LabVIEW®

Picture Control Toolkit Reference Manual

© Copyright 1994 National Instruments Corporation.
All Rights Reserved.

Part Number 320594-01
April 1994

National Instruments Corporate Headquarters
6504 Bridge Point Parkway
Austin, TX 78730-5039
(512) 794-0100
Technical support fax: (512) 794-5678

Branch Offices:
Australia 03 879 9422, Austria 0662 435986, Belgium 02 757 00 20, Canada (Ontario) 519 622 9310, Canada (Québec) 514 694 8521, Denmark 45 76 26 00, Finland 90 527 2321, France 1 48 65 33 70, Germany 089 714 50 93, Italy 02 48301892, Japan 03 3788 1921, Netherlands 01720 45761, Norway 03 846866, Spain 91 640 0085, Sweden 08 730 49 70, Switzerland 056 27 00 20, U.K. 0635 523545

Limited Warranty

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this manual is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

Trademarks

LabVIEW® is a trademark of National Instruments Corporation.

Product and company names listed are trademarks or trade names of their respective companies.

Warning Regarding Medical and Clinical Use of National Instruments Products

National Instruments products are not designed with components and testing intended to ensure a level of reliability suitable for use in treatment and diagnosis of humans. Applications of National Instruments products involving medical or clinical treatment can create a potential for accidental injury caused by product failure, or by errors on the part of the user or application designer. Any use or application of National Instruments products for or involving medical or clinical treatment must be performed by properly trained and qualified medical personnel, and all traditional medical safeguards, equipment, and procedures that are appropriate in the particular situation to prevent serious injury or death should always continue to be used when National Instruments products are being used. National Instruments products are NOT intended to be a substitute for any form of established process, procedure, or equipment used to monitor or safeguard human health and safety in medical or clinical treatment.

Contents

About This Manual

Organization of This Manual ix
Conventions Used in This Manual x
Related Documentation xii
Customer Communication xii

Chapter 1
Getting Started

Installing the LabVIEW Picture Control Toolkit 1-1
Windows 1-1
Sun 1-2
Macintosh 1-3
Basic Concepts 1-3
Coordinate System 1-7
Color Specifications 1-7
Overview of Picture VIs 1-8
Points, Lines, and Pens 1-9
Shapes 1-9
Text 1-10
Empty Picture Constant 1-11
Pixmaps 1-11

Chapter 2
Examples Overview

Example Libraries Organization 2-1
Demonstration Library 2-2
Artificial Strip Chart 2-2
Histogram Plot 2-2
Picture Waterfall Spectrum 2-2
Polar Plot 2-2
Scale Demo 2-3
Simple Smith Plot 2-3
Smith Multi Plot with Styles 2-3
Smith Plot with Zooming 2-3
Waveform & XY Plots 2-3
XY Multi Plot 2-4
XY Scatter Plots 2-4

The demosup.llb 2-4
Graph-Related Libraries 2-4
cartesn.llb 2-4
cartesn2.llb 2-5
polarplt.llb 2-5
scale.llb 2-5
smith.llb 2-5
Miscellaneous Libraries 2-6
pictutil.llb 2-6
robot.llb 2-6

Chapter 3
Picture Controls

Introduction 3-1
Picture Control 3-2
Picture Options 3-3
Picture Attributes 3-4
Picture Custom Controls 3-4
Rectangle Control 3-4
Point Control 3-5
Text Alignment Control 3-5
Font Enum Control 3-6
User-Specified Font Control 3-6

Chapter 4
Picture VIs

Picture VI Descriptions 4-2
Empty Picture 4-2
Draw Point 4-2
Move Pen 4-3
Draw Line 4-4
Draw Multiple Lines 4-5
Draw Rect 4-6
Draw Round Rect 4-7
Draw Oval 4-8
Draw Arc 4-9
Draw Grayed Out Rect 4-11
Draw Text at Point 4-12
Get Text Rect 4-14
Draw Text in Rect 4-16
Draw 1-Bit Pixmap 4-18
Draw 4-Bit Pixmap 4-19
Draw 8-Bit Pixmap 4-20

Chapter 5
Cartesian Graph VIs

Demonstration VIs 5-2
Waveform & XY Plots 5-2
XY Multi Plot 5-2
Artificial Strip Chart 5-2
Histogram Plot 5-2
XY Scatter Plots 5-2
Using the Cartesian Graph Examples as SubVIs 5-3
High-Level Cartesian Graph VIs 5-8
Plot Waveform 5-8
Plot XY 5-12
Plot Multi-XY 5-14

Chapter 6
Polar Graph VIs

Demonstration VI 6-1
Using the Polar Graph Example as SubVIs 6-2
High-Level Polar Graph VI 6-6
Polar Graph 6-6

Chapter 7
Smith Plot VIs

Demonstration VIs 7-2
Simple Smith Plot 7-2
Smith Multi Plot with Styles 7-2
Smith Plot with Zooming 7-2
Smith Plot Overview 7-2
Using the Smith Plot Examples as SubVIs 7-3
High-Level Smith Plot VIs 7-9
Normalize Smith Plot 7-9
Smith Plot 7-10
Smith Multi Plot 7-14

Chapter 8
Graph Scale VIs

Demonstration VIs 8-2
Using the Graph Scale Examples as SubVIs 8-2
High-Level Graph Scale VIs 8-5
Calc Scale Specs 8-5
Draw Scale 8-8

Appendix A
Customer Communication A-1

Glossary G-1

Index I-1

Tables

Table 6-1. Visible Section Values 6-8

Table 8-1. Scale Orientation Options if Horizontal is FALSE 8-6
Table 8-2. Scale Orientation Options if Horizontal is TRUE 8-6
Table 8-3. Draw Bar Options of the Scale 8-7

Introduction

About This Manual

The *LabVIEW Picture Control Toolkit Reference Manual* describes the LabVIEW add-on package you can use for displaying graphical images on your front panels.

Organization of This Manual

This manual is organized as follows:

- Chapter 1, *Getting Started*, describes the installation procedure and introduces you to the basic concepts of the Picture Control Toolkit by leading you through some of the examples.
- Chapter 2, *Picture Control Examples*, gives an overview of the Picture Control Toolkit examples so that you can use them to start building your applications.
- Chapter 3, *Picture Controls*, explains the controls in the **Picture** palette of the **Controls** menu.
- Chapter 4, *Picture VIs*, describes the VIs you use to create pictures that the picture indicator displays.
- Chapter 5, *Cartesian Graph VIs*, describes the VIs located in the `cartesn.llb` and `cartesn2.llb` libraries that you can use to create a Cartesian graph. These Cartesian VIs are a good starting point if you want a Cartesian graph with features that LabVIEW's XY and Waveform Graphs do not have.
- Chapter 6, *Polar Graph VIs*, describes the VIs in the `polarplot.llb` library that you can use to create a polar graph.
- Chapter 7, *Smith Plot VIs*, describes the VIs in the `smith.llb` library that you can use to create a Smith plot.
- Chapter 8, *Graph Scale VIs*, describes the VIs in the `scale.llb` library that you can use to create scales.

- The Appendix, *Customer Communication*, contains forms you can use to request help from National Instruments or to comment on our products and manuals.
- The *Glossary* contains an alphabetical list and description of terms used in this manual, including abbreviations, acronyms, metric prefixes, mnemonics, and symbols.
- The *Index* contains an alphabetical list of key terms and topics in this manual, including the page where you can find each one.

Conventions Used in This Manual

The following conventions are used in this manual:

bold	Bold text denotes menus, menu items, and VI input and output parameters.
italic	Italic text denotes emphasis, a cross reference, or an introduction to a key concept.
bold italic	Bold italic text denotes a note, caution, or warning.
`monospace`	Monospace font denotes text or characters that you enter using the keyboard. File names, directory names, drive names, sections of code, programming examples, syntax examples, and messages and responses that the computer automatically prints to the screen also appear in this font.
Warning:	***This icon to the left of bold italicized text denotes a warning, which alerts you to the possibility of damage to you or your equipment.***
Caution:	***This icon to the left of bold italicized text denotes a caution, which alerts you to the possibility of data loss or a system crash.***
Note:	***This icon to the left of bold italicized text denotes a note, which alerts you to important information.***

LabVIEW Data Types

Each VI description gives a data type picture for each input and output parameter, as illustrated in the following table.

Control	Indicator	Data Type
I16	I16	Signed 16-bit integer
I32	I32	Signed 32-bit integer
U32	U32	Unsigned 32-bit integer
abc	abc	String
TF	TF	Boolean
[I32]	[I32]	Array of signed 32-bit integers
[U8]		Array of unsigned 8-bit integers
[U32]		Array of unsigned 32-bit integers
[TF]		Array of Boolean
		Picture data
[ ]	[ ]	Array of Clusters
		Cluster
		Enumeration

Abbreviations, acronyms, metric prefixes, mnemonics, symbols, and terms are listed in the *Glossary*.

Related Documentation

The following documents contain information that you may find helpful as you read this manual:

- Your LabVIEW tutorial
- Your LabVIEW user manual

Customer Communication

National Instruments wants to receive your comments on our products and manuals. We are interested in the applications you develop with our products, and we want to help if you have problems with them. To make it easy for you to contact us, this manual contains comment and technical support forms for you to complete. These forms are in the appendix, *Customer Communication*, at the end of this manual.

Chapter

1

Getting Started

This chapter describes the installation procedure and introduces you to the basic concepts of the Picture Control Toolkit by leading you through some of the examples.

The Picture Control Toolkit, an add-on package for LabVIEW, features new controls and VIs for displaying images on your front panels. Your diagrams can create these images dynamically using a set of VIs to describe the drawing operations you want to perform.

Using this toolkit, you can create visual displays on your front panels that are not currently built in as front panel controls. For example, you can create graph displays with capabilities that are not supported by LabVIEW graph controls, such as bar plots, polar plots, and Smith charts. You can also create an animated display of objects such as a robot arm, test equipment, or a two-dimensional display of a real world process.

Installing the LabVIEW Picture Control Toolkit

Windows

The LabVIEW for Windows Picture Control Toolkit package contains a set of disks for installing the toolkit.

You can install the Picture Control Toolkit from the DOS prompt, the Windows File Manager, or with the **Run...** command from the **File** menu of the Program Manager.

1. Insert disk 1 of the Picture Control Toolkit into the 3.5-in. disk drive and run the `SETUP.EXE` program using one of the following three methods.

    a. From the DOS prompt, type *X*`:\SETUP` (where *X* is the proper drive designation).

b. From Windows, select **Run...** from the **File** menu of the Program Manager. A dialog box appears. Type *X*`:\SETUP` (where *X* is the proper drive designation).

c. From Windows, launch the File Manager. Click on the drive icon that contains the installation disk. Find `SETUP.EXE` in the list of files on that disk and double-click on it.

2. Follow the instructions on the screen. You will be prompted to insert the subsequent floppy disks.

When you install the LabVIEW for Windows Picture Control Toolkit, your LabVIEW `EXAMPLES` directory should contain a new `PICTURE` subdirectory, and your `VI.LIB` directory should contain a new `PICTURE.LLB` file. When you launch LabVIEW, the **Controls** and **Functions** menus should both contain a **Picture** option. If they do not, verify that the `PICTURE.LLB` file is in your LabVIEW `VI.LIB` directory.

Sun

You can install the LabVIEW for Sun Picture Control Toolkit as described in the following steps. You do not need root privileges to install the toolkit, but you must be able to write to the LabVIEW directory where these libraries will be installed.

1. Insert the first floppy disk into the floppy disk drive.

2. Type the following UNIX command:

```
tar xvf /dev/rfd0a INSTALL
```

3. Run the installation program by typing the following command:

```
./INSTALL
```

4. Follow the instructions on the screen.

When you install the LabVIEW for Sun Picture Control Toolkit, your LabVIEW `Examples` directory should contain a new `Picture` directory, and your `vi.lib` directory should contain a new `Picture.llb` file. If you launch LabVIEW, the **Controls** and **Functions** menus should both contain a **Picture** option. If they do not, verify that the `Picture.llb` file is in your LabVIEW `vi.lib` directory.

Macintosh

The LabVIEW for Macintosh Picture Control Toolkit comes in compressed form on floppy disks. Installing the toolkit requires approximately 3.5 MB.

Some virus detection programs may interfere with the installer program. Check the distribution disks for viruses before you begin installation. Then, turn off the automatic virus checker and run the installer. After installation, check your hard disk for viruses again and turn on the virus checker.

1. Insert disk 1 of the Picture Control Toolkit and double-click on the **Picture Installer** icon.

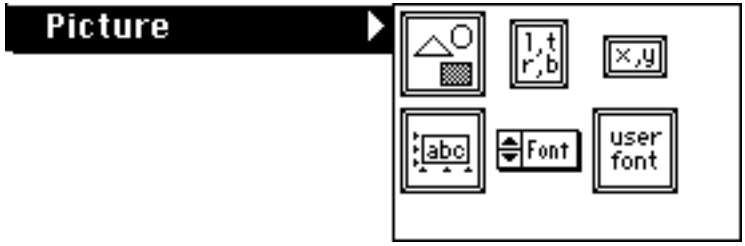
2. Drag the Picture VI library icon to the hard drive that contains your LabVIEW executable file. When you are prompted to select a destination, select your LabVIEW directory.

When you install the LabVIEW for Macintosh Picture Control Toolkit, your LabVIEW `Examples` folder should contain a new `Picture` folder, and your `vi.lib` folder should contain a new `Picture.llb` file. When you launch LabVIEW, the **Controls** and **Functions** menus should both contain a **Picture** option. If they do not, verify that the `Picture.llb` file is in your LabVIEW `vi.lib` folder.

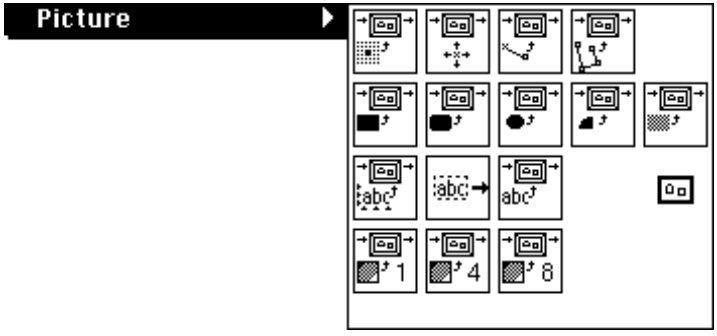
Basic Concepts

The Picture Control Toolkit adds **Picture** palettes to your **Controls** and **Functions** menus. Selecting **Picture** from the Controls palette while in the front panel window brings up the menu shown in the following illustration. You may want to select **Picture** from the menu and look at these features as you read through this section.

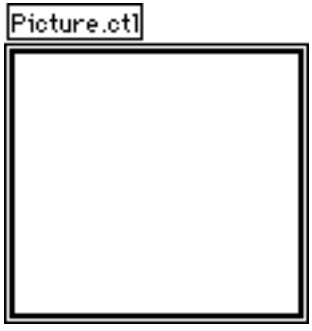
Note: ***The screens illustrated in this manual were taken in the Macintosh environment. If you are using Windows or the Sun SPARCstation, your screens will look slightly different. However, the information on the screens is the same across all three platforms, and the illustrations taken in the Macintosh environment are very similar to those in Windows and on the Sun SPARCstation.***



Selecting **Picture** from the Functions palette while in the diagram window brings up the menu shown in the following illustration.

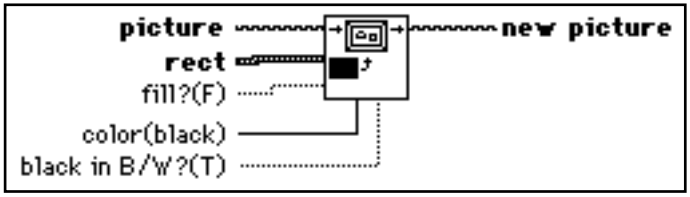


The **Picture** palette from the **Controls** menu contains a Picture control. When the control is placed on a front panel, it appears as a blank rectangular area on the screen, with a corresponding terminal on the block diagram, as shown in the following illustrations.



To display an image in a picture indicator, you must specify a set of drawing instructions. You create these drawing instructions using the VIs in the **Picture** palette of the **Functions** menu. Each VI takes a set of inputs that describe a drawing instruction. Based on these inputs, the VI creates a compact description of these specifications which you can pass to the picture indicator for display.

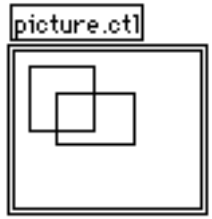
The following illustration shows the help window display for the Draw Rect VI. If you want to open this VI and look at it, it is located in `picture.llb`.

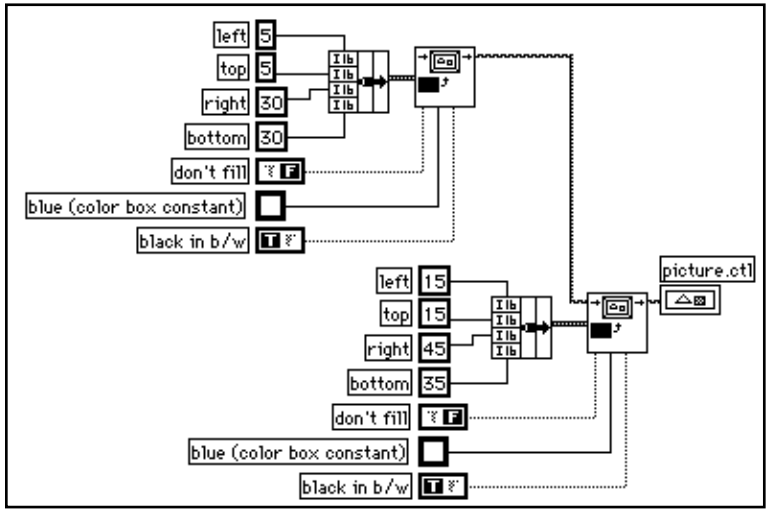


This VI takes the description of the rectangle and produces a picture as an output. A picture is not a string or an array, but a distinct data type. You can connect a picture only to a picture indicator, or to the picture input of a subVI. LabVIEW does not actually draw the rectangle until the data reaches a picture indicator or the picture control input on an open front panel.

Notice that one of the inputs to this VI is a picture. Each VI concatenates its drawing instruction to the previous drawing instruction(s) in this manner. If this input is unwired, it defaults to an empty picture.

The following illustrations of a panel and diagram show how you can use the Draw Rect VI to draw two overlapping rectangles in a picture indicator.





The previous illustration, showing how to use the Draw Rect VI to draw two overlapping rectangles, demonstrates the following:

1. You can use a picture VI to create drawing instructions for a rectangle or some other graphical object.

2. You can take an existing picture, add drawing instructions, and create a new picture that contains the objects from the original picture plus the new object.

Notice the use of color box constants to specify the color of the rectangles. The color box constant is a numeric control from the **Structs & Constants** palette of the **Functions** menu. If you click on it, you can select a color from the color menu.

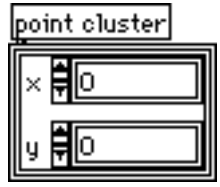
Also notice that the Boolean, black in B/W, specifies what to do if the picture is displayed on a black-and-white monitor. If you create a set of VIs that use the picture control, and you want them to work on all

monitors regardless of whether they are color or black-and-white, you must specify whether each color you draw will correspond to black or to white if shown on a black-and-white monitor.

Coordinate System

The picture control has a pixel-based coordinate system in which the origin (0,0) is the pixel located at the top left corner of the indicator. The horizontal (x) component of a coordinate increases towards the right, and the vertical (y) coordinate increases towards the bottom.

Some of the picture VIs accept a *point* as an input. A point is a cluster of two 16-bit signed integers representing the x and y coordinates of a pixel. The following illustration shows a point cluster. For convenience, a custom control for a point is included in the **Picture** palette of the **Controls** menu. See Chapter 3, *Picture Controls*, for more information on custom controls.



If a picture is too large for the indicator that displays it, the picture is *clipped* (or cropped), meaning that only the pixels that fit within the display region of the indicator are displayed. If you want to resize a control, you can use the parts window of the control editor to size the control to an exact number of pixels. If you are unfamiliar with the parts window, or need a review, see the *Creating a Custom Control* section in Chapter 4, *Introduction to Front Panel Objects*, of your LabVIEW user manual.

If you want to programmatically read the size of the control, you can use the Dimensions attribute from the attribute node. Notice that this is a read-only attribute, because you cannot resize a control at run time.

Color Specifications

Many of the picture VIs have a color input. The easiest way to create a color is to use a color box constant. Using the color box constant, you can

select a color from a popup menu. A color box constant is actually a numeric whose value specifies a color.

In the event that you want to create colors as the result of calculations, instead of through the use of color box constants, you need to understand how a color box specifies a color using a numeric value.

