

LabVIEW® for Macintosh

Version 4.1

These upgrade notes describe the process of upgrading LabVIEW for Macintosh to version 4.1. Start by reading the *System Requirement Changes* section to determine if your system is capable of running LabVIEW 4.1.

For installation instructions, manual corrections and additions, and other important information, read the *LabVIEW Release Notes for Macintosh*.

Contents

| | |
|---|---|
| System Requirement Changes | 2 |
| How to Upgrade | 2 |
| Applibs and Toolkit Users | 3 |
| If You Are Upgrading from LabVIEW 4.0.x | 4 |
| If You Are Upgrading from LabVIEW 3.0.x | 4 |
| If You Are Upgrading from LabVIEW 2.2.1 | 5 |
| If You Are Upgrading from Versions before 2.x | 5 |
| What's New in LabVIEW 4.1 | 5 |
| New DAQ Examples | 5 |
| Basic Palette View | 6 |
| Print Margins | 6 |
| Printing VIs | 6 |
| Support for Template VIs and Controls | 7 |
| Better Support for Toolkit Files | 7 |
| Changes to MPW CIN Tools | 8 |
| Load Memory Usage Option (PowerMac Only) | 8 |

| | |
|---|----|
| Differences between LabVIEW 4.0.1 and LabVIEW 4.1 | 8 |
| Problems Fixed in LabVIEW 4.1 | 9 |
| What's New in LabVIEW 4.0.1 | 14 |
| LabVIEW Application Builder Libraries | 19 |
| Compatibility Issues between Versions 4.0 and 4.1 | 20 |
| Compatibility Issues between Versions 3.1 and 4.1 | 20 |
| Call Instrument VI Compatibility | 20 |
| Analysis VI Changes That Cause Incompatibilities | 20 |
| Example VI Changes That Cause Incompatibilities | 21 |
| Differences between LabVIEW 3.1 and LabVIEW 4.0 | 21 |
| Changes to LabVIEW Prior to Version 4.0 | 44 |
| Compatibility Issues between Versions 3.0.1 and 3.1 | 61 |
| Analysis VI Changes That Cause Incompatibilities | 61 |
| Attribute Changes That Cause Incompatibilities | 62 |
| File Utility VI Changes That Cause Incompatibilities | 63 |
| Incompatibilities Created by Using Counters to Generate Pulses with Am9513 and DAQ-STC Devices | 63 |
| Example VI Changes That Cause Incompatibilities | 64 |
| Changes Introduced between Versions 3.0 and 3.0.1 | 64 |

System Requirement Changes

The initial default memory allocation for LabVIEW is 7 MB; however, you might need more memory to convert your current VIs to version 4.1.

LabVIEW now requires System 7 as the operating system.

How to Upgrade

Upgrading LabVIEW is a fairly automatic process. When you open a VI that was created with an earlier version, LabVIEW automatically converts and compiles the VI.

Conversion is a memory-intensive operation. When LabVIEW loads a VI that was saved with an earlier version, it loads all components of the converted VI (front panel, block diagram, data) into memory, and then compiles the VI in memory. In addition, LabVIEW loads the components of all subVIs needing conversion into memory.

*Before converting VIs, you should increase the memory allocated to LabVIEW. You can increase the memory allocated to LabVIEW from the Finder by selecting the LabVIEW icon and then selecting **Windows»Show VI Info...** from the menu.*

You can estimate how much memory LabVIEW needs to convert VIs by adding up the amount of memory your VIs and all of their subVIs occupy on disk. If they are in VI libraries, you should add approximately 30 percent of the VI library size, because the VIs are compressed. The conversion process might require at least that much memory, plus an additional 2 MB for LabVIEW to use.

If you cannot allocate that much memory to LabVIEW, convert the VIs in stages, by components. Examine your hierarchy of VIs, and begin by loading and saving subVIs in the lower levels of the hierarchy. You can then gradually work up to the higher levels of the hierarchy. You can also select **File»Mass Compile** to convert a directory of VIs. Notice, however, that this option converts VIs in a directory or VI library in alphabetical order. If a high-level VI is encountered first, **Mass Compile** requires approximately the same amount of memory as if you opened the high-level VI first.

