ChartBuilder Properties & Events

These are ChartBuilder's unique properties & events. In addition, ChartBuilder has several standard Visual Basic properties & events which behave in the standard way.

LabelText
LeftTitle
Legend
LegendStyle
LegendText
LineStats
MousePointer
NumPoints
NumSets
Palette
Pattern
PatternData
PatternedLines
Picture
PrintInfo
PrintStyle
QuickData
RandomData
PrintStyle
SDKInfo
SDKMouse
SDKPaint
SeeThru
Symbol
SymbolData
SymbolSize
ThickLines
ThisPoint
ThisSet
TickEvery
Ticks
XAxisMax
XAxisMin
XAxisPos
XAxisStyle
XAxisTicks
XPos
XPosData
YAxisMax
YAxisMin
YAxisPos
YAxisStyle
YAxisTicks

HotHitEvent
SDKHitEvent

SDKPaintEvent
SDKPressEvent
SDKTrackEvent

About
Display About Box

Property Value
Constant text

Description
Read-only, design-time property which triggers the display of the ChartBuilder About Box.

AsciiForms

AsciiColor
AsciiData
AsciiExtra
AsciiFFamily
AsciiFName
AsciiFSize
AsciiFStyle
AsciiLabel
AsciiLegend
AsciiPattern
AsciiSymbol
AsciiXPos

These properties only appear in ASCII form files. They are text equivalents of ChartBuilder's array-type properties. When a graph is saved to an ASCII form its array-type properties are saved as single strings, with each array element separated from its neighbour by a special character.

Columns are separated by the '~' character and the rows of the 2D properties (Data & XPos) are separated by the '^' character. Note that this means you must not use either of these characters within labels or legends if you intend to use ASCII forms.

We have had to implement ASCII form support in this way because, due to the way Visual Basic implements ASCII form files, it is not valid to refer to the numeric array-type properties in an ASCII form file by their usual names.

Help
Request Windows Help

Property Value
Constant text

Description
Read-only, design-time property which activates Windows Help.

If you access this property Windows Help is activated with the ChartBuilder control help file.

AutoInc
Increment Counters

Property Value
0 - Off
1 - On (default)

Description
This property allows **<*array*>Data** and **<*array*>Text** array properties to be set without the necessity of manually incrementing the **ThisPoint** position counter from **ThisPoint** = 1 to **ThisPoint** = **NumPoints**.

A special case is **GraphData**, when **AutoInc** will go through all the points and sets consecutively from **ThisPoint** = 1 to **ThisPoint** = **NumPoints** for each of **ThisSet** = 1 to **ThisSet** = **NumSets.**

When **AutoInc** is set to a new value (0 or 1), the **ThisPoint** and **ThisSet** properties are both re-initialised to 1. Also, when **AutoInc** is On and you switch from setting one of the array properties to a different array property, then once again both **ThisPoint** and **ThisSet** are re-initialised.

Note that **AutoInc** is only effective when setting data values. When getting/using data values **ThisPoint** and **ThisSet** are unaffected.

Example
```
Graph1.ThisSet = 1
for i% = 1 to Graph1.NumSets
        Graph1.ThisPoint = 1
        for j% = 1 to Graph1.NumPoints
                Graph1.GraphData = j%*i%
                If Graph1.ThisPoint < Graph1.NumPoints Then
                        Graph1.ThisPoint = Graph1.ThisPoint + 1
                End If
        next j%
        If Graph1.ThisSet < Graph1.NumSets Then
                Graph1.ThisSet = Graph1.ThisSet + 1
        End If
next i%
Graph1.DrawMode = 2
```

.... may be rewritten as ....

```
Graph1.AutoInc = 1
For i% = 1 To (Graph1.NumSets * Graph1.NumPoints)
    Graph1.GraphData = Graph1.ThisPoint * Graph1.ThisSet
Next i%
Graph1.DrawMode = 2
```

Sadly it is not possible to use **ThisPoint** or **ThisSet** as counters in For statements - Visual Basic will not allow it.

AutoInc works for all the **<*array*>Data** and **<*array*>Text** array properties of the ChartBuilder Control. These are:

GraphData
ColorData
ExtraData
LabelText
LegendText
PatternData

SymbolData
XPosData

Backdrop
File name for backdrop

Property Value
FileSpec (string)

Description
Sets the file name for a bitmap or metafile to be read and displayed as a backdrop to a graph. The appropriate extension (.BMP or .WMF) is appended automatically according to the setting of **BackdropStyle**. Backdrop files are not stored as part of the VB form - ChartBuilder re-reads the file whenever necessary. If the file name is invalid, or the file does not exist, ChartBuilder simply ignores the request to display a backdrop - no error is reported.

No backdrop is displayed if a graph is drawn in monochrome mode. This applies when **DrawStyle** = Monochrome and during printing when **PrintStyle** = Monochrome.

ChartBuilder can be used as a bitmap viewer by setting GraphType = 0 (None) and thus avoiding any graph display. Unlike Visual Basic 1.00, if your hardware supports 256 colors, ChartBuilder will display 256-color bitmaps in their full colors.

BackdropStyle
Select backdrop type and presentation style

Property Value
0 - None (default)
1 - Centred Bitmap
2 - Tiled Bitmap
3 - Stretched Bitmap
4 - Stretched Metafile

Description
Determines the type and style of presentation of a graph backdrop. You can select the type of backdrop independent of DrawMode (Draw or Blit), but you will get slightly better performance if you match like with like ie. use a Bitmap backdrop with Blitting, a Metafile with Drawing. If your hardware supports 256 colors, 256-color bitmaps will be displayed in their full colors.

The file name of the bitmap or metafile in **Backdrop** will given the appropriate extension (.BMP or .WMF) according to the BackdropStyle value.

Background
Select Background Color

Property Value
0 - Black
1 - Blue
2 - Green
3 - Cyan
4 - Red
5 - Magenta
6 - Brown
7 - Light Grey
8 - Dark Grey
9 - Light Blue
10 - Light Green
11 - Light Cyan
12 - Light Red
13 - Light Magenta
14 - Yellow
15 - White (default)

Description
The background color of the ChartBuilder control may be chosen from the above list.

When you change the background color, by default the colors for the components of the graph will be selected on your behalf to give good contrast. However, you can change the **Foreground** property to set the color used for titles, axes etc (the default value of **Foreground** is 'Auto Black/White'). Also, the **ColorData** property sets the colors for individual bars, pie slices etc.

BottomTitle
Title at bottom of graph

Property Value
Text

Description
The text string provided will be placed at the bottom of the graph parallel to the horizontal axis. This property is ignored for Pie charts.

Example
```
Sub Command1_Click ()
graph2.randomdata = 1
If graph2.BottomTitle = "" Then
        graph2.BottomTitle = "Bottom Title"
Else
        graph2.BottomTitle = ""
End If
graph2.DrawMode = 2
End Sub
```

Color (run time array)
Select colors of data sets/points

Property Value
Same as ColorData

Description
Color(i) provides an alternative way of accessing the **ColorData** array at run time. Whereas the ColorData property itself can be accessed at both design time and run time using **ThisPoint** and/or **ThisSet** or the **AutoInc** feature, Color(i) can only be used at run time due to restrictions within Visual Basic. It is, however, a much slicker way of accessing ColorData from your program code.

Color(i) is a 1-based property array ie. Color(3) refers to the third point in the ColorData array.

To set the third point in the ColorData array to Yellow you would previously have written:

```
ThisPoint = 3
ColorData = 14
```

You can now write:

```
Color(3) = 14
```

ColorData (Array)
Select Colors for bars, pie-slices etc

Property Value
0 - Black
1 - Blue
2 - Green
3 - Cyan
4 - Red
5 - Magenta
6 - Brown
7 - Light Grey
8 - Dark Grey
9 - Light Blue
10 - Light Green
11 - Light Cyan
12 - Light Red
13 - Light Magenta
14 - Yellow
15 - White

Description
This allows you to select the colors used to represent each of the data sets on the graph.
The exception is for Pie charts and Bar graphs with **NumSets** = 1, when a color should be allocated for each point, rather than each set.