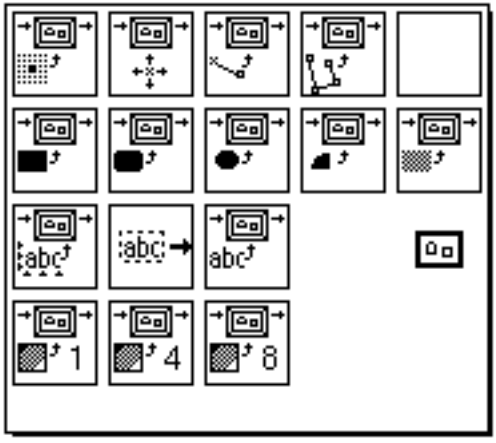
A color is represented by a 32-bit signed integer, with the lower three bytes representing the red, green, and blue components of the color.

For a range of blue colors, create an array of 32-bit integers where only the values of the low bytes change (the low byte contains the blue component).

To create a range of gray colors, you need to create an array of 32-bit integers where the red, green, and blue values of each element are the same.

Overview of Picture VIs

In addition to the Draw Rect VI, the **Picture** palette of the **Functions** menu contains several VIs you can use to draw a variety of different kinds of shapes as well as enter text. The **Picture** palette is shown in the following illustration.

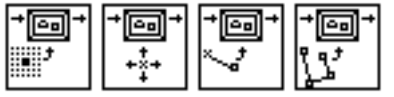


These VIs are described in detail in Chapter 4, *Picture VIs*. The picture VIs are mentioned in the following paragraphs to give you a general

overview of their capabilities. You may want to select and look at these VIs on your screen as you read through the descriptions.

Points, Lines, and Pens

The first row of the **Picture** palette, shown in the following illustration, contains VIs that draw points and lines.



Draw Point VI | Draw Line VI | Move Pen VI | Draw Multiple Lines VI

These VIs use the concept of a graphics *pen*. When you draw, the picture remembers the position of the pen. For most of the Picture VIs, you must specify absolute coordinates—that is, relative to the origin (0,0). With Draw Line VI and Move Pen VI, you can specify either absolute or relative coordinates. Relative coordinates are relative to the current location of the pen. You can use the Move Pen VI to change the location of the pen without drawing.

Only the Draw Point VI, Draw Line VI, Move Pen VI, and Draw Multiple Lines VI change the location of the pen.

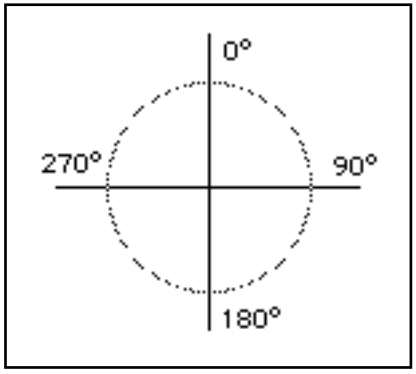
Shapes

The second row of the **Picture** palette contains VIs you can use to draw an optionally filled shape. These VIs are shown in the following illustration.

Draw Rect VI | Draw Round Rect VI | Draw Oval VI | Draw Arc VI | Draw Grayed Out Rect VI

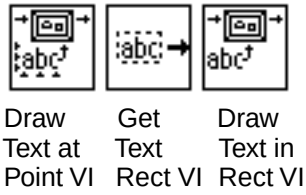
Each of these VIs draws a shape in a rectangular area of a picture. You specify a rectangle as a cluster of four values representing the left, top, right, and bottom pixels.

With the Draw Arc VI, the rectangle describes the size of an oval. Additional parameters specify the portion of the oval (the arc) you want to draw. You must specify the starting angle and the length of the arc. The following illustration is an example of how to specify this information.



Text

The third row of the **Picture** palette primarily contains VIs you can use to draw text in a picture. These VIs are shown in the following illustration.



The Get Text Rect VI does not draw any text. Instead, you can use it to calculate the size of a bounding rectangle of a string.

Empty Picture Constant

The third row of the **Picture** palette also contains the Empty Picture constant, shown in the following illustration. You use this constant whenever you need to start with or change an empty picture.

Pixmaps

The last row of the **Picture** palette contains VIs that draw *pixmaps* in a picture. A pixmap is a two-dimensional array of data, where each value is mapped to a color. The last row of the palette is shown in the following illustration.

Draw 1-Bit Pixmap VI | Draw 4-Bit Pixmap VI | Draw 8-Bit Pixmap VI

In addition to specifying the two-dimensional array of data, you also specify the color table for mapping array values to colors. The two-dimensional array is mapped to a grid of pixels. These VIs convert each value in the data array into a color by using the value as an index into the color table array.

Chapter 2

Examples Overview

This chapter gives an overview of the Picture Control Toolkit examples so that you can use them to start building your applications.

The best way to learn how to use the Picture Control Toolkit is to look at some of the examples that come with your package. Each picture example illustrates how to use the Picture VIs to create either a graph or an animation of a real-world process.

While these working examples show how you might use the picture VIs in your applications, you may decide that you can use the high-level graph VIs in your applications as they are. If you find that you want more functionality, you can customize the examples.

Caution: ***If you decide to customize an example or use it as a subVI in one of your applications, save the example in a new location with a new name. Doing this will prevent a future installation program of LabVIEW from accidentally overwriting your files.***

The examples directory contains several `.llb` files. Where possible, there is at least one example VI designed to use as a subVI in your own applications. For example, the `polarplt.llb` contains a VI called the Polar Graph VI that you can use as a subVI in your applications to create a polar plot.

Example Libraries Organization

Under the Examples directory, you will find several `.llb` files including `demosup.llb`, `cartesn.llb`, `cartesn2.llb`, `demos.llb`, `pictutil.llb`, `polarplt.llb`, `robot.llb`, `scale.llb`, and `smith.llb`. These `.llb` files contain example VIs to draw a waveform graph, plot Smith charts, plot polar plots, and so forth. They also contain supplemental VIs that the higher-level VIs use.

Demonstration Library

The demonstration library, `demos.llb`, contains the high-level examples that illustrate the high-level subVIs in the example libraries. For instance, one demonstration example shows how to use the VIs in the `cartesn.llb` to draw a waveform graph in a picture indicator. Another shows how to use the VIs in the `polarplt.llb` to create a polar plot.

Following are the names of the demonstration VIs from this library and a brief description of what each one illustrates.

Artificial Strip Chart

The Artificial Strip Chart VI uses the Plot XY VI (`cartesn.llb`) to create a strip chart in a picture indicator. Because you cannot scroll an arbitrary region of a picture, the VI maintains a limited history of points, and draws them as a graph with each update. See Chapter 5, *Cartesian Graph VIs*.

Histogram Plot

The Histogram Plot VI uses the Plot Waveform VI (`cartesn.llb`) to draw a bar plot in a picture indicator. For more information on the Plot Waveform VI, see Chapter 5, *Cartesian Graph VIs*.

Picture Waterfall Spectrum

The Picture Waterfall Spectrum VI calls the Draw Multiple Lines VI (**Picture** palette) to draw a waterfall plot. A waterfall plot displays a waveform in the context of previous waveforms. As new waveforms are displayed, older waveforms are still visible, but are scrolled into the background.

Polar Plot

The Polar Plot VI uses the Polar Graph VI (`polarplt.llb`) to draw a polar plot in a picture indicator. For more information on the Polar Graph VI, see Chapter 6, *Polar Graph VIs*.

Scale Demo

The Scale Demo VI uses the Calc Scale Specs VI and the Draw Scale VI (`scale.llb`) to draw a scale of a Cartesian graph in a picture. By changing inputs on the VI, you can change the range, location, and orientation of the scale. The scale VIs are also used by the polar graph VIs and the Cartesian graph VIs to draw the scales for the graphs. For more information on the scale VIs, see Chapter 8, *Graph Scale VIs*.

Simple Smith Plot

The Simple Smith Plot VI uses the Smith Plot VI (`smith.llb`) to draw a Smith plot in a picture indicator. This VI shows how the Smith plot VIs autoscale the displayed data. For more information on the Smith Plot VI, see Chapter 7, *Smith Plot VIs*.

Smith Multi Plot with Styles

The Smith Multi Plot with Styles VI uses the Smith Multi Plot VI and the Normalize Smith Plot VI (`smith.llb`) to draw a Smith plot in a picture indicator. This VI illustrates the line styles available, as well as how to normalize data for display. For more information on the Smith Multi Plot VI, see Chapter 7, *Smith Plot VIs*.

Smith Plot with Zooming

The Smith Plot with Zooming VI uses the Smith Multi Plot VI (`smith.llb`) as well as some support VIs (`_demosup.llb`) to draw a Smith plot with a zooming rectangle superimposed over the plot. Using the scrollbars, you can move and size the rectangle and zoom in on sections of the plot. For more information on the Smith Multi Plot VI, see Chapter 7, *Smith Plot VIs*.

Waveform & XY Plots

The Waveform & XY Plots VI calls the Plot Waveform VI and Plot XY VI (`cartesn.llb`) to draw a waveform and XY graph in a picture indicator. For more information on the Plot Waveform VI and Plot XY VI, see Chapter 5, *Cartesian Graph VIs*.

XY Multi Plot

The XY Multi Plot VI uses the Plot Multi-XY VI (`cartesn.llb`) to draw multiple plots in a single graph, where the graph is drawn in a picture indicator. For more information on the Plot Multi-XY VI, see Chapter 5, *Cartesian Graph VIs*.

XY Scatter Plots

The XY Scatter Plots VI uses the Draw Cartesian Axes VI and Draw XY Data VI (`cartesn.llb`) to draw a scatter plot. Together, these VIs illustrate a capability that is not present in LabVIEW's built-in graphs. That is, when multiple points map to the same point, the points are drawn as boxes, where the box size is proportional to the higher point density. These VIs are subVIs of the Plot XY VI, which is described in more detail in Chapter 5, *Cartesian Graph VIs*.

The demosup.llb

The `_demosup.llb` library contains VIs that are used strictly for the demo VIs. For instance, the Cartesian demos use the Product Squared Distribution VI to generate datasets.

The VIs in this library, which you are unlikely to use directly, include Draw Smith Cursor Coordinates, Generate Signal for Waterfall, Product Squared Distribution, Remap Smith Plot Section, Smith Zoom Stack, Update Smith Zoom Rect, and Waterfall Z-Axis Graphs.

Graph-Related Libraries

The following `.llbs` contain VIs for creating graphs.

cartesn.llb

The Cartesian VIs are a good starting point if you want a Cartesian graph with features that LabVIEW's XY and Waveform Graphs do not have. These VIs draw XY and Waveform Graphs, and have many of the same features as regular LabVIEW graphs. In addition to capabilities found in standard LabVIEW graphs, you can use these VIs to draw bar plots and waveform plots where the plot fills to a baseline.

The high-level VIs in the `cartesn.llb` are Plot XY VI, Plot Waveform VI, and Plot Multi-XY VI. These VIs in turn use other VIs to draw waveforms, scales, and grids.

This library is described in more detail in the Chapter 5, *Cartesian Graph VIs*.

cartesn2.llb

The `cartesn2.llb` library contains subVIs that are used by the VIs in `cartesn.llb`. The library also contains custom controls, which can be used to supply the inputs for the Cartesian graph VIs.

This library is described in more detail in Chapter 5, *Cartesian Graph VIs*.

polarplt.llb

The `polarplt.llb` library contains VIs that implement a polar graph. The high-level VI in `polarplt.llb` is Polar Graph VI. This VI in turn uses other VIs to draw polar grids, plots, and scales.

This library is described in more detail in Chapter 6, *Polar Graph VIs*.

scale.llb

The `scale.llb` library contains VIs that can be used to draw a scale for a graph. The polar plot and Cartesian graph VIs use the `scale.llb` VIs to draw the scales for their graphs. The high-level VIs in the `scale.llb` are Calc Scale Specs VI and Draw Scale VI. These VIs in turn use other VIs to calculate the locations of the markers and also map points to values.

This library is described in more detail in Chapter 8, *Graph Scale VIs*.

smith.llb

The `smith.llb` library contains VIs that implement a Smith plot. The high-level VIs in `smith.llb` are Smith Plot VI, Smith Multi Plot VI, and Normalize Smith Plot VI.

This library is described in more detail in Chapter 7, *Smith Plot VIs*.

Miscellaneous Libraries

pictutil.llb

The `pictutil.llb` library contains the Draw Circle by Radius VI and the Hilite Color VI. The Draw Circle by Radius VI draws a circle by specifying the circles radius and center point. The VI uses this information and calls the Draw Oval VI in the `Picture.llb`. The Hilite Color VI computes two *hilite* colors for a base color, one slightly lighter than the base color, and the other slightly darker. The robot arm VIs use the `pictutil.llb` VIs to create the three-dimensional shading of the various components of the robot arm.

robot.llb

The `robot.llb` library contains VIs that show how you can animate a robotic arm using the picture control. The top-level VI is Robot VI. This VI in turn uses other VIs to draw various components of the robot arm.

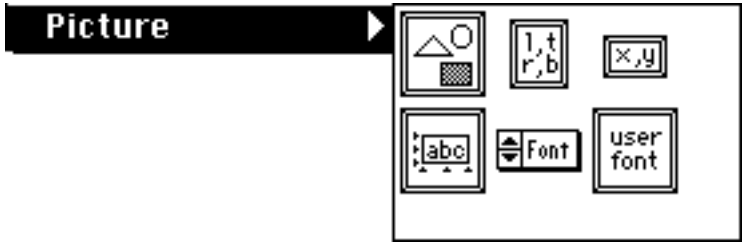
Chapter 3

Picture Controls

This chapter explains the controls in the **Picture** palette of the **Controls** menu.

Introduction

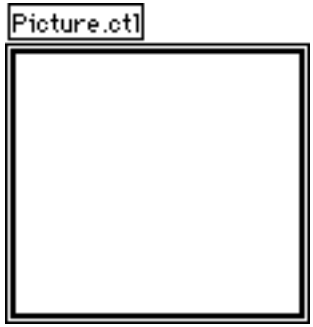
Most of the picture controls are custom controls for data structures that you commonly use in conjunction with the Picture VIs. For example, the rectangle custom control is a cluster of four numeric controls. Because you frequently use this data structure to specify a rectangle in the Picture VIs, LabVIEW includes the rectangle control in the **Controls** menu. These custom controls are included for convenience. If you prefer, you can easily create the same data structure yourself using a bundle function, or a front panel cluster with four numeric values.



The picture control is the only control that could not be built using standard controls in LabVIEW. This control is described in detail in this chapter, along with the attributes you can set for it.

Picture Control

The picture control (`picture.ctl`) displays pictures you create with the Picture VIs. The picture control defaults to an indicator. The picture control is shown in the following illustration.



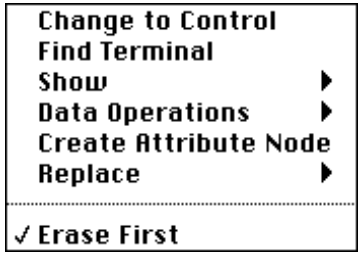
The picture control has a corresponding block diagram terminal as shown in the following illustration.

The picture indicator converts the drawing instructions embedded in the picture data into an image. The picture indicator is strictly a display; it has no interactive options. The only reason to make the picture indicator a control instead of an indicator is if you want to pass a picture as an input to a subVI, which might then concatenate more drawing commands to that picture.

The basics of how a picture control works are discussed in Chapter 1, *Getting Started*. The following is a description of some of the pop-up options, as well as the attribute node options for the picture indicator.

Picture Options

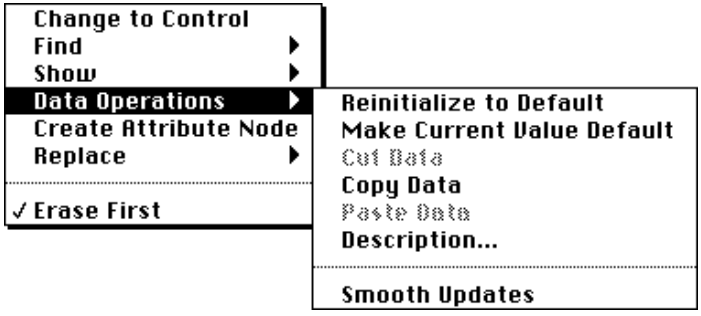
The pop-up menu for the picture indicator is shown in the following illustration. See your LabVIEW user manual for descriptions of most of these menu items.



If the **Erase First** option is on, the picture indicator erases its current image before drawing a new picture. If the **Erase First** option is off, the current image is not erased before drawing a new image.

Note: ***If* Erase First *is off, LabVIEW will not erase an old picture before it draws a new picture. However, LabVIEW does not keep the old picture in memory. Therefore, if you cover the picture indicator with another object, you will erase the covered area.***

The **Data Operations** pop-up menu is shown in the following illustration.



The **Smooth Updates** option acts like the **Smooth Updates** option of the graph indicators. When the **Smooth Updates** option is on, the picture indicator modifies an offscreen buffer before it displays the new image to minimize flashing. This feature may slow the drawing process, depending on the computer and video system you are using.

Picture Attributes

The picture indicator has the same base attributes as controls. These base attributes—Visible, Disabled, and Key Focus—are described in Chapter 15, *Attribute Nodes*, of your LabVIEW user manual.

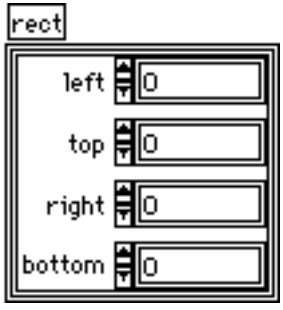
In addition to the base attributes, the picture indicator has the following attributes.

U8	Erase First	0 - Do not erase before drawing new data. 1 - Erase once, immediately. 2 - Erase every time new data is received.
[2D]	Dimensions	Returns the dimensions, in pixels, of the display area of the picture indicator. ***This is a read-only attribute. You cannot set this attribute using an attribute node.***
U32		**Width**, in pixels.
U32		**Height**, in pixels.

Picture Custom Controls

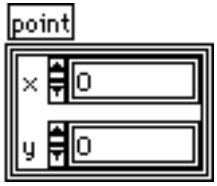
Rectangle Control

The rectangle custom control (`Rectangle.ctl`) is a cluster of four 16-bit signed integers that define the left, top, right, and bottom pixel coordinates of the edges of a rectangular area. This control is shown in the following illustration.



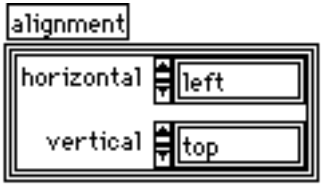
Point Control

The point custom control (`Point.ctl`) is a cluster of two 16-bit signed integers that define the x and y pixel coordinates of a point. This control is shown in the following illustration.

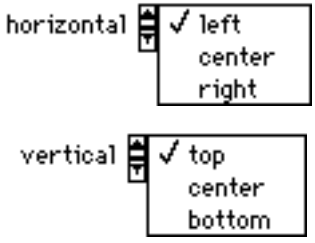


Text Alignment Control

The text alignment custom control (`Text Alignment.ctl`) is a cluster of two enumerations that describe the orientation of the first character of text relative to a specified point. You use this control with the Draw Text at Point VI to specify where you want text to be drawn. This control is shown in the following illustration.

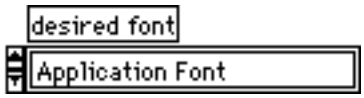


The horizontal and vertical enumerations can have the values shown in the following illustration. If you are not familiar with enumerations, see Chapter 5, *Numeric Controls and Indicators*, in your LabVIEW user manual for a description of how they work.

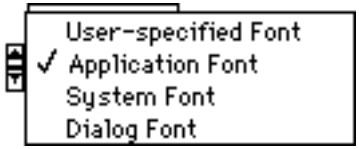


Font Enum Control

The font enum custom control (`Font Enum.ctl`) is an enumeration you use to specify the font you want to use for the text in a picture. This control is shown in the following illustration.



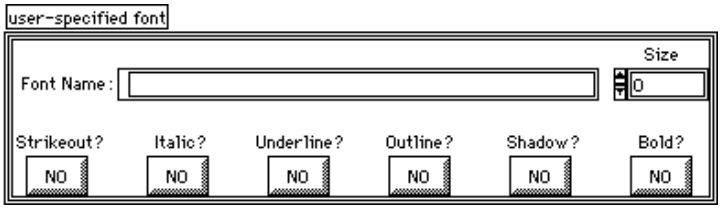
The font enum control enumeration has the following value choices.



If you select **User-specified Font**, you use the user-specified font control to specify a font other than one of LabVIEW's predefined fonts. This control is described next.

User-Specified Font Control

The user-specified font custom control (`User Font.ctl`) is a cluster of several values that you can use to specify a font. This control is shown in the following illustration.

