HiQ™

Version 4.1

These notes contain information about system requirements, installation, new features, upgrade information, and updated documentation for HiQ 4.1.

Contents

321967B-01 *January 1999*

System Requirements

HiQ requires the following minimum system configuration:

- Windows 95 or Windows 98/NT 4.0
- 486 CPU with coprocessor
- 8 MB RAM with Windows 95, 16 MB RAM with Windows 98/NT
- 256-color, 800 by 600 VGA display
- 20–60 MB free disk space (You need 20 MB for HiQ Reader, 40 MB for HiQ, and 60 MB for HiQ Professional.)

The following specifications are the recommended system configuration for HiQ:

- Windows NT 4.0
- Pentium 90 or higher
- 16 MB RAM with Windows 95, 32 MB RAM with Windows 98/NT
- Accelerated True Color, 1024 by 768 display
- 20–60 MB free disk space (You need 20 MB for HiQ Reader, 40 MB for HiQ, and 60 MB for HiQ Professional.)

Installing HiQ

Use the following procedure to install HiQ:

1. Insert the CD in the CD-ROM drive of your computer.
2. Follow the *Setup* instructions you see on your screen.

By default, the HiQ installation routine creates a new folder, `C:\Program Files\National Instruments\HiQ`, that contains the following items:

- Program folder—`HiQ.exe`, HiQ help files, and related files.
- Examples folder—Example Notebooks demonstrating many of the analysis and visualization capabilities of HiQ, organized by category.
- `Readme.txt` file—Late-breaking information about HiQ, known bugs, and corrections.
- Manuals folder—PDF versions of the HiQ manuals.

New Features

HiQ version 4.1 adds two new products—HiQ Professional and HiQ Reader—and several new features, as described below.

HiQ Professional

HiQ Professional 4.1 includes the following:

- HiQ Signal Processing Toolkit—The HiQ Signal Processing Toolkit adds over 40 functions for signal processing including filtering, windowing, transforms, signal generation, time domain analysis, and frequency domain analysis.
- ComponentWorks User Interface ActiveX controls—A collection of user interface controls, including buttons, knobs, sliders, and other controls, for use on the HiQ Notebook.
- ComponentWorks DataSocket ActiveX control—An ActiveX control you can use to access data on the Internet and other DataSocket servers.
- ComponentWorks 3D Graph ActiveX control—An ActiveX control that contains the power of the native HiQ 3D Graph and works in other ActiveX containers.
- Serial Interface ActiveX control—An ActiveX control that enables the Notebook to retrieve data from the serial port.
- HiQ Reader—The HiQ Reader enables others to view your Notebooks and run scripts but protects your Notebook from changes.

HiQ Reader

The HiQ Reader is freely distributable and allows users to view and interact with HiQ Professional Notebooks. You can use the HiQ Reader to share Notebooks with others that do not have HiQ. They can run scripts, interact with the graphs and other objects, save, and print but cannot modify the format of the Notebook, create new objects, or create new notebooks. The HiQ Reader also can be downloaded from the National Instruments Web site (`www.natinst.com`).

Problem Solvers

The HiQ Problem Solvers take advantage of ComponentWorks ActiveX controls to make the problem solving environment easier and more intuitive. You can find the Problem Solvers in `\Program Files\National Instruments\HiQ\Examples\Problem Solvers`.

Notebook Help

You can create customized Notebook help files that users can access through the **Help** menu. If a file exists in the same directory as the current Notebook that has the same name but a different extension, HiQ opens that file when you select **Help»Help on Current Notebook**.

For example, you can create a Web page help file (`.htm`) that opens your default Web browser, a compiled HTML help file (`.chm`) that opens the Microsoft Compiled HTML Help Viewer, or a Windows help file (`.hlp`) that opens a Windows help system.

Notebook help files are useful for describing the background and purpose of a Notebook so that you do not have to include this information in the Notebook itself.

