. Specifying **XRT_XMARKER** is only meaningful on continuous X-axis charts. A marker will not be displayed by Olectra Chart until this property is explicitly set, and the corresponding **MarkerShow** property is **TRUE**.

When positioning markers on combination charts, set **XRT_MARKER_DATASET** appropriately before using these properties.

XRT_XMARKER_POINT	**int**	**undefined**
XRT_XMARKER_SET	**int**	**undefined**

Specifies the indices of the data set and point through which the X-marker should be drawn. This is the only way of specifying a marker on pie charts, and the only way of specifying an X-

marker on discrete X-axis charts. A marker will not be displayed by Olectra Chart until this property is explicitly set, and the corresponding **MarkerShow** property is **TRUE**.

When positioning markers on combination charts, set **XRT_MARKER_DATASET** appropriately before using these properties.

XRT_XMARKER_SHOW	**BOOL**	**FALSE**
XRT_YMARKER_SHOW	**BOOL**	**FALSE**

Specifies whether the corresponding marker should be displayed or not. In addition to setting these Booleans **TRUE**, the marker position must be explicitly set by the program before markers will be shown. Use **XRT_[XY]MARKER** and/or **XRT_XMARKER_SET** and/or **XRT_XMARKER_POINT** to set the marker position.

XRT_XMAX	**double**	**dynamic**
XRT_YMAX	**double**	**dynamic**
XRT_Y2MAX	**double**	**dynamic**

Specifies the value of the high end of the X, Y and Y2 data. Data values greater than this value will not be displayed. A side-effect of setting these properties is that the corresponding **USE_DEFAULT** property is set to **FALSE**.

XRT_[XYY2]MAX is ignored in pie charts. **XRT_XMAX** is ignored on discrete X-axis charts.

In bar charts, attempts to set **XRT_[XYY2]MAX** to a value less than the Y origin will be ignored.

If a relationship exists between the first and second Y-axis, then the default for **XRT_Y2MAX** is calculated from **XRT_YMAX**.

XRT_XMAX_USE_DEFAULT	**BOOL**	**TRUE**
XRT_YMAX_USE_DEFAULT	**BOOL**	**TRUE**
XRT_Y2MAX_USE_DEFAULT	**BOOL**	**TRUE**

When **TRUE**, **XRT_[XYY2]MAX** values are determined by Olectra Chart at render-time. When explicit **XRT_[XYY2]MAX** values are provided, Olectra Chart sets these Booleans to **FALSE**. You cannot set these properties to **FALSE** unless you have previously provided **XRT_[XYY2]MAX** values.

XRT_XMIN	**double**	**dynamic**
XRT_YMIN	**double**	**dynamic**
XRT_Y2MIN	**double**	**dynamic**

Specifies the value of the low end of the X, Y and Y2 data. Data values less than this value will not be displayed. A side-effect of setting these properties is that the corresponding **USE_DEFAULT** property is set to **FALSE**.

XRT_[XYY2]MIN is ignored in pie charts. **XRT_XMIN** is ignored on discrete X-axis charts.

In bar charts, attempts to set **XRT_[XYY2]MIN** to a value greater than the Y origin will be ignored.

If a relationship exists between the first and second Y-axis, then the default for **XRT_Y2MIN** is calculated from **XRT_YMIN**.

XRT_XMIN_USE_DEFAULT	**BOOL**	**TRUE**
XRT_YMIN_USE_DEFAULT	**BOOL**	**TRUE**
XRT_Y2MIN_USE_DEFAULT	**BOOL**	**TRUE**

When **TRUE**, **XRT_[XYY2]MIN** values are determined by Olectra Chart at render-time. When explicit **XRT_[XYY2]MIN** values are provided, Olectra Chart sets these Booleans to **FALSE**. You cannot set these properties to **FALSE** unless you have previously provided **XRT_[XYY2]MIN** values.

XRT_XNUM	**double**	**dynamic**
XRT_YNUM	**double**	**dynamic**
XRT_Y2NUM	**double**	**dynamic**

Specifies the increment between axis-numbering. A side-effect of setting these properties is that the corresponding **USE_DEFAULT** property is set to **FALSE**.

XRT_[XYY2]NUM is ignored when the axis is logarithmic, and in pie charts. **XRT_XNUM** is ignored when the X-axis is discrete or logarithmic.

The default value **XRT_[XYY2]NUM** is derived from the data unless the corresponding **XRT_[XYY2]TICK** has been explicitly set. In this case, the numbering increment will default to twice the tick increment.

XRT_XNUM_USE_DEFAULT	**BOOL**	**TRUE**
XRT_YNUM_USE_DEFAULT	**BOOL**	**TRUE**
XRT_Y2NUM_USE_DEFAULT	**BOOL**	**TRUE**

When **TRUE**, **XRT_[XYY2]NUM** values are determined by Olectra Chart at render-time. When explicit **XRT_[XYY2]NUM** values are provided, Olectra Chart sets these Booleans to **FALSE**. You cannot set these properties to **FALSE** unless you have previously provided **XRT_[XYY2]NUM** values.

XRT_XNUM_METHOD	**XrtNumMethod (enum)**	**XRT_NUM_ROUND**
XRT_YNUM_METHOD	**XrtNumMethod (enum)**	**XRT_NUM_ROUND**
XRT_Y2NUM_METHOD	**XrtNumMethod (enum)**	**XRT_NUM_ROUND**

Specifies the method used to calculate axis numbering. These properties control the default value of the **XRT_[XYY2]NUM** properties. If the value is **XRT_NUM_PRECISION**, the current value of the axis precision (**XRT_[XYY2]PRECISION**) is used to derive the value of **XRT_[XYY2]NUM**. If the value is **XRT_NUM_ROUND**, then rounded numbers are chosen (most significant digit is 1, 2 or 5). In either case, the current value of the axis precision is used to format the numbers.

XRT_XORIGIN	**double**	**dynamic**
XRT_YORIGIN	**double**	**dynamic**

Specifies where the axis should be rendered. For example, setting X origin to 5.0 will cause the Y-axis to cross the X-axis at X=5.0. A side-effect of setting these properties is that the corresponding **USE_DEFAULT** property is set to **FALSE**.

XRT_[XY]ORIGIN is ignored in pie charts. **XRT_XORIGIN** is ignored on discrete X-axis charts.

The Y-origin can only be set on the first Y-axis.

XRT_XORIGIN_USE_DEFAULT **BOOL** **TRUE**
XRT_YORIGIN_USE_DEFAULT **BOOL** **TRUE**

When **TRUE**, Olectra Chart calculates the values of **XRT_[XY]ORIGIN** at render-time. When explicit **XRT_XORIGIN** and **XRT_YORIGIN** values are provided, Olectra Chart sets these Booleans to **FALSE**. You cannot set these properties to **FALSE** unless you have previously provided **XRT_[XY]ORIGIN** values.

XRT_XORIGIN_PLACEMENT **XrtOriginPlacement (enum)** **XRT_ORIGIN_AUTO**
XRT_YORIGIN_PLACEMENT **XrtOriginPlacement (enum)** **XRT_ORIGIN_AUTO**

Specifies the location of the origin. When **XRT_ORIGIN_AUTO**, Olectra Chart places the origin as follows:

- Plot/area charts–origin placed at the axis minimum or at zero if the dataset has both positive and negative values.
- Bar charts–origin placed at zero.

Stacking bar charts always have their origin at zero. Attempts to set it elsewhere are ignored. **XRT_[XY]ORIGIN_PLACEMENT** is ignored in pie charts.

Other values are: **XRT_ORIGIN_ZERO**, which places the origin at zero; **XRT_ORIGIN_MIN**, which places the origin at the axis minimum; and **XRT_ORIGIN_MAX**, which places the origin at the axis maximum.

This property is ignored if you set an explicit origin with **XRT_[XY]ORIGIN**.

XRT_XPRECISION **int** **dynamic**
XRT_YPRECISION **int** **dynamic**
XRT_Y2PRECISION **int** **dynamic**

This property defines the number of digits of precision after the decimal point to be used for axis-numbering. If 0 or less, axis-numbers will all be integers. Olectra Chart will also use the precision to determine working values for the axis' origin, min, max, num and tick increments. If, for example, **XRT_XORIGIN** is set to 1.45 and **XRT_XPRECISION** is 0, Olectra Chart will draw the Y-axis at X=1. If the **XRT_XPRECISION** is subsequently changed to 2, Olectra Chart will draw the Y-axis at X=1.45.

Negative precision values indicates precision in powers of 10. For example, setting **XRT_YPRECISION** to -2 will cause all Y-axis numbering to be multiples of 100.

A side effect of setting these properties is that the corresponding **USE_DEFAULT** property is set to **FALSE**.

XRT_XPRECISION_USE_DEFAULT **BOOL** **TRUE**
XRT_YPRECISION_USE_DEFAULT **BOOL** **TRUE**
XRT_Y2PRECISION_USE_DEFAULT **BOOL** **TRUE**

When **TRUE**, Olectra Chart calculates appropriate values for **XRT_[XYY2]PRECISION** automatically at render-time. When explicit **XRT_[XYY2]PRECISION** values are provided, Olectra Chart sets these Booleans to **FALSE**. You cannot set these properties to **FALSE** unless you have previously provided **XRT_[XYY2]PRECISION** values.

XRT_XTICK	**double**	**dynamic**
XRT_YTICK	**double**	**dynamic**
XRT_Y2TICK	**double**	**dynamic**

Specifies the increment between axis-ticks. A side-effect of setting these properties is that the corresponding **USE_DEFAULT** property is set to **FALSE**.

