NATIONAL INSTRUMENTS™
The Software is the Instrument®

Release Notes

LabVIEW® for UNIX

Version 4.1

Welcome to LabVIEW for UNIX

These release notes introduce you to the contents of LabVIEW for UNIX, describe the system requirements for the LabVIEW software, and contain installation instructions and updated documentation information.

Contents

How to Proceed

If you are upgrading from a previous version of LabVIEW, carefully read the *LabVIEW Upgrade Notes* that are included with your upgrade package. You should read the upgrade notes before continuing with this installation because there are several things you should consider before converting your VIs to this version of LabVIEW.

HiQ®, NI-DAQ® and LabVIEW® are trademarks of National Instruments Corporation.
Product and company names listed are trademarks or trade names of their respective companies.

320541E-01

© 1997 National Instruments Corporation. All rights reserved.

May 1997

Scan the *Required System Configuration* section and then follow the instructions in the *Installing LabVIEW* section of these release notes. If you use GPIB products, read the *GPIB and VXI Installation Notes* section. In addition, you should read the *Manual Clarifications and Additions* section before using LabVIEW 4.1.

Required System Configuration

LabVIEW for Sun and HP-UX ships on CD only. LabVIEW for Concurrent PowerMAX ships on 4 mm DAT tape. They are not available on floppy disks. LabVIEW requires an X Window System server, such as OpenWindows 3, HP-VUE, or X11R6. LabVIEW does not require a specific graphical user interface (GUI) such as Motif or OpenLook, because LabVIEW uses `Xlib` to create its own GUI.

LabVIEW for Sun runs on SPARCstations under SunOS 4.1.3 or later, and Solaris 2.3 or later. LabVIEW for HP-UX runs on Hewlett-Packard Model 9000 Series 700 computers under HP-UX 9.0.3 or later. LabVIEW for Concurrent PowerMAX runs on PowerMAX version 3.1 or later.

The workstation should have 32 MB of RAM, with 32 MB or more of swap space storage. LabVIEW can run on less RAM, but performance will suffer. In addition, the workstation must have a minimum of 65 MB of disk storage space if you want to install the entire LabVIEW package. If you want to save space, install only the VIs you plan to use.

LabVIEW uses a directory for storing temporary files. Some of the temporary files are large, so we recommend that you have several megabytes of disk space available for this temporary directory. The default for the temporary directory is `/tmp`. You can change the temporary directory by selecting **Edit»Preferences...**.

If LabVIEW aborts unexpectedly, it might leave files behind in the temporary directory. Remove old files occasionally to avoid using up your disk space.

(Sun) You can use a TMPFS file system for this directory to improve performance. For Solaris 1.*x*, refer to the Sun *System and Network Administration* manual, part number 800-3805-10, for more information about the TMPFS file system. Solaris 2 uses TMPFS by default.

Operating System Patches on the Sun

If you plan to run LabVIEW under SunOS 4.1.3, you need to obtain the latest revision of the following patch from Sun.

100458-xx: Setitimer sometimes fails to deliver SIGALRM

If you plan to run LabVIEW under Solaris 2.3, you should obtain the latest versions of the following patches from Sun.

101318-*xx*: jumbo patch for kernel
101347-*xx*: fixes for ttcompat
101409-*xx*: jumbo linker patch
101489-*xx*: libthread jumbo patch

There should not be any other patches necessary to run LabVIEW under Solaris 2.4 or later.

Installing LabVIEW

If you are upgrading from an earlier version of LabVIEW, carefully read the *LabVIEW Upgrade Notes* that are included with your upgrade package. You should read the upgrade notes before continuing with this installation.

Installation Procedure

Go through the following steps to install LabVIEW.

Solaris 1

1. To enable superuser privileges, type `su root`. Enter the root password.
2. If the directory `/cdrom` does not exist, type the following UNIX command:

   ```
   mkdir /cdrom
   ```

3. Mount the LabVIEW CD-ROM. Then, type the following command.

   ```
   mount -rt hsfs /dev/sr0 /cdrom
   ```

4. To change to the installation directory, type:

   ```
   cd /cdrom/solaris1.
   ```

5. To run the installation script, type `./INSTALL`. Follow the instructions on the screen.

Solaris 2

1. To enable superuser privileges, type `su root`. Enter the root password.
2. Insert the LabVIEW CD-ROM. Under Solaris 2.*x*, the CD-ROM automatically mounts as soon as the CD is inserted into the drive. If this feature is disabled on your workstation, you have to mount the CD by typing the following command.