Built-in Functions

HiQ 4.1 contains new functions for performing bitwise operations on integers and getting and setting object values referenced by name. The following built-in functions are new:

- `bitAND`
- `bitOR`
- `bitShift`
- `bitXOR`
- `getValue`
- `parity`
- `setValue`

bitAND

Purpose

Computes the bitwise AND operation on two integers.

Usage

```
z = bitAND(x, y)
```

Parameters

Name	Type	Description
x	Integer Scalar	Integer parameter
y	Integer Scalar	Integer parameter
z	Integer Scalar	Result of bitwise ANDing the two inputs

Comments

The input integers are treated as unsigned integers.

See Also

bitOR, bitShift, bitXOR, parity

bitOR

Purpose

Computes the bitwise OR operation on two integers.

Usage

```
z = bitOR(x, y)
```

Parameters

Name	Type	Description
x	Integer Scalar	Integer parameter
y	Integer Scalar	Integer parameter
z	Integer Scalar	Result of bitwise ORing the two inputs

Comments

The input integers are treated as unsigned integers.

See Also

bitAND, bitShift, bitXOR, parity

bitShift

Purpose

Shifts the bits in an integer.

Usage

```
y = bitShift(x, n)
```

Parameters

Name	Type	Description
x	Integer Scalar	Input parameter
n	Integer Scalar	Number of bits to shift
y	Integer Scalar	Result of bit-shifting the input

Comments

If n is positive, the bits are shifted left. If n is negative, the bits are shifted right. The number of bits shifted is n modulo 32. The input integer is treated as an unsigned integer.

See Also

bitAND, bitOR, bitShift, bitXOR

bitXOR

Purpose

Computes the bitwise exclusive or (XOR) operation on two integers.

Usage

```
z = bitXOR(x, y)
```

Parameters

Name	Type	Description
x	Integer Scalar	Integer parameter
y	Integer Scalar	Integer parameter
z	Integer Scalar	Result of bitwise XORing the two inputs

Comments

The input integer is treated as an unsigned integer.

See Also

bitAND, bitOR, bitShift, parity

getValue

Purpose

Gets the value of an object reference by name.

Usage

```
y = getValue(name)
```

Parameters

Name	Type	Description
name	Text	Name of the object
y	Object	Value of the named object

Comments

All objects directly referenced in a script become locked for the duration of that script, which means that you cannot set or get the value of those objects through the user interface or the Command Window while the script is running. If you use `setValue` or `getValue` to reference an object in a script, you can set or get the value of that object through the user interface while the script is running. Because `setValue` and `getValue` reference the object indirectly by name (specified by enclosing the object name in quotes in the script), the object is locked while these functions are getting or setting the value of the object and is unlocked throughout the remainder of the script execution.

Notebook users can attempt to set or get object values through the user interface or Command Window while a script is running, but they can access the object value only when it is unlocked. Because scripts execute quickly, you might need to call the `wait` built-in function to give users time to access the object through the interface, as shown in the following script.

```
//This script performs an operation on an object only when
//the value of the object changes. x is an integer scalar,
//and if the type changes, the script fails.
assume local;
xLast = 0;
x = 0;
repeat forever
   x = getValue("x");
   if x != xLast then
      sin(x);
      xLast = x;
   end if;
   wait(0.2);
end repeat;
```

See Also

setValue

parity

Purpose

Computes the parity of an integer.

Usage

```
i = parity(x)
```

Parameters

Name	Type	Description
x	Integer Scalar	Input parameter
I	Integer Scalar	0 if there is an even number of 1s in the input. 1 if there is an odd number of 1s in the input.

Comments

The input integer is treated as an unsigned integer.

See Also

bitAND, bitOR, bitShift, bitXOR

setValue

Purpose

Sets the value of an object referenced by name.