Since this is an array property, the array element you set is determined by the current value of the **ThisPoint** property.

When entering data, you may make use of the **AutoInc** property. If **AutoInc** is On, then every time you set a new value, the **ThisPoint** counter is automatically incremented. At design time this allows you enter data quickly, at run time it simplifies your programming.

Note
Once one color has been selected, colors should be selected for ALL sets, or they will be shown in black.

CtlVersion
ChartBuilder Control version number

Property Value
Constant Text

Description
This is a read-only property giving the current release of the ChartBuilder control.

CurveOrder

Select polynomial order for curve fitting

Property Value

Integer in range 1 - 9, default value 1 (straight line)

Description

Determines the order of the polynomial being used for curve fitting. It is only applicable when **CurveType** = 0 (Variable Order Polynomial).

A CurveOrder of NumPoints-1 will result in a curve which passes through every point whereas a CurveOrder of 1 will result in a straight line.

CurveSteps
Select resolution of curve display

Property Value
Integer in range 1 - 1000, default value 50

Description
Determines the resolution or 'granularity' of the curve. A curve is actually made up of a number of straight lines joining points calculated to be on the curve defined by the selected mathematical model. The greater the number of calculated points, the smoother the curve will appear. However, the smoother the curve, the longer it takes to draw.

For a VGA display, a value of 50 results in a quickly-drawn, smooth-looking curve. However, if you are displaying or printing on a very high resolution device, you may wish to increase the CurveSteps value.

CurveType
Select curve fit model

Property Value
0 - Variable Order Polynomial (default)
1 - Logarithmic y=a+b*log(x)
2 - Exponential y=a*exp(b*x)
3 - Exponential y=a*x*exp(-b*x)
4 - Power y=a*x^b
5 - Inverse y=a+b/x
6 - Inverse y=a/(b+x)
7 - Inverse y=1/(a+b*x)
8 - Inverse y=x/(a*x+b)
9 - Inverse y=1/(a+b*x)^2

Description
Selects the mathematical model to be used for curve fitting. In all cases least squares regression is used as the fitting alogorithm.

The most popular model is the variable order polynomial. In this case you must also set **CurveOrder**.

The curve is drawn using the current line style ie. it is affected by the **ThickLines** and **PatternedLines** properties and the **PatternData** array. The **CurveSteps** property controls the resolution or 'granularity' of the curve.

Curve fitting only applies to scatter graphs, and then only when **GraphStyle** is set appropriately.

When experimenting with curve fitting, don't be surprised if sometimes no curve is drawn. Some of the models are invalid for x = 0. If you want to see all the different curves, enter some non-zero **XPosData**.

Data (run time 2D array)
Specify values to be graphed

Property Value
Same as GraphData

Description
Data(i) provides an alternative way of accessing the 2-D **GraphData** array at run time. Whereas the GraphData property itself can be accessed at both design time and run time using **ThisPoint** and **ThisSet** or the **AutoInc** feature, Data(i) can only be used at run time due to restrictions within Visual Basic. It is, however, a much slicker way of accessing GraphData within your program code.

Data(i) is a 1-based property array ie. Data(3) refers to the third point in one of the data sets of the GraphData array. Since GraphData is a 2-D array and Visual Basic does not support 2-D arrays directly, you need some way of specifying which data set you are referring to. You do this by setting **ThisSet**.

To set the third point in the second data set of the GraphData array to 125.5 you would previously have written:

```
ThisSet = 2
ThisPoint = 3
GraphData = 125.5
```

You can now write:

```
ThisSet = 2
Data(3) = 125.5
```

DataReset
Reset array data

Property Value
0 - None (default)
1 - GraphData
2 - ColorData
3 - ExtraData
4 - LabelText
5 - LegendText
6 - PatternData
7 - SymbolData
8 - XPosData
9 - All Data
10 - Font Info

Description
This property allows you to remove any or all of the ***<array>*Data** or ***<array>*Text** information which has been supplied to the ChartBuilder control. It will also reset the font selection information as determined by **FontFamily**, **FontStyle** and **FontSize** to their default values.

The 'All Data' option resets all the data and text arrays.

By resetting an array you will reset it to its original empty state. ChartBuilder will then once again take its default action when drawing the graph. For example, in the case of **GraphData**, random test data will once again be generated.

ThisPoint and **ThisSet** are both re-initialised to 1 when **DataReset** is used.

DrawMode
Set the drawing mode

Property Value
0 - No Action
1 - Clear
2 - Draw (default)
3 - Blit
4 - Copy
5 - Print
6 - Write

Description
This defines the drawing mode for the ChartBuilder Control.

DrawModes 0 - 3 are remembered when a graph is saved to disk, and between design-mode and run-mode. DrawModes 4, 5 and 6 are transient values which are used simply to fire the appropriate actions.

At design time, when you change a graph property, the graph is automatically redrawn to show the effect of the change. However, at run time, the graph is only redrawn when you set **DrawMode** to 2 (Draw) or 3 (Blit). This allows you to change as many property values as you wish before displaying the graph.

A special case is at run time when the form containing a graph is first displayed. Then, the graph **is** automatically displayed, according to the current **DrawMode** value.

If **DrawMode** is set to 0, then no action takes place; the control is left blank.
We have provided this option to enable you to leave the control blank until you are ready to display your graph. At design time you can create a graph with whatever property values you wish, and then, before running your program, set **DrawMode** = 0. When you are ready, set **DrawMode** = 2 and your graph will appear just as you designed it.

If **DrawMode** is set to 1, then the control is cleared with the **Background** color, and, if present, the **GraphCaption** text is displayed, centred in the control.

If **DrawMode** is set to 2 (Draw), you will see the components of the graph (its titles, legends, axes etc) appear one by one until the graph is complete. While drawing the graph, Graphics Server builds a metafile of the drawing operations so that later on it can re-paint the graph.

If **DrawMode** is set to 3 (Blit), there will be a brief pause and then the graph will appear all at once. In this mode, Graphics Server builds a hidden bitmap of the graph and then displays it with a BitBlit operation. This mode is useful if you want to draw a graph and then update it with changing data, as the graph will appear to change instantaneously.

DrawMode = 4 copies the image of the Graph to the Clipboard in either Bitmap or Metafile format. If **DrawMode** was set to 3 (Blit), it will be in Bitmap format, otherwise in Metafile format.

DrawMode = 5 allows a high quality image of the Graph to be printed without the form (see **PrintStyle).** At run time, by setting up the **PrintInfo**() array, you can also print a high quality graph into VB's own printer device context, thus mixing standard VB hardcopy with graphs on the same page without the loss of resolution suffered by the VB **PrintForm** command.

DrawMode = 6 writes the image to disk as a Bitmap (.BMP) or Windows Metafile (.WMF). For this option to work, the property **ImageFile** must have been set to provide a name for the file. If **DrawMode** was set to 3 (Blit), a Bitmap will be created, otherwise a Metafile.

DrawStyle
Show graph in monochrome or color

Property Value
0 - Monochrome
1 - Color (default)

Description
By setting this property to monochrome, the ChartBuilder control sets the background to white, all colors to black and will, if no **PatternData**, **SymbolData** or **GraphStyle** have been set, supply default patterns and symbols.

Extra (run time 2D array)
Select additional style options

Property Value
Same as ExtraData

Description
Extra(i) provides an alternative way of accessing the **ExtraData** array at run time. See description of Data(i) for more information.

ExtraData (2D Array)
Additional style data

Property Value
0 <= ExtraData <= 15

Description
The **ExtraData** property has three purposes:

i. 'Explode' Pie chart segment(s) (**0 - not exploded, 1 - exploded**)
ii. Color of the sides of a 3-D bar chart (using colors **0 - 15** as for **ColorData**).
iii. For Line, Log/Lin, Scatter & Tape graphs, mark the corresponding GraphData & XPosData values as missing **(0 - present** (default)**, 1 - missing)**. This feature enables the drawing of graphs with unequal sized data sets - set **NumPoints** to the size of the largest set, and mark the missing points of the remaining sets using the **ExtraData** array. Missing points are not only not drawn, but are also excluded from stats calculations & curve fitting. In the case of Line, Log/Lin & Tape graphs a missing point embedded within a data set (ie. not a leading or trailing point) results in a gap in the line/tape.