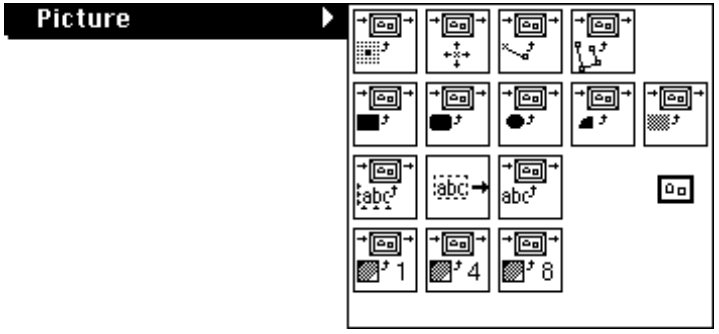


Chapter 4

Picture VIs

This chapter describes the VIs you use to create pictures that the picture indicator displays.

You get these VIs from the **Functions Picture** palette shown in the following illustration.

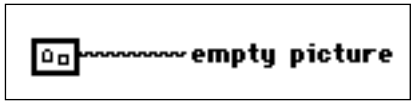


Many of these picture VIs use clusters to define points and rectangles. The VIs related to drawing text use clusters and enumerations to describe the font choice and the positioning of the text. These data types are described in Chapter 3, *Picture Controls*.

Picture VI Descriptions

Empty Picture

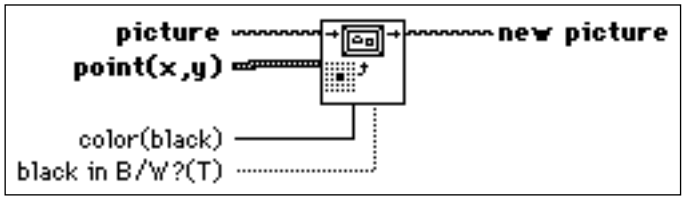
Returns an empty picture. Notice that the input picture for all of the picture VIs defaults to an empty picture.



empty picture is the output.

Draw Point

Sets a pixel in a picture to a specified color.



picture is the collection of drawing instructions to which you add an instruction to set a pixel.

point (x, y) is the coordinate of the pixel to set in the resulting picture with the origin (0,0) at the top-left corner of the indicator.

I16 **x** is the horizontal coordinate that increases to the right.

I16 **y** is the vertical coordinate that increases to the bottom.

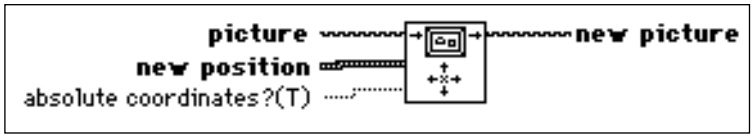
U32 **color** is a color numeric that specifies the color of the pixel. The default is black.

black in B/W? specifies whether the pixel appears as black or as white if you display it on a black-and-white monitor. The default is black (TRUE).

new picture is the modified picture.

Move Pen

Changes the current pen location of a LabVIEW picture to the specified position, or by the specified amount if the position is relative.



picture is the picture whose current pen location you want to move. It defaults to an empty picture.

new position is the point to which the pen moves in the new picture. If **absolute coordinates?** is TRUE, the pen moves to the absolute coordinate specified by **new position**. If **absolute coordinates?** is FALSE, the pen moves relative to the current pen location in **picture**.

x is the horizontal coordinate that increases to the right.

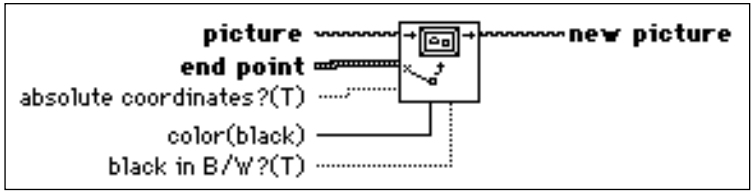
y is the vertical coordinate that increases to the bottom.

absolute coordinates? specifies whether the point **new position** is in absolute or relative coordinates. If you do not wire it, it defaults to TRUE (absolute).

new picture is the modified picture.

Draw Line

Draws a line from the current pen position to the specified location in a LabVIEW picture. The **end point** is either absolute or relative to the current position, depending on the value of the **absolute coordinates?** input.



picture is the picture to which you want to add the line. It defaults to an empty picture.

end point is the end point of the new line. If **absolute coordinates?** is TRUE, the line draws to the absolute coordinate that **end point** specifies. If **absolute coordinates?** is FALSE, the line draws relative to the current pen location in **picture**.

I16 **x** is the horizontal coordinate that increases to the right.

I16 **y** is the vertical coordinate that increases to the bottom.

TF **absolute coordinates?** specifies whether the point **end point** is in absolute or relative coordinates. If you do not wire it, it defaults to TRUE (absolute).

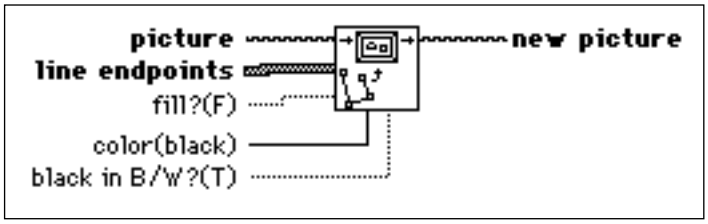
U32 **color** is a color numeric that specifies the color of the line. The default is black.

TF **black in B/W?** specifies whether the line appears as black or as white if you display it on a black-and-white monitor. The default is black (TRUE).

new picture is the modified picture.

Draw Multiple Lines

Draws multiple connected lines into a LabVIEW picture. You can also use it to draw color-filled polygons by passing TRUE to the fill input. All points are absolute coordinates.



picture is the picture to which you want to add lines. It defaults to an empty picture.

line endpoints is an array of points describing the lines to draw. The lines are drawn in the specified color, connecting the points in order. All coordinates are absolute.

I16 **x** is the horizontal coordinate that increases to the right.

I16 **y** is the vertical coordinate that increases to the bottom.

fill? specifies whether to fill the interior. It defaults to drawing only the lines (FALSE).

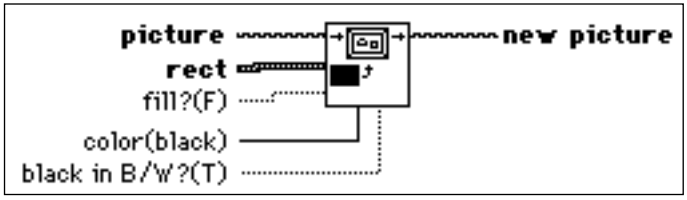
color is a color numeric that specifies the color of the lines. If the interior is filled, the VI uses this color for the interior. The default is black.

black in B/W? specifies whether the lines appear as black or as white if you display it on a black-and-white monitor. The default is black (TRUE).

new picture is the modified picture.

Draw Rect

Draws a rectangle, optionally filling the interior.



picture is the picture to which you want to add the rectangle. It defaults to an empty picture.

rect is a cluster containing four numerics that describe the left, top, right, and bottom coordinates of the outer edges of the rectangle.

I16 **left** is the horizontal coordinate of the left edge of the rectangle.

I16 **top** is the vertical coordinate of the top edge of the rectangle.

I16 **right** is the horizontal coordinate of the right edge of the rectangle.

I16 **bottom** is the vertical coordinate of the bottom edge of the rectangle.

TF **fill?** specifies whether to fill the rectangle. It defaults to drawing only the frame (FALSE).

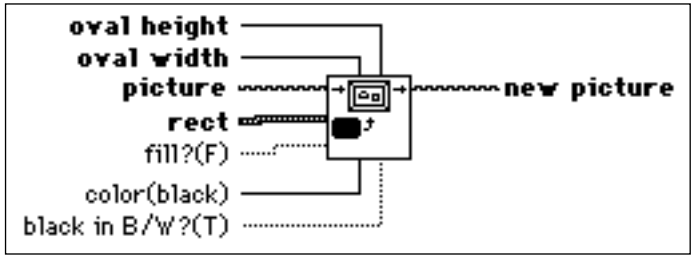
U32 **color** is a color numeric that specifies the color of the rectangle border. If the rectangle is filled, the VI also uses this color for the interior. The default is black.

TF **black in B/W?** specifies whether the rectangle should be black or white if you display it on a black-and-white monitor. The default is black (TRUE).

new picture is the modified picture.

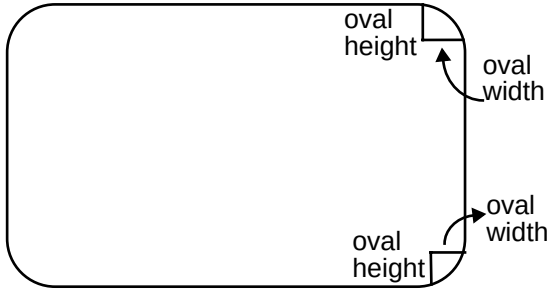
Draw Round Rect

Draws a rounded rectangle into a LabVIEW picture. The curvature of the corners is defined by the **oval width** and **oval height** inputs.



oval width is the width of an oval that defines the amount of curvature for the corners. See the discussion of how this parameter works in the description of **oval height**.

oval height is the height of an oval that defines the amount of curvature for the corners. If **oval height** and **oval width** are zero, you get a normal rectangle. If they are greater than zero, they describe how the corners of the rectangle should be rounded, as shown in the following illustration.



picture is the picture to which you want to add the rectangle. It defaults to an empty picture.

rect is a cluster containing four numerics that describe the left, top, right, and bottom coordinates of the outer edges of a rectangle that circumscribes the rounded rectangle. All coordinates are absolute.

left is the horizontal coordinate of the left edge of the rectangle.

top is the vertical coordinate of the top edge of the rectangle.

right is the horizontal coordinate of the right edge of the rectangle.

bottom is the vertical coordinate of the bottom edge of the rectangle.

fill? specifies whether to fill the oval rectangle. It defaults to drawing only the frame (FALSE).

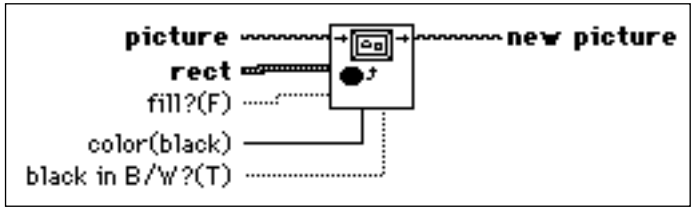
color is a color numeric that specifies the color of the rectangle. If the rectangle is filled, the VI uses this color also for the interior. The default is black.

black in B/W? specifies whether the oval rectangle should be black or white if you display it on a black-and-white monitor. The default is black (TRUE).

new picture is the modified picture.

Draw Oval

Draws an oval in the rectangle you specify, optionally filling the interior.



picture is the picture to which you want to add the oval. It defaults to an empty picture.

rect is a cluster containing four numerics that describe the left, top, right, and bottom coordinates of the outer edges of the rectangle.

left is the horizontal coordinate of the left edge of the rectangle.

top is the vertical coordinate of the top edge of the rectangle.

right is the horizontal coordinate of the right edge of the rectangle.

bottom is the vertical coordinate of the bottom edge of the rectangle.

fill? indicates whether the oval should be filled when drawn. It defaults to drawing only the frame, or outline, (FALSE).

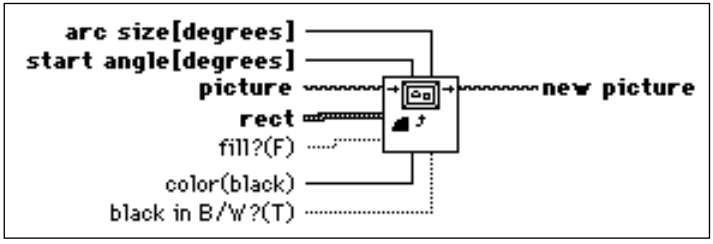
color is a color numeric that specifies the color of the oval. If the oval is filled, the VI uses this color for the interior. The default is black.

black in B/W? specifies whether the oval should be black or white if you display it on a black-and-white monitor. The default is black (TRUE).

new picture is the resulting picture.

Draw Arc

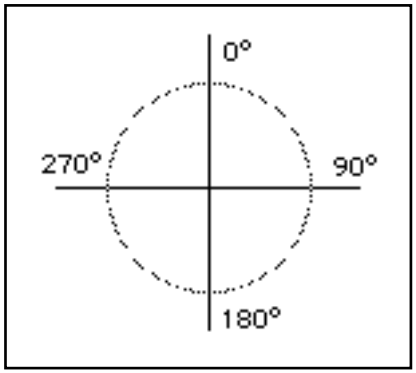
Draws an arc in a LabVIEW picture.



A rectangle specifies the size of an oval that contains the arc. Thus, the arc may not fill the whole rectangle. LabVIEW specifies the angles that describe the arc in degrees. It can range from -360˚ to 360˚, with positive angles appearing clockwise, and negative angles appearing counterclockwise. Zero degrees is straight up.

LabVIEW measures all angles as though the enclosing rectangle were square. Thus, a line from the center to the upper-right corner describes an

angle of 45˚. The following illustration shows the conventions used in specifying the angle of a point of an arc.



I16 **arc size [degrees]** specifies the length of the arc.

I16 **start angle [degrees]** is where the arc begins.

picture is the picture to which you want to add the arc. It defaults to an empty picture if you do not wire it.

rect is a cluster containing four numerics that describe the left, top, right, and bottom coordinates of the outer edges of the rectangle.

- I16 **left** is the horizontal coordinate of the left edge of the rectangle.
- I16 **top** is the vertical coordinate of the top edge of the rectangle.
- I16 **right** is the horizontal coordinate of the right edge of the rectangle.
- I16 **bottom** is the vertical coordinate of the bottom edge of the rectangle.

TF **fill?** specifies whether to fill the rectangle. If FALSE (default), the VI draws only the arc component (it does not draw interior bounds of the wedge).

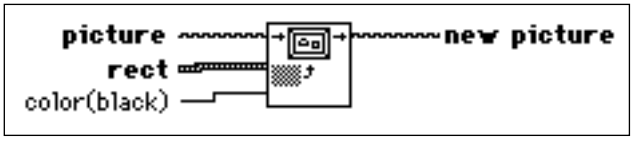
U32 **color** is a color numeric that specifies the color of the arc. If the arc is filled, the VI uses this color for the interior. The default is black.

black in B/W? specifies whether the arc should be black or white if you display it on a black-and-white monitor. The default is black (TRUE).

new picture is the modified picture.

Draw Grayed Out Rect

Draws a patterned rectangle in the specified picture to create the effect of graying-out a section of the picture. An underlying picture shows through the holes in the pattern. The VI does not frame the rectangle.



picture is the picture to which you want to add the rectangle. It defaults to an empty picture if you do not wire it.

rect is a cluster containing four numerics that describe the left, top, right, and bottom coordinates of the outer edges of the rectangle.

left is the horizontal coordinate of the left edge of the rectangle.

top is the vertical coordinate of the top edge of the rectangle.

right is the horizontal coordinate of the right edge of the rectangle.

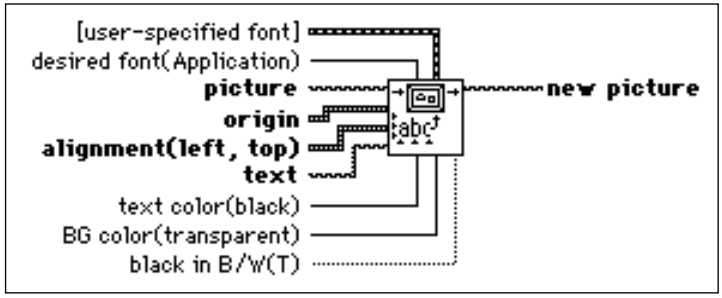
bottom is the vertical coordinate of the bottom edge of the rectangle.

color indicates the color of the pattern. In black and white, the VI uses white.

new picture is the modified picture.

Draw Text at Point

Draws a string into a picture. This VI will automatically calculate the bounding rectangle of the text and position the text with respect to a specified point.



user-specified font is a cluster containing the specific font characteristics for your text. The VI uses this input only if you set the **desired font** input to zero (user-specified font). The data type for the **user-specified font** is the same as the User-Specified Font control described in Chapter 3, *Picture Controls*.

font name selects the font to use for the string.

font size selects the size of the font in points.

strikeout? If true, text appears in ~~strikeout~~.

italic? If true, text appears in *italic*.

underline? If true, text appears underlined.

outline? If true, text appears in outline.

shadow? If true, text appears in shadow.

bold? If true, text appears in **bold**.

desired font specifies the font for the text you draw. You can choose from the following settings:

0: user-specified font. Use the font that **User-Specified Font** describes.
1: Application Font (default).
2: System Font.
3: Dialog Font.

picture is the picture to which you add the text string. It defaults to an empty picture.

alignment specifies the location of the text relative to the specified **origin**.

horizontal is an enumeration that you use to specify the horizontal position of the text with respect to the origin. It has one of the following values.

- left: The horizontal component of origin specifies the left edge of the text.
- center: The horizontal component of the origin specifies the center of the text.
- right: The horizontal component of origin specifies the right of the text.

vertical is an enumeration that you use to specify the vertical position of the text with respect to the origin.

- top: The vertical component of origin specifies the top edge of the text.
- center: The vertical component of the origin specifies the center of the text.
- bottom: The vertical component of origin specifies the bottom of the text.

origin is the coordinate of the starting point of the text.

x is the horizontal coordinate that increases to the right.

y is the vertical coordinate that increases to the bottom.

text specifies the string of text to be drawn.

text color is a color numeric that specifies the color of the text. The default is black.

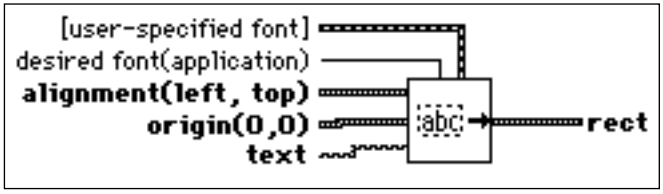
BG color is a color numeric that specifies the color of the background of the text. The default is transparent.

black in B/W? specifies whether the text should be black or white if you choose to display in on a black-and-white monitor. The default is black (TRUE). If you set the text to black, its background will be white, and vice versa.

new picture is the modified picture.

Get Text Rect

Returns the bounding rectangle of a string.



user-specified font is a cluster containing the specific font characteristics for the text you draw. The VI uses this input only if you set the **desired font** input to zero (user-specified font). The data type for the **user-specified font** is the same as the User-Specified Font control described in Chapter 3, *Picture Controls*.

font name selects the font to use for the string.

font size selects the size of the font in points.

strikeout? If true, text appears in ~~strikeout~~.

italic? If true, text appears in *italic*.

underline? If true, text appears underlined.

outline? If true, text appears in outline.

shadow? If true, text appears in shadow.

bold? If true, text appears in **bold**.

desired font specifies the font for the text you draw. You can choose from the following settings:

0: user-specified font. Use the font that **User-Specified Font** describes.
1: Application Font (default).
2: System Font.
3: Dialog Font.

alignment specifies the location of the text relative to the specified **origin**.

horizontal is an enumeration that you use to specify the horizontal position of the text with respect to the origin. It has one of the following values.

- left: The horizontal component of origin specifies the left edge of the text.
- center: The horizontal component of the origin specifies the center of the text.
- right: The horizontal component of origin specifies the right of the text.

vertical is an enumeration that you use to specify the vertical position of the text with respect to the origin.

- top: The vertical component of origin specifies the top edge of the text.
- center: The vertical component of the origin specifies the center of the text.
- bottom: The vertical component of origin specifies the bottom of the text.

origin is the coordinate of the starting point of the text.

x is the horizontal coordinate that increases to the right.

y is the vertical coordinate that increases to the bottom.

text specifies the string of text to be drawn.

rect is a cluster containing four numerics that describe the left, top, right, and bottom coordinates of the outer edges of the rectangle.

left is the horizontal coordinate of the left edge of the rectangle.

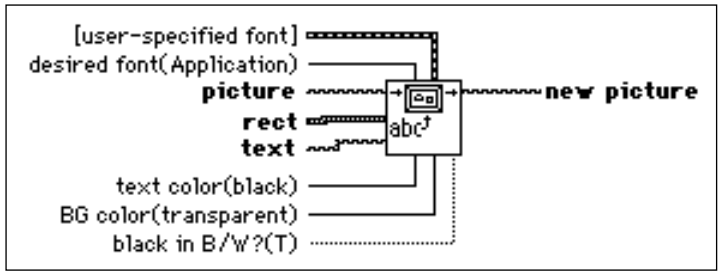
top is the vertical coordinate of the top edge of the rectangle.

right is the horizontal coordinate of the right edge of the rectangle.

bottom is the vertical coordinate of the bottom edge of the rectangle.