   ```
   mount -o ro -F hsfs /dev/dsk/c0t6d0s2 /cdrom
   ```

3. If your CD was automatically mounted, type the following command.

   ```
   pkgadd -d /cdrom/cdrom0/solaris2
   ```

   Otherwise, if you used the mount command as in step 2, type the following command.

   ```
   pkgadd -d /cdrom/solaris2
   ```

   Follow the instructions on the screen.

HP-UX

1. To enable superuser privileges, type `su root`. Enter the root password.
2. Mount the LabVIEW CD ROM on the `/cdrom` directory with the "sam" system administration utility.
3. To change to the installation directory, type:

   ```
   cd /cdrom/hp-ux.
   ```

4. To run the installation script, type `./INSTALL`. Follow the instructions on the screen.

PowerMAX

1. Insert the 4 mm DAT tape into the tape drive.
2. To create the directory in which you will install LabVIEW, type the following command:

   ```
   mkdir /opt/lv41
   ```

3. To change to the new directory, type:

   ```
   cd /opt/lv41
   ```

4. Extract the files from the tape by typing the following command:

   ```
   tar xv
   ```

5. To run the installation script, type `./INSTALL`. Follow the instructions on the screen.

After you have completely installed LabVIEW, it is ready to run.

☞ **Note:** ***The function material from the*** *Code Interface Reference Manual* ***and the*** *VXI VI Reference Manual* ***are available online in Adobe Acrobat format on the LabVIEW CD. For more information on installing and viewing these online manuals, please refer to the*** `manuals\readme.txt` ***on the LabVIEW CD.***

☞ **Note:** ***If you are upgrading from a previous version of LabVIEW, you should read the*** *What Is New in LabVIEW* ***section of the*** *LabVIEW Upgrade Notes.* ***If you have one of the add-on packages, such as the Test Executive Toolkit or the Picture Control Toolkit, you may want to install those files at this time.***

Where to Go from Here

The following resources can help get you up to speed with LabVIEW 4.1 more quickly.

- If you are a new LabVIEW user, work through the *LabVIEW Tutorial*. This manual teaches basic LabVIEW concepts and tasks.
- The `examples` directory contains a VI called `readme.vi`. With this VI, you can look at the available examples. when you select a VI, you can see the documentation that was entered for that VI by choosing **Window»Show VI Info...**. to open a VI, choose **File»Open...**.

GPIB and VXI Installation Notes

The LabVIEW installer prompts you to choose the NI-488.2M drivers for the GPIB hardware you are using (SB-GPIB-TNT, GPIB-ENET, or GPIB-SCSI-A). The installer then installs that driver for you. The sole exception is the Solaris 1 NI-488.2M driver for the GPIB-SCSI-A, which you must install separately.

If you have a GPIB-SCSI-A, follow the installation instructions in the getting started manual that came with your original GPIB-SCSI-A hardware and software kit, including the *Getting Started with Your GPIB-SCSI-A and the NI-488.2M Software for the Sun SPARCstation* manual.

☞ **Note:** ***LabVIEW does not work with the GPIB-1014 series (VME) devices or the original GPIB-SCSI box. It does work with the newer GPIB-SCSI-A box.***

The VXI device drivers for Solaris 1.*x* and 2.*x* are included on floppy disks with the LabVIEW for Sun VXI Upgrade and with the VXI/VME-SB2020 kit. A VXI device driver must be installed on your system to perform VXIbus operations from LabVIEW. Install the appropriate device driver (Solaris 1.*x* or Solaris 2.*x*) before beginning development. To install the VXI device driver, refer to the *Getting Started with Your VXI-SB2020 and the NI-VXI Software for SunOS* manual.

☞ **Note:** ***National Instruments updates drivers for GPIB and VXI occasionally. If you add new GPIB or VXI hardware for use with LabVIEW, the included drivers may supersede those sent with LabVIEW. Compare the version numbers and use the driver with the higher number.***

Common LabVIEW Launch Errors

The following table lists common errors that occur when you launch LabVIEW for UNIX. See the *Required System Configuration* section of these release notes for more information on solving these and other installation problems.