Since this is a two-dimensional array property, the array element you set is determined by the current value of the **ThisPoint** and **ThisSet** properties.

When entering data, you may make use of the **AutoInc** property. If **AutoInc** is On, then every time you set a new value, the **ThisPoint** counter is automatically incremented. When it reaches its maximum value (**NumPoints**), the **ThisSet** counter is incremented and **ThisPoint** reset to 1. If **ThisSet** reaches its maximum value (**NumSets**) then it also is reset to 1. At design time this allows you enter data quickly, at run time it simplifies your programming.

Example
```
graph1.ThisPoint = 2
graph1.ExtraData = 1
graph1.ThisPoint = 4
graph1.ExtraData = 1
```

applied to a Pie chart with five 'slices' will give a Pie chart with its second and fourth slice exploded.

FontFamily
Select font family

Property Value
0 - Roman (default)
1 - Swiss
2 - Modern

Description
Selects the font family in which the text specified by the current setting of **FontUse** will be displayed.

ChartBuilder uses font family rather than type face as its method of font selection to avoid the problems of having to enumerate the available fonts (which may vary from machine to machine). That type of enumeration is not really suitable for a programmatic interface. Selecting by font family guarantees a font of the requested generic type will always be available, regardless of the Windows configuration of a particular machine.

FontName
Select font name

Property Value
Text

Description
Selects the font name to be associated with a font family.

As you know, ChartBuilder selects fonts by family (Roman, Swiss or Modern) using the **FontFamily** property applied to the different text components of your graph. By default ChartBuilder will ask Windows for a TrueType font within the selected family (Times New Roman, Arial & Courier New respectively). However, if you wish, you can override these defaults with your own font names. For example, you could make ZapfChancery the Modern family font by setting FontFamily to Modern and setting FontName to "ZapfChancery". Subsequently any text component of your graph displayed with FontFamily set to Modern will use the ZapfChancery font. There is no requirement for the font name to actually belong to the family. If you override the defaults in this way you are using FontFamily simply as a selection mechanism for choosing between three different named fonts rather than using it in a formal Windows sense. Note that font names are **not** case-sensitive.

If the font name you have associated with a font family does not exist then ChartBuilder will ask Windows for any font belonging to the family. In this case FontFamily takes on a formal meaning. Windows will supply whatever it decides is most appropriate, based on the other requested font characteristics (family, style & size).

FontName is really a 3-element array. At design time the font name displayed is the one associated with the current setting of FontFamily. At run time you should use the true property array **FName(i)** to set font names. Note that FName(i) is a 0-based property array ie. FName(0) refers to the first element (which corresponds to the Roman family).

For example graph1.FName(1) = "Helvetica" sets the font name associated with the Swiss family to Helvetica.

FontSize
Select font size

Property Value
Integer in range 50 - 500 (% of system font size), default value depends on **FontUse**

Description
Determines the approximate font size in which the text specified by the current setting of **FontUse** will be displayed.

ChartBuilder uses FontSize as a guide rather than an absolute setting. It uses FontSize as its starting point, and will if necessary reduce it in order to fit the text into the space available.

The default values for FontSize are:

Graph Title	200%
Other Titles	150%
Labels	100%
Legend	100%

FontStyle
Select font style

Property Value
0 - Default
1 - Italic
2 - Bold
3 - Bold Italic
4 - Underlined
5 - Underlined Italic
6 - Underlined Bold
7 - Underlined Bold Italic

Description
Selects the style (italic, bold and/or underlined) in which the text specified by the current setting of **FontUse** will be displayed.

FontUse
Select graph text component for font selection

Property Value
0 - Graph Title (default)
1 - Other Titles
2 - Labels
3 - Legend
4 - All Text

Description
Selects the graph text component to which **FontFamily**, **FontStyle** and **FontSize** will be applied. These three properties are really arrays, each array holding four values (one for each of the graph text components). By selecting a particular text component using FontUse, you can then set the font family, style and/or size for that component.

If you select 'All Text' you can set values for the font family, style and/or size which then will be applied to all the text components ie. it is equivalent to selecting and setting each of the components in turn. It gives you a quick way in which to set all the text to, for example, a Swiss font. You can then select the individual components to make further changes. Note that at design time the font family, style and size values you see on the Property Bar when you have selected 'All Text' are the values for the Graph Title (which may or may not be the settings of the other components, according to whether individual changes have been made).

Foreground
Set foreground color

Property Value
0 - Black
1 - Blue
2 - Green
3 - Cyan
4 - Red
5 - Magenta
6 - Brown
7 - Light Grey
8 - Dark Grey
9 - Light Blue
10 - Light Green
11 - Light Cyan
12 - Light Red
13 - Light Magenta
14 - Yellow
15 - White
16 - Auto Black/White (default)

Description
The foreground color of the ChartBuilder control may be chosen from the above list.

The foreground color is the color used by ChartBuilder for its titles, labels, legends and axes. The colors of bars, pie slices etc are determined by the **ColorData** property.

By default ChartBuilder will automatically use black or white as its foreground, dependent on the background color set. It will pick whichever gives the best contrast.

GraphCaption
Graph Control Caption text

Property Value
Text

Description
This property accepts a single line of text which is then displayed in the ChartBuilder control when **DrawMode** = 1 (Clear).

Example
Graph1.**Caption** = "Graphics Server"
Graph1.**DrawMode** = 1

This displays a box with "Graphics Server" in the centre. ChartBuilder automatically chooses an appropriate font size. The colors of the text and the background can be selected using the **Foreground** and **Background** properties.

GraphData (2D Array)
Data values to be graphed

Property Value
Any user-supplied number of type **REAL** (positive/negative/integer/decimal)

Description
This property is used to set the data to be graphed.

Since this is a two-dimensional array property, the array element you set is determined by the current value of the **ThisPoint** and **ThisSet** properties.

When entering data, you may make use of the **AutoInc** property. If **AutoInc** is On, then every time you set a new value, the **ThisPoint** counter is automatically incremented. When it reaches its maximum value (**NumPoints**), the **ThisSet** counter is incremented and **ThisPoint** reset to 1. If **ThisSet** reaches its maximum value (**NumSets**) then it also is reset to 1. At design time this allows you enter data quickly, at run time it simplifies your programming.

Example 1
```
Graph1.ThisSet = 1
for i% = 1 to Graph1.NumSets
        Graph1.ThisPoint = 1
        for j% = 1 to Graph1.NumPoints
                Graph1.GraphData = j%*i%
                If Graph1.ThisPoint < Graph1.NumPoints Then
                        Graph1.ThisPoint = Graph1.ThisPoint + 1
                End If
        next j%
        If Graph1.ThisSet < Graph1.NumSets Then
                Graph1.ThisSet = Graph1.ThisSet + 1
        End If
next i%
Graph1.DrawMode = 2
```

Example 2
```
Graph1.AutoInc = 1
For i% = 1 To (Graph1.NumSets * Graph1.NumPoints)
    Graph1.GraphData = Graph1.ThisPoint * Graph1.ThisSet
Next i%
Graph1.DrawMode = 2
```

The resulting graphs from both examples are the same.