You can monitor your memory usage using the **Help»About LabVIEW...** option, which summarizes the amount of memory you have available.

Applibs and Toolkit Users

If you have an existing toolkit, it should work with LabVIEW without any problems. However, you need to move the VIs so that they show up in the menus. LabVIEW 4.1 is compatible with toolkits designed for 3.1 with the following exceptions.

LabVIEW Test Executive

The LabVIEW 3.1.1 Test Executive has been recompiled for use with LabVIEW 4.x. If you are an existing user of the LabVIEW 3.1.1 Test Executive and want to use LabVIEW 4.1, you must upgrade to the LabVIEW 4.0 Test Executive. This upgrade is free to existing users of the LabVIEW 3.1.1 Test Executive. Contact your local National Instruments sales office for availability.

LabVIEW Application Builder Libraries

If you are upgrading from 4.0 or 4.0.1, the updater will update your Application Builder Libraries. If you are upgrading from an earlier version, you will receive the standard installation, which might not include the Application Builder Libraries. The upgrade is free to existing users of the LabVIEW Application Builder Libraries. Contact your local, National Instruments sales office for availability.

Adding Options to the Controls and Functions Palettes

Many toolkits, such as the Picture Control Toolkit, the SPC Toolkit, and the PID Toolkit, add new VIs and controls to the **Controls** and **Functions** palettes. In previous versions of LabVIEW, the toolkits did this by installing VIs in `vi.lib`. National Instruments now discourages placing VIs in `vi.lib` because your files might be overwritten accidentally when new versions of these VIs are installed as you upgrade LabVIEW. Instead, National Instruments recommends that you place these VIs in `vi.lib\addons`, `user.lib`, or `instr.lib`, so that they automatically appear at the top level of the **Control** or **Functions** palette. Or, you can use **Edit»Edit Control and Function Palettes** to add them to the palette of your choice.

If you have a previous version of the Picture Control Toolkit, the SPC Toolkit, or the PID Toolkit, you should move the associated libraries to `user.lib`. For more information on customizing the **Controls** and **Functions** palettes, see Chapter 8, *Customizing Your LabVIEW Environment*, in the *LabVIEW User Manual*.

If You Are Upgrading from LabVIEW 4.0.x

LabVIEW 4.1 contains a number of new features and fixes a number of bugs. There are only a few compatibility issues between LabVIEW 4.0 and LabVIEW 4.1, so upgrading should not be very difficult. The improvements and fixes found in LabVIEW 4.1 are described in the *What's New in LabVIEW 4.1* section of these notes.

If You Are Upgrading from LabVIEW 3.0.x

LabVIEW 4.1 contains a number of new features and corrects a number of bugs. Since the release of version 3.0.x, there are only a few compatibility issues between LabVIEW 3.0.x and LabVIEW 4.1, so upgrading should not be very difficult.

If You Are Upgrading from LabVIEW 2.2.1

LabVIEW 4.1 does not support VIs created prior to version 3.0. If you are upgrading VIs from an earlier version, you must first upgrade to version 3.x and then convert them to LabVIEW. You should order the *LabVIEW 3.x Conversion Package*, part number 776874C-21.

If You Are Upgrading from Versions before 2.x

If you are upgrading from LabVIEW 1.x, you must first upgrade your VIs to LabVIEW 2.0. From there, you must upgrade the VIs to LabVIEW 3.0, and then upgrade them to LabVIEW 4.0.

If you are upgrading from LabVIEW 1.x, you must upgrade to LabVIEW 2.2.1 before upgrading to LabVIEW 4.0. You also need the *LabVIEW 1.x Conversion Package*, part numbers 776617-01 and 850001-01. This package assists in the conversion process, and documents the differences between the versions.

What's New in LabVIEW 4.1

This section describes the features that were added between 4.0 and 4.1. To help you learn more about LabVIEW, Version 4.1 comes with extensive online documentation, which you can access by selecting the **Help»Online Reference...** option.