Draw Text in Rect

Draws a string into a LabVIEW picture. The string is clipped, or cropped, to the area of a specified rectangle.



user-specified font is a cluster containing the specific font characteristics for the text you draw. The VI uses this input only if you set the **desired font** input to zero (user-specified font). The data type for the **user-specified font** is the same as the User-Specified Font control described in Chapter 3, *Picture Controls*.

font name selects the font to use for the string.

font size selects the size of the font in points.

strikeout? If true, text appears in ~~strikeout~~.

italic? If true, text appears in *italic*.

underline? If true, text appears underlined.

outline? If true, text appears in outline.

shadow? If true, text appears in shadow.

bold? If true, text appears in **bold**.

desired font specifies the font for the text you draw. You can choose from the following settings:

0: user-specified font. Use the font that **user-specified font** describes.
1: Application Font (default).
2: System Font.
3: Dialog Font.

picture is the picture to which you add the text string. It defaults to an empty picture.

rect is a cluster containing the upper-left and lower-right coordinates that describe the rectangle that should contain the text. All coordinates are absolute.

left is the horizontal coordinate of the left edge of the rectangle.

top is the vertical coordinate of the top edge of the rectangle.

right is the horizontal coordinate of the right edge of the rectangle.

bottom is the vertical coordinate of the bottom edge of the rectangle.

text specifies the string of text to be drawn.

text color is a color numeric that specifies the color of the text. The default is black.

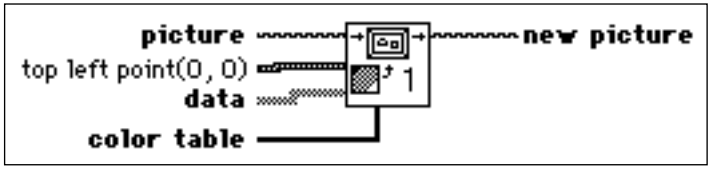
BG color is a color numeric that specifies the color of the background of the text. The default is transparent.

black in B/W? specifies whether the text should be black or white if you choose to display it on a black-and-white monitor. The default is black (TRUE). If you set the text to black, its background will be white, and vice versa.

new picture is the modified picture.

Draw 1-Bit Pixmap

Draws a one-bit pixmap into a LabVIEW picture. This VI has an input that is a 2D array of Booleans (where each Boolean represents a single pixel).



picture is the picture to which you want to add the pixmap. It defaults to an empty picture.

top left point is a cluster of two 16-bit integers that specifies where to place the top-left corner of the pixmap in the picture. It is in absolute coordinates.

I16 **x** is the horizontal coordinate that increases to the right.

I16 **y** is the vertical coordinate that increases to the bottom.

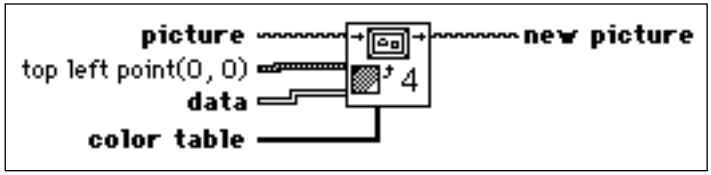
data is a 2D array of Booleans that LabVIEW draws as a bitmap in the picture. Elements that are FALSE map to element 0 in the **color table**, and elements that are TRUE map to element 1 in the **color table**.

color table is an array of two colors to which the array of data maps. If unwired, it defaults to black for data values of FALSE, and white for data values of TRUE.

new picture is the modified picture.

Draw 4-Bit Pixmap

Draws a four-bit pixmap into a LabVIEW picture. This VI has an input that is a 2D array of bytes (where each byte represents a single pixel).



picture is the picture to which you want to add the pixmap. It defaults to an empty picture.

top left point is a cluster of two 16-bit integers that specifies where to place the top left corner of the pixmap in the picture. It is in absolute coordinates.

I16 **x** is the horizontal coordinate that increases to the right.

I16 **y** is the vertical coordinate that increases to the bottom.

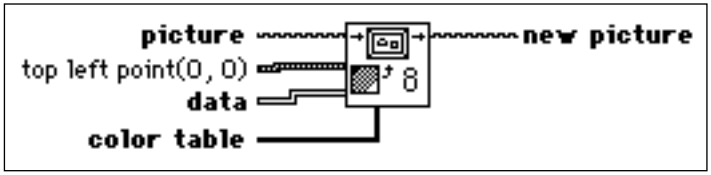
data is a 2D array of unsigned eight-bit integers that LabVIEW draws as a pixmap in the picture. The VI uses only the lower four bits of each byte, meaning that each point can have a range of zero to 15. Each data value maps to a value in **color table**.

color table is an array of up to 16 colors to which the array of data maps. If unwired, the picture uses LabVIEW's default 16 color palette.

new picture is the modified picture.

Draw 8-Bit Pixmap

Draws an eight-bit pixmap into a LabVIEW picture. This VI has an input that is a 2D array of bytes (where each byte represents a single pixel).



picture is the picture to which you want to add the pixmap. It defaults to an empty picture.

top left point is a cluster of two 16-bit integers that specifies where to place the top left corner of the pixmap in the picture. It is in absolute coordinates.

I16 **x** is the horizontal coordinate that increases to the right.

I16 **y** is the vertical coordinate that increases to the bottom.

data is a 2D array of unsigned eight-bit integers that LabVIEW draws as a pixmap in the picture. Each data value maps to a value in **color table**.

color table is an array of up to 256 colors to which the array of data maps. If unwired, the picture uses LabVIEW's default 256 color palette.

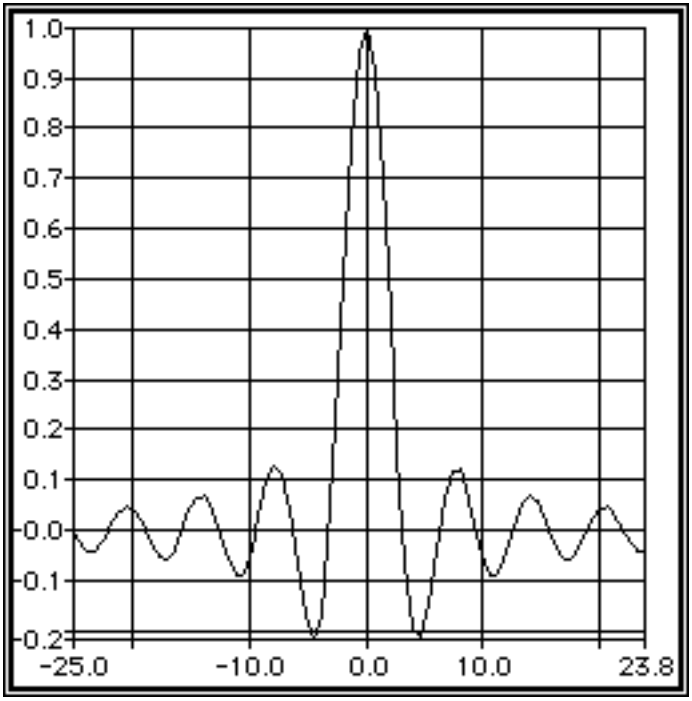
new picture is the modified picture.

Chapter 5

Cartesian Graph VIs

This chapter describes the VIs located in the `cartesn.llb` and `cartesn2.llb` libraries that you can use to create a Cartesian graph. These Cartesian VIs are a good starting point if you want a Cartesian graph with features that LabVIEW's XY and Waveform Graphs do not have.

An example of a Cartesian graph is shown in the following illustration.



The `cartesn2.llb` library contains subVIs and custom controls the VIs in `cartesn.llb` use.

The high-level VIs in the `cartesn.llb` are Plot XY VI, Plot Waveform VI, and Plot Multi-XY VI. These VIs in turn use other VIs to draw waveforms, scales, and grids.

Demonstration VIs

The following demonstration VIs are in the `demos.llb` and illustrate the Cartesian graph VIs.

Waveform & XY Plots

The Waveform & XY Plots VI uses the Plot Waveform VI and Plot XY VI (`cartesn.llb`) to draw a waveform graph and an XY graph in a picture indicator.

XY Multi Plot

The XY Multi Plot VI uses the Plot Multi-XY VI (`cartesn.llb`) to draw multiple plots in a single graph, where the graph is drawn in a picture indicator.

Artificial Strip Chart

The Artificial Strip Chart VI uses the Plot XY VI (`cartesn.llb`) to create a strip chart in a picture indicator. Because you cannot scroll an arbitrary region of a picture, the VI maintains a limited history of points, and draws them as a graph with each update.

Histogram Plot

The Histogram Plot VI uses the Plot Waveform VI (`cartesn.llb`) to draw a bar plot in a picture indicator.

XY Scatter Plots

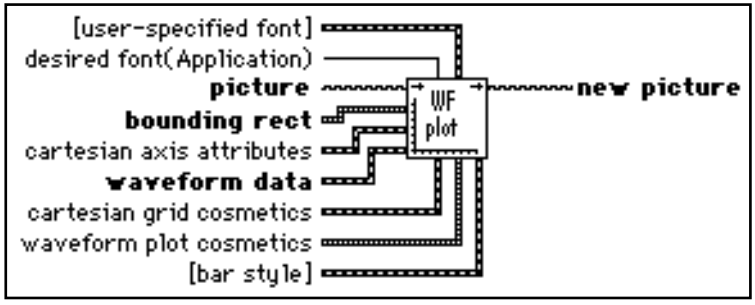
The XY Scatter Plots VI uses the Draw Cartesian Axes VI and Draw XY Data VI (`cartesn.llb`) to draw a scatter plot. Together, these VIs illustrate a capability that is not present in LabVIEW's built-in graphs. When multiple points map to the same point, the points are drawn as boxes, where the box size is proportional to the higher point density. These VIs are subVIs of the Plot XY VI. The Plot XY VI is described in more detail later in this chapter.

Using the Cartesian Graph Examples as SubVIs

The high-level graph VIs are general purpose VIs that provide a lot of functionality. Many of these VIs have a number of clusters for inputs. Using these VIs as subVIs may seem somewhat intimidating at first because of the clusters, but it is relatively simple if you use default values and custom controls.

For example, consider the Plot Waveform VI. This VI emulates the behavior of the built-in waveform graph. The Plot Waveform VI has the ability to draw waveforms in a variety of styles (including points, connected lines, bars), and as a comb plot. In addition, you can specify the color of a variety of the components, optionally have a grid, and specify range and format for the scales.

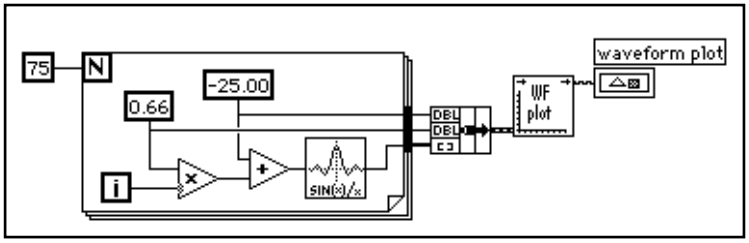
The help window for the Plot Waveform VI is shown in the following illustration.



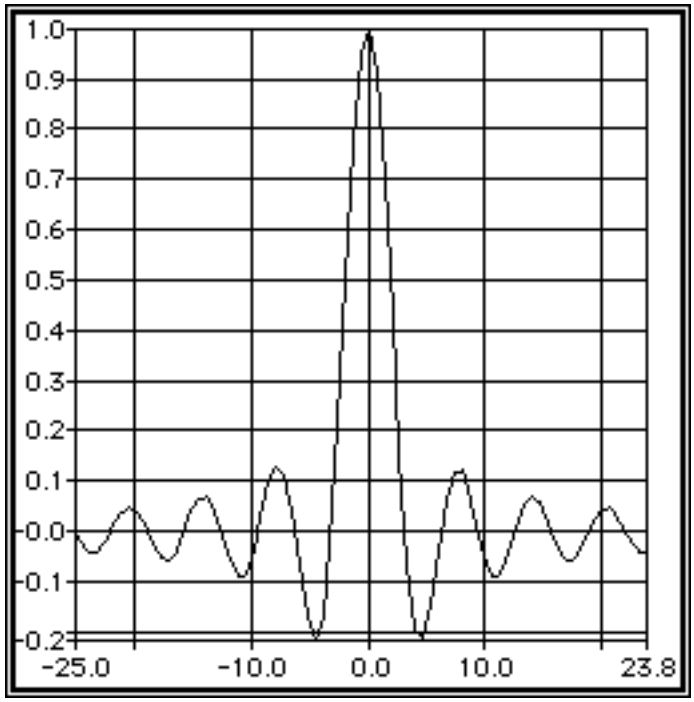
Notice that almost all of the inputs for this VI are clusters. Fortunately, most of them default to reasonable values, so you may not need to wire them at all. The inputs in bold are the ones you commonly wire. The inputs in brackets [ ] are rarely used. For instance, you only need to wire the **user-specified font** if you connect an input to **desired font** that indicates you want to use one of the built-in fonts (Application, System, or Dialog).

The example libraries contain custom controls for most of the inputs and outputs of the examples. Instead of creating the cluster on the diagram, you can place the control on the front panel, configure the input the way you want it, make the current value of the cluster the default value, and then hide the control.

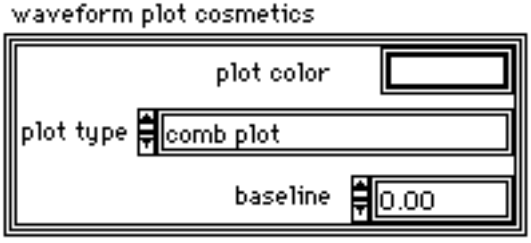
For example, with the Plot Waveform VI, you could use the following diagram to create waveform data and then plot it.



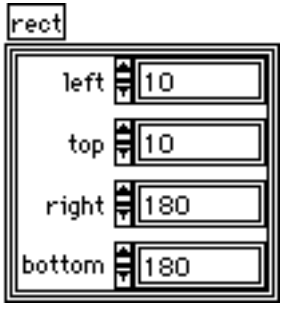
The resulting graph is shown in the following illustration.



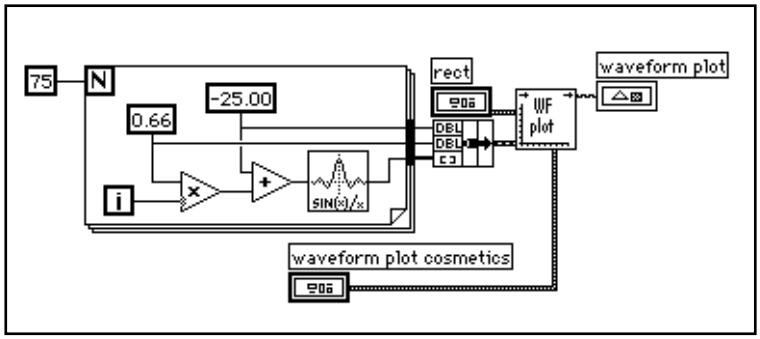
By default, the VI connects the points of the plot with lines. You can use the plot cosmetics input to this VI to override this setting. The easiest way to do this is to use the `Waveform Plot Cosmetics.ctl` to specify this information (use **Control…** to select the control from the file system). To make the graph draw as a comb plot, change the control to look like the one in the following illustration, and then make the current value the default value for the control.



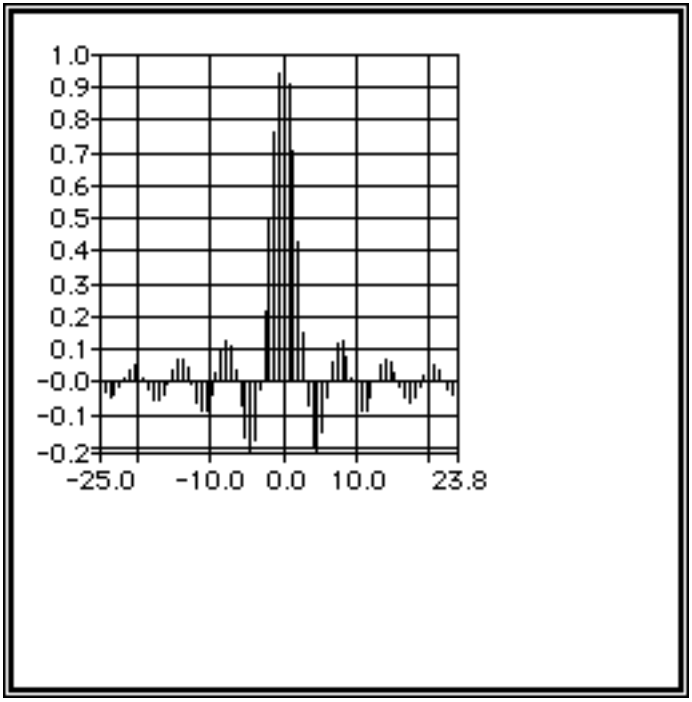
You may want to specify a different bounding rectangle for the graph. To do this, create the rectangle by using a control or by bundling data on the diagram, as shown in the following illustration.



The diagram with these changes is shown in the following illustration.

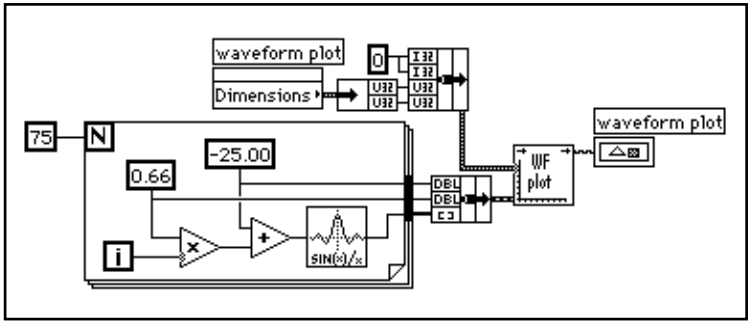


The resulting plot is shown in the following illustration.



If you want to make the rectangle the same size as the indicator that will display the data, you have two ways to determine the size of a picture indicator. One way is to use the parts window of the control editor to determine the size. Programmatically, you can use the **Dimensions** attribute from the picture indicator attribute node to determine the size.

Using the attribute node has an added advantage; if you resize the picture indicator, the VI will fill the new size automatically. An example is shown in the following illustration.

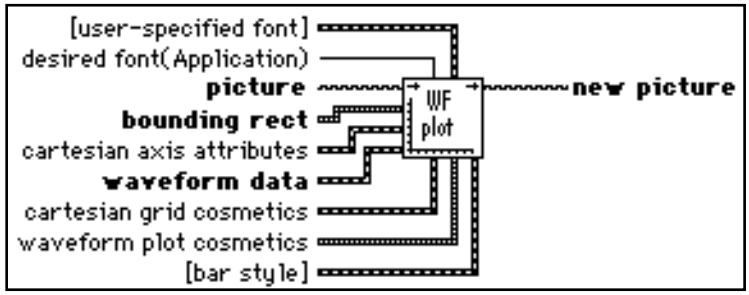


The Plot XY VI and Plot multi-XY VI are very similar to the Plot Waveform VI. The primary difference is the data type. In addition, you use different controls to specify the cosmetic appearance of the plot, because the XY plotting VIs have two additional plot styles. With the XY plotting VIs, you can add two scatter plot styles and a plot style where a line is drawn at each unique X position marking the minimum and maximum Y values for that X value.

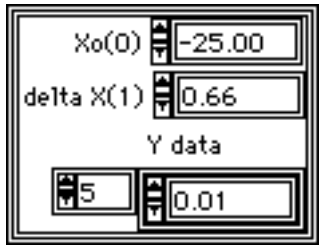
High-Level Cartesian Graph VIs

Plot Waveform

Takes uniformly distributed data and a picture, and creates a new picture that represents a waveform graph of the data. You specify the data, the starting point, and the separation between points. The graph can have scales and grids, and you can specify the style of the plots (bar, point, line, and so on).



The **waveform data** input is a cluster containing the initial X value, the spacing between points along the X axis, and the Y data that you want to plot. This cluster is shown in the following illustration.



Most of the remaining controls are clusters. As described at the beginning of this chapter, the simplest way to specify data for these inputs is to use custom controls that correspond to the given input.

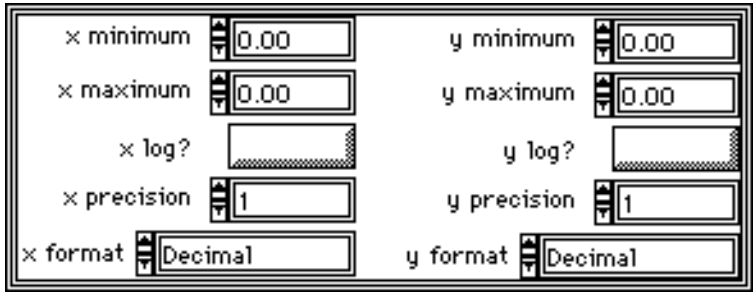
The **desired font** and **user-specified font** inputs determine the font that LabVIEW uses for the scales. Their data types are described in Chapter 3, *Picture Controls*. Corresponding controls are in the **Picture**

palette of the **Controls** menu. If you do not specify them, the fonts default to the Application Font.