XRT_[XYY2]TICK is ignored in pie charts or if the axis is logarithmic. **XRT_XTICK** is ignored when the X-axis is discrete or logarithmic.

The default **XRT_[XYY2]TICK** value is derived from the data unless the corresponding **XRT_[XYY2]NUM** has been explicitly set. In this case, the tick increment will default to half the numbering increment.

When the X annotation method is Value-labels or Time-labels, and if **XRT_XTICK_USE_DEFAULT** is **TRUE**, ticks will display at each label. Otherwise **XRT_XTICK_USE_DEFAULT** is **FALSE**, so the specified X tick increment will be honored. If the increment is zero, no ticks will display.

XRT_XTICK_USE_DEFAULT	**BOOL**	**TRUE**
XRT_YTICK_USE_DEFAULT	**BOOL**	**TRUE**
XRT_Y2TICK_USE_DEFAULT	**BOOL**	**TRUE**

When **TRUE**, **XRT_[XYY2]TICK** values are determined by Olectra Chart at render-time. When explicit **XRT_[XYY2]TICK** values are provided, Olectra Chart sets these Booleans to **FALSE**. You cannot set these properties to **FALSE** unless you have previously provided **XRT_[XYY2]TICK** values.

XRT_XTITLE	**char ***	**NULL**
XRT_YTITLE	**char ***	**NULL**
XRT_Y2TITLE	**char ***	**NULL**

Specifies the title to display on the X-, Y- and Y2-axis. These properties are ignored on pie charts.

XRT_XTITLE_ROTATION	**XrtRotate (enum)**	**XRT_ROTATE_NONE**
XRT_YTITLE_ROTATION	**XrtRotate (enum)**	**XRT_ROTATE_NONE**
XRT_Y2TITLE_ROTATION	**XrtRotate (enum)**	**XRT_ROTATE_NONE**

Use this property to rotate the title text. It can only be applied to vertically-oriented axes (Y or Y2 unless **XRT_INVERT_ORIENTATION** is **TRUE**). When set to **XRT_ROTATE_NONE**, the title is centered above or below the axis. When set to **XRT_ROTATE_90** or **XRT_ROTATE_270**, the title is rotated either 90 or 270 degrees counterclockwise and is centered to the left or right of the vertical axis.

Rotation is not allowed if the axis font (specified by **XRT_AXIS_FONT**) is a raster font. Using rotated text can slow down the real-time performance of the chart control dramatically, and therefore is not recommended for charts that need to be updated many times per second.

XRT_YAXIS_CONST **double** **0.0**

Used in conjunction with **XRT_YAXIS_MULT** to specify a relationship between the first and second Y-axis. To indicate that no relationship exists between the two Y-axes, set **XRT_YAXIS_MULT** to 0.0.

For example, if the first Y-axis is degrees Celsius, and the second Y-axis is degrees Fahrenheit, set **XRT_YAXIS_MULT** to 9.0/5.0, and set **XRT_YAXIS_CONST** to 32.0.

XRT_YAXIS_MULT **double** **0.0**

Used in conjunction with **XRT_YAXIS_CONST** to specify a relationship between the first and second Y-axes. To indicate that no relationship exists between the two Y-axes, set **XRT_YAXIS_MULT** to 0.0.

For example, if the first Y-axis is degrees Celsius, and the second Y-axis is degrees Fahrenheit, set **XRT_YAXIS_MULT** to 9.0/5.0, and set **XRT_YAXIS_CONST** to 32.0.

B

Procedures and Methods Reference

This appendix lists the Olectra Chart procedures and methods in alphabetical order.

XrtArrCheckAxisBounds()

Determines if additional array data will require a chart repaint.

```
int
XrtArrCheckAxisBounds(
    HXRT2D      hChart,
    int         dataset,
    int         numpoints
    )
```

hChart is the chart handle. *dataset* indicates whether data is being added to the chart's first or second dataset and should be either 1 or 2. *numpoints* is the number of points that have been added to each set in the dataset.

Before calling this procedure, a chart control must have already been created and must have an array type **XrtData** structure attached to it using either the **XRT_DATA** or **XRT_DATA2** property.

To use this procedure, first add *numpoints* additional data values to each set in the array. **XrtArrDataAppendPts**() is a useful convenience routine to use for this purpose. Be sure to add the points to the **XrtData** structure currently being pointed to by the control's **XRT_DATA** or **XRT_DATA2** property.

Once you've added the additional data, call **XrtArrCheckAxisBounds**(). If all of the additional data falls within the current axes bounds, this routine will return 0. Otherwise, a bit pattern is returned indicating which axis bounds were exceeded by the new data:

```
#define XRT_LTX 0x01 /* New data < X axis min */
#define XRT_GTX 0x02 /* New data > X axis max */
#define XRT_LTY 0x04 /* New data < Y axis min */
#define XRT_GTY 0x08 /* New data > Y axis max */
```

The returned value can be used in a scrolling strip chart application to determine when to scroll the "data window."

XrtArrDataAppendPts()

Appends a new point to each of the sets in an Array structure.

```
int
XrtArrDataAppendPts(
    XrtData     *data,
    double      xval,
    double      *yvector
    )
```

xval is the new X-value. *yvector* points to an array of Y-values (one for each set). **XrtArrDataAppendPts**() returns 1 on success and 0 on failure.

XrtArrDataFastUpdate()

Displays additional array data in an existing chart as quickly as possible.

```
int
XrtArrDataFastUpdate(
    HXRT2D      hChart,
    int         dataset,
    int         numpoints
    )
```

hChart is the chart handle. *dataset* indicates whether data is being added to the chart's first or second dataset and should be either 1 or 2. *numpoints* is the number of points that have been added to each set in the dataset.

Before calling this procedure, a chart control must have already been created and must have an array type **XrtData** structure attached to it using either the **XRT_DATA** or **XRT_DATA2** property.

To use this procedure, first add *numpoints* additional data values to each set in the array. **XrtArrDataAppendPts**() is a useful convenience routine for this purpose. Be sure to add the points to the **XrtData** structure currently being pointed to by the control's **XRT_DATA** or **XRT_DATA2** property.

Once you've added the additional data, call **XrtArrDataFastUpdate**(). This routine will graph the newly added points. If a fast data update is possible, the new points will be added extremely quickly, otherwise a chart repaint will take place (as if you had simply reset the **XRT_DATA** or **XRT_DATA2** property). This routine returns 0 if a chart repaint was required, and 1 if a fast update was possible.

You can determine ahead of time if a fast update is possible by using the **XrtArrCheckAxisBounds**() procedure.

XrtArrDataRemovePts()

Removes the point indexed by *point* from all the sets in an Array structure.

```
int
XrtArrDataRemovePts(
    XrtData     *data,
    int         point
    )
```

XrtArrDataRemovePts() returns 1 on success and 0 on failure.

XrtArrDataShiftPts()

Copies a block of points in an Array structure. Points in all sets are shifted. Shifted-out points are set to the hole value.

```
int
XrtArrDataShiftPts(
    XrtData     *data,
    int         dest,
    int         src,
    int         npoints
    )
```

src specifies the first point in the block to be shifted; *npoints* specifies the size of the block. *dest* specifies the point to shift the block to. *src* and *dest* are zero-based.

The **npoints** element of the **XrtData** structure is not changed. This function returns 1 on success and 0 on failure.

XrtAttachWindow()

Attaches a window to a chart. Returns **True** if the attach was successful.

```
BOOL
XrtAttachWindow(
    HXRT2D      hChart,
    HWND        hWnd,
    )
```

hChart is the chart handle. *hWnd* is the window handle.

XrtCallAction()

Calls an action explicitly at a given window coordinate. All coordinates should be within the graph area of the control. Any notification messages normally triggered by this action are called.

```
void
XrtCallAction(
    HXRT2D      hChart,
    XrtAction   action,
    int         x,
    int         y
    )
```

hChart is the chart handle. *action* is the action to be called. *x* and *y* specify the window coordinate.

XrtComputePalette()

Asks the chart to recompute its palette based on the current system palette.

```
void
XrtComputePalette(
    HXRT2D      hChart,
    )
```

hChart is the chart handle.

XrtCreate()

Creates a chart without creating a window for it. Returns the chart handle.

```
HXRT2D
XrtCreate(void)
```

XrtCreateWindow()

Creates a window and a chart and attaches the two. Returns the chart handle.

```
HXRT2D
XrtCreateWindow(
    LPCTSTR     lpWindowName,
    int         x, y,
    int         nWidth, nHeight,
    HWND        hWndParent,
    HINSTANCE   hInstance
    )
```

lpWindowName is a pointer to the window name. *x* and *y* are the horizontal and vertical position of the window. *nWidth* and *nHeight* are the width and height of the window. *hWndParent* is a handle to the parent or owner window. *hInstance* is a handle to the application instance.

XrtDataConcat()

Returns a new **XrtData** structure created by concatenating two existing **XrtData** structures pointed to by *data1* and *data2*.

```
XrtData *
XrtDataConcat(
    XrtData     *data1,
    XrtData     *data2
    )
```

XrtDataConcat() returns **NULL** on failure. The type of the result is dependent on the types of *data1* and *data2*:

data1	data2	Result
NULL	anything	data2
anything	**NULL**	data1
General	General	General
General	Array	General
Array	General	General
Array	Array	General or Array

When concatenating an Array with an Array, an Array will be returned if both arrays have the same number of points per set, and both have the same X-values. Otherwise a General will be returned.

If either *data1* or *data2* has zero points or sets, it will be treated as if it were a **NULL** pointer.