New DAQ Examples

This release includes a number of new data acquisition examples. These examples are broken into examples that illustrate techniques and more complete examples called solutions.

The technique examples are based on the DAQ examples that shipped with previous releases, but there are many new and improved examples for analog input, analog output, simultaneous analog input and output, and more.

- The analog input examples include digital triggering, analog hardware and software triggering, using DAQ Occurrences for asynchronous acquisition, and other techniques.
- The analog output examples include digital triggering, external clocking, generating data from file, and other techniques.

- The simultaneous analog input/output examples include buffered acquisition and generation examples for E-series boards (like the PCI-MIO-16E-1), as well as legacy boards (such as the AT-MIO-16 or NB-MIO-16).

The solution examples are more complete examples that show how you can integrate data acquisition into your application. They include examples for the following applications:

- Bench top instruments such as oscilloscopes and waveform generators
- Control applications with PID loops and alarms, data logging applications such as logging to spreadsheet files and high speed logging
- Applications for transducers such as thermocouples and strain gauges

Basic Palette 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.

Print Margins

Now, you can set margins on printouts, both globally to LabVIEW, and for specific VIs. Margins can be set in either inches or millimeters. To set margins globally to LabVIEW, use the Printing page of the Preference dialog box. To set margins for a single VI, use the Execution page of VI Setup dialog box, accessed from the connector pane of the front panel. Settings for a single VI 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 you can use to print VIs programmatically. This palette consists of the following VIs.

Print Panel.vi

Use this VI to print a VI 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 panel or only the visible part of the 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 you can select the format for the printout. If you choose not to display the dialog box, the printout uses 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 retrieve a panel image programmatically in a format that you can pass to one of the VIs in Picture Control Toolkit or write to disk. You can specify whether you want the entire panel or only the visible panel and also the color depth for the data. The VI returns arrays of data and color table information and the rectangle describing the image.

Support for Template VIs and Controls

Now 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). You can also use the “Stationery Pad” checkbox in the VI “Get Info” dialog box in the Finder to change a VI to a template. When you close the VI, LabVIEW prompts you to save the file.

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

Better Support for Toolkit Files

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 older toolkits that installed files elsewhere, you can move them to the addons directory if you want to make them easier to access. If you have VIs of your own that you want

to add to the palettes, we recommend placing them in `user.lib` or adding them to a custom palette set.

Changes to MPW CIN Tools

The MPW script `cinmake (:cintools:MPW:cinmake)` has been modified to have its default behavior be to use the MrC compiler to compile the cin for the PowerPC platform. A new command line option ("`-C`") was put in to allow users to specify that the CIN should be compiled using the older C compiler for the 68K platform.

Load Memory Usage Option (PowerMac Only)

A new preference has been added that allows you to control whether LabVIEW should compact memory as it loads VIs. It can be set in the Performance and Disk page of the Preferences dialog box. By default, this preference is on, and the preference is presented as `Less memory fragmentation / Slower loading`.

The reason for this feature is the Modern Memory Manager, which, although faster for most operations, allows memory to become fragmented more easily.

Compacting memory while loading VIs reduces this memory fragmentation but slows down the loading of VIs. Notice that switching the preference to `Faster loading / Increased memory fragmentation` can cause your application to run out of memory sooner than it would otherwise, but can dramatically speed up loading for some applications, especially those with lots of re-entrant VIs.

Differences between LabVIEW 4.0.1 and LabVIEW 4.1

This section contains major changes since the previous version of LabVIEW and lists of some of the major problems that have been corrected in this version. Additional problems that could not easily be characterized or that were fairly cosmetic in nature are not listed.

Problems Fixed in LabVIEW 4.1

The following is a list of the bugs corrected in LabVIEW 4.1.