Error Message/Description	Probable Cause/Solution
"Xlib: connection to :0.0 refused by server" OR "client is not authorized to connect to server" OR "internal error during connection authorization check"	Probable Cause—Trying to run LabVIEW as a user who does not have permission to open a window on the display server. Typically seen after running the `su` command to temporarily become a different user, such as root (superuser). Solution—Exit the `su` command and launch LabVIEW as the login user.
"Executable version doesn't match resource file"	Probable Cause—Version of LabVIEW executable does not match version of `labview.rsc`. Solution—Check to see whether the `appResFilePath` parameter in the configuration file correctly sets the path to the `labview.rsc` file.

Setting Up LabVIEW Windows

Setting Up LabVIEW with HP VUE

If you use the HP VUE window system, you can change environment settings so the HP VUE window manager (`Vuewm`) interacts better with LabVIEW. By default, `Vuewm` does not honor the window position requests of an application. This means that LabVIEW windows, such as the panel, diagram, help, and file dialog windows, do not appear in consistent locations on your screen. To change the `Vuewm` behavior, use the `xrdb` command to set two `Vuewm` settings:

```
Vuewm.clientAutoPlace: False
Vuewm.positionIsFrame: False
```

You can also manually edit your `$HOME/.vue/sessions/home/vue.resources` and `$HOME/.vue/sessions/current/vue.resources` files to add these two entries.

Setting Up LabVIEW with Motif

If you use the Motif Window Manager (`mwm`), you can change environment settings so that `mwm` interacts better with LabVIEW. By default, `mwm` does not honor the window position requests of an application. This means that LabVIEW windows, such as the panel, diagram, help, and file dialog windows, do not appear in consistent locations on your screen. To change the behavior of `mwm`, use the `xrdb` command to set two `mwm` settings:

```
mwm.clientAutoPlace: False
mwm.positionIsFrame: False
```

You can also manually edit your `$HOME/.Xdefaults` file to add these two entries.

Notice to Sun SPARCstation 5 Owners

A bug exists in some early revisions of the SPARCstation 5. This bug can cause LabVIEW and other programs to hang the system when executing certain floating point operations. When this condition occurs, it is necessary to physically reset the computer to recover. The problem exists in the firmware of the computer, and can occur when running SunOS 4.1.3_U1, SunOS 4.1.4, and Solaris 2.*x*.

☞ **Note:** ***This bug has been reported only on early revisions of the 70 MHz and 85 MHz SPARCstation 5.***

To find out if your SPARCstation 5 is afflicted, perform the following steps. (Following these steps temporarily interrupts the operation of your computer, so be sure to warn anyone who may be using the computer remotely.)

1. From your SPARCstation 5 console, hold down the `<Stop/L1>` key (located near the upper left corner of your keyboard) and press the `<A>` key to break into the PROM monitor.
2. You should see one of two prompts. You will see either

   ```
   Type b (boot), c (continue), or n (new
   command mode)>
   ```

or

```
Type 'go' to resume ok
```

In the first case, select "n" to go to new command mode, where you will see an "ok" prompt.

If you already have an "ok" prompt, skip to step 3.

3. At the "ok" prompt, type

```
module-info
```

You should then see something like this:

```
CPU FMI,MB86904 Rev. 2.5 : 70.0 MHz
SBus (Divide By 3)       : 23.3 MHz
```

4. Type "go" to exit the monitor and resume operation of your system.

If your CPU Revision number (2.5 in this example) is earlier than 3.2, *and* your CPU clock speed (70.0 MHz in this example) is less than 110 MHz, then your computer has this problem. Contact Sun and ask to have your CPU firmware upgraded to swift_pg 3.2 or later. (Swift is Sun's code-name for the SPARCstation 5 firmware.) The Sun Bug ID number for this problem is 1151654.

If you have a SPARCstation 5 with this bug, National Instruments highly recommends upgrading your firmware.

☞ **Note:** ***This problem can affect other programs besides LabVIEW. Notably, the GNU C compiler can also produce code that hangs your system in versions prior to 2.6.0.***

Notice to Concurrent PowerMAX Users

Because there is no online documentation available for LabVIEW for PowerMAX at this time, a printed version of the *LabVIEW Function and VI Reference Manual* is included in your kit instead of being available primarily online. You should disregard links to any online documentation or references to online documentation.