XrtDataCopy()

Returns an exact replica of the **XrtData** structure passed in.

```
XrtData *
XrtDataCopy(
    XrtData     *data
    )
```

XrtDataCopy() returns **NULL** on failure.

XrtDataExtractSet()

Returns a new **XrtData** structure containing a copy of the set indexed by *set* of the structure pointed to by *data*.

```
XrtData *
XrtDataExtractSet(
    XrtData     *data,
    int         set
    )
```

The returned structure is the same type (Array or General) as the structure pointed to by *data*. **XrtDataExtractSet**() returns **NULL** on failure.

XrtDataRemoveSet()

Removes the set indexed by *set* from either an Array or General data structure.

```
int
XrtDataRemoveSet(
    XrtData     *data,
    int         set
    )
```

XrtDataSort()

Sorts the points in each set by increasing X-value.

```
int
XrtDataSort(
    XrtData     *data
    )
```

XrtDataSort() returns 0 on failure and 1 on success.

XrtDeleteNthPointLabel()
XrtDeleteNthPointLabel2()

Deletes the *n*th Point-label. **XrtDeleteNthPointLabel2**() operates on the second set of Point-labels used in combination charts.

```
void
XrtDeleteNthPointLabel(
    HXRT2D      hChart,
    int         index,
    )
```

hChart is the chart handle. The (zero-based) *index* specifies the index of the Point-label to be deleted. All Point-labels with index greater than *index* are shifted down.

XrtDeleteNthSetLabel()
XrtDeleteNthSetLabel2()

Deletes the *n*th Set-label. **XrtDeleteNthSetLabel2**() operates on the second set of Set-labels used in combination charts.

```
void
XrtDeleteNthSetLabel(
    HXRT2D      hChart,
    int         index,
    )
```

hChart is the chart handle. The (zero-based) *index* specifies the index of the Set-label to be deleted. All Set-labels with index greater than *index* are shifted down.

XrtDestroyData()

Destroys an **XrtData** structure. Use it only for structures created with *XrtMakeData()* and *XrtMakeDataFromFile()*.

```
void
XrtDestroyData(
    XrtData     *data,
    BOOL        all
    )
```

data is the structure to be destroyed. If *all* is **TRUE**, then any space allocated for data values is also freed.

If *data* is in use by any chart, it will be automatically detached.

XrtDetachWindow()

Detaches a chart from its window. Returns the window handle.

```
HWND
XrtDetachWindow(
    HXRT2D      hChart
    )
```

hChart is the chart handle.

XrtDrawToClipboard()

Copies a chart to the Windows clipboard as a bitmap or metafile.

```
BOOL
XrtDrawToClipboard(
    HXRT2D          hChart,
    XrtDrawFormat   format
    )
```

hChart is the chart handle to output. *format* is the graphics format to use and can be: **XRT_DRAW_BITMAP** for a Windows bitmap (BMP); **XRT_DRAW_METAFILE** for a Windows metafile (WMF); or **XRT_DRAW_ENHMETAFILE** (***Windows 95*** and ***Windows NT*** applications only) for an enhanced metafile (EMF).

XrtDrawToDC()

Outputs a chart to any device context as a bitmap or metafile.

```
BOOL
XrtDrawToDC(
    HXRT2D          hChart,
    HDC             hdc,
    XrtDrawFormat   format,
    XrtDrawScale    scale,
    long            left, top, width, height
    )
```

hChart is the chart handle to output. *hdc* is a device context handle. *format* is the graphics format to use and can be: **XRT_DRAW_BITMAP** for a Windows bitmap (BMP); **XRT_DRAW_METAFILE** for a Windows metafile (WMF); or **XRT_DRAW_ENHMETAFILE** (***Windows 95*** and ***Windows NT*** applications only) for an enhanced metafile (EMF). *scale* specifies the scaling to perform when printing, and is one of **XRT_DRAWSCALE_NONE** (no scaling), **XRT_DRAWSCALE_TOWIDTH** (scale to width specified by *width*, preserving aspect ratio and ignoring *height*), **XRT_DRAWSCALE_TOHEIGHT** (scale to height specified by *height*, preserving aspect ratio and ignoring *width*), **XRT_DRAWSCALE_TOFIT** (scale to minimum of *height* or *width*, preserving aspect ratio), **XRT_DRAWSCALE_TOMAX** (enlarge to size of page regardless of aspect ratio). *left* is the offset from the left of the page. *top* is the offset from the top of the page. *width* specifies the width to scale to; to use the existing window width, set *width* to 0. *height* specifies the height to scale to; to use the existing window height, set *height* to 0.

XrtDrawToFile()

Outputs a chart to a file as a bitmap or metafile.

```
BOOL
XrtDrawToFile(
    HXRT2D          hChart,
    char            *filename,
    XrtDrawFormat   format
    )
```

hChart is the chart handle to output. *format* is the graphics format to use and can be: **XRT_DRAW_BITMAP** for a Windows bitmap (BMP); **XRT_DRAW_METAFILE** for a Windows metafile (WMF); or **XRT_DRAW_ENHMETAFILE** (***Windows 95*** and ***Windows NT*** applications only) for an enhanced metafile (EMF).

XrtDupDataStyles()

Duplicates an array of **XrtDataStyle** structures.

```
XrtDataStyle **
XrtDupDataStyles(
    XrtDataStyle    **ds
    )
```

XrtDupDataStyles() returns a pointer to the duplicate array.

XrtDupStrings()

Duplicates an array of strings.

```
char **
XrtDupStrings(
    char            **s
    )
```

XrtDupStrings() returns a pointer to the duplicate array.

XrtDupValueLabels()

This routine returns a copy of the passed Value-labels.

```
XrtValueLabel **
XrtDupValueLabels(
    XrtValueLabel   **label
    )
```

XrtFreeDataStyles()

Frees the memory used by an array of **XrtDataStyle** structures.

```
void
XrtFreeDataStyles(
    XrtDataStyle    **ds
    )
```

XrtFreePropString()

Frees the memory used by a string allocated by **XrtGetPropString**().

```
void
XrtFreePropString(
    char            *str
    )
```

XrtFreeStrings()

Frees the memory used by an array of strings.

```
void
XrtFreeStrings(
    char            **s
    )
```

XrtFreeTextHandles()

Frees the memory used by an array of handles to text areas.

```
void
XrtFreeTextHandles(
    XrtTextHandle   *list
    )
```

XrtFreeValueLabels()

Frees the memory used by a set of Value-labels.

```
void
XrtFreeValueLabels(
    XrtValueLabel   **label
    )
```

XrtGenCheckAxisBounds()

Determines if additional general data will require a chart repaint.

```
int
XrtGenCheckAxisBounds(
    HXRT2D      hChart,
    int         dataset,
    int         set,
    int         numpoints
    )
```

hChart is the chart handle. *dataset* indicates whether data is being added to the chart's first or second dataset and should be either 1 or 2. *numpoints* is the number of points that have been added to the set indexed by *set*.

Before calling this procedure, a chart control must have already been created and must have a general type **XrtData** structure attached to it using either the **XRT_DATA** or **XRT_DATA2** property.

To use this procedure, first add *numpoints* additional data values to one of the sets in the general dataset. **XrtGenDataAppendPt**() is a useful convenience routine for this purpose. Be sure to add the points to the **XrtData** structure currently being pointed to by the control's **XRT_DATA** or **XRT_DATA2** property.

Once you've added the additional data, call **XrtGenCheckAxisBounds**(). If all of the additional data falls within the current axes bounds, this routine will return 0. Otherwise, a bit pattern is returned indicating which axes bounds were exceeded by the new data:

```
#define XRT_LTX 0x01 /* New data < X axis min */
#define XRT_GTX 0x02 /* New data > X axis max */
#define XRT_LTY 0x04 /* New data < Y axis min */
#define XRT_GTY 0x08 /* New data > Y axis max */
```

The returned value can be used in a scrolling strip chart application to determine when to scroll the "data window."

XrtGenDataAppendPt()

Appends the (x,y) value *xval* and *yval* to the end of the set indexed by *set*.

```
int
XrtGenDataAppendPt(
    XrtData     *data,
    int         set,
    double      xval,
    double      yval
    )
```

XrtGenDataAppendPt() returns 1 on success and 0 on failure. *data* must point to a General structure.

XrtGenDataFastUpdate()

Displays additional general data in an existing chart as quickly as possible.

```
int
XrtGenDataFastUpdate(
    HXRT2D      hChart,
    int         dataset,
    int         set,
    int         numpoints
    )
```

hChart is the chart handle. *dataset* indicates whether data is being added to the chart's first or second dataset and should be either 1 or 2. *numpoints* is the number of points that have been added to the set indexed by *set*.

Before calling this procedure, a chart control must have already been created and must have a general type **XrtData** structure attached to it using either the **XRT_DATA** or **XRT_DATA2** property.

To use this procedure, first add *numpoints* additional data values to one of the sets in the general dataset. **XrtGenDataAppendPt**() is a useful convenience routine for this purpose. Be sure to add the points to the **XrtData** structure currently being pointed to by the control's **XRT_DATA** or **XRT_DATA2** property.

Once you've added the additional data, call **XrtGenDataFastUpdate**(). This routine will graph the newly added points. If a fast data update is possible, the new points will be added extremely quickly, otherwise a chart repaint will take place (as if you had simply reset the **XRT_DATA** or **XRT_DATA2** property.) This routine returns 0 if a chart repaint was required, and 1 if a fast update was possible.

You can determine ahead of time if a fast update is possible by using the *XrtGenCheckAxisBounds()* procedure.