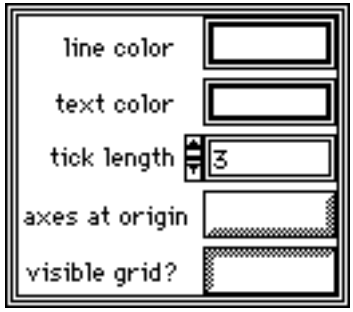
The **bounding rect** input defines the bounds of the graph rectangle, including the scales. The data type for a rectangle is described in Chapter 3, *Picture Controls*. If not specified, the VI uses a rectangle of 250 by 250 pixels.

The **cartesian axis attributes** input is a cluster that specifies the range and format for the x and y axes. It includes the following information on the minimum and maximum values for the scale—a Boolean indicating whether the scale should be logarithmic, the precision of the numbers, and the format (decimal, scientific, engineering, binary, octal, or hex).

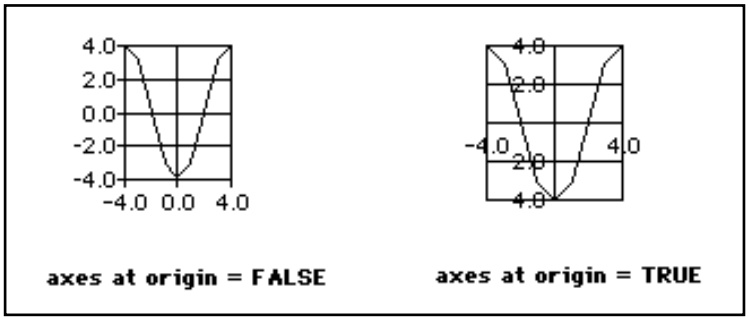
You can use the `Cartesian Axis Attributes.ctl` from the `cartesn2.llb` to specify the information for this input. If the minimum and maximum values are the same, or if the **cartesian axis attributes** are not wired, then the minimum and maximum values are automatically calculated from the data. The `Cartesian Axis Attributes.ctl` is shown in the following illustration.



The **cartesian grid cosmetics** input is a cluster specifying the style for the grid of the graph. It includes the following information on the line and text color—the length of a tick mark (mark on the edge of a scale next to the text), the Booleans that indicate whether the axes should be drawn at the origin or at the left edge of the plot, and whether the grid is visible. This cluster is shown in the following illustration.



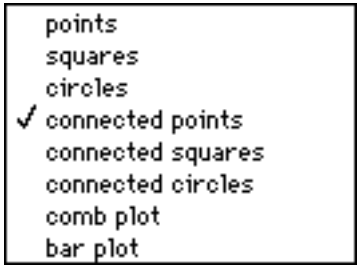
The **cartesian grid cosmetics** cluster contains a Boolean called **axes at origin**. If **axes at origin** is TRUE, LabVIEW always draws scales crossing at 0,0. If it is FALSE, LabVIEW always draws scales at the left and bottom edges of the plot. The difference between these two options is shown in the following illustration.



You can use `Cartesian Grid Cosmetics.ctl` from the `cartesn2.llb` to specify the information for **grid cosmetics**. If you do not specify this information, the line color and text color default to green, the tick length is at three, the axes are not set at the origin (they draw along the edges of the plot), and the grid is visible. This grid style looks best against a black background.

The **waveform plot cosmetics** input is a cluster that you can use to specify the plot color, the style for the plot, and the baseline. LabVIEW uses the baseline only if you specify a style of comb plot or bar plot. This cluster is shown in the following illustration.

The plot style is a ring that has the following options.



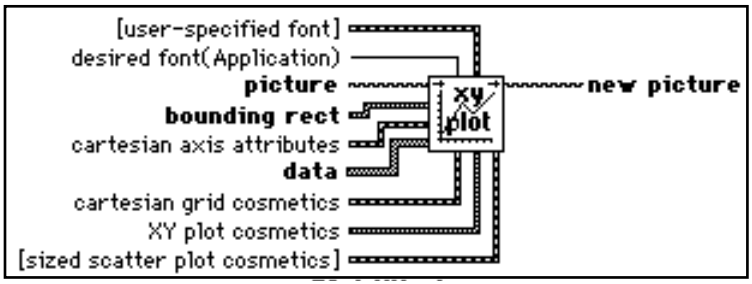
You can use the `Waveform Plot Cosmetics.ctl` from the `cartesn2.llb` to specify the information for the **waveform plot cosmetics** input. By default, the plot color is yellow, the style is connected points, and the baseline is zero. This plot style looks best against a black background.

The **bar style** input is a cluster you can use to specify the style for bar plots, assuming you select bar plot as the style for the graph. It contains inputs for the width of the bar, a Boolean that indicates whether you want the bar framed, and the color for the bar.

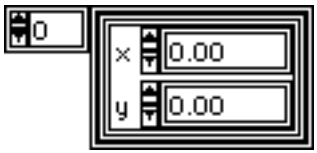
You can use the `Bar Style.ctl` from the `cartesn2.llb` to specify the information for this input. By default, bar width is 100 percent; that is, the width is equal to the interval between the points. The bar is centered around the specified point and is framed in black by default. This control is shown in the following illustration.

Plot XY

Takes a picture and an array of points, and appends a picture that represents an XY graph of the data. You specify both the x and y values for each point. The graph can have scales and grids, and you can specify the style of the plots—bar, point, line, and so on.



The **data** input is an array of points, where each point is a cluster of x and y pixel coordinates. This array is shown in the following illustration.



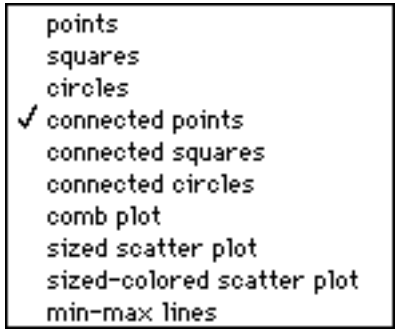
Most of the remaining controls are clusters. As described at the beginning of this chapter, the simplest way to specify these inputs is to use custom controls that are included in various locations, as described in the following paragraphs.

The **user-specified font**, **desired font**, **bounding rect**, **cartesian axis attributes**, and **cartesian grid cosmetics** inputs are identical to the inputs for the Plot Waveform VI.

The **XY plot cosmetics** input is different from its Plot Waveform VI counterpart. With the **XY plot cosmetics** cluster, you can specify the plot color, the style for the plot, and the baseline. LabVIEW uses the baseline only if you specify a style of comb plot. This cluster is shown in the following illustration.

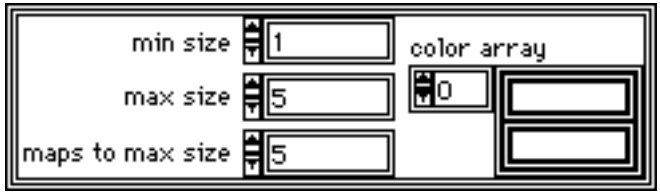
This input is different from its Plot Waveform VI counterpart in that the plot style ring has a different set of options. As with the Plot Waveform VI plot cosmetics, you can select basic point and line styles and a comb plot. You can also show distributions of data using a scatter plot where points that map to the same value are drawn as larger dots, a variation of the same plot with colors used to illustrate different distribution frequencies, and a style with lines at each x point showing you the minimum and maximum values for that point.

The options for the ring are shown in the following illustration.



You can use the `XY Plot Cosmetics.ctl` from the `cartesn2.llb` to specify the information for this input. By default, the plot color is yellow, the style is connected points, and the baseline is zero. This plot style looks best against a black background.

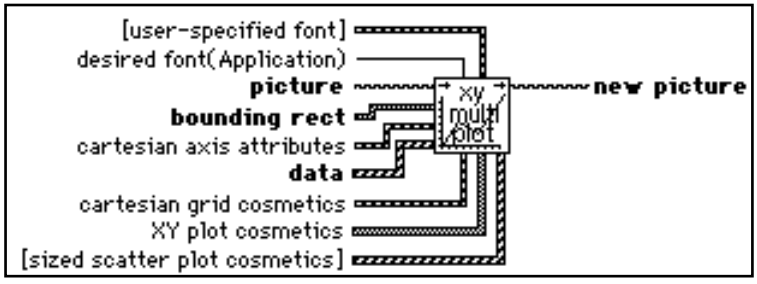
The **sized scatter plot cosmetics** input is a cluster that you can use to specify the range of sizes for frequency distribution points, assuming you select sized scatter plot or sized-colored scatter plot as the style for the graph. This cluster contains inputs for the minimum and maximum size for points, and a color array that contains the colors each size should map to. This cluster is shown in the following illustration.



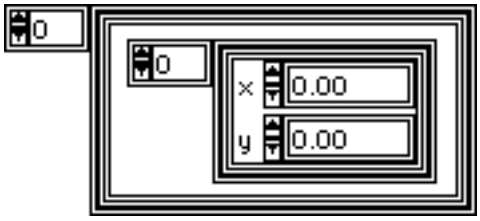
You can use the `Sized Scatter Plot Cosmetics.ctl` from the `cartesn2.llb` to specify the information for this input.

Plot Multi-XY

Takes a picture and an array of plots, and appends a picture of an XY graph of the data with the plots overlaid on the same graph. Each plot is an array of points. The graph can have scales and grids, and you can specify the style of the plots (bar, point, line, and so on).



The **data** input is an array of clusters of plots, where each plot is an array of points. Each point is represented by a cluster of x and y pixel coordinates. This array is shown in the following illustration.



Most of the remaining controls are clusters. As described at the beginning of this chapter, the simplest way to specify data for these inputs is to use custom controls that are included in various locations. These custom controls are described in the following paragraphs.

The **user-specified font**, **desired font**, **bounding rect**, **cartesian axis attributes**, and **cartesian grid cosmetics** inputs are identical to the inputs for the Plot Waveform VI.

The **XY plot cosmetics** input is an array of clusters of information that you can use to specify the plot color, the style for the plot, and the baseline. See the description of the **XY plot cosmetics** input of the Plot XY VI for more information on the data structure of the cluster.

You can use an array of the `XY Plot Cosmetics.ctl` from the `cartesn2.llb` to specify the information for this input. By default, the plot color is yellow, the style is connected points, and the baseline is zero. This plot style looks best against a black background.

The **sized scatter plot cosmetics** input is an array of clusters that you can use to specify the range of sizes and colors for frequency distribution points, assuming you select sized scatter plot or sized-colored scatter plot as the style for the graph. See the description of the **sized scatter plot cosmetics** input of the Plot XY VI for more information on the data structure of the cluster.

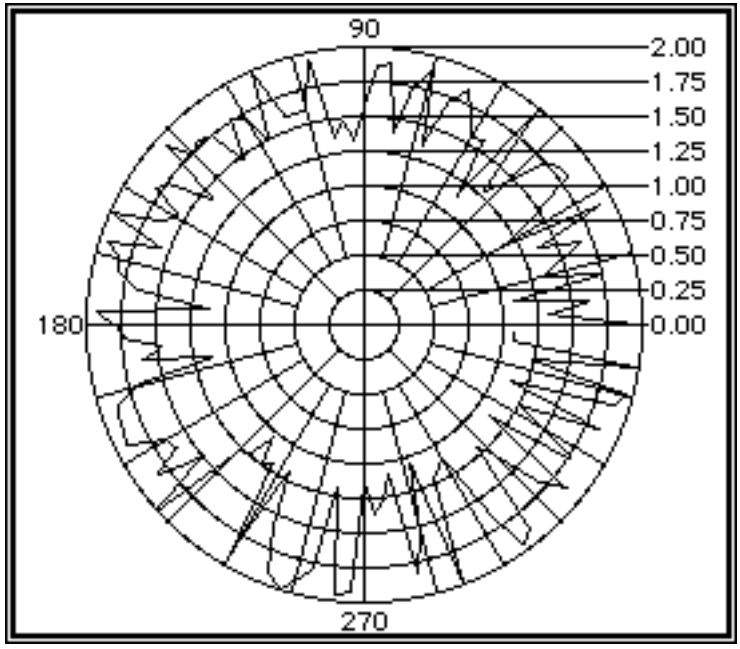
You can use the `Sized Scatter Plot Cosmetics.ctl` from the `cartesn2.llb` to specify the information for this input.

Chapter 6

Polar Graph VIs

This chapter describes the VIs in the `polarplt.llb` library that you can use to create a polar graph.

The following illustration is an example of a polar graph you can create using the Polar Graph VIs.



The high-level VI in the `polarplt.llb` is the Polar Graph VI. This VI in turn uses other VIs to draw polar grids, plots, and scales.

Demonstration VI

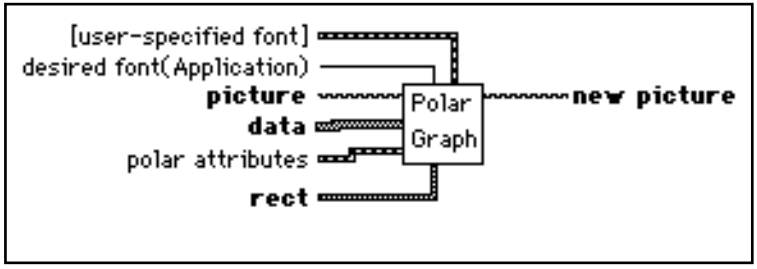
The Polar Plot VI in the `demos.llb` uses the Polar Graph VI (`polarplt.llb`) to draw a polar plot in a picture indicator.

Using the Polar Graph Example as SubVIs

The high-level Polar Graph VI has a lot of functionality and, consequently, a number of clusters for inputs. Using these VIs as subVIs may seem somewhat intimidating at first because of the clusters, but it is actually relatively simple if you use default values and custom controls.

The Polar Graph VI can draw specific, contiguous quadrants of a polar graph, or the entire graph at once. In addition, you can specify the color of a variety of the components, choose to have a grid, and specify range and format for the scales.

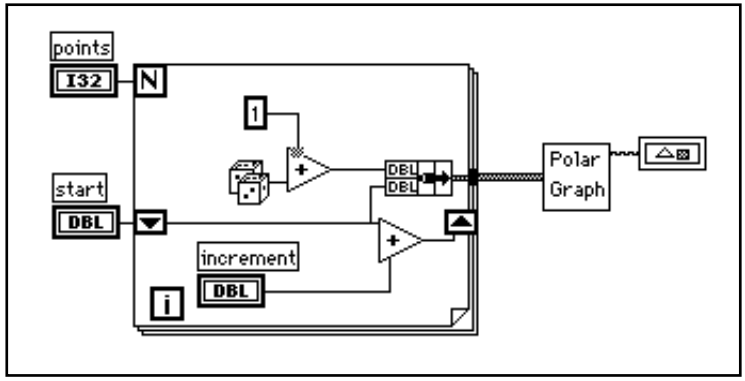
The help window for the Polar Graph VI is shown in the following illustration.



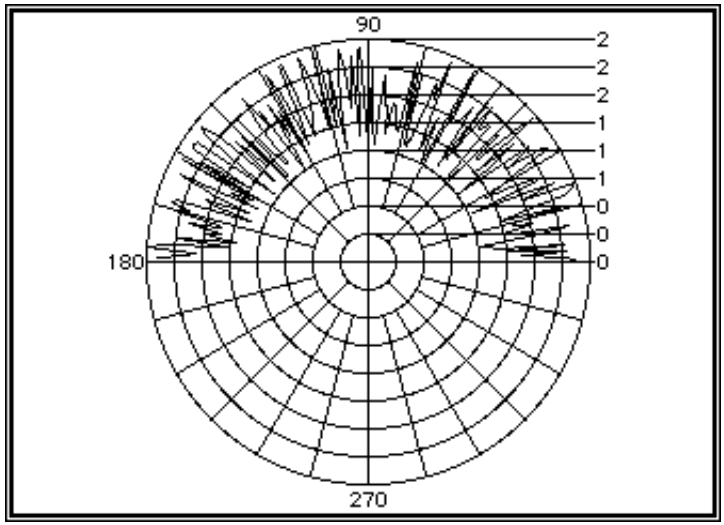
Notice that almost all of the inputs for this VI are clusters. Fortunately, most of them default to reasonable values, so you may not need to wire them at all. The inputs in bold are the inputs you commonly wire. You will rarely use the inputs in brackets [ ]. For instance, you only need to wire the **user-specified font** if you connect an input to **desired font** that indicates you want to use one of the built-in fonts (Application, System, or Dialog).

The `polarplt.llb` contains custom controls for most of the inputs and outputs of this VI. Thus, instead of creating the cluster on the diagram, you can place the control on the front panel, configure the input the way you want it, make the current value of the cluster the default value, and then hide the control.

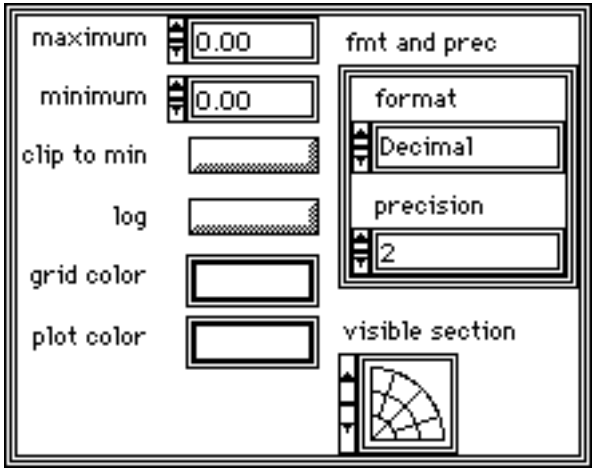
For example, with the Polar Graph VI, you could use the following diagram to create a set of data and plot it. This example is similar to the Polar Plot VI in the `demos.llb` example library.



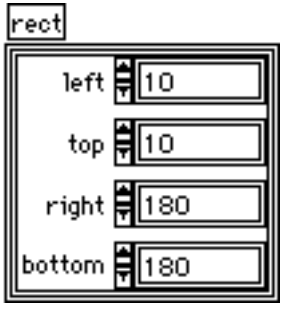
The resulting graph is shown in the following illustration.



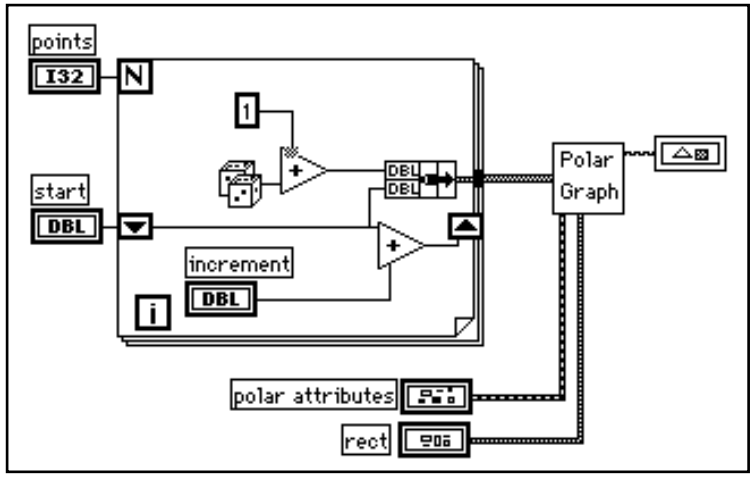
By default, the VI draws all four quadrants of the plot. In addition to other attributes, you can use the polar attributes input to specify the quadrant(s) you want to view. The simplest way to do this is to use the `Polar Attributes.ctl` from the `polarplt.llb` to specify this information (use **Control...** to select the control from the file system). To make the graph draw only the first quadrant, you would change the polar attributes to look like the following illustration, and then make the current value the default value for the control.



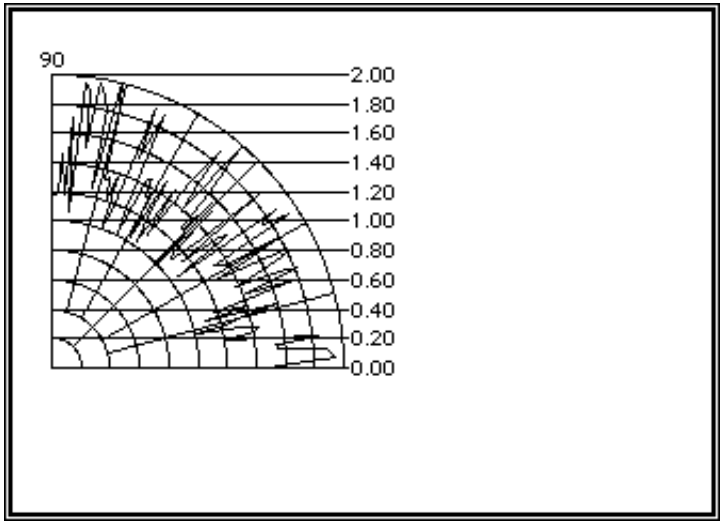
You will want to specify a different bounding rectangle for the graph. To do this, create the rectangle using a control or by bundling data on the diagram.



The block diagram with these changes is shown in the following illustration.