GraphStyle

Select Graph Style

Property Value

0 <= **GraphStyle** <= 7, depending on GraphType (see below)

Description

GraphType	GraphStyle	Notes
Pie	0 - Default 1 - No Label Lines 2 - Colored Labels 3 - Colored Labels without Lines 4 - % Labels 5 - % Labels without Lines 6 - % Colored Labels 7 - % Colored Labels without Lines	Default: lines joining labels to pie. If LabelText values are set, then those labels are used, otherwise the data values are used as labels.
Bar	0 - Default 1 - Horizontal 2 - Stacked 3 - Horizontal Stacked 4 - Stacked % 5 - Horizontal Stacked % 6 - Z-Clustered 7 - Horizontal Z-Clustered	Default: vertical bars, clustered if NumSets > 1. If NumSets = 1, then the bars will be differently colored. If NumSets > 1, then the sets will be differently colored. (3D only) (3D only)
Gannt	0 - Default 1 - Spaced Bars	Default: non-spaced bars. Spaced bars have a small gap between each other.
Line & Polar	0 - Default 1 - Symbols 2 - Sticks 3 - Sticks and Symbols 4 - Lines 5 - Lines and Symbols 6 - Lines and Sticks 7 - Lines and Sticks and Symbols	Default: lines. Thick or patterned lines are created by setting the ThickLines or PatternedLines properties to 1 (on).
Area	0 - Default 1 - Absolute 2 - Percentage	Default: stacked data sets. Absolute means relative to the X axis. Percentage shows data sets as percentage of total.
Scatter	0 - Default 1- Curve 2 - Symbols 3 - Curve & Symbols	Scatter graph needs XPosData. Only graph to have curve fitting.
HLC	0 - Default 1 - No Close Bar 2 - No High-Low Bars 3 - No Bars	Default: all bars shown.
Tape	0 - Default	

Bubble	0 - Default	Bubble chart needs 2-D XPosData.

GraphTitle
Main title at top of graph

Property Value
Text

Description
This property will place a the text string at the top of the graph.

Example

```
graph2.randomdata = 1
If graph2.GraphTitle = "" Then
        graph2.GraphTitle = "Graph Title"
Else
        graph2.GraphTitle = ""
End If
graph2.DrawMode = 2
```

GraphType
Select Graph Type

Property Value
0 - None
1 - 2D Pie
2 - 3D Pie
3 - 2D Bar (default)
4 - 3D Bar
5 - Gantt
6 - Line
7 - Log/Lin
8 - Area
9 - Scatter
10 - Polar
11 - HLC
12 - Bubble
13 - Tape
14 - 3D Area

Description
Choose the type of graph to best show your data.

For each graph type there are many unique style options (see **GraphStyle**), in addition to all the generic options.

GridStyle
Place Grids on the axes

Property Value
0 - None (default)
1 - Horizontal
2 - Vertical
3 - Both

Description
This places reference grids on the graph axes.

Note that, for Polar graphs, the Horizontal axes are concentric circles,and Vertical axes are radial lines ("spokes").

HCtlWin
Handle of Graph Control window

Property Value
Windows handle

Description
HCtlWin is a read-only run time property which provides the Windows handle of the Visual Basic Graph Control window. The Graphics Server window (see **HWin**) in which your graph is displayed is a child of this window.

Hot
Enable/disable hot graphs

Property Value
0 - Off (default)
1 - On

Description
The **Hot** property controls ChartBuilder's unique **HotGraph** system. It can be set at design time or run time, but it only triggers its associated custom event **HotHit** at run time.

If you set **Hot** to 1 (On), then when your graph is next drawn (at run time) all the bars, pie-slices, lines, points etc (depends upon **GraphType**) become hot and will respond to being clicked on. The **HotHit** event is then triggered, and will provide you with the data set and point of the data item that's been hit. This simple, but very powerful, facility enables you to create all kinds of interesting applications - your graphs can be used as sophisticated push buttons for navigating around databases etc, entirely graphically!

When you enable the HotGraph system, the standard Visual Basic mouse events (Click, DblClick etc) are disabled.

You can turn **Hot** off any time you wish, but don't forget that turning it back on again will only re-enable **HotHit** events if you also re-draw the graph.

Note that there is no distinction between single and double clicks within the HotGraph system.

Hwin
Handle of graph window

Property Value
Windows handle

Description
HWin is a read-only run time property which provides the Windows handle of the Graphics Server-created window in which your ChartBuilder graph is displayed.

ImageFile
File name for image output

Property Value
FileSpec (Text)

Description
Sets a file name to which the bitmap or metafile will be written when **DrawMode** is set to 6. If a path is not specified, then the current directory will be used. The appropriate extension (.BMP or .WMF) will be appended automatically according to whether DrawMode was set to 3 (Blit), in which case a Bitmap will be created, otherwise a Metafile will be created.

IndexStyle
Set the data array index style

Property Value
0 - Standard (default)
1 - Enhanced

Description
The standard way in which ChartBuilder's one-dimensional arrays are accessed is via the **ThisPoint** property, regardless as to whether the the array is being applied to data sets or to data points. Some arrays are always applied to data sets (**SymbolData**), some always to data points (**LabelText**), but some (**ColorData**, **ExtraData**, **LegendText** & **PatternData**) vary according to context - for Pie Charts and Bar Graphs with only one data set they are applied to data points, otherwise to data sets. ChartBuilder's two-dimensional arrays (**GraphData** & now **XPosData**) are always accessed via **ThisSet** and **ThisPoint** together.

In order to achieve consistency regardless of graph type, we originally odopted the approach of accessing all the one-dimensional arrays through **ThisPoint**. This remains the default.

However, some of our users have found this confusing, so we have provided an enhanced way of accessing the arrays which **is** context-sensitive, and is thus more intuitive to use. However, be careful when writing your programs - make sure you are aware of your graph type and the number of your data sets when setting up your array data.

Set **IndexStyle** = 1 to activate this enhanced style. Note that it has effect at both design time and run time.

When the enhanced **IndexStyle** is in effect, ChartBuilder's arrays are accessed as follows:

GraphData	ThisSet & ThisPoint (2-D array)
ColorData	ThisSet or ThisPoint
ExtraData	ThisSet or ThisPoint
LabelText	ThisPoint
LegendText	ThisSet or ThisPoint
PatternData	ThisSet or ThisPoint
SymbolData	ThisSet
XPosData	ThisSet & ThisPoint (2-D array)

ThisSet or **ThisPoint** is used according to the following rule:
If the current graph type is a Pie Chart (2- or 3-D), a Bubble Chart or a single data set Bar Graph (2- or 3-D), then **ThisPoint** is used, otherwise **ThisSet** is used. This is because Pie Charts, Bubble Charts and single data set Bar Graphs function differently from other graph types - they display legends per point rather than per data set.

If you use the **AutoInc** facility, then **IndexStyle** will probably be of no concern to you. **AutoInc** increments **ThisSet** and/or **ThisPoint** correctly irrespective of **IndexStyle**.

Note also that this issue is only relevant to setting up the data arrays. Once the data arrays have been created, graphs are drawn as normal, irrespective of **IndexStyle**.

Example
```
Graph1.GraphType = 6                    ' line graph
Graph1.IndexStyle = 1                   ' enhanced index style

For i% = 1 To Graph1.NumSets
    Graph1.ThisSet = i%
```

```
    For j% = 1 To Graph1.NumPoints
        Graph1.ThisPoint = j%
        Graph1.GraphData = <your data value>
        Graph1.XPosData = <your data value>
    Next
Next

For i% = 1 To Graph1.NumSets
    Graph1.ThisSet = i%                         ' use ThisSet as index
    Graph1.LegendText = "Data set " + Str$(i%)
    Graph1.ExtraData = <your data value>
    Graph1.ColorData = <your data value>
    Graph1.PatternData = <your data value>
    Graph1.SymbolData = <your data value>
Next

For i% = 1 To Graph1.NumPoints
    Graph1.ThisPoint = i%
    Graph1.LabelText = "Data point " + Str$(i%)
Next

Graph1.DrawMode = 2
```

OR...

```
Graph1.GraphType = 6                            ' line graph
Graph1.IndexStyle = 0                           ' standard index style

For i% = 1 To Graph1.NumSets
    Graph1.ThisSet = i%
    For j% = 1 To Graph1.NumPoints
        Graph1.ThisPoint = j%
        Graph1.GraphData = <your data value>
        Graph1.XPosData = <your data value>
    Next
Next

For i% = 1 To Graph1.NumSets
    Graph1.ThisPoint = i%                       ' use ThisPoint as index
    Graph1.LegendText = "Legend " + Str$(i%)
    Graph1.ExtraData = <your data value>
    Graph1.ColorData = <your data value>
    Graph1.PatternData = <your data value>
    Graph1.SymbolData = <your data value>
Next

For i% = 1 To Graph1.NumPoints
    Graph1.ThisPoint = i%
    Graph1.LabelText = "Label " + Str$(i%)
Next

Graph1.DrawMode = 2
```

Label (run time array)

Specify label text

Property Value

Same as LabelText

Description

Label(i) provides an alternative way of accessing the **LabelText** array at run time. See description of Color(i) for more information.