XrtGenDataRemovePt()

Removes the value indexed by *set* and *point* from a General structure.

```
int
XrtGenDataRemovePt(
    XrtData     *data,
    int         set,
    int         point
    )
```

XrtGenDataRemovePt() returns 1 on success and 0 on failure.

XrtGenDataShiftPts()

Copies a block of points in one set of a General structure. Shifted-out points are set to the hole value.

```
int
XrtGenDataShiftPts(
    XrtData     *data,
    int         set,
    int         dest,
    int         src,
    int         npoints
    )
```

set is the set to shift points within; *src* specifies the first point in the block to be shifted; *npoints* specifies the size of the block. *dest* specifies the point to shift the block to. *set*, *dest*, and *src*, are all zero-based.

The sets's **npoints** element of the **XrtData** structure is not changed. This function returns 1 on success and 0 on failure.

XrtGetAction()

Returns the action that is bound to this window event. If there is no action bound, **XRT_ACTION_NONE** is returned.

```
XrtAction
XrtGetAction(
    HXRT2D      hChart,
    UINT        msg,
    UINT        modifier,
    UINT        keycode
    )
```

hChart is the chart handle. *msg* is the message for this window event. The following messages are recognized:

WM_LBUTTONDBLCLK	double-click left mouse button
WM_MBUTTONDBLCLK	double-click both mouse buttons
WM_RBUTTONDBLCLK	double-click right mouse button
WM_LBUTTONDOWN	press left mouse button
WM_MBUTTONDOWN	press both mouse buttons
WM_RBUTTONDOWN	press right mouse button
WM_LBUTTONUP	release left mouse button
WM_MBUTTONUP	release both mouse buttons
WM_RBUTTONUP	release right mouse button
WM_MOUSEMOVE	move mouse
WM_KEYDOWN	press key
WM_KEYUP	release key

modifier specifies the modifier flags, if any. The following modifier flags are recognized:

MK_LBUTTON	left mouse button
MK_MBUTTON	both mouse buttons
MK_RBUTTON	right mouse button
MK_ALT	Alt key
MK_SHIFT	Shift key
MK_CONTROL	Ctrl key

All actions are normalized to match the event sent by Microsoft Windows. For example, **MK_LBUTTON** is added to the modifier flags if a **WM_LBUTTONDOWN** message is sent.

keycode is the keycode. Any valid **VK_** value is treated as a recognized keycode. All alphabetic characters are forced to upper case. **MK_SHIFT** must appear in the modifier if capitals are desired. The CapsLock key toggles the meaning of the **MK_SHIFT** modifier.

XrtGetActionList()

Returns a pointer to the entire list of actions used by the control. The list is stored as a linked list, and must not be modified in any way.

```
XrtActionItem *
XrtGetActionList(
    HXRT2D      hChart,
    )
```

hChart is the chart handle.

XrtGetHandle()

Retrieves the chart attached to a window.

```
HXRT2D
XrtGetHandle(
    HWND        hWnd
    )
```

hWnd is the window handle.

XrtGetNthDataStyle()
XrtGetNthDataStyle2()

Returns a pointer to the *n*th data style currently being used by the chart.

```
XrtDataStyle *
XrtGetNthDataStyle(
    HXRT2D      hChart,
    int         index
    )
```

hChart is the chart handle.The (zero-based) *index* specifies which data style should be returned. The default data style for *index* will be returned if *index* is larger than the number of currently defined data styles. The pointer returned by this method should be considered *read-only*.

XrtGetNthDataStyle2() operates on the second set of data styles used in combination charts.

XrtGetNthFooterString()

Returns a pointer to the *n*th footer string currently being used by the chart.

```
char *
XrtGetNthFooterString(
    HXRT2D      hChart,
    int         index
    )
```

hChart is the chart handle. The (zero-based) *index* specifies which footer string should be returned. The default footer string for *index* will be returned if *index* is larger than the number of currently defined footer strings. The pointer returned by this method should be considered *read-only*.

XrtGetNthHeaderString()

Returns a pointer to the *n*th header string currently being used by the chart.

```
char *
XrtGetNthHeaderString(
    HXRT2D      hChart,
    int         index
    )
```

hChart is the chart handle. The (zero-based) *index* specifies which header string should be returned. The default header string for *index* will be returned if *index* is larger than the number of currently defined header strings. The pointer returned by this method should be considered *read-only*.

XrtGetNthPointLabel()
XrtGetNthPointLabel2()

Returns a pointer to the *n*th Point-label currently being used by the chart.

```
char *
XrtGetNthPointLabel(
    HXRT2D      hChart,
    int         index
    )
```

hChart is the chart handle. The (zero-based) *index* specifies which Point-label should be returned. If *index* is larger than the number of currently defined Point-labels, then **NULL** is returned. The pointer returned by this method should be considered *read-only*.

XrtGetNthPointLabel2() operates on the second set of Point-labels used in combination charts.

XrtGetNthSetLabel()
XrtGetNthSetLabel2()

Returns a pointer to the *n*th Set-label currently being used by the chart.

```
char *
XrtGetNthSetLabel(
    HXRT2D      hChart,
    int         index
    )
```

hChart is the chart handle. The (zero-based) *index* specifies which Set-label should be returned. If *index* is larger than the number of currently defined Set-labels, then **NULL** is returned. The pointer returned by this method should be considered *read-only*.

XrtGetNthSetLabel2() operates on the second set of Set-labels used in combination charts.

XrtGetPalette()

Retrieves a chart's color palette.

```
HPALETTE
XrtGetPalette(
    HXRT2D          hChart
    )
```

hChart is the chart handle.

XrtGetPropString()

Retrieves the current value of a chart property as a string.

```
BOOL
XrtGetPropString(
    HXRT2D          hChart,
    int             property,
    char            **str          /* Returned */
    )
```

hChart is the chart handle. *property* specifies any Olectra Chart property; *str* is a pointer to a string returned with the value of the property. *str* must be freed after use by calling **XrtFreePropString**().

XrtGetTextHandles()

Determines the text handles currently defined for the chart. **XrtGetTextHandles**() returns the total number attached.

```
int
XrtGetTextHandles(
    HXRT2D          hChart,
    XrtTextHandle   **list  /* Returned */
    )
```

hChart is the chart handle. *list* is a **NULL**-terminated list of text handles returned. This list must be freed after use by calling **XrtFreeTextHandles**().

XrtGetValueLabel()

Determines the Value-label closest to the label you specify, on either the X-, Y-, or Y2-axis.

```
XrtValueLabel *
XrtGetValueLabel(
    HXRT2D          hChart,
    XrtAxis         axis,
    XrtValueLabel   *label
    )
```

hChart is the chart handle. *axis* must be either **XRT_AXIS_X**, **XRT_AXIS_Y**, or **XRT_AXIS_Y2**; the *value* element in the **XrtValueLabel** structure pointed to by *label* should be filled in before calling this procedure. This procedure returns a pointer to an existing **XrtValueLabel** structure that has the closest *value* (to the one passed in). If an invalid axis is specified, or there are no Value-labels defined, **NULL** is returned. The pointer returned by this procedure should be considered *read-only*.

XrtGetValues()

Retrieves the current value of one or more chart properties.

```
void
XrtGetValues(
    HXRT2D      hChart,
    ...,
    NULL
    )
```

hChart is the chart handle. ... is one or more *property-value* pairs. Each pair consists of a *property* name and a pointer to a variable for its *value*. Olectra Chart writes the current value of the property to this variable. The list is terminated by **NULL**.

XrtGetWindow()

Retrieves the window attached to a chart.

```
HWND
XrtGetWindow(
    HXRT2D      hChart
    )
```

hChart is the chart handle.

XrtInsertNthDataStyle()

Inserts a new data style which becomes the *n*th data style.

```
void
XrtInsertNthDataStyle(
    HXRT2D       hChart,
    int          index,
    XrtDataStyle *ds
    int          dataset
    )
```

hChart is the chart handle. The (zero-based) *index* specifies the index of the new data style. All data styles with index greater than or equal to *index* are shifted up. *ds* points to an **XrtDataStyle** structure containing the new data style to be used. Olectra Chart makes its own copy of the data style pointed to by *ds*. *dataset* is either 1 or 2, indicating that either the first or second set of data styles is to be operated on.

A side effect of using this method is that **XRT_DATA_STYLES_USE_DEFAULT** is set to **FALSE**.

XrtInsertNthPointLabel()
XrtInsertNthPointLabel2()

Inserts a new Point-label which becomes the *n*th Point-label. **XrtInsertNthPointLabel2**() operates on the second set of Point-labels used in combination charts.

```
void
XrtInsertNthPointLabel(
    HXRT2D      hChart,
    int         index,
    char        *s
    )
```

hChart is the chart handle. The (zero-based) *index* specifies the index of the new Point-label. All Point-labels with index greater than or equal to *index* are shifted up. *s* is the string to be used. Olectra Chart makes its own copy of the characters pointed to by *s*.

XrtInsertNthSetLabel()
XrtInsertNthSetLabel2()

Inserts a new Set-label which becomes the *n*th Set-label. **XrtInsertNthSetLabel2**() operates on the second set of Set-labels used in combination charts.

```
void
XrtInsertNthSetLabel(
    HXRT2D      hChart,
    int         index,
    char        *s
    )
```

hChart is the chart handle. The (zero-based) *index* specifies the index of the new Set-label. All Set-labels with index greater than or equal to *index* are shifted up. *s* is the string to be used. Olectra Chart copies the characters pointed to by *s*.