Usage

```
setValue(name, x)
```

Parameters

Name	Type	Description
`name`	Text	Name of the object whose value to set
`x`	Object	New value of the object

Comments

All objects directly referenced in a script become locked for the duration of that script, which means that you cannot set or get the value of those objects through the user interface or the Command Window while the script is running. If you use `setValue` or `getValue` to reference an object in a script, you can set or get the value of that object through the user interface while the script is running. Because `setValue` and `getValue` reference the object indirectly by name (specified by enclosing the object name in quotes in the script), the object is locked while these functions are getting or setting the value of the object and is unlocked throughout the remainder of the script execution.

Notebook users can attempt to set or get object values through the user interface or Command Window while a script is running, but they can access the object value only when it is unlocked. Because scripts execute quickly, you might need to call the `wait` built-in function to give users time to access the object through the interface, as shown in the following script.

```
//This script performs an operation on an object only when
//the value of the object changes. x is an integer scalar,
//and if the type changes, the script fails.
assume local;
xLast = 0;
x = 0;
repeat forever
   x = getValue("x");
   if x != xLast then
      sin(x);
      xLast = x;
   end if;
   wait(0.2);
end repeat;
```

See Also

`getValue`

Script Error Messages

Script run-time error messages are now displayed in the Log Window. The last error that occurs is still displayed in a dialog box, however the Log Window displays a complete traceback of the error. This is very useful when an error occurs in a function that exists in a script other than the one that is executing.

Automation Methods

Two new automation methods have been added to HiQ 4.1: `CompileScript` and `RunScriptEx`.

CompileScript

Compiles a script and returns any errors that occur.

```
Status = Notebook.CompileScript(ScriptName as String, ErrorBegin as
  Integer, ErrorEnd as Integer, ErrorMessage as String)
```

Parameter	Description
`ScriptName`	Name of the script to compile
`ErrorBegin`	Zero-based index into the script text where the error begins
`ErrorEnd`	Zero-based index into the script text one character beyond where the error ends
`ErrorMessage`	HiQ error message
`Status`	Result of the operation If `Status` is not equal to zero, an error occurred.

For example, the following Visual Basic code launches HiQ, loads the `Automation.HiQ` Notebook, sets the value of two vector objects, sets the value of the script object `newScript`, compiles the script, and saves the notebook.

```
Dim HiQApp As Object
Dim Notebook As Object
Dim Script As String
Dim x(10) As Double
Dim y(10) As Double
Dim errorStart As Long
Dim errorend As Long
Dim errorMessage As String

Set HiQApp = CreateObject("HiQ.Application")
HiQApp.Visible = True
Set Notebook = HiQApp.Open("c:\my documents\Automation.HiQ")
Status = Notebook.setdata("x", x)
```

```
Status = Notebook.setdata("y", y)
Script = "addPlot(myGraph,x,y)"
Status = Notebook.setscript("newScript", Script)
If (Notebook.CompileScript("newscript", errorStart, errorend,
  errorMessage)) Then
    MsgBox ("Compilation error: " + errorMessage + vbCrLf + Mid(Script,
      errorStart + 1, errorend - errorStart))
End If
If (Notebook.RunScriptEx("newscript", errorStart, errorend,
  errorMessage)) Then
    MsgBox ("Run-time error: " + errorMessage + vbCrLf + Mid(Script,
      errorStart + 1, errorend - errorStart))
End If
```

RunScriptEx

Runs a script and returns any errors that occur.

```
Status = Notebook.RunScriptEx(ScriptName as String, ErrorBegin as
  Integer, ErrorEnd as Integer, ErrorMessage as String)
```

Parameter	Description
ScriptName	Name of the script to run
ErrorBegin	Zero-based index into the script text where the error begins
ErrorEnd	Zero-based index into the script text one character beyond where the error ends
ErrorMessage	HiQ error message
Status	Result of the operation If Status is not equal to zero, an error occurred.