- LabVIEW sometimes crashed when converting VIs that used the LabVIEW 3 VISA transition library.
- **(HP-UX)** Several code generator errors existed that could cause LabVIEW to crash.
- **(Windows)** LabVIEW could crash when saving VIs that included enhanced metafiles.
- If a negative number was passed to the plot index or cursor index of a graph attribute node, LabVIEW sometimes crashed.
- Graphs with multiple plots that used fill styles sometimes caused LabVIEW to crash if the legend had more plots than the data.
- Indicators that adapt to their source input type (graphs, charts, and so on) might not have updated the types of their local variable references when they changed type. Running the VI sometimes caused LabVIEW to crash.
- If you selected part of a diagram that contained a local variable and chose the SubVI from Selection menu item, LabVIEW sometimes crashed.
- If you use the `DbgPrintf` function (for CINs) with a format string that contained `%g`, LabVIEW would report an error message and quit. Now, LabVIEW treats `%g` as if it were `%e`.
- LabVIEW no longer crashes when compiling large VIs that changed from a bad to good state.
- **(Windows)** If an unexpected error was reported by the file system when listing a directory, LabVIEW could display an error message and quit.
- Pasting diagram objects onto a new front panel could cause LabVIEW to crash.
- **(Windows NT)** LabVIEW might have hung if you tried to move a window while a VI was running.
- **(Windows NT)** LabVIEW could crash if you dragged an unsaved custom control out of the hierarchy window onto a panel.
- Writing data to a string indicator that used backslash mode sometimes caused LabVIEW to crash.
- Various edits to controls inside datalog refnums could draw incorrectly or cause LabVIEW to display an error message and quit.

- When scrolling a diagram while doing cluster ordering, LabVIEW could display an error message and quit.
- If you configured a Boolean typedef to blink, LabVIEW would display an error message and quit.
- Popping up on an Unbundle By Name function that had no names sometimes caused LabVIEW to crash.
- If you have an Intensity Chart with autoscaling in the X dimension, and the X dimension contained only one point, LabVIEW sometimes crashed.
- Format Into String could have caused LabVIEW to crash with %^g (engineering string) format.
- **(UNIX)** LabVIEW sometimes displayed an error message and refused to launch with some 24-bit X display servers.
- LabVIEW sometimes displayed an error message and quit when compiling a VI that passed an array to a Call Library function.
- Replacing controls with a color box could have caused LabVIEW to crash.
- **(Windows)** Windows bitmap printing now defaults to a higher resolution.
- If text was selected on the block diagram while printing, LabVIEW sometimes displayed an error message and quit.
- **(Windows)** If you typed a menu accelerator key (such as ctrl-H) while a menubar palette menu (such as “Unopened SubVIs”) was open, LabVIEW sometimes crashed.

Incorrect Behavior

The following lists the incorrect behavior that has been fixed in LabVIEW 4.1.

- **(Windows)** Standard PC printing now does a much better job adapting to the differences between screen and printer fonts.
- Font attributes were ignored when using PostScript printing.
- PostScript printing with decimal point set to comma did not work.
- Graphs or charts with only 2 data points could print incorrectly when using PostScript printing.
- We now print grayscale (instead of black & white) printouts on black & white printers.

- **(UNIX)** Scaling of printouts with standard printing did not work correctly, so scaling is now only allowed with PostScript printing.
- Transparent flat circles printed all black when using Postscript printing.
- The GPIB SendIFC function returned errors incorrectly.
- GPIB primary board addresses only could be set to 0, 1, 16 or 17.
- Traditional GPIB Read and 488.2 Receive, Find Listeners and Read Response Message could report out of memory if there was a problem communicating to the driver.
- **(Windows)** Power PC performance improvements (alignment).
- We no longer call `ibonl(bd,0)` when quitting LabVIEW. This improves cooperation with other applications that might be using GPIB.
- **(Windows)** Send break on serial port incorrectly returned errors.
- **(Windows)** VISA reads and writes are executed synchronously by default, to improve performance.
- VISA could report “Unable to queue asynchronous operation” error erroneously.
- Wait on occurrence could fail to test “ignore previous” correctly.
- **(Concurrent PowerMAX)** The Wait MS Multiple function waited longer than it should have.
- The first grid line on a graph or chart with arbitrary markers was not drawn.
- A chart or graph scale's “editable” attribute did not prevent you from editing the scale.
- Hidden controls would sometimes draw anyway.
- VIs that are loaded and unloaded dynamically no longer flash their windows when opening and closing.
- Data pasted into a diagram constant did not mark the VI so that it needed to be saved.
- The profiling window might have shown out of date information when VIs were edited.
- Dynamically loading VIs while profiling was paused caused problems.
- Spreadsheet String To Array might have appended an extra element to the array if the array had more than 2 dimensions.