XrtMakeData()

Procedure used to allocate an empty **XrtData** structure. **XRT_HUGE_VAL** is used as the hole value.

```
XrtData *
XrtMakeData(
    XrtDataType data_style,
    int         nsets,
    int         npoints,
    BOOL        all
    )
```

data_style must be either **XRT_DATA_GENERAL** or **XRT_DATA_ARRAY**. If *all* is **TRUE**, then space for the data values is also allocated, otherwise only the structure and pointers are allocated. When the style is **XRT_DATA_GENERAL**, *npoints* refers to the maximum number of points in each set.

XrtMakeDataFromFile()

Procedure which will allocate an **XrtData** structure and load it with data read in from a text file.

```
XrtData *
XrtMakeDataFromFile(
    char        *filename,
    char        *errbuf
    )
```

filename is the filename of the data file. If **XrtMakeDataFromFile**() encounters any errors, it will return **NULL**, and will write an error message in *errbuf*. If *errbuf* is **NULL**, error messages will be written to a debug window. *errbuf* should be at least 100 bytes.

Lines in the file beginning with a "**#**" symbol in the first position are treated as comments and ignored. The first non-comment line must begin with either

"**ARRAY**" or "**GENERAL**" followed by 2 integers specifying the number of sets (*nsets*) and the number of points (*npoints*) in each set. (If **GENERAL** is specified, the second integer specifies the *maximum* number of points in each set.) The two integers may optionally be followed by the character "T" to indicate that the data which follows is transposed.

The second non-comment line *can optionally* specify a hole value with the text "**HOLE <value>**". When Olectra Chart sees this value in the dataset, it treats that point as missing. The default hole value is **XRT_HUGE_VAL**.

If **ARRAY** is specified, the next *npoints* values are treated as the X-values. The following *npoints* values are treated as the first set's Y-values. The following *npoints* values are treated as the second set's Y-values, etc.

ARRAY DATA

An example of an **ARRAY** specification follows:

```
# The Michelle's Microchips mm63 file
ARRAY 2 4
HOLE 10000
# X-values
1.0 2.0 3.0 4.0
# Y-values line 1 - Expenses
150.0 175.0 160.0 170.0
# Y-values line 2
125.0 100.0 225.0 300.0 - Revenue
```

ARRAY DATA with "T"

An example of the same array data in transposed form follows:

```
# The Michelle's Microchips mm63 file transposed
ARRAY 2 4 T
# X-values  Y1 ExpensesY2 Revenues
1.0         150.0          125.0
2.0         175.0   100.0
3.0         160.0   225.0
4.0         170.0   300.0
```

If **GENERAL** is specified, each of the set definitions begins with an integer specifying the number of points in the set, followed by each of the set's X-values, followed by each of the set's Y-values.

GENERAL DATA

An example of a **GENERAL** specification follows:

```
# 3 lines, max 5 points each
GENERAL 3 5
HOLE 10000
# line 1, 3 points
3
1 3 5
5 14 24
# line 2, 5 points
5
1 2 3 4 5
11 15 18 21 22.
# line 3, 2 points
2
2 4
16 15.2
```

GENERAL DATA with "T"

GENERAL data may also be specified with the "T" transpose option.

```
# 3 lines, max 5 points each, transposed
GENERAL 3 5 T
# line 1, 3 points
3
# X     Y
1       5
3       14
5       24
# line 2, 5 points
5
# X     Y
1       11
2       15
3       18
4       21
5       22
# line 3, 2 points
2
# X     Y
2       16
4       15.2
```

XrtMakeTime()

Converts time information into a long integer (representing the number of seconds since January 1, 1970).

```
long
XrtMakeTime(
    int             yr,
    int             mon,
    int             day,
    int             hr,
    int             min,
    int             sec,
    )
```

yr is the year minus 1900. *mon* is a value from 0 to 11 representing the month. *day* is the day of the month. *hr* is a value from 0 to 23 representing the hour of the day. *min* and *sec* are the minute and second.

XrtMap()

Maps a pixel coordinate to a chart coordinate.

```
XrtRegion
XrtMap(
    HXRT2D          hChart,
    int             yaxis,
    int             pix_x,
    int             pix_y,
    XrtMapResult    *map
    )
```

Used by an application to determine the chart coordinates corresponding to *pix_x* and *pix_y* (in the chart whose handle is specified by *hChart*). Set *yaxis* to 1 for the coordinate system defined by the first Y-axis, or set to 2 for the second Y-axis. Results are returned in *map*. Typically used in an event handling procedure.

XrtPick()

Picks the displayed data that is closest to the given pixel coordinate.

```
XrtRegion
XrtPick(
    HXRT2D          hChart,
    XrtDsGroup      ds_group,
    int             pix_x,
    int             pix_y,
    XrtPickResult   *pick,
    XrtFocus        focus
    )
```

Used by an application to determine what data (in terms of *set* and *point*) is displayed closest to the pixel at (*pix_x*, *pix_y*) in the chart whose handle is specified by *hChart*. Results are returned in *pick*. Typically used in an event handling procedure.

The *ds_group* parameter is used to tell **XrtPick**() which dataset to use in combination charts. When calling **XrtPick**() on a single chart, set this to **XRT_DATASET1**. On combination charts, set it to **XRT_DATASET1** or **XRT_DATASET2** to restrict pick results to one of the datasets. Setting it to **XRT_DATASET1 | XRT_DATASET2** causes **XrtPick**() to consider both datasets.

The *focus* parameter indicates the method used by the **XrtPick**() routine to determine the closest set and point. Setting focus to **XRT_XFOCUS** means that only the distance along the X-axis is used to determine "closeness". Setting it to **XRT_YFOCUS** means that only the distance along the Y-axis is used. Setting it to **XRT_XFOCUS | XRT_YFOCUS** means that both the X- and Y-axis distance is used.

XrtPrint()

Outputs a chart to a printer using the Windows Print dialog box.

```
BOOL
XrtPrint(
    HXRT2D          hChart,
    XrtDrawFormat   format,
    XrtDrawScale    scale,
    long            left, top, width, height
    )
```

hChart is the chart handle. *format* is the graphics format to use and can be: **XRT_DRAW_BITMAP** for a Windows bitmap (BMP); **XRT_DRAW_METAFILE** for a Windows metafile (WMF); or **XRT_DRAW_ENHMETAFILE** (***Windows 95*** and ***Windows NT*** applications only) for an enhanced metafile (EMF). *scale* specifies the scaling to perform when printing, and is one of **XRT_DRAWSCALE_NONE** (no scaling), **XRT_DRAWSCALE_TOWIDTH** (scale to width specified by *width*, preserving aspect ratio and ignoring *height*), **XRT_DRAWSCALE_TOHEIGHT** (scale to height specified by *height*, preserving aspect ratio and ignoring *width*), **XRT_DRAWSCALE_TOFIT** (scale to minimum of *height* or *width*, preserving aspect ratio), **XRT_DRAWSCALE_TOMAX** (enlarge to size of page regardless of aspect ratio). *left* is the offset from the left of the page. *top* is the offset from the top of the page. *width* specifies the width to scale to; to use the existing window width, set *width* to 0. *height* specifies the height to scale to; to use the existing window height, set *height* to 0.

XrtRemoveNthDataStyle()

Removes the *n*th data style.

```
void
XrtRemoveNthDataStyle(
    HXRT2D          hChart,
    int             index,
    int             dataset
    )
```

hChart is the chart handle. The (zero-based) *index* specifies the index of the data style to be removed. All data styles with index greater than *index* are shifted down. *dataset* is either 1 or 2, indicating that either the first or second set of data styles is to be operated on.

XrtSaveDataToFile()

Writes data stored in an **XrtData** structure to a text file in a format suitable for use with **XrtMakeDataFromFile**().

```
int
XrtSaveDataToFile(
    XrtData         *data,
    char            *filename,
    char            *errbuf
    )
```

data is the structure containing the data to be written. *filename* is the filename of the file to be written. If any errors are encountered, messages will be written to *errbuf* (which should be at least 100 bytes long). If *errbuf* is **NULL**, messages will be written

to a debug window. This procedure returns 1 on success and 0 on failure. **XrtSaveDataToFile**() writes the hole value in the **XrtData** structure to the second line of the file.

XrtSetAction()

Programs an action for the Microsoft Windows event. Any previous action for this event is replaced.

```
void
XrtSetAction(
    HXRT2D      hChart,
    UINT        msg,
    UINT        modifier,
    UINT        keycode,
    XrtAction   action
    )
```

hChart is the chart handle. *msg* is the message for this window event. *modifier* specifies the modifier flags, if any. *keycode* is the keycode. *action* is the new action for this event; if *action* is **XRT_ACTION_NONE**, the action mapping is completely removed.

XrtSetNthDataStyle()
XrtSetNthDataStyle2()

Sets the *n*th data style. **XrtSetNthDataStyle2**() operates on the second set of data styles used in combination charts.

```
void
XrtSetNthDataStyle(
    HXRT2D      hChart,
    int         index,
    XrtDataStyle *ds
    )
```

hChart is the chart handle. The (zero-based) *index* specifies which data style should be changed. *ds* points to an **XrtDataStyle** structure containing the new data style to be used. Olectra Chart makes its own copy of the data style pointed to by *ds*.

A side effect of using this method is that **XRT_DATA_STYLES_USE_DEFAULT** is set to **FALSE**.

XrtSetNthFooterString()

Sets the *n*th footer string.

```
void
XrtSetNthFooterString(
    HXRT2D      hChart,
    int         index,
    char        *s
    )
```

hChart is the chart handle. The (zero-based) *index* specifies which footer string should be changed. *s* is the new string to be used. Olectra Chart makes its own copy of the characters pointed to by *s*.