For example, the following Visual Basic code launches HiQ, loads the `Automation.HiQ` Notebook, sets the value of two vector objects, sets the value of the script object `newScript`, runs the script, and saves the notebook.

```
Dim HiQApp As Object
Dim Notebook As Object
Dim Script As String
Dim x(10) As Double
Dim y(10) As Double
Dim errorStart As Long
Dim errorend As Long
Dim errorMessage As String

Set HiQApp = CreateObject("HiQ.Application")
HiQApp.Visible = True
Set Notebook = HiQApp.Open("c:\my documents\Automation.HiQ")
Status = Notebook.setdata("x", x)
Status = Notebook.setdata("y", y)
Script = "addPlot(myGraph,x,y)"
Status = Notebook.setscript("newScript", Script)
If (Notebook.CompileScript("newscript", errorStart, errorend,
  errorMessage)) Then
    MsgBox ("Compilation error: " + errorMessage + vbCrLf + Mid(Script,
      errorStart + 1, errorend - errorStart))
End If
If (Notebook.RunScriptEx("newscript", errorStart, errorend,
  errorMessage)) Then
    MsgBox ("Run-time error: " + errorMessage + vbCrLf + Mid(Script,
      errorStart + 1, errorend - errorStart))
End If
```

Block Comment and Tab

You can comment, uncomment, tab, and untab multiple lines of script in Script objects. To comment, uncomment, tab, or untab multiple lines of script, use the following steps:

1. Select the lines of script.
2. Right click on the script or pull down the **Script** menu.
3. Select the appropriate action.

Script and Text Properties

Script and text objects have two new properties.

Property	Data Type	Description
`numlines`	Integer	Number of lines in a text or script object
`line(`*n*`)`	Text	Sets or returns the text or script on line *n* of the object

LabVIEW HiQ-Script Node

LabVIEW 5.1 has a HiQ-Script node that allows you to take advantage of the analysis and visualization capabilities of HiQ-Script on the block diagram of a virtual instrument. This script node requires HiQ 4.1 to be installed.

Automatic User Functions

HiQ automatically executes special user functions (if present) when specific Notebook events occur. The user functions `onOpen()`, `onSave()`, and `onClose()` are executed when the Notebook is opened, saved, or closed, respectively. These functions do not take any parameters and do not return a value. You can use these functions to automatically perform tasks such as initializing ActiveX controls, logging save messages, or returning objects to default values, as in the following examples.

```
function onOpen()
   assume project;
   myComboBox.additem("Select One…");
   myComboBox.additem("Low Pass");
   myComboBox.additem("Band Pass");
   myComboBox.additem("High Pass");
end function;
```

```
function onSave()
   assume project;
   logMessage("last saved on "+date()+" at "+time());
end function;

function onClose()
   assume project;
   x = xDefault;
   y = yDefault;
end function;
```

Complex Objects with ActiveX Controls

Complex objects now can be passed to and from ActiveX controls in the HiQ Notebook. HiQ packages a complex object as a SAFEARRAY containing three VARIANT elements as defined in Tables 1–3.

Table 1. Complex Scalar

VARIANT Data Type	Description
Integer (VT_I4)	Type identifier. Currently the only valid value is 13, indicating complex data.
Real (VT_R8)	Real part of the complex scalar.
Real (VT_R8)	Imaginary part of the complex scalar.

Table 2. Complex Vector

VARIANT Data Type	Description
Integer (VT_I4)	Type identifier. Currently the only valid value is 13, indicating complex data.
1D Real Array (VT_R8 | VT_ARRAY)	Real part of the complex vector.
1D Real Array (VT_R8 | VT_ARRAY)	Imaginary part of the complex vector.

Table 3. Complex Matrix

VARIANT Data Type	Description
Integer (VT_I4)	Type identifier. Currently the only valid value is 13, indicating complex data.
2D Real Array (VT_R8 | VT_ARRAY)	Real part of the complex matrix.
2D Real Array (VT_R8 | VT_ARRAY)	Imaginary part of the complex matrix.

Open in Reader

As you develop Notebooks that you want to share with others using the HiQ Reader, preview the Notebooks in the HiQ Reader to verify that they look and behave as you expect. HiQ Professional Notebooks can be opened in the HiQ Reader from the HiQ Professional version. Choose **Edit»Open in Reader** from HiQ Professional to preview the current Notebook in the HiQ Reader.