LabelEvery
Label every nth data point

Property Value
Integer in range 1 - 1000, default value 1

Description
Selects the frequency with which labels are displayed on the X axis.

LabelEvery only has effect when **XPosData** is **not** present. Note that this means that LabelEvery never has any effect on Scatter graphs and Bubble charts, which always have XPosData (either user-entered or generated on your behalf).

For example, imagine you have a graph with 5 points and that LabelText has been set to "Jan", "Feb", "Mar", "Apr" & "May".

LabelEvery = 1 means that all 5 labels are displayed as normal.
LabelEvery = 2 means that the 1st, 3rd & 5th labels are displayed ie. "Jan", "Mar" & "May".
LabelEvery = 3 means that the 1st & 4th labels are displayed ie. "Jan" & "Apr".

Labels
Enable/disable axis labels

Property Value
0 - Off (default)
1 - On
2 - X labels only
3 - Y labels only

Description
Selects whether labels are displayed. You can turn labels on/off separately for the X & Y axes.

This option operates independently of the **Ticks** option.

LabelText (Array)
Provide Labels for the axes

Property Value
One text string for each data point

Description
This property allows label text to be entered. If no text has been entered, then the labels will show the value of **ThisPoint** for all graphs except Pie charts, which will show the magnitude of the slices.

Since this is an array property, the array element you set is determined by the current value of the **ThisPoint** property.

When entering text, you may make use of the **AutoInc** property. If **AutoInc** is On, then every time you set a new string, the **ThisPoint** counter is automatically incremented. At design time this allows you enter data quickly, at run time it simplifies your programming.

LeftTitle
Title at left of graph

Property Value
Text

Description
The text string provided will be placed on the left of the graph parallel to the horizontal axis. This property is ignored for Pie charts.

Example
```
If graph2.LeftTitle = "" Then
        graph2.LeftTitle = "Left Title"
Else
        graph2.LeftTitle = ""
End If
graph2.DrawMode = 2
```

Legend (run time array)
Specify legend text

Property Value
Same as LegendText

Description
Legend(i) provides an alternative way of accessing the **LegendText** array at run time. See description of Color(i) for more information.

LegendStyle

Select monochrome or colored legends

Property Value

0 - Monochrome (default)
1 - Color

Description

This property gives the option of the legend text taking the same color as the data it is representing. This is in addition to the colored symbols or patterns.

LegendText (Array)
Provide Legends

Property Value
Text

Description
This property allows you to enter text for legends. There should be one string for each data set, except for Pie charts and Bar graphs with one data set which should have a string for each data point.

Since this is an array property, the array element you set is determined by the current value of the **ThisPoint** property.

When entering text, you may make use of the **AutoInc** property. If **AutoInc** is On, then every time you set a new string, the **ThisPoint** counter is automatically incremented. At design time this allows you enter data quickly, at run time it simplifies your programming.

LineStats

Statistics options for line & scatter graphs

Property Value

0 - None (default)
1 - Mean
2 - MinMax
3 - Mean and MinMax
4 - StdDev
5 - StdDev and Mean
6 - StdDev and MinMax
7 - StdDev and MinMax and Mean
8 - BestFit
9 - BestFit and Mean
10 - BestFit and MinMax
11 - BestFit and MinMax and Mean
12 - BestFit and StdDev
13 - Bestfit and StdDev and Mean
14 - Bestfit and StdDev and MinMax
15 - All

Description

The LineStats property allows statistics lines to be superimposed on the graph. This property is valid for Line, Log/Lin & Scatter graphs only.

MousePointer

Set mouse pointer shape

Property Value

0 - Arrow
1 - I-Beam
2 - Hourglass
3 - Crosshair
4 - Up Arrow
5 - Size
6 - Icon

Description

This property sets the mouse pointer shape. It can be set at design time or run time, but only has effect at run time when the graph window is enabled ie. when either the **Hot** property or the **SDKMouse** property is set On. The mouse pointer takes on the selected shape when the pointer is within the graph window.

NumPoints
Data points per set

Property Value
Minimum value 2
Default value 5
Maximum value of NumPoints * NumSets is 3800.

Description
This property specifies the number of data points in each data set.

NumPoints can be changed at any time. If you reduce **NumPoints**, then any excess array data is discarded. If you increase **NumPoints**, then additional, null-value data is created.

NumSets
Number of data sets

Property Value
Minimum value 1
Default value 1
Maximum value of NumPoints * NumSets is 3800.

Description
Specifies the number of data sets to be graphed (default = 1).

NumSets can be changed at any time. If you reduce **NumSets**, then any excess array data is discarded. If you increase **NumSets**, then additional, null-value data is created.

Pie charts will only use the first data set, even if **NumSets** > 1.

Palette
Set color palette

Property Value
0 - Default
1 - Pastel
2 - Grayscale

Description
This property selects the color palette to be used. Individual colors are selected as normal using the standard set of 16 color names, but the colors they refer to come from the selected palette. Currently only 2 alternate palettes are available - Pastel & Grayscale.

Pattern (run time array)
Select fill/line style

Property Value
Same a PatternData

Description
Pattern(i) provides an alternative way of accessing the **PatternData** array at run time. See description of Color(i) for more information.

PatternData (Array)
Select line styles or fill patterns

Property Value
0 <= **PatternData** <= 15

Description
Pattern data - one value per Set (or per point for pie or bar charts with **NumSets** = 1).
Selects pattern for solid fills, line pattern for patterned lines, or line thickness (in pixels) for thick lines.

Since this is an array property, the array element you set is determined by the current value of the **ThisPoint** property.

When entering data, you may make use of the **AutoInc** property. If **AutoInc** is On, then every time you set a new value, the **ThisPoint** counter is automatically incremented. At design time this allows you enter data quickly, at run time it simplifies your programming.

PatternedLines
Select line style

Property Value
0 - Off (default)
1 - On

Description
When On, gives dotted lines of pattern 1 unless **PatternData** is set. See **PatternData** for the different pattern styles.

Picture
Get Picture handle

Property Value
Read-only run-time picture handle.

Description
This is a read-only property, only available at run time. It can be used to pass a graph image direct to a picture control.

Example
picture1.picture = graph1.picture

This will put a copy of the graph currently displayed in graph1 into picture1. If picture1 has a different aspect ratio from graph1 then the graph image will be stretched/compressed accordingly.

PrintInfo

Specify print control information

Property Value

PrintInfo(1) = VB Printer Device Context (set = Printer.hDC).
PrintInfo(2) = X coordinate of top left of graph.
PrintInfo(3) = Y coordinate of top left of graph.
PrintInfo(4) = Width of graph.
PrintInfo(5) = Height of graph.
PrintInfo(6) = Scale Left (set = Printer.ScaleLeft).
PrintInfo(7) = Scale Top (set = Printer.ScaleTop).
PrintInfo(8) = Scale Width (set = Printer.ScaleWidth).
PrintInfo(9) = Scale Height (set = Printer.ScaleHeight).

Description

You can use the PrintInfo run time array to set up the information required to make ChartBuilder print a graph at high resolution on the same sheet of paper as other Visual Basic printed output.

You can print a graph in three ways:

1. As part of a form at run time using the VB **PrintForm** method. This is in effect a screen dump and does not take advantage of the resolution of your printer.

2. Using ChartBuilder's **DrawMode** property at design time. DrawMode = 5 prints your graph on its own sheet of paper, centred at the top of the page, at the full resolution of your printer.

3. Using ChartBuilder's **DrawMode** property at run time. Unless you have set up the PrintInfo array, DrawMode = 5 has the same effect as at design time.

   However, if you have set up the PrintInfo array, your graph will be printed into the VB Printer Object's Device Context. This means that it will appear on the same page as any other printed output you have created using the Printer Object **at the full resolution of your printer**.