Because Code Interface Nodes (CINs) work essentially the same for Concurrent PowerMAX as for Sun and HP-UX, you can use the Sun and HP-UX instructions in the *LabVIEW Code Interface Reference Manual* for creating your CINs. See the *Special Instructions for CIN Programming with Concurrent PowerMAX* section of these release notes for information on the minor exceptions to the Sun and HP-UX instructions.

Attempting to use 7-bit data with the LabVIEW Serial VIs may crash your machine because the operating system only supports 8-bit characters.

Although the LabVIEW Serial VIs allow you to select the hardware flow control option, serial line hardware flow control is not supported by the operating system and therefore will not work.

Special Instructions for CIN Programming with Concurrent PowerMAX

Supported Languages

External code is compiled as a shared library. Only the PowerMAX compiler is supported.

Compiling CIN Source Code

You can follow the instructions for compiling source code for Solaris and HP-UX in the *LabVIEW Code Interface Reference Manual*.

Debugging External Code

It is not possible at this time to use the PowerMAX debugger to debug CINs. You can use standard C `printf` calls, or the `DbgPrintf` function mentioned in the *Debugging External Code* section of Chapter 1, *CIN Overview*, in the *LabVIEW Code Interface Reference Manual*.

External Subroutines

The techniques for creating and calling shared external subroutines described in Chapter 4, *External Subroutines*, of the *LabVIEW Code Interface Reference Manual*, are the same for PowerMAX as for other LabVIEW UNIX platforms.

Manual Clarifications and Additions

This section contains information that was not included in the LabVIEW documentation and corrections to the LabVIEW manuals. For information on new features to this particular version of LabVIEW, please read the *LabVIEW Upgrade Notes*.

General Interface Changes

The following changes were made to the LabVIEW interface, but were not explained in the printed or online documentation.

Print Margins

You now can set margins on printouts, both globally for LabVIEW and for specific VIs. Margins can be set in inches or millimeters. To set margins for all LabVIEW printouts, use the Printing page of the Preference dialog box. To set margins for a single VI, use the Execution page of the VI Setup dialog box, which can be accessed from the connector pane of the front panel. Settings for a single VI will override the global setting. Margins can be specified arbitrarily small, but will be limited by the device settings on actual printouts.

Printing VIs

The **Functions»Advanced»Printing** palette contains VIs that you can use to print VIs programmatically. This palette consists of the following VIs:

- `Print Panel.vi`—Use this VI to print a VI front panel in the same format as if you selected **File»Print Window** or you enabled programmatic printing. You can specify whether you want the entire front panel or only the visible part of the front panel.
- `Print Documentation.vi`—Use this to print the components of a VI as though you selected **File»Print Documentation**. You can specify whether the VI should display the Print Documentation dialog box so that you can select the format for the printout. If you choose not to display the dialog box, the printout will use the same settings as the last VI printed or the default settings if you have not printed any VIs.
- `Get Panel Image.vi`—Use this VI to programmatically retrieve a front panel image in a format that you could pass to one of the VIs in the Picture Control Toolkit or write to disk. You can specify whether you want the entire front panel or only the visible

part of the front panel. You can also specify the color depth for the data. The VI returns arrays of data and color table information in the rectangle describing the image.

Support for Template VIs and Controls

You can save commonly used VIs and controls as templates. To create a template VI, save a VI with a `.vit` extension (or `.ctt` extension for typedefs). When you open a template VI or control, the new file you create is named automatically using your template name and a number corresponding to the number of times it has been opened. When you finish editing the VI and try to save it, LabVIEW prompts you to enter a new name for the file.

If you want to modify a template, you should open it, make your changes, and then save over the `.vit` (or `.ctt`) file that you originally created.

Better Support for Toolkits

Files that are installed in `vi.lib\addons` will automatically show up at the top level of the **Control** and **Functions** palettes. This feature can be used by new toolkits to make them more accessible after installation. If you already have toolkits that installed files elsewhere, you can move them to the `addons` directory for easier access. If you want to add your own VIs to the palettes, we recommend placing them in `user.lib` or adding them to a custom palette set.

Debugging and Modal VIs

When LabVIEW stops a VI from executing (typically, when the user clicks on the pause button or because the program encounters a breakpoint) all VIs whose window style is “Dialog” temporarily change their window style to be non-modal. When all of the VIs have finished executing, LabVIEW returns the window style to modal.

VI Control Changes

The Call Instrument VI, which is located in **Advanced»VI Control**, has had several enhancements to allow it to perform more error checking and to execute calls more quickly.