XrtSetNthHeaderString()

Sets the *n*th header string.

```
void
XrtSetNthHeaderString(
    HXRT2D      hChart,
    int         index,
    char        *s
    )
```

hChart is the chart handle. The (zero-based) *index* specifies which header string should be changed. *s* is the new string to be used. Olectra Chart makes its own copy of the characters pointed to by *s*.

XrtSetNthPointLabel()
XrtSetNthPointLabel2()

Sets the *n*th Point-label. **XrtSetNthPointLabel2**() operates on the second set of Point-labels used in combination charts.

```
void
XrtSetNthPointLabel(
    HXRT2D      hChart,
    int         index,
    char        *s
    )
```

hChart is the chart handle. The (zero-based) *index* specifies which Point-label should be changed. *s* is the new string to be used. Olectra Chart makes its own copy of the characters pointed to by *s*.

XrtSetNthSetLabel()
XrtSetNthSetLabel2()

Sets the *n*th Set-label. **XrtSetNthSetLabel2**() operates on the second set of Set-labels used in combination charts.

```
void
XrtSetNthSetLabel(
    HXRT2D      hChart,
    int         index,
    char        *s
    )
```

hChart is the chart handle. The (zero-based) *index* specifies which Set-label should be changed. *s* is the new string to be used. Olectra Chart copies the characters pointed to by *s*.

XrtSetPropString()

Sets a chart property to the value represented by a string.

```
BOOL
XrtSetPropString(
    HXRT2D      hChart,
    int         property,
    char        *str
    )
```

hChart is the chart handle. *property* specifies any Olectra Chart property; *str* is a pointer to a string representation to set the property to.

XrtSetValueLabel()

Defines a new Value-label, on either the X-, Y-, or Y2-axis.

```
void
XrtSetValueLabel(
    HXRT2D          hChart,
    XrtAxis         axis,
    XrtValueLabel   *label
    )
```

hChart is the chart handle. *axis* must be either **XRT_AXIS_X**, **XRT_AXIS_Y**, or **XRT_AXIS_Y2**; the *value* and *string* elements in the **XrtValueLabel** structure pointed to by *label* should be filled in before calling this procedure. This procedure adds the new Value-label to the chart. Olectra Chart copies the **XrtValueLabel** pointed to by label.

XrtSetValues()

Sets one or more chart properties.

```
void
XrtSetValues(
    HXRT2D          hChart,
    ...,
    NULL
    )
```

hChart is the chart handle. ... is one or more *property-value* pairs. Each pair consists of a *property* name and a variable (must be the same data type as the property) containing its new *value*. Olectra Chart sets the property to the value. The list is **NULL**-terminated.

XrtTextAttach()

Attaches a text area to a chart.

```
void
XrtTextAttach(
    HXRT2D          hChart,
    XrtTextHandle   handle
    )
```

XrtTextCreate()

Used to create a new text area. Returns the text area's handle. See section 6.3 on page 85 for information on text areas.

```
XrtTextHandle
XrtTextCreate(
    HXRT2D          hChart,
    XrtTextDesc     *textd
    )
```

XrtTextDestroy()

Destroys a text area.

```
void
XrtTextDestroy(
    HXRT2D          hChart,
    XrtTextHandle   handle
    )
```

XrtTextDetach()

Detaches a text area from a chart.

```
void
XrtTextDetach(
    HXRT2D          hChart,
    XrtTextHandle   handle
    )
```

XrtTextDetail()

Retrieves text area information. Returns 1 if *handle* is valid, and 0 otherwise.

```
int
XrtTextDetail(
    HXRT2D          hChart,
    XrtTextHandle   handle,
    XrtTextDesc     *textd
    )
```

XrtTextUpdate()

Updates text area information.

```
void
XrtTextUpdate(
    HXRT2D          hChart,
    XrtTextHandle   handle,
    XrtTextDesc     *textd
    )
```

XrtTimeToValue()

Converts a **time_t** value to a value representing its position on the Time-axis.

```
double
XrtTimeToValue(
    HXRT2D          hChart,
    time_t          value
    )
```

hChart is the chart handle. *value* is the time value to be converted.

XrtUnmap()

Unmaps from a chart coordinate to a pixel coordinate.

```
void
XrtUnmap(
    HXRT2D          hChart,
    int             yaxis,
    double          x,
    double          y,
    XrtMapResult    *map
    )
```

Used by an application to determine the pixel coordinates corresponding to chart coordinate *(x, y)* in the chart whose handle is specified by *hChart*. Set *yaxis* to 1 to unmap from the coordinate system defined by the first Y-axis, or set to 2 for the second Y-axis. Results are returned in the *pix_x* and *pix_y* elements of the structure pointed to by *map*. *pix_x* and *pix_y* values of -1 are returned for unmap requests that are out of range. In non-transposed discrete charts, *pix_x* will be -1. In transposed discrete charts, *pix_y* will be -1.

XrtUnpick()

Determines a pixel coordinate given a dataset *set* and *point*.

```
void
XrtUnpick(
    HXRT2D          hChart,
    int             dataset,
    int             set,
    int             point,
    XrtPickResult   *pick
    )
```

Used by an application to determine the pixel coordinates at which a data element is displayed (in the chart whose handle is specified by *hChart*).

Set *dataset* to **XRT_DATASET1** unless unpicking from the second dataset of a combination chart. *set* and *point* are indices into the dataset.

Results are returned in the *pix_x* and *pix_y* elements of the structure pointed to by *pick*. Both *pix_x* and *pix_y* will be -1 if the data is out of range.

XrtValueToTime()

Converts a position on the Time-axis to its equivalent **time_t** value.

```
time_t
XrtValueToTime(
    HXRT2D          hChart,
    double          value
    )
```

hChart is the chart handle. *value* is the Time-axis position to be converted.

C

Macros

The following macros are provided for convenience in accessing various portions of the **XrtData** structure. They are defined in OLCH2DCM.H, located in Olectra Chart's \INCLUDE directory.

Macro Name	Definition
arr_nsets	a.nsets
arr_npoints	a.npoints
arr_data	a.data
arr_hole	a.hole
arr_xdata	arr_data.xp
arr_ydata	arr_data.yp
arr_xel(j)	arr_xdata[j]
arr_yel(i, j)	arr_ydata[i][j]
gen_nsets	g.nsets
gen_data	g.data
gen_hole	g.hole
gen_npoints(i)	gen_data[i].npoints
gen_xdata(i)	gen_data[i].xp
gen_ydata(i)	gen_data[i].yp
gen_xel(i, j)	gen_xdata(i)[j]
gen_yel(i, j)	gen_ydata(i)[j]
XrtGetDataType(xrt_data)	(xrt_data->g.type)

D

Message Reference

This appendix lists the Olectra Chart messages in alphabetical order.

XRTN_MODIFY_END

Sent to a chart's parent window to indicate that user interaction has ended:

```
XRTN_MODIFY_END:
hWnd = (HWND) wParam;
```

wParam is the window handle.

XRTN_MODIFY_START

Sent to a chart's parent window to indicate that a user interaction is about to begin:

```
XRTN_MODIFY_START:
hWnd = (HWND) wParam;
mcb = (XrtModifyCallbackStruct *) lParam;

typedef struct {
    BOOL doit;
} XrtModifyCallbackStruct;
```

wParam is the window handle. *lParam* is a pointer to a modification message structure. In this structure, the *doit* element indicates whether the user interaction is to be permitted; set *doit* to **FALSE** to disallow this user interaction.

XRTN_PALETTECHANGED

Sent to a chart's parent window after the chart control has changed its color palette.

```
XRTN_PALETTECHANGED:
hWnd = (HWND) wParam;
```

wParam is the window handle.

XRTN_PROPERTIES

Sent to a chart's parent window to indicate that the user has requested to activate the property page:

```
XRTN_PROPERTIES:
hWnd = (HWND) wParam;
pcb = (XrtPropertiesCallbackStruct *) lParam;

typedef struct {
    int x;
    int y;
} XrtPropertiesCallbackStruct;
```

wParam is the window handle. *lParam* is a pointer to a modification message structure. In this structure, the *x* and *y* elements indicate the coordinates at which the event occurred.

XRTN_REPAINTED

Sent to a window after the chart control has been redrawn:

```
XRTN_REPAINTED:
hWnd = (HWND) wParam;
cb = (XrtCallbackStruct *) lParam;

typedef struct {
    HDC         hdc;
    RECT        rectDamaged;
} XrtCallbackStruct;
```

wParam is the window handle. *lParam* is a pointer to a repaint message structure. In this structure, *hdc* is the handle to the device context, and *rectDamaged* is the rectangle that has been repainted.

XRTN_RESIZED

Sent to a window after the chart control has changed size:

```
XRTN_RESIZED:
hWnd = (HWND) wParam;
rcb = (XrtResizeCallbackStruct *) lParam;

typedef struct {
    int         width;
    int         height;
} XrtResizeCallbackStruct;
```

wParam is the window handle. *lParam* is a pointer to a resize message structure. In this structure, *width* is the new width of the control, and *height* is the new height of the control.

XRTN_ROTATE

Sent to a window when the user attempts to perform a rotation operation:

```
XRTN_ROTATE:
hWnd = (HWND) wParam;
rcb = (XrtRotateCallbackStruct *) lParam;

typedef struct {
    int         rotation;
    int         inclination;
    BOOL        doit;
} XrtRotateCallbackStruct;
```

wParam is the window handle. *lParam* is a pointer to a rotation message structure. In this structure, *rotation* and *inclination* are the proposed values for the **XRT_GRAPH_ROTATION** and **XRT_GRAPH_INCLINATION** properties; they can be modified by the message handler. The *doit* element indicates whether the rotation is to be permitted; set *doit* to **FALSE** to disallow this rotation.