   To do this you must set PrintInfo(1) = Printer.hDC and PrintInfo(6)-(9) to the Printer Object's scale factors. You can then set the location and size of the printed graph using the same units as for your other printed output. If necessary, ChartBuilder will stretch/compress the graph to fit the specified size. If you leave both PrintInfo(4) & (5) = 0, then ChartBuilder will print the graph at the actual size it appears on your screen. If you set PrintInfo(4) but leave PrintInfo(5) = 0, then ChartBuilder will use your specified width and calculate the height such that the graph's aspect ratio is maintained. Similarly, you can set PrintInfo(5) but leave PrintInfo(4) = 0.

   Using this technique you can print multiple graphs on a single sheet of paper surrounded by other Visual Basic text and graphics.

PrintStyle
Select Print style options

Property Value
0 - Monochrome
1 - Color
2 - Monochrome with border
3 - Color with border

Description
Selects the print style options when printing the control (**DrawMode** = 5).

The default option will temporarily convert the **DrawStyle** to Monochrome before printing.
If you are using a color printer, or have a printer capable of printing grey scales, then you will need to set **PrintStyle = 1**.

Using these options in conjunction with **DrawMode = 5**, the graph is printed via Graphics Server, which will give a hardcopy graph printed to the best of the ability of your printer, rather than the bitmap image generated by the **PrintForm** command.

QuickData
Get/Set the GraphData array in a single operation

Property Value
String consisting of tab-delimited, numeric values representing the two-dimensional **GraphData** array.

Description
This is a run time-only property, designed to allow the entire **GraphData** array to be set (or retrieved) in a single operation.

The format of the string is simple. Each point within a data set is separated by a TAB character (chr$(9)), and each data set is separated by a CR+LF (chr$(13) + chr$(10)), as follows:

Set1, Point1 TAB Set1, Point2 TAB Set1, Point3 CR LF
Set2, Point1 TAB Set2, Point2 TAB Set2, Point3 CR LF
Set3, Point1 TAB Set3, Point2 TAB Set3, Point3 CR LF

This is the format used by the Grid control's **Clip** property. The **QuickData** property can thus be used for simplifying the exchange of numeric data between the two controls.

Example
graph1.QuickData = grid1.Clip

When using **QuickData** to set the **GraphData** array, **NumPoints** and **NumSets** are set automatically, according to the number of points and sets within the **QuickData** string.

If the format of the string is incorrect (for example, the data sets do not contain the same number of points as each other) then an error will be reported and the **GraphData** array, **NumPoints** and **NumSets** will not be set.

QuickData will always contain at least one data set with at least two points.

RandomData
Generate random test data

Property Value
0 - Off
1 - On (default)

Description
When this property is On, the control will generate random data to be graphed. This is mainly of use at design time, when, as the designer, you want to see how the graph will appear at run time.

RandomData is automatically set Off if **GraphData** is present. However, you can override this if you wish by setting **RandomData** back On. Setting it Off again reinstates the **GraphData** values.

Note
The random numbers generated are never negative. If you want to see the effect of negative values you must enter your own.

SDKInfo

Get ChartBuilder Graphing Information

Property Value

SDKInfo(1) = X axis max (your data units)
SDKInfo(2) = X axis min (your data units)
SDKInfo(3) = Y axis max (your data units)
SDKInfo(4) = Y axis min (your data units)
SDKInfo(5) = X axis length (Graphics Server units)
SDKInfo(6) = Y axis length (Graphics Server units)
SDKInfo(7) = X origin (Graphics Server units)
SDKInfo(8) = Y origin (Graphics Server units)
SDKInfo(9) = Label font size (% of system font)

Description

The SDKInfo run time array provides internal Graphics Server information about the last graph displayed. It is intended primarily for Graphics Server SDK users, but can be used without the SDK to relate mouse clicks back to the data on which the graph was based.

Example

The following example displays, in your data units, the XY values of points clicked (with the left mouse button) on your graph. All you need to do in addition to the above is to set **SDKMouse** = 1 to enable the **SDKPress** event.

```
Sub Graph1_SDKPress (PressStatus As Integer, Pressx As Double, Pressy As Double)

If PressStatus = 1 Then
        xdata = (graph1.sdkinfo(1) - graph1.sdkinfo(2)) *
                        (Pressx - graph1.sdkinfo(7)) / graph1.sdkinfo(5) +
                        graph1.sdkinfo(2)
        ydata = (graph1.sdkinfo(3) - graph1.sdkinfo(4)) *
                      (Pressy - graph1.sdkinfo(8)) / graph1.sdkinfo(6) +
                      graph1.sdkinfo(4)
        label1.caption = Str$(Int(xdata + .5)) + " " + Str$(Int(ydata + .5))
End if

End Sub
```

SDKMouse

Enable/disable SDKHit and SDKPress custom events

Property Value

0 - Off (default)
1 - On

Description

By setting the property SDKMouse = 1 (On), the **SDKHit**, **SDKPress** and **SDKTrack** events are enabled. These are mouse events with which you can monitor for mouse clicks/movement in your graph window. Without the Graphics Server SDK these events will probably not be of much use to you (but see the SDKHit, SDKPress & SDKTrack descriptions below).

When you set SDKMouse On, the standard Visual Basic mouse events (Click, DblClick etc) are disabled. Note also that the ChartBuilder **HotGraph** system takes priority over SDK mouse events (they cannot both be active at the same time).

SDKPaint

Enable/disable SDKPaint custom event

Property Value

0 - Off (default)
1- On

Description

By setting the property SDKPaint = 1 (On), the **SDKPaint** event is enabled. This event is intended primarily for users of the Graphics Server SDK, but may be of use to you even if you do not have a copy of the SDK. When enabled the SDKPaint event occurs after each redraw (like the Picture control's **Paint** event).

SeeThru

Select 'See-thru graph' option

Property Value
0 - Off (default)
1 - On

Description
When this property is On, the graph background is not cleared. Instead, whatever was there before shows through. This can be used to draw a graph over a picture control containing a bitmap to create special effects.

We have made this is a run time-only property, because at design time its effect can be rather confusing! Also, in order to function correctly, some Visual Basic programming must be done (see example below). Otherwise, the graph will not be re-drawn if it is covered and then un-covered by another window.

Example
At design time create a picture (Picture1) and then create a graph (Graph1), **NOT** as a child of the picture but direct on your form, and then move it over the top of the picture, making sure the graph does not entirely cover the picture (leave a little border all the way round). This is to ensure the picture receives paint messages. Don't forget to set the **BorderStyle** property to 'none', otherwise your see-thru graph will still have a black line around it. Note the slight differences in the code required between VB1 & VB2.

```
Dim flag As Integer

Sub Form_Load ()
flag = 0
picture1.ZOrder 0                    ' bring the picture box to the top (VB2 only)
graph1.SeeThru = 1
End Sub

Sub Picture1_Paint ()
If flag = 1 Then
    flag = 0
    picture1.Refresh                 ' refresh the picture box (VB1 only) OR...
    picture1.ZOrder 0                ' bring the picture box to the top (VB2 only)
    graph1.Refresh
Else
    flag = 1
End If
End Sub
```

The effect of this is that when the picture receives a paint message, it refreshes both itself and the graph, ensuring the graph is still on top of the picture with the picture showing through. The 'flag' is necessary to prevent a re-entrant loop (picture1.Refresh itself fires a Paint event).

Symbol (run time array)
Select symbols for display

Property Value
Same as SymbolData

Description
Symbol(i) provides an alternative way of accessing the **SymbolData** array at run time. See description of Color(i) for more information.

SymbolData (Array)
Select Symbols for line graphs

Property Value
0 <= **SymbolData** <=9

Description
Selects symbols to be used for Lin, Log/Lin, Scatter and Polar graphs. One symbol to be selected per data set.

Since this is an array property, the array element you set is determined by the current value of the **ThisPoint** property.

When entering data, you may make use of the **AutoInc** property. If **AutoInc** is On, then every time you set a new value, the **ThisPoint** counter is automatically incremented. At design time this allows you enter data quickly, at run time it simplifies your programming.