- The flattened data types component of the **requested outputs** input was previously ignored. The Call Instrument VI now checks the requested types against the types of the subVI that you are calling and returns an error if they are incompatible.

- In previous versions of LabVIEW, if you did not specify any specific outputs for the Call Instrument VI, all outputs were returned. This feature did not allow for any type checking and often introduced runtime errors if you tried to unflatten the wrong data. Secondly, returning all of the outputs of a VI consumes considerable system time and memory, especially with VIs that contain large numbers of indicators. The Call Instrument VI defaults to returning only the outputs that you request. If you do not request any outputs, none are returned. If you want LabVIEW to return all VI outputs, wire a `True` Boolean to the **Return All Outputs** input of the VI.
- If you do not specify a path, the Call Instrument VI does not attempt to load and release the VI, which significantly increases execution speed. In this case, you should preload the VI yourself using the Preload Instrument VI, located in **Advanced**»**VI Control**, before calling the Call Instrument VI.
- You can use the new Get Panel Size VI to find out the size and location of a VI's front panel. The VI must be in memory, but its front panel does not have to be open. One way you might use this is in combination with the Resize Panel VI to position a front panel before calling the Open Panel VI to display it.
- Both the Get Panel Size and Resize Panel VIs have a Boolean input, **panel bounds**, that lets you specify whether you want the bounds to reflect the size of the front panel or the window. The size of the front panel only includes the visible part of the panel and does not include the window's title bar, the scrollbars, the toolbar, or the menu bar.
- The Open Panel VI has a new input that lets you specify whether you want it to reflect VI Setup settings (for instance, modality, hidden components, and so on). This input defaults to `True`. You might set it to `False` if you are opening a VI that you do not plan to immediately run, so that you can edit the VI or look at its block diagram.

Update VXI*plug&play* Drivers Menu Option

VXI*plug&play* installs files in a special `vxipnp` directory on your system. This directory is created if you install the NI-VISA driver. If LabVIEW detects this directory, it adds the **Update VXI*plug&play* Drivers** menu option to the **File** menu, which makes it easy to import these drivers into LabVIEW's **Functions** palette.

When you select **File**»**Update VXI*plug&play* Drivers**, LabVIEW scans the `vxipnp` directory to find new instrument drivers. LabVIEW looks for libraries and CVI Function Panels and displays lists of the new files that were found, asking which function panels you want to convert and which libraries you want to copy to the `instr.lib` directory. For every function panel you select, LabVIEW converts the function panel as if you had selected **File**»**Convert FP File**.

Updated CVI Panel Converter for Easier Handling of Arrays and Strings

The documentation for the CVI Function Panel converter indicates that the main reason you might have problems in converting a file is if the functions return arrays or strings. In these cases, LabVIEW needs to preallocate a buffer large enough to hold the data. With previous versions of LabVIEW, you had to modify a constant on the diagram if the panel had the potential for returning a larger buffer. With LabVIEW 4.1, the buffer size can be specified as an input to the subVI instead of a constant. Consequently, there are very few cases in which you would need to modify the diagrams or panels of a converted function panel.

Alternate Palette Menu Views

The *LabVIEW User Manual* and *LabVIEW Tutorial* describe the default **Controls** and **Functions** palettes. In addition to these palettes, there are two alternate, customized views that tailor the setup for users of LabVIEW for test and measurement and basic use. These views are called the T & M and Basic views. If you are interested in customizing LabVIEW to one of these views, see Chapter 8, *Customizing Your LabVIEW Environment*, in the *LabVIEW User Manual* for more information.

These alternate views contain the same functions as the default view, but the functions are rearranged so that the options that you use more frequently are more accessible. These views also contain some customized controls, which were created using the Control Editor, that are set up to minimize the amount of configuration you have to do yourself for many applications. You can switch between views quickly by choosing **Edit**»**Select Palette Set**.

T & M View

The Test and Measurement (T & M) view emphasizes functions used in test and measurement. The VISA functions appear at the top level of the **Functions** palette, with the GPIB and Serial functions listed below that palette. The **Controls** palette contains a submenu of controls that are frequently used in creating instrument drivers, which also allows you to create instrument drivers more consistently.