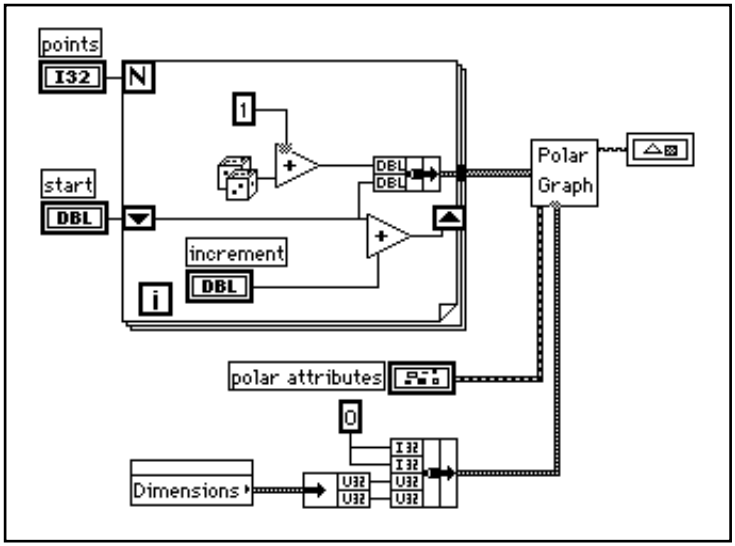


The resulting plot is shown in the following illustration.



Instead of creating the rectangle arbitrarily, you might want to make it the same size as the indicator that will display the data. There are two ways to determine the size of a picture indicator. One way is to use the parts window of the control editor to determine the size.

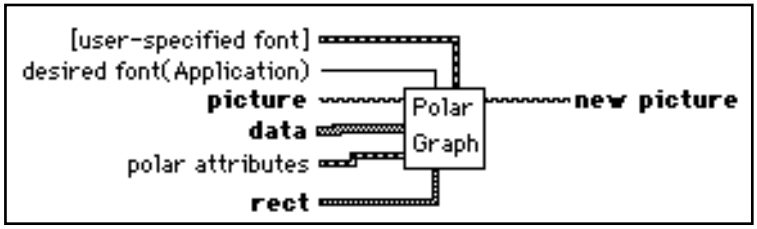
Programmatically, you can use the **Dimensions** attribute from the attribute node of the picture indicator to determine the size. Using the attribute node has the added advantage that if the indicator changes its size, the rectangle is automatically updated.



High-Level Polar Graph VI

Polar Graph

Takes a picture and polar data, and appends a picture that represents a polar graph of the data. Polar graphs always have scales and grids, and can only draw lines between points. You can specify the colors for the grid and plot, the format for the scale, and the section of the plot you want to view.



The **data** input is an array of points, where each point is a cluster containing a magnitude and a phase.

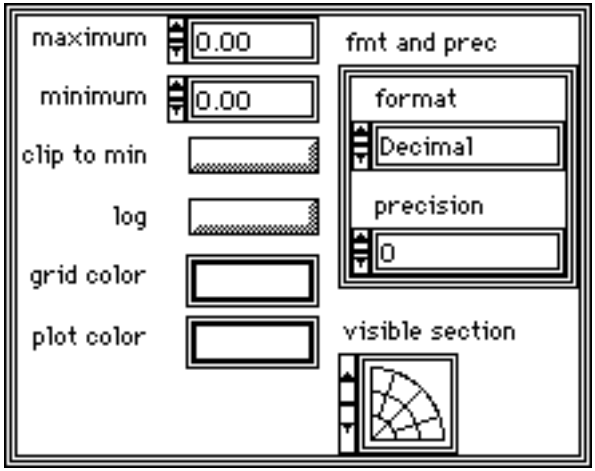
Most of the remaining controls are clusters. As described at the beginning of this chapter, the simplest way to give data for these inputs is to use custom controls that are included in various locations, as described in the following paragraphs.

The **desired font** and **user-specified font** inputs determine the font that is used for the scales. Their data types are described in Chapter 3, *Picture Controls*. Corresponding controls are given in the **Picture** palette of the **Controls** menu. If you do not specify which fonts to use, the font inputs default to the Application font.

The **bounding rect** input is the bounding rectangle for the graph, including the scales. The data type for a rectangle is described in Chapter 3, *Picture Controls*. If you do not specify a boundary, the VI uses a rectangle of 234 by 234 pixels.

The **polar attributes** input is a cluster that describes the format for the grid and scale of the polar plot, and indicates how the graph should handle negative magnitude data (clip the data or draw it). The **polar attributes** input contains numerics that describe the minimum and maximum values for the scale, a Boolean that indicates whether negative magnitude data should be clipped or drawn, a Boolean that indicates whether the scale should be logarithmic, the colors for the plot and the grid, the format (decimal, scientific, engineering, binary, octal, or hex) and precision for the scale numbers, and a ring you can use to specify the quadrant you want to view.

This cluster is shown in the following illustration.



You can use the `Polar Attributes.ctl` from the `polarplt.llb` to specify the information for this input. If the minimum and maximum values are the same, or if the **polar attributes** are not wired, then the minimum and maximum values are automatically calculated from the data. The remaining defaults are shown in the previous illustration.

With the **visible section** ring inside of the **polar attributes** cluster, you can select which quadrant(s) of the polar plot you want to view. The following table shows the possible values.

Table 6-1. Visible Section Values

value	quadrant(s) visible
0	
1	

Table 6-1. Visible Section Values (Continued)

value	quadrant(s) visible
2	
3	
4	
5	
6	
7	
8	

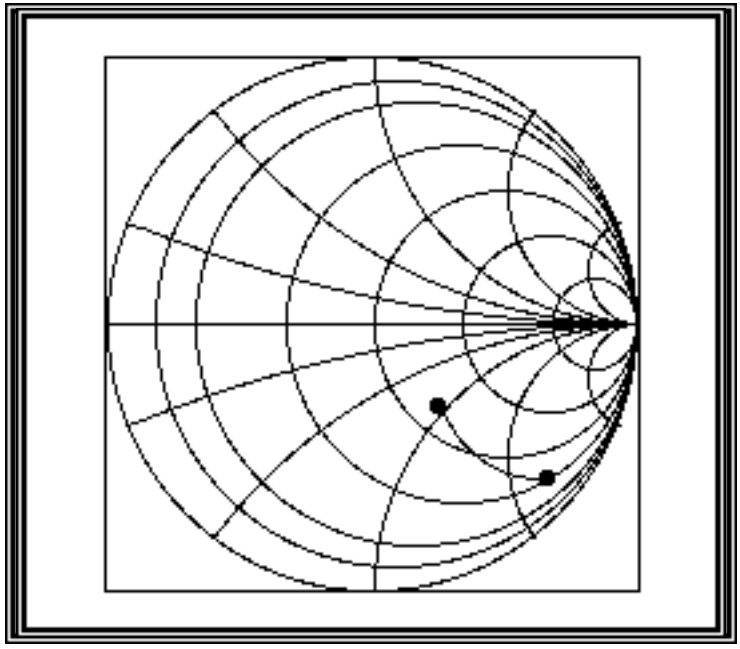
Chapter

7

Smith Plot VIs

This chapter describes the VIs in the `smith.llb` library that you can use to create a Smith plot.

The `smith.llb` library contains VIs that show you how to create a Smith plot using the picture control. Smith plots are frequently used in engineering applications to analyze the load effects of a media on a signal being transferred along that media. An example Smith plot is shown in the following illustration.



The top-level VIs are the Smith Plot VI and the Smith Multi Plot VI. These VIs in turn uses other VIs to draw various components of the Smith plot.

Demonstration VIs

The following three demonstration VIs are located in the `smith.llb` and relate to Smith plot VIs.

Simple Smith Plot

The Simple Smith Plot VI uses the Smith Plot VI to draw a Smith plot in a picture indicator. This VI shows how the Smith plot VIs autoscale the displayed data. See the section *High-Level Smith Plot VIs* later in this chapter for more detailed information about this VI.

Smith Multi Plot with Styles

The Smith Multi Plot with Styles VI uses the Smith Multi Plot VI and the Normalize Smith Plot VI to draw a Smith plot in a picture indicator. This VI illustrates the line styles available, as well as how to normalize data for display. See the section *High-Level Smith Plot VIs* later in this chapter for more detailed information about this VI.

Smith Plot with Zooming

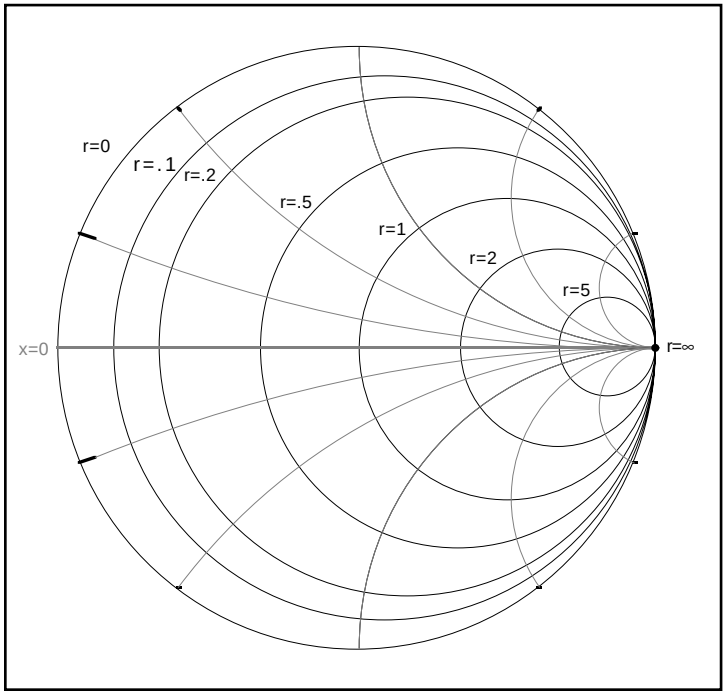
The Smith Plot with Zooming VI uses the Smith Multi Plot VI as well as some support VIs (located in the `_demosup.llb`) to draw a Smith plot with a zooming rectangle superimposed over the plot. Using the scrollbars, you can move and size the rectangle and zoom in on sections of the plot. See the section *High-Level Smith Plot VIs* later in this chapter for more detailed information about this VI.

Smith Plot Overview

You use Smith plots to study transmission line behavior. A transmission line is a medium through which energy and signals are transmitted. A transmission line might be a wire, or it might the atmosphere through which a signal is transmitted. Each of these transmission lines has an effect on the signal that is being transmitted. This effect, called the impedance of the transmission line, can attenuate or phase shift an AC signal.

A transmission line has an associated impedance, a measure of the resistance and the reactance of the line. The impedance is commonly listed as a complex number of the form z = r + jx. In this equation, z is a complex number that represents the impedance, and contains both resistance (r) and reactance (x) components.

You use the Smith plot to plot impedances of transmission lines. The plot consists of circles of constant resistance and reactance. An example of a Smith plot is shown in the following illustration.



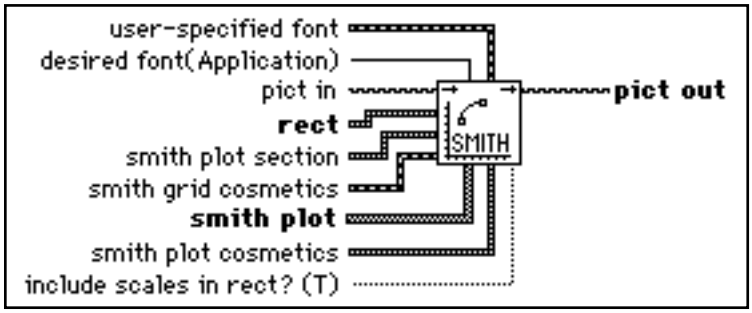
You can plot a given impedance, r + jx, by locating the intersection of the appropriate r circle and x circle. Once plotted, you can use the Smith plot as a visual aid to match impedance and to calculate the reflection coefficient of a transmission line.

Using the Smith Plot Examples as SubVIs

The high-level Smith plot VIs are very general, meaning that they provide a lot of functionality in a single VI. Consequently, many of these VIs have a number of complicated clusters for inputs. Using these VIs as subVIs may seem somewhat intimidating at first because of the

complexity of the inputs, but it is relatively simple if you use default values and custom controls.

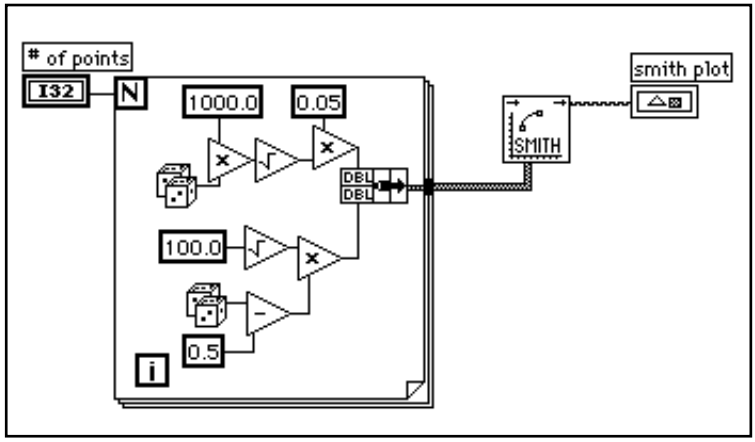
For instance, consider the Smith Plot VI. The help window for the Smith Plot VI is shown in the following illustration. This VI is discussed in detail later in this chapter in the *High-Level Smith Plot VIs* section.



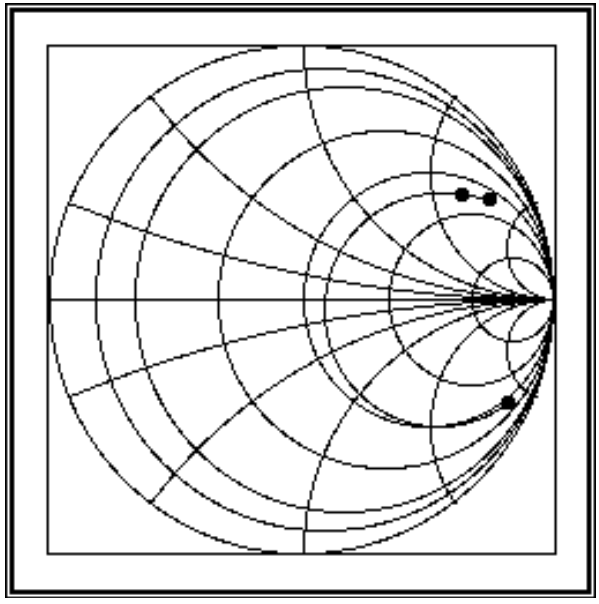
Notice that almost all of the inputs for this VI are clusters. Fortunately, most of them default to reasonable values, and you may not need to wire them at all. The inputs in **bold** are the inputs that you probably need to wire. You only need to wire the **user-specified font** if you connect an input to **desired font** that indicates you want to use one of the built-in fonts (Application, System, or Dialog).

Also, the example libraries contain custom controls for most of the inputs and outputs of the examples. Thus, instead of creating the cluster on the diagram, you can place the control on the front panel, configure the input the way you want it, make the current value of the cluster be the default value, and then hide the control.

If you are graphing load impedances, you can represent an impedance as a complex number of the r + jx. For the **smith plot** data input, you specify an array of points, where each point is a cluster containing an r value and an x value. For example, with the Smith Plot VI, you could use the following diagram to create data and plot it.

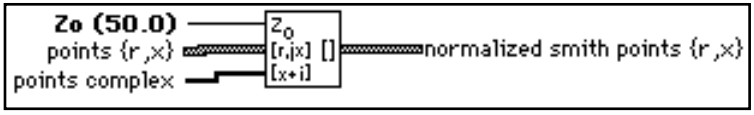


The resulting graph looks like the one in the following illustration.

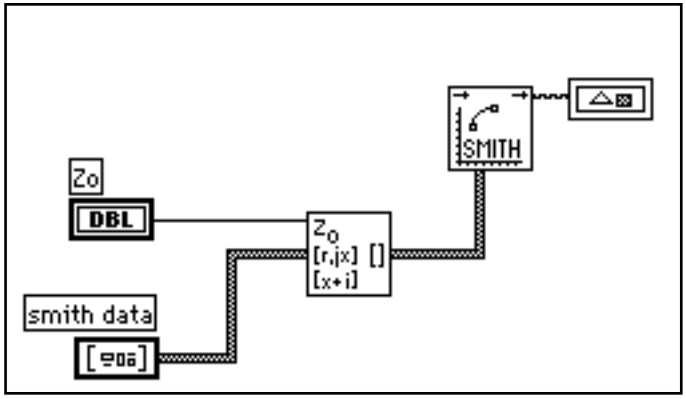


The data in the preceding example was random. Also, it was calculated so that the points had r values between zero and five, the most visible section of the Smith plot. For larger values of r, the points end up bunching together so closely that all detail is lost. To avoid this problem, you will probably want to normalize your data, scaling it relative to a

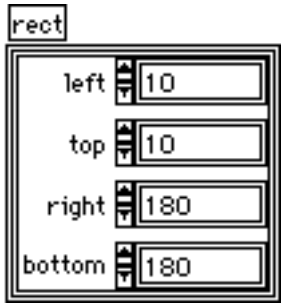
known value. In practice, you usually scale Smith plot data with respect to the characteristic impedance (Z_0) of the system. In LabVIEW, you can use the Normalize Smith Plot VI to normalize your data. The help window for this VI is shown in the following illustration. See the *High-Level Smith Plot VIs* section later in this chapter for more detailed information about this VI.



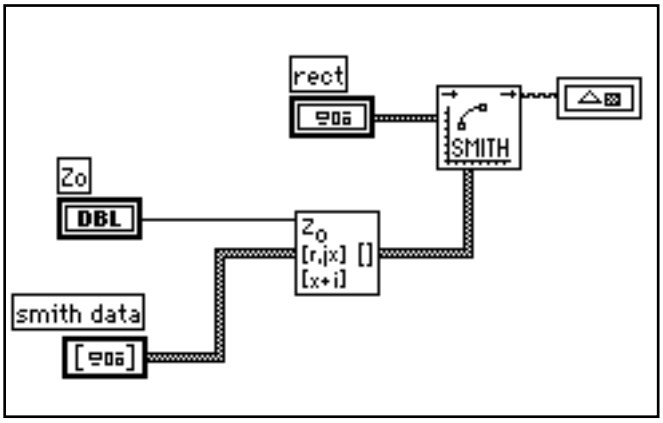
With the Normalize Smith Plot VI, you can specify the data you want to normalize. The resulting normalized data can be passed directly to the Smith Plot VI. An example of how to normalize data for a Smith plot is shown in the following illustration.



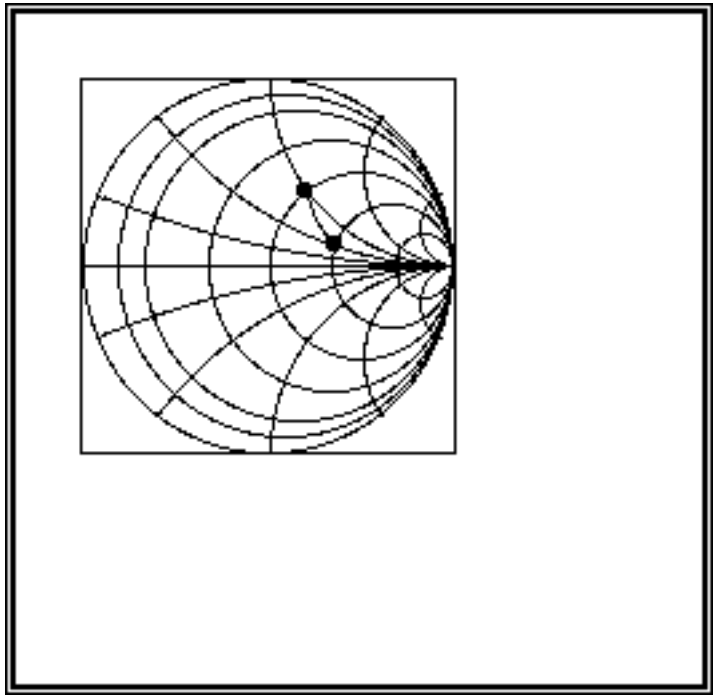
In addition to normalizing the data you want to display, you probably will want to specify a different bounding rectangle for the graph. One method for doing this would be to create the rectangle using either a control or by bundling data on the diagram. An example is shown in the following illustration.



An example of how to specify a bounding rect for a Smith plot is shown in the following illustration.

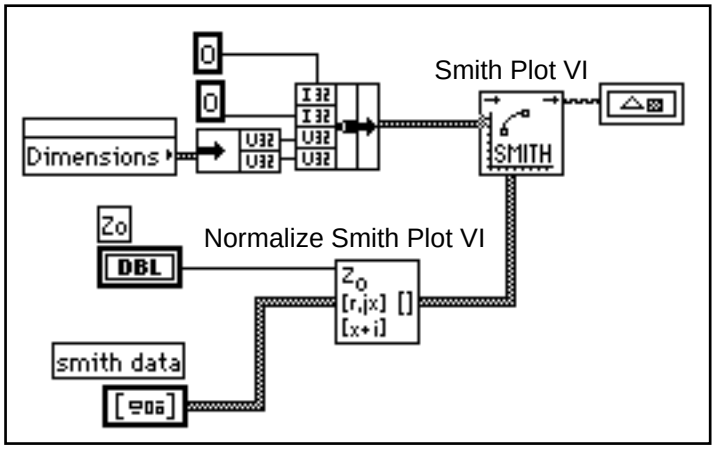


The resulting plot looks like the one in the following illustration.