SymbolSize
Select symbol size

Property Value
Integer in range 10 - 1000 (% of standard symbol size). Default is 100%.

Description
Determines the size of displayed symbols in line and scatter graphs.

In the case of 'hot' line, scatter and HLC graphs, SymbolSize affects the size of the hot regions regardless as to whether symbols are actually displayed or not. You an use this facility to create the desired level of click resolution. Note that a large value of SymbolSize may cause the hot regions to overlap one another.

ThickLines
Select thick lines

Property Value
0 - Off (default)
1 - On

Description
When On, gives lines 3 pixels thick unless **PatternData** is entered. If **DrawStyle** = 0 (**Monochrome**), the line thicknesses between 2 and 7 will be selected.

ThisPoint
Current data point

Property Value
1 <= **ThisPoint** <= **NumPoints**

Description
ThisPoint sets the current point number manually, so that a particular data point may be changed (this will override the **AutoInc** setting).

Example
```
Graph1.Numpoints = 5
Graph1.NumSets = 1
Graph1.AutoInc=1
for i% = 1 to 5
        Graph1.GraphData = i%
next i%
Graph1.ThisPoint = 3
Graph1.GraphData = 10
Graph1.GraphType = 4
Graph1.DrawMode = 2
```

ThisSet
Current data set

Property Value
1 <= **ThisSet** <= **NumSets**

Description
ThisSet sets the current Set number manually, so that a particular data Set may be changed (this will override the **AutoInc** setting).

This gives the ability to absolutely address any individual data point when dealing with multiple data sets.

Example
```
Graph1.Numpoints = 5
Graph1.NumSets = 3
Graph1.AutoInc=1
For i% = 1 To Graph2.NumPoints * Graph2.NumSets
        Graph1.GraphData = 5
next i%
Graph1.ThisSet = 2
Graph1.ThisPoint = 3
Graph1.GraphData = 10
Graph1.GraphType = 4
Graph1.DrawMode = 2
```

TickEvery
Tick every nth data point

Property Value
Integer in range 1 - 1000, default value 1

Description
Selects the frequency with which ticks are displayed on the X axis.

TickEvery only has effect when **XPosData** is **not** present. Note that this means that TickEvery never has any effect on Scatter graphs and Bubble charts, which always have XPosData (either user-entered or generated on your behalf).

A special effect of TickEvery is that when **NumPoints** is less than TickEvery, the X axis of your graph will be extended to the value of TickEvery. Also, since there must always be an integral number of ticks, the X axis will, if necessary, be extended to a multiple of TickEvery.

For example, if NumPoints = 127 and TickEvery = 50, then the X axis will be extended to 150.

Ticks
Enable/disable axis ticks

Property Value
0 - Off (default)
1 - On
2 - X ticks only
3 - Y ticks only

Description
Selects whether ticks are displayed. You can turn ticks on/off separately for the X & Y axes.

This option operates independently of the **Labels** option. Note that it has no effect on 3-D graphs drawn with a cage effect.

XAxisMax
Specify X axis maximum value

Property Value
Real number

Description
Specifies the maximum X axis value.

XAxisMax is used in combination with **XAxisMin** & **XAxisTicks** and only has effect when **XAxisStyle** = 2 (User-defined). For a description of its use see **XAxisStyle** below.

XAxisMin
Specifiy X axis minimum value

Property Value
Real number

Description
Specifies the minimum X axis value.

XAxisMin is used in combination with **XAxisMax** & **XAxisTicks** and only has effect when **XAxisStyle** = 2 (User-defined). For a description of its use see **XAxisStyle** below.

XAxisPos
Select X axis position

Property Value
0 - Default
1 - Top
2 - Bottom

Description
Selects the position of the X axis on your graph.

By default, the X axis is positioned automatically by ChartBuilder according to your **GraphData** values. When your values are all positive, the X axis will be at the bottom. However, if you were to specify all-negative **GraphData** values, the X axis would be at the top.

XAxisPos allows you to override ChartBuilder's automatic action and place the X axis wherever you wish.

XAxisStyle
Select X axis method of scaling & ranging

Property Value
0 - Default
1 - Variable Origin
2 - User-defined

Description
Selects the method for X axis scaling & ranging.

Exactly how XAxisStyle affects your graph's appearance depends upon whether you have set any **XPosData** values. Note that Scatter graphs and Bubble charts always have XPosData (either user-entered or generated on your behalf).

Without XPosData
Without XposData your graph represents a series of **NumPoints** data points (if **NumSets** > 1 then a number of sets of such points).

In this situation XAxisStyle = 1 (Variable Origin) has no effect.

XAxisStyle = 2 (User-defined) enables you to specify the number of ticks for the X axis using **XAxisTicks** (in this situation **XAxisMax** & **XAxisMin** have no effect). If XAxisTicks is less than NumPoints then it has no effect. However, if XAxisTicks is greater than NumPoints, then the X axis is extended to the XAxisTicks value. The main use of this feature is to create graphs where the data is incomplete. For example, you may wish to have your X axis labelled Jan-Dec but only have data available for Jan-May.

A side effect of this feature is that you may need to enter more than NumPoints labels. If XAxisStyle = 2 (User-defined) you can enter as many labels as you wish (normally ThisPoint is checked to be <= NumPoints).

With XPosData
With XposData your graph is a true XY graph. The simplest XY graph is the Scatter graph, where every point always has an explicit X,Y value. In fact, ChartBuilder allows you apply XPosData to all graph types except the Gantt Chart. When XPosData is present, XAxisStyle provides the same control over the X axis as YAxisStyle does over the Y axis.

By default, ChartBuilder automatically calculates the X axis range based on your **XPosData** values. The maximum X axis value will be a suitable value >= the maximum data value and the minimum X axis value will be 0 or, if the data includes negative values, a suitable value <= the minimum data value. This means that the X axis **always** includes the 0 origin.

When XAxisStyle = 1 (Variable Origin), ChartBuilder attempts to 'zoom in' on your **XPosData** values. The maximum X axis value will again be a suitable value >= the maximum data value, but this time the minimum X axis value will be a suitable value <= the minimum data value, whether the data includes negative values or not. This means that the X axis **may not include** the 0 origin.

The major benefit of the Variable Origin style occurs when you have **XPosData** values with a small variation around a non-zero value.

When XAxisStyle = 2 (User-defined), **XAxisMax**, **XAxisMin** & **XAxisTicks** together control exactly how the X axis is drawn. **XAxisTicks** specifies the number of ticks from the origin to **XAxisMax**. Since there must always be an integral number of ticks on an axis, ChartBuilder can sometimes override the **XAxisMin** value.

For example, if XAxisMax = 300, XAxisMin = -10 & XAxisTicks = 3, then ChartBuilder will place ticks 100

units apart and the actual XAxisMin value displayed will be -100 rather than -10.

To be precise, ChartBuilder behaves in a slightly more complex fashion than this. XAxisTicks actually specifies the number of ticks from the origin to the greater of XAxisMax & XAxisMin, regardless of sign. Thus in some situations the above can be reversed.

For example, if XAxisMax = 10, XAxisMin = -300 & XAxisTicks = 3, then ChartBuilder will make the actual XAxismax value 100 rather than 10.

XAxisTicks

Specify X axis number of ticks

Property Value

Integer in range 1 - 100, default value 1

Description

Specifies the number of ticks on the X axis.

XAxisTicks is used in combination with **XAxisMax** & **XAxisMin** and only has effect when **XAxisStyle** = 2 (User-defined). For a description of its use see **XAxisStyle**.

XPos (run time 2D array)
Specify X data

Property Value
Same as XPosData

Description
XPos(i) provides an alternative way of accessing the 2-D **XPosData** array at run time. See description of Data(i) for more information.

XPosData (2D Array)
Set independent X-variable data

Property Value
Any user-supplied number of type **REAL**

Description
XPosData provides an independent X value. It is normally used with line and scatter graphs, but can be applied to all graph types except Gantt Charts.

Since this is a 2-D array property, the array element you set is determined by the current value of the **ThisSet** and **ThisPoint** properties.