- To Hexadecimal and To Octal functions now create correctly-sized strings for negative 16 bit and 18 bit integer values.
- **(Macintosh)** 68K - Size attribute on arrays did not update frame size correctly.
- The history buffer attribute on a chart could have insane type descriptor if the chart had a name.
- The increment value for controls with nonlinear units was handled incorrectly.
- The Knot unit used the British value. Now it uses the US value.
- **(Macintosh)** The Close Serial Port VI would not close both incoming and outgoing ports.
- **(Macintosh)** The Memory Monitor Example VI now displays free memory instead of used memory.
- The Power Spectrum VI did not work correctly for arrays with a size greater than 32768 that was not a power of 2.
- The Real FFT, Inverse Real FFT, and Power Spectrum did not work correctly for arrays of one element.
- The Spline Interpolation VI did not work correctly for certain decreasing X values.
- The Singular Value Decomposition VIs did not work correctly for some data sets.
- The Median Filter VI was slow for large ranks.
- The Convolution and CrossCorrelation VI now uses frequency-domain techniques to improve performance.
- The AC & DC Estimator now does better AC estimation for signals with low DC levels.
- **(UNIX)** Some valves in the automation symbol toolkit drew incorrectly on platforms that use X Windows.
- **(Macintosh)** Because of conflicts with various INITs, the file menu and other menus could draw their items with the wrong text.
- If you saved a VI that had breakpoints set on its diagram without that block diagram, the copy of the VI (without a diagram) would run slowly because LabVIEW continually tried to open the diagram.
- If you disconnect a control from a strict typedef, LabVIEW now keeps the attribute nodes.

- **(Windows)** In the Windows Explorer, if you dragged a VI with a long filename to the LabVIEW executable, LabVIEW opened the VI with the 8.3 filename instead of the long filename.
- LabVIEW did not allow you to resize structures that were several thousand pixels wide or tall. Now you can make them smaller.
- Programmatically closing a VI that was suspended could cause the caller VI to draw its execution palette incorrectly.
- If you used the keyboard to access a picture ring menu, the menu might have appeared in the wrong location.
- **(Macintosh)** LabVIEW refused to open a serial port if it was open already by another application.
- **(Windows and Macintosh)** When LabVIEW installed a serial number, it did not flush the data to disk immediately, so if LabVIEW subsequently crashed, the `labview.rsc` file might have been corrupted.
- Unbundle by Name on clusters would not always update the names correctly if the source names changed.
- Now, palette menus are hidden when a modal dialog box appears.
- **(Windows and UNIX)** Deny modes in the file I/O functions work better in UNIX.
- **(Windows)** The Volume Information function now supports UNC filenames.
- **(Macintosh)** TCP Get Raw Net Object did not work.
- **(Macintosh)** TCP read on a closed connection with a zero timeout did not report an error.
- Stacked charts with more plots to draw than charts to draw on would draw the last chart incorrectly.
- LabVIEW incorrectly allowed an array of a cluster of one number to be wired to a graph.
- Clear in the Edit menu did not work to delete objects.
- **(Windows)** In certain situations, Enums would sometimes behave as if they were signed numbers.
- **(Macintosh)** The characters in the strings of enums were mapped incorrectly from the Macintosh character set to ISO-8859-1 character set when VIs changed platforms.
- You could not copy search results text onto the clipboard.

- **(Windows)** Now, LabVIEW accounts for the Windows 95 task bar when resizing windows to make them tiled or full screen.
- A block diagram popup on a refnum could show the cluster tools menu incorrectly.
- Find on VIs with no block diagram could report out of memory errors incorrectly.
- The tabbing hilite box did not update correctly for single-line strings.
- A problem with using an infinite timeout on the UDP Read VI has been corrected.
- **(Windows)** References to DLLs can be relative paths now.