Modified Features

ActiveX Visible Property

Setting the HiQ application `visible` property to true now brings the HiQ window to the top.

Complex Objects with ActiveX Clients

Complex objects now can be passed to and from ActiveX clients and servers. HiQ packages a complex object as a SAFEARRAY containing three VARIANT elements as defined in Tables 4–6.

The HiQ Notebook object method `SetData` creates a complex object if the `Value` parameter is a VARIANT SAFEARRAY as defined in Tables 4–6.

Table 4. Complex Scalar

VARIANT Data Type	Description
Integer (VT_I4)	Type identifier. Currently the only valid value is 13, indicating complex data.
Real (VT_R8)	Real part of the complex scalar.
Real (VT_R8)	Imaginary part of the complex scalar.

Table 5. Complex Vector

VARIANT Data Type	Description
Integer (VT_I4)	Type identifier. Currently the only valid value is 13, indicating complex data.
1D Real Array (VT_R8 | VT_ARRAY)	Real part of the complex vector.
1D Real Array (VT_R8 | VT_ARRAY)	Imaginary part of the complex vector.

Table 6. Complex Matrix

VARIANT Data Type	Description
Integer (VT_I4)	Type identifier. Currently the only valid value is 13, indicating complex data.
2D Real Array (VT_R8 | VT_ARRAY)	Real part of the complex matrix.
2D Real Array (VT_R8 | VT_ARRAY)	Imaginary part of the complex matrix.

Built-in Functions

The built-in functions `random` and `createView` have new functionality. `Random` now creates random vectors and matrices. `CreateView` now places and sizes the object view on the screen. The following function definitions are updated to reflect these changes.

random

Purpose

Generates a random number.

Usage

Generates a random number between zero and one.

```
y = random()
```

Generates a random number with uniform distribution within the specified range.

```
y = random(a, b, <uniform>)
y = random(dim, a, b, <uniform>)
```

Generates a random number with normal distribution.

```
y = random(xMean, xStddev, <normal>)
y = random(dim, xMean, xStddev, <normal>)
```

Generates a random number with exponential distribution.

```
y = random(k, <exp>)
y = random(dim, k, <exp>)
```

Generates a random number with Bernoulli distribution.

```
y = random(p, <bernoulli>)
y = random(dim, p, <bernoulli>)
```

Parameters

Name	Type	Description
a	Real Scalar	The minimum value for the uniform distribution.
b	Real Scalar	The maximum value for the uniform distribution.
dim	Integer Vector or Matrix	The dimension of the vector or matrix to create.
xMean	Real Scalar	The mean of the normal distribution.
xStddev	Real Scalar	The standard deviation of the normal distribution.
k	Real Scalar	The reciprocal of the average of the exponential distribution.

Name	Type	Description
p	Real Scalar	The probability of ones occurring in the distribution.
y	Real Scalar, Vector, or Matrix	A real random number.

Comments

The usage `random()` generates a random number over the interval [0, 1] using the fast Knuth algorithm.

If `dim` is a vector, `y` is a vector of random numbers. If `dim` is a matrix, `y` is a matrix of random numbers. The following table identifies how `random` creates `y` based on `dim`. The parameter `dim` must contain at most two elements.

dim		y
object type	**example**	
vector	`{v:m}`	random vector containing `m` elements
vector	`{v:m,n}`	random vector containing `m*n` elements
matrix	`{m}`	random matrix containing `m` rows and 1 column
matrix	`{m,n} or {m;n}`	random matrix containing `m` rows and `n` columns

HiQ automatically seeds the random number generator before a script executes. You should manually seed the random number generator once with the `seed` function when you want to duplicate a random sequence.