When entering data, you may make use of the **AutoInc** property. If **AutoInc** is On, then every time you set a new value, the **ThisSet** and **ThisPoint** counters are automatically incremented. At design time this allows you enter data quickly, at run time it simplifies your programming.

In ChartBuilder 1, **XPosData** was implemented as a one dimensional array. The enhancement to 2-D allows you to display multiple sets of X,Y data in Line and Scatter graph form rather than having to share a single set of X values. It is compatible with previous versions in that **XPosData** stored in the old format will be loaded as the first set of new **XPosData**.

If you have multiple sets of **GraphData**, but only store one set of **XPosData**, then ChartBuilder automatically applies the single set of **XPosData** to each set of **GraphData** ie. it behaves as it did in previous versions.

Note that 2-D XPosData only applies to Line Graphs, Scatter Graphs & Bubble Charts (for which 2 sets of XPosData are **always** required).

Example
```
Graph1.AutoInc = 0
For i% = 1 To Graph1.NumSets
    Graph1.ThisSet = i%
    For j% = 1 To Graph1.NumPoints
        Graph1.ThisPoint = j%
        Graph1.XPosData = <your data value>
    Next
Next
Graph1.DrawMode = 2
```

OR...

```
Graph1.AutoInc = 1
For i% = 1 To Graph1.NumSets
    For j% = 1 To Graph1.NumPoints
        Graph1.XPosData = <your data value>
    Next
Next
Graph1.DrawMode = 2
```

YAxisMax
Specify Y axis maximum value

Property Value
Real number

Description
Specifies the maximum Y axis value.

YAxisMax is used in combination with **YAxisMin** & **YAxisTicks** and only has effect when **YAxisStyle** = 2 (User-defined). For a description of its use see **YAxisStyle** below.

YAxisMin
Specify Y axis minimum value

Property Value
Real number

Description
Specifies the minimum Y axis value.

YAxisMin is used in combination with **YAxisMax** & **YAxisTicks** and only has effect when **YAxisStyle** = 2 (User-defined). For a description of its use see **YAxisStyle** below.

YAxisPos
Select Y axis position

Property Value
0 - Default
1 - Left
2 - Right

Description
Selects the position of the Y axis on your graph.

By default, the Y axis is positioned automatically by ChartBuilder according to your **XPosData** values. When your values are all positive, the Y axis will be on the left. However, if you were to specify all-negative **XPosData** values, the Y axis would be on the right.

YAxisPos allows you to override ChartBuilder's automatic action and place the Y axis wherever you wish.

YAxisStyle
Select Y axis method of scaling & ranging

Property Value
0 - Default
1 - Variable Origin
2 - User-defined

Description
Selects the method for Y axis scaling & ranging.

By default, ChartBuilder automatically calculates the Y axis range based on the data to be graphed. The maximum Y axis value will be a suitable value >= the maximum data value and the minimum Y axis value will be 0 or, if the data includes negative values, a suitable value <= the minimum data value. This means that the Y axis **always** includes the 0 origin.

When YAxisStyle = 1 (Variable Origin), ChartBuilder attempts to 'zoom in' on the data to be graphed. The maximum Y axis value will again be a suitable value >= the maximum data value, but this time the minimum Y axis value will be a suitable value <= the minimum data value, whether the data includes negative values or not. This means that the Y axis **may not include** the 0 origin.

The major benefit of the Variable Origin style occurs when you are graphing data with a small variation around a non-zero value. The variation will be visible, whereas with the default style such data may appear as a straight line.

When YAxisStyle = 2 (User-defined), **YAxisMax**, **YAxisMin** & **YAxisTicks** together control exactly how the Y axis is drawn. Use this style when you know something about the data you are graphing and want it presented in a certain way. For example, you may want a graph to have a Y axis range of -1000 to +1000 even though sometimes the data is entirely positive. When producing a series of graphs for comparison this can be very useful. However, be careful, because if your data exceeds the limits of your Y axis range, ChartBuilder will scale the graph according to what you have specified and will draw a graph outside the bounds of your axes. This can result in some very strange effects! Don't worry though, it won't crash your system.

YAxisTicks specifies the number of ticks from the origin to **YAxisMax**. Since there must always be an integral number of ticks on an axis, ChartBuilder can sometimes override the **YAxisMin** value.

For example, if YAxisMax = 300, YAxisMin = -10 & YAxisTicks = 3, then ChartBuilder will place ticks 100 units apart and then actual YAxisMin value displayed will be -100 rather than -10.

In fact, ChartBuilder behaves in a slightly more complex fashion than this. YAxisTicks actually specifies the number of ticks from the origin to the greater of YAxisMax & YAxisMin, regardless of sign. Thus in some situations the above can be reversed.

For example, if YAxisMax = 10, YAxisMin = -300 & YAxisTicks = 3, then ChartBuilder will make the actual YAxismax value 100 rather than 10.

YAxisTicks

Specify Y axis number of ticks

Property Value

Integer in range 1 - 100, default value 1

Description

Specifies the number of ticks on the Y axis.

YAxisTicks is used in combination with **YAxisMax** & **YAxisMin** and only has effect when **YAxisStyle** = 2 (User-defined). For a description of its use see **YAxisStyle**.

HotHitEvent (HitSet, HitPoint)

The HotHit event is triggered when a bar/pie-slice/line/point etc of a hot graph is clicked on at run time.

ChartBuilder's unique **HotGraph** system must first be enabled by setting the **Hot** property. This must be done **before** drawing your graph at run time, or alternatively at design time.

According to the graph type (see table below), the data set and/or point of the data item clicked on is provided by the HitSet & HitPoint arguments.

Graph type	**Hot item**	**Information provided**
Pie (2-D & 3-D)	Pie-slice	Point only
Bar (2-D & 3-D)	Bar	Set & Point
Gantt	Bar	Set & Point
Line	Point on line	Set & Point
Scatter	Symbol	Set & Point
Area (2-D & 3-D)	Filled area	Set only
Polar	Point on line	Point only (1 data set only)
HLC	High/Low/Close points	Set & Point
Bubble	Bubble	Point only
Tape	Tape face	Set & Point

SDKHitEvent (HitRegion)

If it has been enabled by setting the **SDKMouse** property, the SDKHit event is triggered whenever a hot region within your graph window is clicked on. If you have the Graphics Server SDK you can create as many hot regions as you wish of whatever size you wish.

HitRegion provides the number of the region hit. It refers to the number returned to you by the Graphics Server function when you originally created the hot reqion.

SDKPaintEvent ()

If it has been enabled by setting the **SDKPaint** property the SDKPaint event is triggered after every redraw. If you have the Graphics Server SDK you can insert Graphics Server DLL calls to draw additional graphical objects into your graph window. By making such calls here you can be sure your graph window is always up-to-date.

SDKPressEvent (PressStatus, PressX, PressY, PressDataX, PressDataY)

If it has been enabled by setting the **SDKMouse** property, the SDKPress event is triggered whenever the mouse is press/released in your graph window.

PressStatus provides the following button press information:

0	Button released
1	Left button pressed
2	Middle button pressed
4	Right button pressed

PressX & **PressY** provide the mouse coordinates in terms of Graphics Server units for your graph window. In this version of ChartBuilder, the height of a ChartBuilder graph window is 1000 units, the width depending on the aspect ratio of the window (this may change in later releases). **PressDataX** & **PressDataY** provide the mouse coordinates in terms of your data units (they are set to zero for Pie and Polar graphs).
Users of the Graphics Server SDK can make more extensive use of this event.

SDKTrackEvent (TrackX, TrackY, TrackDataX, TrackDataY)

If it has been enabled by setting the **SDKMouse** property, the SDKTrack event is triggered whenever the mouse is moved in your graph window.

TrackX & **TrackY** provide the mouse coordinates in terms of Graphics Server units for your graph window. In this version of ChartBuilder, the height of a ChartBuilder graph window is 1000 units, the width depending on the aspect ratio of the window (this may change in later releases). **TrackDataX** & **TrackDataY** provide the mouse coordinates in terms of your data units (they are set to zero for Pie and Polar graphs).
Users of the Graphics Server SDK can make more extensive use of this event.