Basic View

The new basic control and functions palette view provides a simplified set of palettes, including only the most commonly used functions. This palette can be useful for those who are learning LabVIEW, because it emphasizes only the functions you might need for beginner applications. To switch to the Basic view, select **Edit»Select Palette Set»Basic**.

LabVIEW Preferences

The **Miscellaneous** page in **Edit**»**Preferences...** has an option for creating constants using a pop-up menu that lets you specify whether the constant name should be visible.

There are preferences that are not a part of the **Preferences...** dialog box. If you want to enable one of these preferences, you can edit your LabVIEW Preferences file to add the option.

- By default, LabVIEW looks for the `menus` directory inside of your library directory (the directory that also contains `vi.lib`, `instr.lib`, and `user.lib`). The `menus` directory contains folders of files describing the **Control** and **Function** palette views that are available to you. If you are running off of a network, you may want to define individual `menus` directories for each user. You can add a line to your preference file that sets the `menusDir` preference to an alternative path, one that would be unique for each user's preference file.

Adding VIs to the Project and Help Menus

You can now add VIs to the **Project** and **Help** menus by placing them inside of the project or help directories in the LabVIEW directory. One way you could use this would be to provide quick access to VIs that act as tools in your system. National Instruments uses this feature to make the Tech Support VIs accessible from the **Help** menu.

Any VI placed at the top level of the project or help directory will be directly appended to the corresponding menu. If you create a subdirectory, a submenu will be appended. If you place a library (LLB) inside of one of these directories, only VIs that are marked as top level will be appended to the menu bar.

Applibs and Toolkit Users

National Instruments plans to upgrade all of our toolkits for LabVIEW 4.1. In most cases, any existing toolkits will work without problems with LabVIEW 4.1. Version 4.1 is compatible with all toolkits designed for Version 4.0 with the following exception.

LabVIEW Application Builder Libraries

If you have the LabVIEW Application Builder Libraries and want to use LabVIEW 4.1, you must upgrade your Application Builder Libraries as well. The upgrade is free to existing users. Contact your local National Instruments sales office for availability.

Adding Options to the Controls and Functions Menus

Many toolkits such as the Picture Control Toolkit, the SPC toolkit, and the PID toolkit add new VIs and controls to the **Control** and **Function** palettes. In previous versions, toolkits installed VIs in `vi.lib`. National Instruments discourages placing VIs in `vi.lib` anymore, because you can overwrite your files when you install new versions of VIs as you upgrade your software. Instead, toolkit VIs should be placed in `vi.lib\addons`, as described earlier. VIs of your own should be placed in `user.lib` or `instr.lib`, where they will automatically appear in the user libraries or instrument drivers submenus of the **Functions** menu. Or, you can use the **Edit**»**Edit Control and Function Palettes** menu option to add them anywhere to the menus.

For more information on customizing the **Control** and **Function** palettes, refer to Chapter 8, *Customizing Your LabVIEW Environment*, of the *LabVIEW User Manual*.

LabVIEW User Manual Corrections

In Chapter 11, *Boolean Controls and Indicators*, in the *LabVIEW User Manual*, the manual states:

> You can also remove the Boolean text from both states by selecting **Hide Boolean Text** from the pop-up menu.

This sentence should read as the following:

> You can also remove the Boolean text from both states by toggling the **Boolean Text** option on the **Show** submenu.

In Chapter 11, *Boolean Controls and Indicators*, in the *LabVIEW User Manual*, the manual states:

> If you want to change the font of either the name label or the Boolean text without changing both, select what you want to change with the Labeling tool, and then use the **Text** menu options to make the changes you want.

This sentence should read as the following:

> If you want to change the font of either the name label or the Boolean text without changing both, select what you want to change with the Labeling tool, and then use the **Font** ring options to make the changes you want.

The *Documenting VIs with the Get Info... Option* section of Chapter 7, *Printing and Documentation*, in the *LabVIEW User Manual* should be replaced with the following section.

Documenting VIs with the Show VI Info… Option

Selecting **Windows»Show VI Info...** displays the information dialog box for the current VI. You can use the information dialog box to perform the following functions.

- Enter a description of the VI. The description window has a scrollbar so you can edit or view lengthy descriptions.
- Lock or unlock the VI. You can execute but not edit a locked VI.
- See the current revision number.
- View the path of the VI.
- See how much memory the VI uses. The **Memory Usage** portion of the information box displays the disk and system memory used by the VI. (This figure applies only to the amount of memory the

VI is using and does not reflect the memory used by any of its subVIs.)