Usage	Probability Density
`random(a, b, <uniform>)`	$\frac{1}{b-a}, a < x < b$ 0, otherwise
`random(a, b, <normal>)`	$\frac{1}{(2\pi b^2)^{0.5}} e^{-\left(\frac{(x-a)^2}{2b^2}\right)}$

Usage	Probability Density
`random(k, <exp>)`	$\frac{1}{k}e^{-\frac{x}{k}}$
`random(p, <bernoulli>)`	$\frac{1}{\sqrt{2\pi}b}e^{(x-a)^2/(2b^2)}$

See Also

`createMatrix, createVector, seed`

createView

Purpose

Creates a view of an object in a separate window.

Usage

`createView(x, pause, position, size)`

Parameters

Name	Type	Description
`x`	Object	Object to be viewed.
`pause`	HiQ Constant	Specifies whether to pause the current script while the view is still visible. (Optional. Default = `false`). Valid values include `true` and `false`.
`position`	Integer Vector	Position, in pixels, of the upper-left corner of the view. (Optional.)
`size`	Integer Vector	Size, in pixels, of the view. (Optional.)

Comments

This function does not work for objects with local scope.

If *`pause`* is `false`, the script continues to execute. If *`pause`* is `true`, the view has a **continue** button and the script pauses execution until the **continue** button is pressed. The view closes when the **continue** button is pressed.

The parameter *`position`* must contain two elements specifying the pixel location of the upper-left corner of the window from the upper-left corner of the screen. The parameter *`size`* must contain two elements specifying the pixel width and height of the view.

See Also

`updateViews, wait`

Improvements

File Open/Save

Notebook opening and saving performance has improved. Large Notebooks now open and save much faster.

3D Graphs

The calculation of surface normals has improved, which affects the surface-normal plot style and lighting. You might see some differences between graphs in HiQ version 4.0 and version 4.1 if you are using these features.

The calculation of shaded color maps has improved. In some cases, you might see some differences between shaded graphs in HiQ version 4.0 and version 4.1 if the fill color has a luminescence greater than 120. You can check this value from the Fill property page (on the Plot property page) by clicking **Other** on the fill color.

Using HiQ with LabVIEW

You can combine the data acquisition and instrument control capabilities of LabVIEW with the powerful data analysis, 3D visualization, and report generation features of HiQ.

HiQ includes a high-level ActiveX automation interface that allows LabVIEW applications to control HiQ through a set of virtual instruments (VIs) that support operations such as the following:

- Launching HiQ
- Opening a notebook
- Setting and getting scalar, vector, matrix and text data
- Running a script
- Printing a notebook

These VIs appear in the Communication functions palette in LabVIEW.

Refer to the LabVIEW manuals and online reference for complete information about the HiQ VIs. In addition, there are several example VIs and HiQ Notebooks in the LabVIEW `Examples\Comm\HiQ` folder that demonstrate this link.

☞ **Note** ***You must use LabVIEW 5.0 or greater.***

Upgrade Issues

Loading Notebooks

You can open Notebooks created with HiQ 3.*x* and HiQ 4.0 in HiQ 4.1. However, Notebooks saved with HiQ 4.1 cannot be opened with HiQ 3.*x* or HiQ 4.0. If you load a HiQ 3.*x* Notebook into HiQ 4.*x* that contains multiple views of the same numeric, text, or script object, you might notice changes in one or more of these views because multiple views of the same object in HiQ 4.*x* now share the same set of properties.

Zooming Graphs

HiQ 4.*x* now uses the <Alt> key instead of the <Ctrl> key to zoom a 3D graph using the mouse. Use the <Ctrl> key and your mouse to move views of an object.

Root Built-In Function

In the usages for the Newton and Muller options for the built-in function `root`, if you specify any parameter to the right of `ftolr`, you also must provide `ftolr`.

Using Vector Objects in Matrix Initialization and Linear Algebra

HiQ 4.*x* interprets vector objects as single-column matrices in linear algebra operations and matrix initialization. For example, given three five-element vectors `v1`, `v2`, and `v3`, the following HiQ 4.*x* script creates a 5-by-3 matrix whose columns are `v1`, `v2`, and `v3`:

```
M = {v1, v2, v3};
```

In HiQ 3.1, the above script creates a 3-by-5 matrix whose rows are `v1`, `v2`, and `v3`.