XRTN_TRANSFORM

Sent to a window when the user attempts to perform a scaling, translation, or zooming operation:

```
XRTN_TRANSFORM:
hWnd = (HWND) wParam;
tcb = (XrtTransformCallbackStruct *) lParam;

typedef struct {
    BOOL        reset;/* Read-only */
    int         left_margin;
    int         right_margin;
    int         top_margin;
    int         bottom_margin;
    BOOL        doit;
} XrtTransformCallbackStruct;
```

wParam is the window handle. *lParam* is a pointer to a transformation message structure. In this structure, *left_margin*, *right_margin*, *top_margin* and *bottom_margin* are the proposed values for the Margin properties (**XRT_GRAPH_MARGIN_BOTTOM**, etc.); they can be modified to limit the scope of the transformation. The *doit* element indicates whether the transformation is to be permitted; set *doit* to **FALSE** to disallow this transformation.

E

Data Types

This appendix lists the Olectra Chart data types in alphabetical order. The C language definition of structures is also provided.

XrtAdjust

Enumeration used by the **XRT_FOOTER_ADJUST** and **XRT_HEADER_ADJUST** properties:

```
XRT_ADJUST_LEFT
XRT_ADJUST_RIGHT
XRT_ADJUST_CENTER
```

XrtAlign

Enumeration used with the **XRT_LEGEND_ORIENTATION** property:

```
XRT_ALIGN_VERTICAL
XRT_ALIGN_HORIZONTAL
```

XrtAnchor

Enumeration used with the **XRT_LEGEND_ANCHOR** property and for text areas:

```
XRT_ANCHOR_NORTH
XRT_ANCHOR_SOUTH
XRT_ANCHOR_EAST
XRT_ANCHOR_WEST
XRT_ANCHOR_NORTHWEST
XRT_ANCHOR_NORTHEAST
XRT_ANCHOR_SOUTHWEST
XRT_ANCHOR_SOUTHEAST
XRT_ANCHOR_HOME
XRT_ANCHOR_BEST
```

XRT_ANCHOR_HOME and **XRT_ANCHOR_BEST** are for use in text areas only.

XrtAnnoMethod

Enumeration used to define methods for X-, Y-, and Y2-axis annotation. Used with the **XRT_[XYY2]ANNOTATION_METHOD** property:

```
XRT_ANNO_VALUES
XRT_ANNO_POINT_LABELS
XRT_ANNO_VALUE_LABELS
XRT_ANNO_TIME_LABELS
```

XrtAnnoPlacement

Enumeration used with the **XRT_[XY]ANNO_PLACEMENT** property to select the location of axis annotation:

```
XRT_ANNO_AUTO
XRT_ANNO_ORIGIN
XRT_ANNO_MIN
XRT_ANNO_MAX
```

XrtArray

The structure defining **XrtData** when type is **XRT_DATA_ARRAY**:

```
typedef struct {
    XrtDataType     type;/*=XRT_DATA_ARRAY */
    double          hole;
    int             nsets;
    int             npoints;
    XrtArrayData    data;
} XrtArray;
```

XrtArrayData

The structure defining data values within an **XrtArray** structure:

```
typedef struct {
    double          *xp;
    double          **yp;
} XrtArrayData;
```

XrtAttachType

Enumeration used to specify the type of text area attachment:

```
XRT_TEXT_ATTACH_PIXEL
XRT_TEXT_ATTACH_VALUE
XRT_TEXT_ATTACH_DATA
XRT_TEXT_ATTACH_DATA_VALUE
```

XrtAxis

Enumeration used to specify an axis. Used with **XrtGetValueLabel**() and **XrtSetValueLabel**():

```
XRT_AXIS_X
XRT_AXIS_Y
XRT_AXIS_Y2
```

XrtBorder

Enumeration used by **XRT_HEADER_BORDER**, **XRT_FOOTER_BORDER**, **XRT_LEGEND_BORDER**, and **XRT_GRAPH_BORDER** and text areas:

```
XRT_BORDER_NONE
XRT_BORDER_3D_OUT
XRT_BORDER_3D_IN
XRT_BORDER_ETCHED_IN
XRT_BORDER_ETCHED_OUT
XRT_BORDER_SHADOW
XRT_BORDER_PLAIN
```

XrtCallbackStruct

Structure that defines the information passed to the message handler when the chart control has been redrawn:

```
typedef struct {
    HDC                 hdc;
    RECT                rectDamaged;
} XrtCallbackStruct;
```

XrtData

The structure defining data for use with **XRT_DATA**. It is a union of either array data or general data:

```
typedef union {
    XrtArray            a;
    XrtGeneral          g;
} XrtData;
```

XrtDataStyle

The structure defining how a particular set of data will appear when graphed:

```
typedef struct {
    XrtLinePattern      lpat;           /* line pattern */
    XrtFillPattern      fpat;           /* fill pattern */
    COLORREF            color;          /* color */
    int                 width;          /* line width */
    XrtPoint            point;          /* point style */
    COLORREF            pcolor;         /* point color */
    int                 psize;          /* point size */
    COLORREF            resh;           /* reserved */
    COLORREF            ress;           /* reserved */
} XrtDataStyle;
```

XrtDataType

Enumeration used for defining the type of an **XrtData** structure:

```
XRT_DATA_ARRAY
XRT_DATA_GENERAL
```

XrtDrawFormat

Enumeration used to specify chart output format with the **XrtDrawToClipboard**(), **XrtDrawToDC**(), **XrtDrawToFile**(), and **XrtPrint**() procedures:

```
XRT_DRAW_BITMAP
XRT_DRAW_METAFILE
XRT_DRAW_ENHMETAFILE
```

XrtDrawScale

Enumeration used to specify the scaling factor to use when printing the chart:

```
XRT_DRAWSCALE_NONE
XRT_DRAWSCALE_TOWIDTH
XRT_DRAWSCALE_TOHEIGHT
XRT_DRAWSCALE_TOFIT
XRT_DRAWSCALE_MAX
```

XrtFillPattern

Enumeration of various fill patterns used within an **XrtDataStyle** structure:

```
XRT_FPAT_NONE
XRT_FPAT_SOLID
XRT_FPAT_25_PERCENT
XRT_FPAT_50_PERCENT
XRT_FPAT_75_PERCENT
XRT_FPAT_HORIZ_STRIPE
XRT_FPAT_VERT_STRIPE
XRT_FPAT_45_STRIPE
XRT_FPAT_135_STRIPE
XRT_FPAT_DIAG_HATCHED
XRT_FPAT_CROSS_HATCHED
XRT_WFPAT_BDIAGONAL
XRT_WFPAT_FDIAGONAL
XRT_WFPAT_HORIZONTAL
XRT_WFPAT_VERTICAL
XRT_WFPAT_CROSS
XRT_WFPAT_DIAGCROSS
```

XrtGeneral

The structure defining **XrtData** when type is **XRT_DATA_GENERAL**:

```
typedef struct {
    XrtDataType     type; /*=XRT_DATA_GENERAL*/
    double          hole;
    int             nsets;
    XrtGeneralData  *data;
} XrtGeneral;
```

XrtGeneralData

The structure defining the data values within an **XrtGeneral** structure:

```
typedef struct {
    int             npoints;
    double          *xp;
    double          *yp;
} XrtGeneralData;
```

XrtLinePattern

Enumeration of various line patterns, used within an **XrtDataStyle** structure:

```
XRT_LPAT_NONE
XRT_LPAT_SOLID
XRT_LPAT_LONG_DASH
XRT_LPAT_DOTTED
XRT_LPAT_SHORT_DASH
XRT_LPAT_LSL_DASH
XRT_LPAT_DASH_DOT
```

XrtMapResult

Structure used to pass information about mapped pixel coordinates:

```
typedef struct {
    int          pix_x, pix_y;
    int          yaxis;
    double       x, y;
} XrtMapResult;
```

XrtNumMethod

Enumeration used with the **XRT_[XYY2]NUM_METHOD** property to determine how to calculate axis annotation:

```
XRT_NUM_PRECISION
XRT_NUM_ROUND
```

XrtOriginPlacement

Enumeration used to select the way in which the default origin is chosen:

```
XRT_ORIGIN_AUTO
XRT_ORIGIN_ZERO
XRT_ORIGIN_MIN
XRT_ORIGIN_MAX
```

XrtPickResult

Structure used to pass information about picked pixel coordinates:

```
typedef struct {
    int          pix_x, pix_y;
    int          dataset;
    int          set, point;
    int          distance;
} XrtPickResult;
```

XrtPieOrder

Enumeration of pie ordering styles:

```
XRT_PIEORDER_ASCENDING
XRT_PIEORDER_DESCENDING
XRT_PIEORDER_DATA_ORDER
```

XrtPieThresholdMethod

Enumeration of pie threshold methods:

```
XRT_PIE_SLICE_CUTOFF
XRT_PIE_PERCENTILE
```

XrtPoint

Enumeration of various point styles used in **XrtDataStyle**:

```
XRT_POINT_NONE
XRT_POINT_DOT
XRT_POINT_BOX
XRT_POINT_TRI
XRT_POINT_DIAMOND
XRT_POINT_STAR
XRT_POINT_VERT_LINE
XRT_POINT_HORIZ_LINE
XRT_POINT_CROSS
XRT_POINT_CIRCLE
XRT_POINT_SQUARE
```

XrtPropertiesCallbackStruct

Structure that defines the information passed to the message handler when the user clicks the right mouse button to activate the property page:

```
typedef struct {
    int         x;
    int         y;
} XrtPropertiesCallbackStruct;
```