The memory usage is divided into space required for the front panel and the block diagram, VI code, and data space. The memory usage can vary widely, especially as you edit and execute the VI. The block diagram usually requires the most memory. When you are not editing the diagram, save the VI and close the block diagram to free space for more VIs. Saving and closing subVI panels also frees memory.

LabVIEW Analysis VI Reference Manual Additions

The following information supplements the *LabVIEW Analysis VI Reference Manual*, the *LabVIEW Function and VI Reference Manual*, and the *Online Help*.

Complex QR Factorization

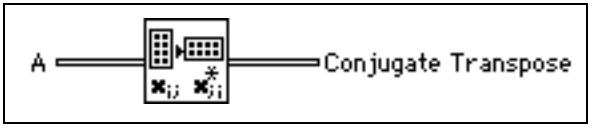
The Complex QR Factorization VI, located in **Analysis**»**Linear Algebra**»**Complex Linear Algebra**»**Advanced Complex Linear Algebra**, only supports the Householder algorithm.

General LS Linear Fit

The General LS Linear Fit VI, which is found in **Analysis**»**Curve Fitting**, has an additional input, called **covariance selector**. This **covariance selector** input bypasses the covariance computation.

Complex Conjugate Transpose

The Complex Conjugate Transpose VI, located in **Analysis**»**Linear Algebra**, takes the complex conjugate transpose of the input matrix A, forming the output complex matrix Conjugate Transpose.



A refers to the input matrix to the complex conjugate transpose operation

Conjugate Transpose is the complex conjugate transpose of the input matrix **A**. The Conjugate Transpose C of a complex matrix **A** is defined as:

$$C = A^H \Rightarrow c_{ij} = a^*_{ji}$$

LabVIEW Code Interface Reference Manual Additions

The example code in the *Computing the Cross Product of Two Two-Dimensional Arrays* section (found in the *Examples with Variably-Sized Data* section of Chapter 2, *CIN Parameter Passing*) produces the wrong result if the array B is not square. The line at the bottom of this code that reads:

```
*resultElmtp += aElmtp[i*k + l] + bElmtp[l*k + j];
```

should read:

```
*resultElmtp += aElmtp[i*k + l] + bElmtp[l*cols + j];
```

LabVIEW Instrument I/O VI Reference Manual Additions

The following information supplements the *LabVIEW Instrument I/O VI Reference Manual* and the *Online Help*.

VISA Attribute Descriptions

The following attributes are available with LabVIEW 4.1, but were not included in the *LabVIEW Instrument I/O VI Reference Manual*. Each attribute description includes the range, default value, and access privileges. *Local* applies to the current session only. *Global* refers to all sessions to the same VISA resource.

Interface Number of Parent

Specifies the board number of the parent device.

Range: 0 to FFFFh
Default: 0
Access Privilege: Read Only Global

Number of Bytes at Serial Port

Specifies the number of bytes currently available at the serial port used by this session.

Range: 0 to FFFFFFFFh
Default: N/A
Access Privilege: Read Only Global

Serial Baud Rate

U32

Specifies the baud rate of the given communications port.

Range: System Dependent (any 32-bit value, usually from 300 to 38400)
Default: 9600
Access Privilege: Read/Write Global

Serial Data Bits

U16

Specifies the number of data bits contained in each frame.

Range: 5 to 8
Default: 8
Access Privilege: Read/Write Global

Serial End Mode for Reads

U16

Specifies the method used to terminate read operations.

Range: VI_ASRL_END_NONE(0), VI_ASRL_END_LAST_BIT(1), VI_ASRL_END_TERMCHAR(2)
Default: 2
Access Privilege: Read/Write Local

Serial End Mode for Writes

Specifies the method used to terminate write operations.

Range:	VI_ASRL_END_NONE(0), VI_ASRL_END_LAST_BIT(1), VI_ASRL_END_BREAK(3)
Default:	0
Access Privilege:	Read/Write Local

Serial Flow Control

Specifies the flow control method used for both transmitting and receiving data.

Range:	VI_ASRL_FLOW_NONE(0), VI_ASRL_FLOW_XON_XOFF(1), VI_ASRL_FLOW_RTS_CTS(2)
Default:	0
Access Privilege:	Read/Write Global

Serial Parity

Specifies the parity used with every frame that is transmitted or received.