The following HiQ 4.*x* script creates the same 3-by-5 matrix:

```
M = {v1';v2';v3'};
```

Notes for Windows 95 Users

The following limitations exist on Windows 95:

- You cannot use more than 32 different fonts per page.
- You cannot set the virtual page size greater than 32x32 inches (800x800 millimeters).
- Concurrent use of HiQ and other graphic-intensive programs might exhaust system resources.

☞ **Note** ***These limitations do not exist on Windows NT.***

Notes for Windows NT 3.51 Users

HiQ 4.*x* uses Microsoft OpenGL version 4.0 for advanced 3D visualization. Because this version of OpenGL is not supported by Microsoft Windows NT 3.51, you must upgrade to Microsoft Windows NT 4.0 or Windows 95.

Important Issues

The following sections list important issues about HiQ performance regarding graphics, printing, ActiveX support, and font objects.

Graphics

- 3D graphs draw slowly on some systems running in 64k color mode. To improve performance, try switching to 256 colors or True Color (24-bit or higher).
- Contour labels are not drawn when a 3D plot style is set to surface-contour.
- Certain display adapters, including the Matrox Millennium, exhibit incorrect behavior when the **Use 3D Acceleration** option is selected in the Preferences dialog box. If you experience problems, deselect this option.

Printing

- Printing large 3D graphs to high-resolution output devices might require a long print time. Although certain printers do not correctly support this feature, the **Compress Bitmap Images** option in the Preferences dialog box can drastically reduce the print time.
- If you experience problems printing in HiQ, verify that you have the latest printer driver installed on your system. You can request the latest driver from the manufacturer of the printer. If your problem persists, contact National Instruments for technical support.
- Certain PCL printer drivers do not support some operations that HiQ requires in order to produce accurate, high-resolution output. If you experience a problem and your printer supports Postscript, use the Postscript driver instead. Often you can improve output by configuring the printer to print text as graphics.

ActiveX Support

- HiQ does not support the **Create From File** option when embedding a HiQ Notebook into another application.

- Do not interact with an instance of HiQ that is being controlled by another application via ActiveX automation.

Creating Font Objects Programmatically

If you create Font objects programmatically, do not use compound font names that specify both the face and the style of the font. Instead, specify only the face name and set the style property separately to prevent errors when the system maps the font name to an actual font. For example, use

```
myFont = {font: "Arial",10};
myFont.Style = <bold>;
```

rather than

```
myFont = {font: "Arial Bold",10};
```

User Feedback

National Instruments is interested in your applications, suggestions, and any problems you might encounter while using HiQ 4.*x*. To report a problem or suggestion, complete the following steps.

1. Complete the Technical Support Form located in the online help.
2. Email the form to `hiq_support@natinst.com` or fax to (512) 794-5499.

HiQ Reference Manual Clarifications and Additions

- You can set the font style for the title of a graph, as in the following example. The `title.font` property should have appeared in Table 3-2, *Graph Properties*, and Table 4-15, *Graph Object Properties*.

  ```
  graph.title.font = {font: "Arial", 10};
  ```

- Table 3-3, *Plot Properties*, and Table 4-18, *Plot Object Properties*, incorrectly list the data type for `contour.interval`, which is Real.
- Table 3-3, *Plot Properties*, and Table 4-18, *Plot Object Properties*, do not list `<linePoint>` as a valid value for 3D plots. `<linePoint>` is a valid plot style constant for both 2D and 3D plots.
- In Table 4-3, *Numeric Matrix Object Properties*, `labels.column(`*n*`)` should be `label.column(`*n*`)`.
- For the most current information about HiQ built-in functions, refer to the online help, which you can access in HiQ by selecting **HiQ Help Topics** from the **Help** menu.

321967B-01 Jan99