XrtRectangle

Structure used to define the bounding box of a text area:

```
typedef struct {
    int         x;
    int         y;
    int         width;
    int         height;
} XrtRectangle;
```

XrtRegion

Enumeration of map or pick results. Returned by **XrtMap**() and **XrtPick**():

```
XRT_RGN_NOWHERE
XRT_RGN_IN_GRAPH
XRT_RGN_IN_LEGEND
XRT_RGN_IN_FOOTER
XRT_RGN_IN_HEADER
```

XrtResizeCallbackStruct

Structure that defines the information passed to the message handler when the chart control changes size:

```
typedef struct {
    int         width;
    int         height;
} XrtResizeCallbackStruct;
```

XrtRotate

Enumeration used to define counter-clockwise rotation in degrees. Used with title and annotation rotation properties:

```
XRT_ROTATE_NONE
XRT_ROTATE_90
XRT_ROTATE_270
```

XrtRotateCallbackStruct

Structure that defines the information passed to the message handler when the user rotates a chart:

```
typedef struct {
    int             rotation;
    int             inclination;
} XrtRotateCallbackStruct;
```

XrtTextDesc

Structure used to specify a text area:

```
typedef struct {
    XrtTextPosition  position;
    char **          strings;
    XrtAnchor        anchor;
    int              offset;
    int              connected;
    XrtAdjust        adjust;
    COLORREF         fore_color;
    COLORREF         back_color;
    XrtBorder        border;
    int              border_width;
    HFONT            font;
    XrtRectangle     coords; /* read-only */
} XrtTextDesc;
```

XrtTextPosition

Union used to specify the position of a text area:

```
typedef union {
    struct {
    XrtAttachType   type;
    int             x, y;
} pixel;
struct {
    XrtAttachType   type;
    int             dataset;
    double          x, y;
} value;
struct {
    XrtAttachType   type;
    int             dataset;
    int             set, point;
} data;
struct {
    XrtAttachType   type;
    int             dataset;
    int             set, point;
    double          y;
} data_value;
} XrtTextPosition;
```

XrtTimeUnit

Enumeration used to define time units. Used with **XRT_TIME_UNIT**:

```
XRT_TMUNIT_SECONDS
XRT_TMUNIT_MINUTES
XRT_TMUNIT_HOURS
XRT_TMUNIT_DAYS
XRT_TMUNIT_WEEKS
XRT_TMUNIT_MONTHS
XRT_TMUNIT_YEARS
```

XrtTransformCallbackStruct

Structure that defines the information passed to the message handler when the user scales, translates or zooms a chart:

```
typedef struct {
    int             left_margin;
    int             right_margin;
    int             top_margin;
    int             bottom_margin;
    BOOL            doit;
} XrtTransformCallbackStruct;
```

XrtType

Enumeration used to define chart types. Used with **XRT_TYPE**:

```
XRT_TYPE_PLOT
XRT_TYPE_BAR
XRT_TYPE_PIE
XRT_TYPE_STACKING_BAR
XRT_TYPE_AREA
```

XrtValueLabel

The structure defining a single Value-label for the X-, Y-, or Y2-axis. Used with the **XRT_[XYY2]LABELS** property.

```
typedef struct {
    double      value;
    char        *string;
} XrtValueLabel;
```

F

Sample Code

PANEL.C

This appendix describes Olectra Chart's sample code and provides a listing of one program.

C Examples

Olectra Chart provides the following examples, located in Olectra Chart's \CHART\2D\DEMOS\DLL\SDK directory. There is a directory for each program.

- PLOT1 displays a very basic chart that you can use to learn some Olectra Chart basics. It is the starting point for the discussion in Chapter 1.
- PANEL displays a simple chart that the user can manipulate by clicking on several buttons. It is the ending point for the discussion in Chapter 1.
- CASHFLOW displays an income and expenses report broken down by category.
- STOCK displays a simple stock-market simulation using two charts (one for stock prices, the other for volume). This program also shows how to create a combination chart and use the Fast Update functions to quickly add points to it.
- TIME shows how to label the X-axis using Time-axis labels. The chart interprets X-values as months.

MFC Example

Olectra Chart provides a simple program that uses the MFC classes. It is located in Olectra Chart's \CHART\2D\DEMOS\DLL\MFC directory.

OWL Example

Olectra Chart provides a simple program that uses the OWL classes. It is located in Olectra Chart's \CHART\2D\DEMOS\DLL\OWL directory.

F.1 PANEL.C

The PANEL.C program defines a window containing several button controls. A chart is created below the buttons. Message handlers are registered for the buttons.

When the user clicks a button, the message handler runs and changes a chart property.

When run, the window in Figure 7 on page 14 displays.

```
#include <windows.h>
#include <olch2d.h>
#include "panel.h"

/* Text Strings for the graph */
char *header_strings[] = { "Michelle's Microchips", NULL };
char *footer_strings[] = { "1963 Quarterly Results", NULL };

/* Label Strings */
char *set_labels[] = { "Expenses", "Revenue", NULL };
char *point_labels[] = { "Q1", "Q2", "Q3", "Q4", NULL };


long WINAPI
WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)
{
    static HWND hwndXrt2D;
    static HXRT2D hChart;
    static XrtData *my_data;
    UINT wCheck;

    switch (msg) {
    case XRTN_PALETTECHANGED:
        SendMessage(XrtGetWindow(hChart), WM_QUERYNEWPALETTE, 0, 0);
        break;

    case WM_QUERYNEWPALETTE:
    case WM_PALETTECHANGED:
        SendMessage(XrtGetWindow(hChart), msg, wParam, lParam);
        break;

    case WM_INITDIALOG:
        hwndXrt2D = GetDlgItem(hWnd, IDGRAPH);
        hChart = XrtCreate();
        XrtAttachWindow(hChart, hwndXrt2D);
        my_data = XrtMakeDataFromFile((LPSTR) "mm63.dat", NULL);
        XrtSetValues(hChart,
                XRT_DATA,       my_data,
                XRT_HEADER_STRINGS,header_strings,
                XRT_FOOTER_STRINGS, footer_strings,
                XRT_SET_LABELS, set_labels,
                XRT_POINT_LABELS,point_labels,
                XRT_XANNOTATION_METHOD,XRT_ANNO_POINT_LABELS,
                NULL);

        CheckRadioButton(hWnd, IDBAR, IDPLOT, IDPLOT);
        SetFocus(hWnd);
        break;

    case WM_COMMAND:
        /* check which button clicked and set graph properties appropriately */
        switch (wParam) {
        case IDBAR:
            /* change the graph to a bar chart */
            CheckRadioButton(hWnd, IDBAR, IDPLOT, wParam);
```

```
            XrtSetValues(hChart, XRT_TYPE, XRT_TYPE_BAR, NULL);
            break;

        case IDPLOT:
            /* change the graph to a plot graph */
            CheckRadioButton(hWnd, IDBAR, IDPLOT, wParam);
            XrtSetValues(hChart, XRT_TYPE, XRT_TYPE_PLOT, NULL);
            break;

        case IDTRAN:
            /* transpose the points and sets of the graph */
            wCheck = IsDlgButtonChecked(hWnd, IDTRAN);
            CheckDlgButton(hWnd, IDTRAN, !wCheck);
            XrtSetValues(hChart, XRT_TRANSPOSE_DATA, !wCheck, NULL);
            break;

        case IDINV:
            /* invert the axes of the graph */
            wCheck = IsDlgButtonChecked(hWnd, IDINV);
            CheckDlgButton(hWnd, IDINV, !wCheck);
            XrtSetValues(hChart, XRT_INVERT_ORIENTATION, !wCheck, NULL);
            break;
        }
        break;

    case WM_SIZE:
        {
            int width;
            int height;
            RECT rect;

            width = LOWORD(lParam);
            height = HIWORD(lParam);

            GetWindowRect(hwndXrt2D, &rect);
            SetWindowPos(hwndXrt2D, NULL, rect.left, rect.top,
                        width , height - rect.top + 23, SWP_NOMOVE);
        }
        break;

    case WM_CLOSE:
        XrtDestroyData(my_data, TRUE);
        DestroyWindow(hWnd);

    case WM_DESTROY:
        PostQuitMessage(0);
        break;

    default:
        return FALSE;
    }
    return TRUE;
}

int PASCAL
WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpszCmdLine, int nCmdShow)
{
    static char szAppName[] = "panel";
    HWND    hWnd;
    MSG     msg;
```

```
    WNDCLASS wc;
    DLGPROC dlgprc;
    if (!hPrevInstance) {
        wc.style            = CS_HREDRAW | CS_VREDRAW;
        wc.lpfnWndProc      = WndProc;
        wc.cbClsExtra       = 0;
        wc.cbWndExtra       = 0;
        wc.hInstance        = hInstance;
        wc.hIcon            = LoadIcon (hInstance, IDI_APPLICATION);
        wc.hCursor          = LoadCursor (NULL, IDC_ARROW);
        wc.hbrBackground    = GetStockObject (WHITE_BRUSH);
        wc.lpszMenuName     = NULL;
        wc.lpszClassName    = szAppName;
        if (!RegisterClass(&wc)) return FALSE;
    }
    dlgprc = (DLGPROC) MakeProcInstance((FARPROC)WndProc, hInstance);
    hWnd = CreateDialog(hInstance, szAppName, 0, dlgprc);
    ShowWindow(hWnd, nCmdShow);
    while (GetMessage(&msg, NULL, 0, 0)) {
        if (!IsDialogMessage(hWnd, &msg)) {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
        }
    }
    return msg.wParam;
}
```

Index

D

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z