Instead of creating the rectangle arbitrarily, you might want to make it the same size as the indicator that will display the data. There are two ways to determine the size of a picture indicator. One way is to use the parts window of the control editor to determine the size. Programmatically, you can use the **Dimensions** attribute from the attribute node of the Picture indicator to determine the size. Using the attribute node has the added advantage that if the indicator changes its size, the rectangle is automatically updated.

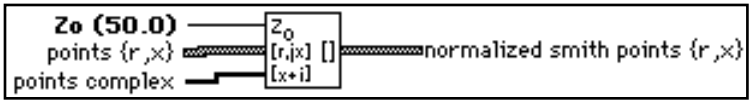
An example of how to use an attribute node to calculate the bounding rectangle is shown in the following illustration.



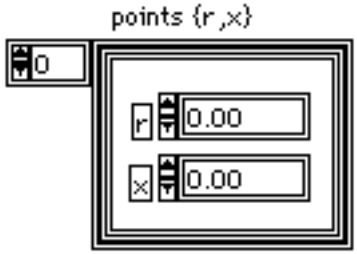
High-Level Smith Plot VIs

Normalize Smith Plot

Takes data for a Smith plot and normalizes it, meaning that the data is scaled relative to a known value.



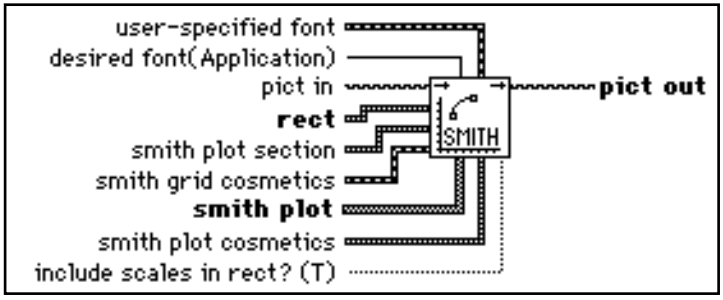
The data can be expressed as either an array of points, where each point is a cluster of an r and an x value, or an array of complex data. The VI detects which one of the data inputs is connected and uses it. If both are connected, then the VI normalizes the array of points, as shown in the following illustrations.



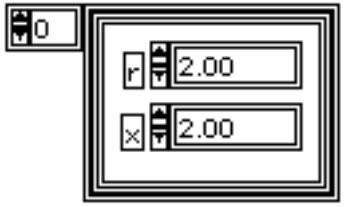
This VI normalizes the data with respect to a specified characteristic impedance (**Z_0**) of the system. Then it returns the resulting data as an array of clusters of points, where each cluster contains an r and an x value. You can pass the resulting normalized data directly to the Smith Plot VI.

Smith Plot

Takes a picture and the data for a Smith plot and appends a picture that represents the Smith plot of the data. You can specify whether r circles and x circles should be drawn for the grids. You can also specify whether the points should be drawn as single points or connected lines, and you can specify a section of the plot to zoom in on.



The **smith plot** input is an array of points, where each point is a cluster containing the r and x values that make up the complex impedance r + jx. An example of this cluster is shown in the following illustration.

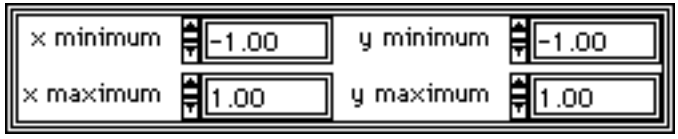


Most of the remaining controls are clusters. As described at the beginning of this chapter, the simplest way to provide data for these inputs is to use custom controls that are included in various locations, as described in the following paragraphs.

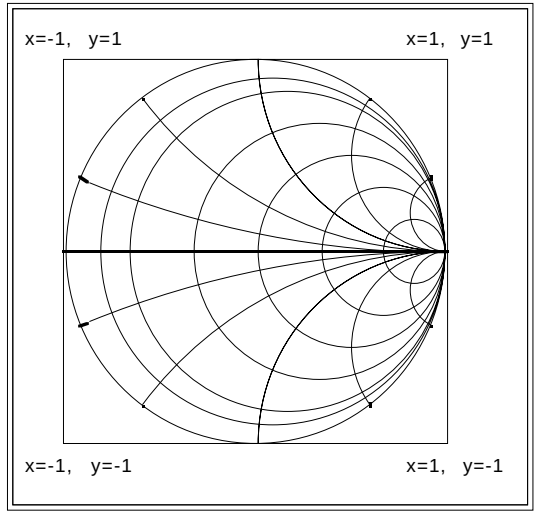
The **desired font** and **user-specified font** inputs determine the font that the scales use. You can view these scales if you zoom in on a section of the plot. The data types of **desired font** and **user-specified font** are described in Chapter 3, *Picture Controls*. You can find the corresponding controls in the **Picture** palette of the **Controls** menu. If you do not specify a font, these inputs default to the Application font.

The **rect** input is the bounding rectangle for the graph, including the scales, which are only shown if you zoom in on a section of the graph. The data type for a rectangle is described in Chapter 3, *Picture Controls*. If you do not specify the size, the VI uses a rectangle of 250 by 250 pixels.

The **smith plot section** input is a cluster that describes the portion of the graph you want to display. It contains four values that represent a rectangle superimposed over the Smith plot. You can use the `Smith Plot section.ctl` from the `polarplt.llb` to specify the information for this input. This cluster is shown in the following illustration.

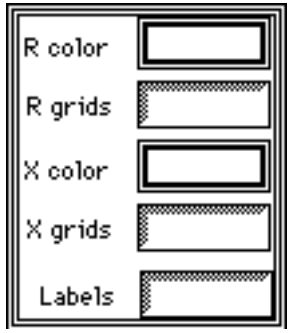


If the minimum x and y values are -1, and the maximum values are 1, the entire plot is displayed. To display the top right quadrant, you could specify minimum x and minimum y values of 0, and maximum x and y values of 1. If you specify an empty region, such as a cluster where all four values are zero, the entire plot is displayed. The following illustration shows the range of x and y values you can use.



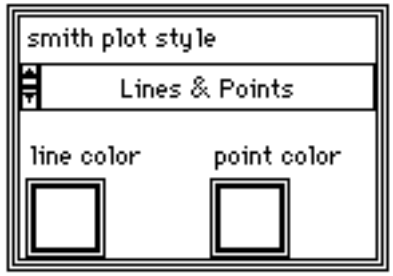
The **smith grid cosmetics** input is a cluster that describes the style to use for the grids and whether to draw labels in the plot corners when you zoom in on a section of the plot. The **smith grid cosmetics** cluster contains color numerics that specify the colors for the r circles and the x circles and Booleans that specify whether you want the r circles and x circles to be drawn.

The **smith grid cosmetics** cluster also contains a Boolean that specifies the format and color for the grid, and indicates whether labels should be drawn for the plot corners if you zoom in on a region of the plot. You can use the `Smith Grid Cosmetics.ctl` from the `smith.llb` to specify the information for this input. The **smith grid cosmetics** cluster is shown in the following illustration.

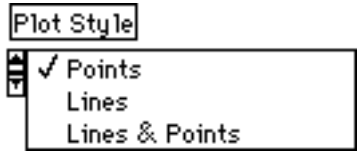


The **smith plot cosmetics** input is a cluster that specifies the style to use in drawing the plot. This cluster contains a ring with which you can select whether you want the points drawn with no connecting lines, lines only, or both lines and points. It also contains color numerics with which you can select the line color and the point color.

You can use the `Smith Plot Cosmetics.ctl` from `smith.llb` to specify the information for this input. If **smith plot cosmetics** is not wired, then the Smith Plot VI draws the plot by drawing lines and points and uses a yellow color for the lines and points This cluster is shown in the following illustration.



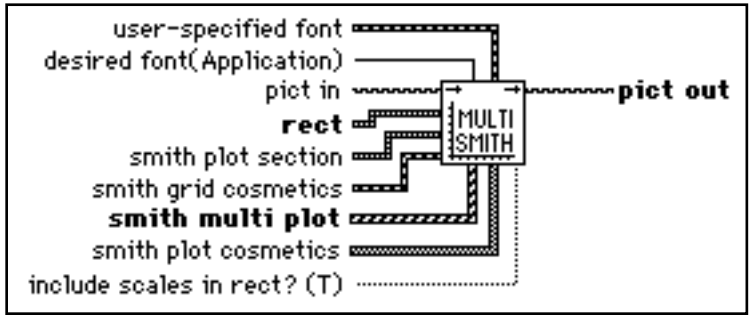
The plot style is a ring that has the options shown in the following illustration.



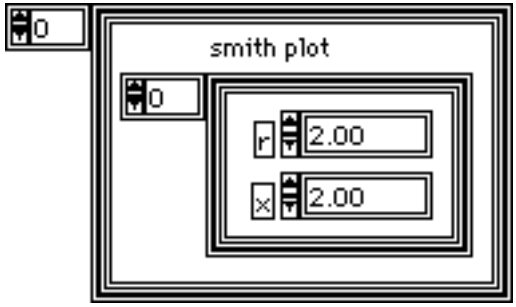
The **include scales in rect?** input is a Boolean that determines whether the **rect** input includes the scales, in addition to the plot area. If FALSE, the **rect** input describes the plot area, and the scales are drawn outside of the rectangle. If TRUE, the scales are drawn inside of **rect** and the plot area is inset enough to make room for the text. By default, this input has a value of True.

Smith Multi Plot

Takes a picture and an array of Smith plots and appends a picture that represents the superimposed Smith plots of the data. You can specify whether r circles and x circles should be drawn for the grids. You can also specify whether the points should be drawn as single points or connected lines, and you can specify a section of the plot to zoom in on.



The **smith multi plot** input is an array of clusters, where each cluster contains an array of points. A point is represented by a cluster containing the r and x values that make up the complex impedance r + jx. This cluster is shown in the following illustration.



Most of the remaining controls are clusters. As described at the beginning of this chapter, the easiest way to provide data for these inputs is to use custom controls that are included in various locations. These are described in the following paragraphs.

The **desired font** and **user-specified font**, **smith plot section**, **smith grid cosmetics**, and **smith plot cosmetics** are the same as those in the Smith Plot VI described in the previous section of this chapter. The only difference is that in the MultiSmith Plot VI, the `Smith Plot Cosmetics.ctl` is an array of controls, one for each plot.

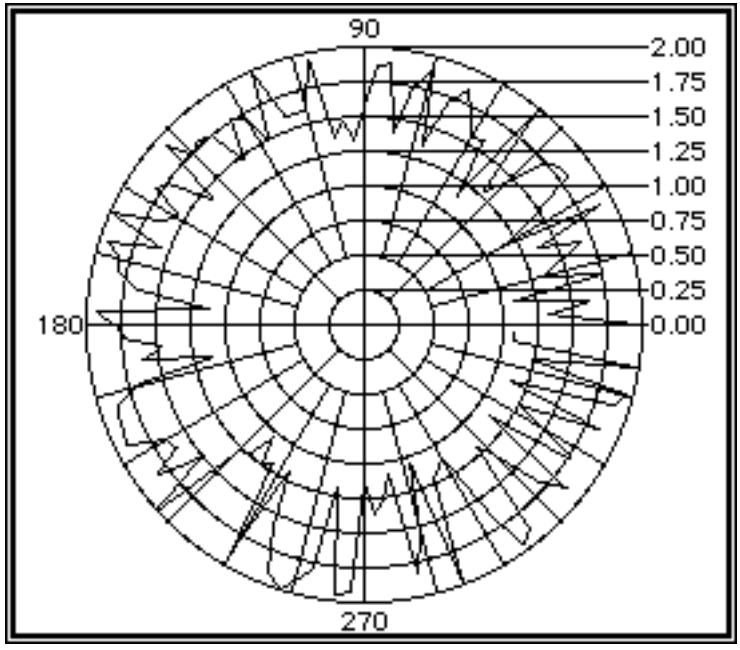
Chapter

8

Graph Scale VIs

This chapter describes the VIs in the `scale.llb` library that you can use to create scales.

The polar plot VIs and the Cartesian plot VIs use the graph scale VIs to calculate and draw scales. An example graph scale is shown in the following illustration.



These scale VIs are a good starting point if you want a Cartesian scale for a graph you are designing. You can specify the orientation, fonts, logarithmic versus linear, and the format and precision. You can also specify the fonts, color, whether you want tick marks, and whether you want a bar adjacent to and connecting the scale numbers.

The high-level VIs in the `scale.llb` are Calc Scale Specs VI and Draw Scales VI. These VIs in turn use other VIs to calculate the locations of the markers and to map points to values.

Demonstration VIs

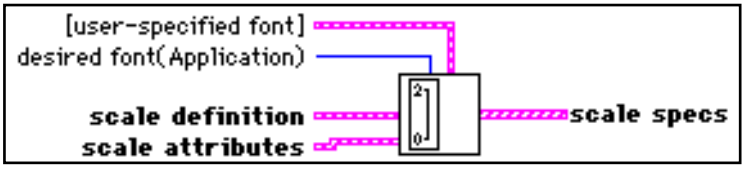
The Cartesian Scale VI in the `demos.llb` uses the Calc Scale Specs VI and Draw Scale VI (`scale.llb`) to draw a scale in a picture indicator.

The polar graph VIs (`polar.llb`) and the Cartesian graph VIs (`cartesn.llb` and `cartesn2.llb`) also use the scale VIs in the context of drawing a specific type of graph.

Using the Graph Scale Examples as SubVIs

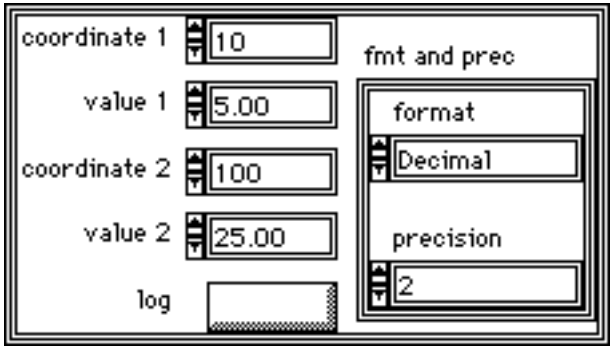
To draw a scale, you first call Calc Scale Specs VI with a description of the scale (range, bar, text format, and so on). Pass the resulting scale specification to the Draw Scale VI, and it will draw the scale into a picture control at the location you specify.

The help window for the Calc Scale Specs VI is shown in the following illustration.



This **scale definition** cluster specifies the pixel location of both endpoints of the scale, and the range of values to display on the scale. It also indicates whether you want the scale to be logarithmic or linear, and the format and precision of the markers of the scale.

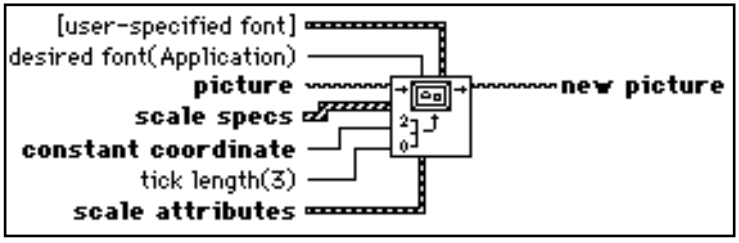
The following illustration shows the format for this input. The cluster in this illustration specifies a scale that starts at the 10th pixel. At that endpoint, it has a value of `5.00`. The other endpoint of the scale is at the 100th pixel, and it has a value of `25.00`. In addition, the scale is linear, and the scale values are displayed as decimals with two places of precision.



If the scale range and position are constant, the easiest way to create this cluster is to us the `Scale Definition.ctl` from the `scale.llb` to specify the information for this input (use **Control...** to select the control from the file system). After placing it on the front panel, configure the control to have the correct values, and then make the current value the default value.

The **scale definition** cluster specifies the range and location for the scale. You use the **scale attributes** cluster to specify the orientation (horizontal or vertical) and the colors of the scale. You can use the `Scale Attributes.ctl` to specify the value for this control.

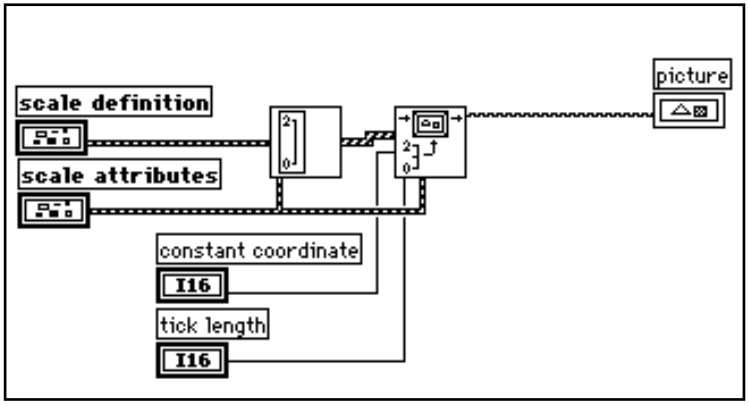
The **scale specs** output of the Calc Scale Specs VI is an array of marker values and their coordinates. You pass this information to the Draw Scale VI, which draws the scale into a picture. The help window for the Draw Scale VI is shown in the following illustration.



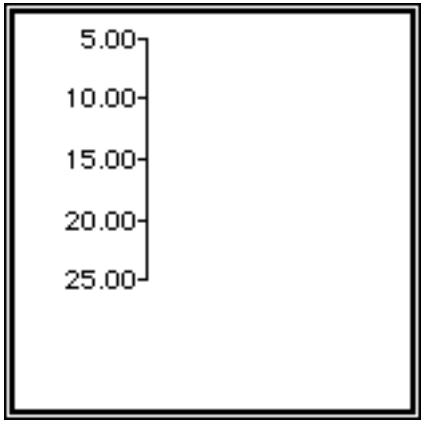
The **constant coordinate** input specifies the location of the scale. If you specify that the scale is a horizontal scale, then the **constant coordinate** specifies the y coordinate of the scale. If you specify that the scale is a vertical scale, then the **constant coordinate** specifies the x coordinate of the scale.

The **tick length** input specifies the size, in pixels, of the tick mark that is adjacent to each marker on the scale.

The following diagram shows how you can use these VIs to create a scale and draw it into a picture. In practical applications, you would concatenate the picture of the scale with the picture of a graph waveform and possibly other scales.



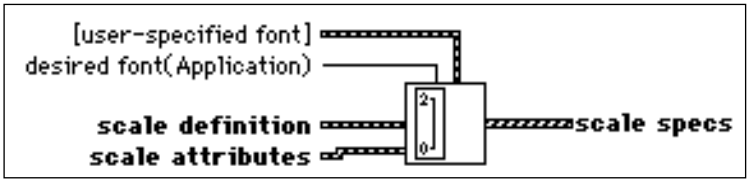
The resulting graph is shown in the following illustration.



High-Level Graph Scale VIs

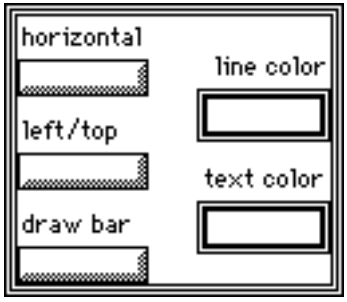
Calc Scale Specs

Takes a specification for a Cartesian style scale and calculates the coordinates and labels to use for the scale. This VI works in conjunction with the Draw Scale VI to draw a scale.



The **desired font** and **user-specified font** inputs determine the font that is used for the scales. Their data types are described in Chapter 3, *Picture Controls*. Corresponding controls are given in the **Picture** palette of the **Controls** menu. If not specified, they default to the Application Font.

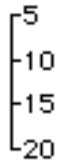
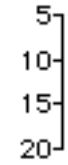
The **scale attributes** is a cluster that describes the orientation and color of the scale, as well as whether you want to draw a bar connecting the scale markers. You can use the `Scale Attributes.ctl` from the `scale.llb` to specify the information for this input. If you do not specify this information, the VI assumes the scale is vertical, with the scale numbers to the right of the scale.



The **horizontal** and **left/top** Booleans inside of the **scale attributes** cluster indicate the orientation of the scale. If **horizontal** is FALSE, the scale is drawn vertically. If **horizontal** is TRUE, the scale is drawn horizontally.

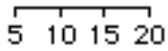
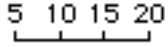
If the scale is vertical (**horizontal** is FALSE) and **left/top** is FALSE, the scale is drawn with the numbers to the right of the scale. If the scale is vertical (**horizontal** is FALSE) and **left/top** is TRUE, the scale is drawn with the numbers to the left of the scale. This is shown in the following table.