Range:	VI_ASRL_PAR_NONE(0), VI_ASRL_PAR_ODD(1), VI_ASRL_PAR_EVEN(2), VI_ASRL_PAR_MARK(3), VI_ASRL_PAR_SPACE(4)
Default:	0
Access Privilege:	Read/Write Global

VISA Classes

The following table shows which VISA classes are supported by each VISA operation

☞ **Note:** ***The VISA Read and VISA Write operations are synchronous on the Sun.***

Operations	VISA Classes				
	Instr	**GPIB Instr**	**VXI/ GPIB-VXI MBD Instr**	**VXI/ GPIB-VXI RBD Instr**	**ASRL Instr**
VISA Assert Trigger	√	√	√	√	
VISA Clear	√	√	√		
VISA Disable Event	√	√	√	√	√
VISA Discard Events	√	√	√	√	√
VISA Enable Event	√	√	√	√	√
VISA In 8/16/32	√		√	√	
VISA Lock	√	√	√	√	√
VISA Map Address	√		√	√	
VISA Mem Allocate	√		√	√	
VISA Mem Free	√		√	√	
VISA Move In 8/16/32	√		√	√	
VISA Move Out 8/16/32	√		√	√	
VISA Out 8/16/32	√		√	√	
VISA Peek 8/16/32	√		√	√	
VISA Poke 8/16/32	√		√	√	
VISA Read	√	√	√		√
VISA Read STB	√	√	√		

Operations	VISA Classes				
	Instr	GPIB Instr	VXI/ GPIB-VXI MBD Instr	VXI/ GPIB-VXI RBD Instr	ASRL Instr
VISA Wait On Event	√	√	√	√	√
VISA Write	√	√	√		√
VISA Unmap Address	√		√	√	
VISA Unlock	√	√	√	√	√

The following table shows which VISA classes are supported by each VISA attribute.

Attributes	VISA Classes				
	Instr	GPIB Instr	VXI/ GPIB-VXI MBD Instr	VXI/ GPIB-VXI RBD Instr	ASRL Instr
Bytes at Serial Port	√				√
Commander Logical Address	√		√	√	
Fast Data Channel Number	√		√		
Fast Data Channel Mode	√		√		
Fast Data Channel Pairs	√		√		
Fast Data Channel Signals	√		√		
GPIB Primary Address	√	√	√	√	
GPIB Secondary Address	√	√	√	√	
Immediate Servant	√		√	√	
Increment Destination Count	√		√	√	
Increment Source Count	√		√	√	

Attributes	VISA Classes				
	Instr	GPIB Instr	VXI/ GPIB-VXI MBD Instr	VXI/ GPIB-VXI RBD Instr	ASRL Instr
Interface Number	√	√	√	√	√
Interface Number of Parent	√		√	√	
Interface Type	√	√	√	√	√
IO Protocol	√	√	√		√
Mainframe Logical Address	√		√	√	
Manufacturer Identification	√		√	√	
Maximum Queue Length	√	√	√	√	√
Model Code	√		√	√	
Resource Lock State	√	√	√	√	√
Resource Manufacturer Name	√	√	√	√	√
Resource Manufacturer ID	√	√	√	√	√
Resource Name	√	√	√	√	√
Send End Enable	√	√	√		√
Serial Baud Rate	√				√
Serial Data Bits	√				√
Serial End Mode for Reads	√				√
Serial End Mode for Writes	√				√
Serial Flow Control	√				√
Serial Parity	√				√
Serial Stop Bits	√				√

Attributes	VISA Classes				
	Instr	GPIB Instr	VXI/ GPIB-VXI MBD Instr	VXI/ GPIB-VXI RBD Instr	ASRL Instr
Slot	√		√	√	
Suppress End Enable	√	√	√		√
Termination Character	√	√	√		√
Termination Char Enable	√	√	√		√
Timeout Value	√	√	√	√	√
Trigger Identifier	√	√	√	√	
User Data	√	√	√	√	√
Version of Implementation	√	√	√	√	√
Version of Specification	√	√	√	√	√
VXI Logical Address	√		√	√	
VXI Memory Address Space	√		√	√	
VXI Memory Base Address	√		√	√	
VXI Memory Size	√		√	√	
Window Access	√		√	√	
Window Base Address	√		√	√	
Window Size	√		√	√	

320541E-01
May 1997