Table 8-1. Scale Orientation Options if Horizontal is FALSE

Orientation if Left/Top is FALSE	**Orientation if Left/Top is TRUE**
	

If **horizontal** is TRUE, and **left/top** is FALSE, the Calc Scale Specs VI draws the scale with the numbers below it. If **horizontal** and **left/top** are TRUE, the Calc Scale Specs VI draws the scale with the numbers above it. These two conditions are shown in the following table.

Table 8-2. Scale Orientation Options if Horizontal is TRUE

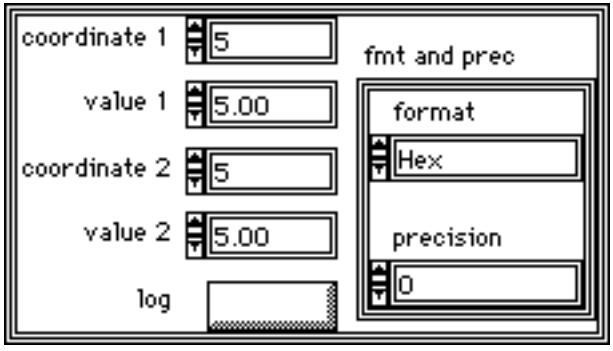
Orientation if Left/Top is FALSE	**Orientation if Left/Top is TRUE**
	

The **draw bar** Boolean inside of the **scale attributes** cluster indicates whether a bar should be drawn connecting the scale markers. The effect of this Boolean is shown in the following table.

Table 8-3. Draw Bar Options of the Scale

Appearance if Draw Bar is FALSE	**Appearance if Draw Bar is TRUE**
-5 -10 -15 -20	5 10 15 20

The **scale definition** is a cluster that describes the location and range of the scale.The **scale definition** cluster contains numerics that describe the location and value of both ends of the scale. A Boolean value indicates whether you want the scale to be logarithmic or linear. Finally, a cluster describes the format and precision of the markers of the scale. You can use the `Scale Definition.ctl` from the `scale.llb` to specify the information for this input. This cluster is shown in the following illustration.



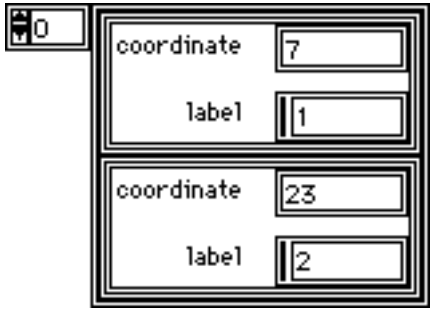
The **coordinate 1** and **coordinate 2** numerics in the **scale definition** cluster describe the location of the scale within the picture. If the scale is a horizontal scale, **coordinate 1** and **coordinate 2** describe the left and right endpoints for the scale. If the scale is a vertical scale, **coordinate 1** and **coordinate 2** describe the top and bottom endpoints for the scale.

The **value 1** and **value 2** numerics specify the range of the scale.

The **log** Boolean of the **scale definition** cluster must be FALSE if you want a linear scale, and TRUE if you want a logarithmic scale.

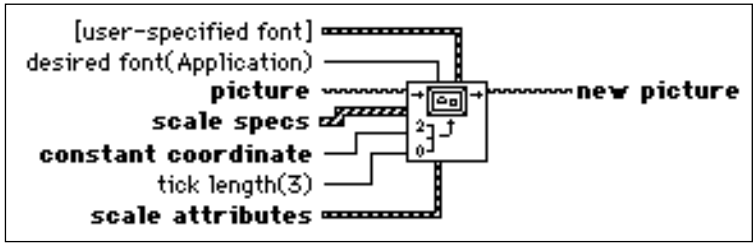
The **fmt and prec** cluster inside of the **scale definition** cluster describes the format (decimal, scientific, engineering, binary, octal, or hex) and precision (number of decimal places) of the scale numbers.

The **scale specs** is an array of clusters where each cluster describes the coordinate of a scale marker and the label to draw for that marker. This output can be passed to the Draw Scale VI to draw the scale. This array of clusters is shown in the following illustration.



Draw Scale

Takes a specification for a Cartesian style scale and a picture, and draws the scale in the picture. You use this VI in conjunction with the Calc Scale Specs VI to draw a scale.



The **desired font** and **user-specified font** inputs determine the font for the scales. Their data types are described in Chapter 3, *Picture Controls*. Corresponding controls are in the **Picture** palette of the **Controls** menu. If not specified, they default to the Application Font.

The **scale specs** is an array of clusters where each cluster describes the coordinate of a scale marker and the label to draw for that marker. This input can be created using the Calc Scale Specs VI. The `Scale Spec.ctl` is a control in the `scale.llb` with the same data structure as this input. This array is shown in the following illustration.

The **constant coordinate** input specifies the location of the scale. If you specify that the scale is a horizontal scale, then the **constant coordinate** specifies the y coordinate of the scale. If you specify that the scale is a vertical scale, then the **constant coordinate** specifies the x coordinate of the scale.

The **tick length** input specifies the length of the scale tick marks (lines on the scale next to each scale marker). By default, the **tick length** is three pixels.

The **scale attributes** input is a cluster. See the Calc Scale Specs VI earlier in this chapter for a description of this cluster.

Appendix

Customer Communication

For your convenience, this appendix contains forms to help you gather the information necessary to help us solve technical problems you might have as well as a form you can use to comment on the product documentation. Filling out a copy of the Technical Support Form before contacting National Instruments helps us help you better and faster.

National Instruments provides comprehensive technical assistance around the world. In the U.S. and Canada, applications engineers are available Monday through Friday from 8:00 a.m. to 6:00 p.m. (central time). In other countries, contact the nearest branch office. You may fax questions to us at any time.

Corporate Headquarters
(512) 795-8248
Technical Support fax: (512) 794-5678

Branch Offices	**Phone Number**	**Fax Number**
Australia	03 879 9422	03 879 9179
Austria	0662 435986	0662 437010 19
Belgium	02 757 00 20	02 757 03 11
Denmark	45 76 26 00	45 76 71 11
Finland	90 527 2321	90 502 2930
France	1 48 65 33 00	1 48 65 19 07
Germany	089 7 14 50 93	089 7 14 60 35
Italy	02 48301892	02 48301915
Japan	03 3788 1921	03 3788 1923
Netherlands	01720 45761	01720 42140
Norway	32 846866	32 846860
Spain	91 640 0085	91 640 0533
Sweden	08 730 49 70	08 730 43 70
Switzerland	056 27 00 20	056 27 00 25
U.K.	0635 523545	0635 523154
	or 0800 289877 (in U.K. only)	

Technical Support Form

Photocopy this form and update it each time you make changes to your software or hardware, and use the completed copy of this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

If you are using any National Instruments hardware or software products related to this problem, include the configuration forms from their user manuals. Include additional pages if necessary.

Name _____

Company _____

Address _____

_____

Fax (___) _____ Phone (___) _____

Computer brand _____ Model _____ Processor _____

Operating system (include version number) _____

Clock speed _____MHz RAM _____MB Display adapter _____

Mouse ___yes ___no Other adapters installed _____

Hard disk capacity _____MB Brand _____

Instruments used _____

_____

Boards installed (include revision and configuration) _____

_____

_____

LabVIEW Version ____ VI Libraries installed (other than standard libraries) _____

_____

The problem is: _____

_____

_____

List any error messages: _____

_____

The following steps reproduce the problem:_____

_____

_____

Documentation Comment Form

National Instruments encourages you to comment on the documentation supplied with our products. This information helps us provide quality products to meet your needs.

Title: **LabVIEW® Picture Control Toolkit Reference Manual**

Edition Date: **April 1994**

Part Number: **320594-01**

Please comment on the completeness, clarity, and organization of the manual.

_____

_____

_____

_____

_____

If you find errors in the manual, please record the page numbers and describe the errors.

_____

_____

_____

_____

_____

Thank you for your help.

Name _____

Title _____

Company _____

Address _____

_____

_____

Phone (____) _____

Mail to: Technical Publications
National Instruments Corporation
6504 Bridge Point Parkway, MS 53-02
Austin, TX 78730-5039

Fax to: Technical Publications
National Instruments Corporation
MS 53-02
(512) 794-5678

Glossary

Prefix	Meaning	Value
M-	mega-	10^6

A

absolute coordinates — Picture coordinates relative to the origin (0,0) of the picture indicator.

array — Ordered, indexed set of data elements of the same type.

B

Boolean — Front panel controls and indicators, such as switches, buttons and LEDs, that you use to manipulate and display or input and output Boolean (TRUE or FALSE) data.

C

cluster — A set of ordered, unindexed data elements of any data type including numeric, Boolean, string, array, or cluster. The elements must be all controls or all indicators.

D

default — A preset value. Many VI inputs use a default value if you do not specify one.

I

indicator — Front panel object that displays output.

integer — Any of the natural numbers, their negatives, or zero.

I/O — Input/output. The transfer of data to or from a computer system involving communications channels, operator interface devices, and/or data acquisition and control interfaces.

K

Kwords	1,024 words (16 bits) of memory.

L

LabVIEW	Laboratory Virtual Instrument Engineering Workbench.

M

MB	megabytes of memory.

P

picture	Series of graphics instructions that a picture indicator uses to create a picture.
picture indicator	A general-purpose graphics display indicator.
Picture VI palette	A palette of Picture VIs that you access by selecting **Picture** from the **Functions** menu.
pixel	The smallest unit of a digitized picture.
point	A cluster containing two 16-bit integers that represent horizontal and vertical coordinates.

R

rectangle	A cluster containing four 16-bit integers. The first two values describe the vertical and horizontal coordinates of the top left corner. The last two values describe the vertical and horizontal coordinates of the bottom right corner.
relative coordinates	Picture coordinates relative to the current location of the pen.

S

string	A connected sequence of characters of bits treated as a single data item.
subVI	VI used in the block diagram of another VI; comparable to a subroutine.

V

virtual instrument	LabVIEW program; so called because it models the appearance and function (VI) of a physical instrument.

Index

A

arcs, drawing, 1-10, 4-9 to 4-10
Artificial Strip Chart demonstration VI, 2-2, 5-2. *See also* Plot XY VI.

C

Calc Scale Specs VI, 8-5 to 8-8. *See also* Scale Demo demonstration VI.
Cartesian graph VIs
  Cartesian graph (illustration), 5-1
  demonstration VIs
    Artificial Strip Chart, 5-2
    cartesn2.llb library, 2-5
    cartesn.llb library, 2-4 to 2-5
    Histogram Plot, 5-2
    using as subVIs, 5-3 to 5-7
    Waveform & XY Plots, 5-2
    XY Multi Plot, 5-2
    XY Scatter Plots, 5-2
  high-level VIs
    Plot Multi-XY, 5-14 to 5-15
    Plot Waveform, 5-8 to 5-12
    Plot XY, 5-12 to 5-14
clipped pictures, 1-7
color box constant, 1-6, 1-8
color specifications, 1-7 to 1-8
Controls menu
  availability of Picture palette, 1-3
  Picture palette (illustration), 1-4
coordinate system, 1-7
custom controls. See picture custom controls.
customer communication, xii, A-1

D

Data Operations option, 3-3
data types, icons for, xi
demonstration VIs. See example VIs.
demosup.llb library, 2-4
documentation
  conventions used in manual, x-xi
  organization of manual, ix-x
  related documentation, xii
Draw 1-Bit Pixmap VI, 4-18
Draw 4-Bit Pixmap VI, 4-19
Draw 8-Bit Pixmap VI, 4-20
Draw Arc VI, 1-10, 4-9 to 4-10
Draw Cartesian Axes VI, 5-2
Draw Circle by Radius VI, 2-6
Draw Grayed Out Rect VI, 4-11
Draw Line VI, 4-4
Draw Multiple Lines VI, 2-2, 4-5
Draw Oval VI, 4-8 to 4-9
Draw Point VI, 4-2 to 4-3
Draw Rect VI
  description, 4-6
  drawing two overlapping rectangles (example), 1-5 to 1-7
  help window display (illustration), 1-5
  purpose, 1-5
Draw Round Rect VI, 4-7 to 4-8
Draw Scale VI, 8-8 to 8-9. *See also* Scale Demo demonstration VI.
Draw Text at Point VI, 4-12 to 4-14
Draw Text in Rect VI, 4-16 to 4-17
Draw XY Data VI, 5-2

E

Empty Picture constant, 1-11
Empty Picture VI, 4-2
Erase First option, 3-3
example VIs
  demonstration library, 2-2
  demonstration VIs
    Artificial Strip Chart, 2-2, 5-2
    graph scale VIs, 8-2
    Histogram Plot, 2-2, 5-2
    Picture Waterfall Spectrum, 2-2
    Polar Graph, 6-2 to 6-6
    Polar Plot, 2-2
    Scale Demo, 2-3
    Simple Smith Plot, 2-3, 7-2
    Smith Multi Plot with Styles, 2-3, 7-2
    Smith Plot with Zooming, 2-3, 7-2
    using as subVIs, 7-3 to 7-9
    Waveform & XY Plots, 2-3, 5-2
    XY Multi Plot, 2-4, 5-2
    XY Scatter Plots, 2-4, 5-2
  demosup.llb library, 2-4
  graph-related libraries
    cartesn2.llb, 2-5
    cartesn.llb, 2-4 to 2-5
    polarplt.llb, 2-5
    scale.llb, 2-5
    smith.llb, 2-5
  miscellaneous libraries
    pictutil.llb, 2-6
    robot.llb, 2-6
  organization of example libraries, 2-1
  Plot Waveform VI, 5-3 to 5-7
  saving customized VIs in new location (caution), 2-1
  using as subVIs
    Cartesian graph examples, 5-3 to 5-7
    graph scale VIs, 8-2 to 8-4
    Polar Graph VI, 6-2 to 6-6
    Smith plot VIs, 7-3 to 7-9

F

fax technical support, A-1
font control, user-specified, 3-6
font enum control, 3-6
Functions menu
  availability of Picture palette, 1-3
  Picture palette (illustration), 1-4, 4-1
  Structs & Constants palette, 1-6

G

Get Text Rect VI, 4-14 to 4-16
graph scale VIs
  demonstration VIs, 8-2
  graph scale (illustration), 8-1
  high-level VIs
    Calc Scale Specs, 8-5 to 8-8
    Draw Scale, 8-8 to 8-9
  scale.llb library, 2-5
  using examples as subVIs, 8-2 to 8-4
grayed-out rectangle, drawing, 4-11

H

Hilite Color VI, 2-6
Histogram Plot demonstration VI, 2-2, 5-2. *See also* Plot Waveform VI.

I

installation
  Macintosh computers, 1-3
  Sun SPARCstations, 1-2 to 1-3
  Windows, 1-1 to 1-2

L

LabVIEW Picture Control Toolkit. See Picture Control Toolkit.
libraries. See example VIs.
lines, drawing
  Draw Line VI, 4-4
  Draw Multiple Lines VI, 2-2, 4-5

M

Macintosh computers
installing Picture Control Toolkit, 1-3
manual. See documentation.
Move Pen VI, 4-3

N

Normalize Smith Plot VI
description, 7-9 to 7-10
using as subVI, 7-6

O

oval, drawing, 4-8 to 4-9

P

pen
concept of, 1-9
moving, 4-3
VIs capable of moving, 1-9
picture attributes, 3-4
Picture Control Toolkit
basic concepts, 1-3 to 1-8
color specifications, 1-7 to 1-8
coordinate system, 1-7
definition, 1-1
installation
Macintosh computers, 1-3
Sun workstations, 1-2 to 1-3
Windows environment, 1-1 to 1-2
picture controls. *See also* picture custom controls.
block diagram terminal (illustration), 3-2
coordinate system, 1-7
definition, 3-2
illustration, 1-4, 3-2
menu selection (illustration), 3-1
overview, 3-1
picture custom controls
font enum control, 3-6
point control, 3-5
rectangle control, 3-4
text alignment control, 3-5
user-specified font control, 3-6
picture indicator
displaying image, 1-5
purpose and use, 3-2
picture options
Data Operations, 3-3
Erase First, 3-3
pop-up menu (illustration), 3-3
Smooth Updates, 3-3
Picture palette
availability on Controls and Functions menus, 1-3
illustration, 1-4, 1-8, 4-1
picture VIs
Draw 1-Bit Pixmap, 4-18
Draw 4-Bit Pixmap, 4-19
Draw 8-Bit Pixmap, 4-20
Draw Arc, 4-9 to 4-10
Draw Grayed Out Rect, 4-11
Draw Line, 4-4
Draw Multiple Lines, 4-5
Draw Oval, 4-8 to 4-9
Draw Point, 4-2 to 4-3
Draw Rect, 4-6
Draw Round Rect VI, 4-6 to 4-7
Draw Text at Point, 4-12 to 4-14
Draw Text in Rect, 4-16 to 4-17
Empty Picture, 4-2
Function Picture palette (illustration), 4-1
Get Text Rect, 4-14 to 4-16
Move Pen, 2-2, 4-3
overview, 1-8 to 1-11
empty picture constant, 1-11
lines, 1-9
pens, 1-9
Picture palette (illustration), 1-8
pixmaps, 1-11
points, 1-9
shapes, 1-9 to 1-10
text, 1-10
Picture Waterfall Spectrum, 2-2

pictutil.llb library, 2-6
pixmap VIs
  Draw 1-Bit Pixmap, 4-18
  Draw 4-Bit Pixmap, 4-19
  Draw 8-Bit Pixmap, 4-20
pixmaps
  definition, 1-11
  VIs for drawing, 1-11
Plot Multi-XY VI
  description, 5-14 to 5-15
  similarity with Plot Waveform VI, 5-7
Plot Waveform VI
  description, 5-8 to 5-12
  similarity with Plot XY and Plot multi-XY VIs, 5-7
  using as subVI, 5-3 to 5-7
Plot XY VI
  description, 5-12 to 5-14
  similarity with Plot Waveform VI, 5-7
  subVIs
    Draw Cartesian Axes VI, 5-2
    Draw XY Data VI, 5-2
point control, 3-5
points
  definition, 1-7
  drawing, 4-2 to 4-3
polar graph (illustration), 6-1
Polar Graph VI
  description, 6-6 to 6-9
  polarplt.llb library, 2-5
  using as subVI, 6-2 to 6-6
  visible section values, 2-8 to 2-9
Polar Plot demonstration VI, 2-2

R

rectangle control, 3-4
rectangle VIs
  Draw Grayed Out Rect VI, 4-11
  Draw Rect VI, 4-6
  Draw Round Rect VI, 4-6 to 4-7
  Draw Text in Rect, 4-16 to 4-17
  Get Text Rect, 4-14 to 4-16
robot.llb library, 2-6
rounded rectangle, drawing, 4-6 to 4-7

S

Scale Demo demonstration VI, 2-3
scale VIs. See graph scale VIs.
shapes, VIs for drawing, 1-9 to 1-10
Simple Smith Plot demonstration VI, 2-3, 7-2
Smith Multi Plot VI, 7-14 to 7-15
Smith Multi Plot with Styles demonstration VI, 2-3, 7-2
Smith Plot VI
  description, 7-10 to 7-14
  using as subVI, 7-4 to 7-5
Smith plot VIs
  demonstration VIs
    Simple Smith Plot, 7-2
    Smith Multi Plot with Styles, 7-2
    Smith Plot with Zooming, 7-2
    using examples as subVIs, 7-3 to 7-9
  high-level VIs
    Normalize Smith Plot, 7-9 to 7-10
    Smith Multi Plot, 7-14 to 7-15
    Smith Plot, 7-10 to 7-14
    overview, 7-2 to 7-3
  Smith plot (illustration), 7-1
  smith.llb library, 2-5
Smith Plot with Zooming demonstration VI, 2-3, 7-2
Smooth Updates option, 3-3
Structs & Constants palette, 1-6
subVIs, using examples as. See example VIs.
Sun SPARCstations
  installing Picture Control Toolkit, 1-2 to 1-3

T

technical support, A-1
text alignment control, 3-5
text VIs
  Draw Text at Point, 4-12 to 4-14
  Draw Text in Rect, 4-16 to 4-17

Get Text Rect VI, 4-14 to 4-16
VIs for drawing text, 1-10

U

user-specified font control, 3-6

V

visible section values, Polar Graph VI, 2-8 to 2-9

W

waterfall plot, 2-2
Waveform & XY Plots demonstration VI, 2-3, 5-2. *See also* Plot Waveform VI; Plot XY VI.
Windows environment
installing Picture Control Toolkit, 1-1 to 1-2

X

XY graph. See Plot Multi-XY VI; Plot XY VI.
XY Multi Plot demonstration VI, 2-4, 5-2. *See also* Plot Multi-XY VI.
XY Scatter Plots demonstration VI, 2-4, 5-2. *See also* Draw Cartesian Axes VI; Draw XY Data VI.