What's New in LabVIEW 4.0.1

This section describes the changes between 4.0 and 4.0.1. Many of the problems listed below can be easily worked around, but we recommend that you install the update so that you will not encounter any of these problems.

LabVIEW Version 4.0.1 Development System corrects the following problems:

- If you changed any graph pop-up options while a VI is executing, the VI incorrectly needed to be saved.
- If the Close File function received an error as an input, it would not close the file specified by the file refnum input.
- Sort 1D array would not work for an array of clusters if the cluster contained path or picture components.
- The Initialize Array and Resize Array functions would give a Memory Full Error when the dimension specified was a negative value.
- The Open File+.vi function passed a value to Open File to only open files as read only.
- If you highlighted any password displayed text and choose Find (ctrl-F), the Find dialog will show the password text.
- When you selected **Find»Locals** or **Find»Attribute Nodes** from a pop-up menu it would fail to locate any locals or attribute nodes for a strict type definition.
- Unit labels could not be edited while running a VI.

- Using the “traditional” GPIB functions, secondary addressing failed unless the bus address was also included.
- LabVIEW could have crashed if a path control was set to blink using its attribute node.
- The Refnum to Path function caused a crash if it was passed an array of file refnums.
- If you dragged an empty string constant along with a numeric constant together into an empty cluster constant, LabVIEW would crash.
- If you passed a string parameter larger than 32K in size to Format Into String function, LabVIEW would give a memory full error, stop the VI, or crash.
- If you made more than one call to the Call Chain function, LabVIEW could have crashed.
- Removing a cluster from an array constant caused LabVIEW to crash.
- If you tried to place a subVI in a cluster constant on the block diagram, LabVIEW would crash.
- The Write File+[I16] utility function did not have the pos mode input wired to the Write File function in it's block diagram. This caused the function to always overwrite data in a file.
- If you tried to use a type definition recursively, LabVIEW would crash.
- The Close Instrument function was modified to have the option to not abort the VI when called.
- A new VI Control function has been added called Abort Instrument.vi. Also, Close Panel.vi has a new option to allow you to close a panel without aborting the VI.
- Functions for converting numbers to complex numbers were mistakenly left out of the Functions palette. They have been added to the Numeric Conversions palette
- LabVIEW did not send Remote Enable (REN) when 488.2 functions were used. Some instruments required REN to be sent or they would not work properly.
- The 488.2 GPIB function Set Timeout did not correctly change the time out value for the GPIB board.

- If you performed a GPIB read with an EOS value and then another with a different EOS value, you would not be able to perform a read with the original EOS value again until you quit LabVIEW.
- Using the Create Constant pop-up option on inputs that were an enum type produced a constant that did not have the correct enum strings.
- Knobs set as strict type definitions would still allow their scales to be modified.
- **(Windows)** By default, when you copy a picture from LabVIEW under Windows 95 and Windows NT, LabVIEW exports a bitmap to the clipboard. Under LabVIEW 4.0.1, you can now choose to export an enhanced metafile instead if you prefer. An enhanced metafile is a picture format that records the original drawing instructions so that it can scale and print more precisely. If you want LabVIEW to export enhanced metafiles instead of bitmaps, add the following line to your LabVIEW.INI file in the [LabVIEW] section.

```
exportMetaFile = TRUE
```

LabVIEW exports bitmaps by default instead of metafiles for two reasons. First, under Windows 95 some programs import the metafile and lose track of the scale of the metafile (it becomes huge when it is imported). Second, under both Windows 95 and Windows NT, the bottom and right edges of the picture are lost in the resulting metafile.

- **(Windows)** The LabVIEW installation under Windows 3.1 would not launch correctly unless a TEMP directory was specified for the system.
- **(Windows)** The Seconds to Date/Time function returned the incorrect number of days in the year for leap years.
- **(Windows)** LabVIEW could not be launched under Windows 3.1 by double-clicking on a VI or VI library file in the File Manager.
- **(Windows)** The VI Setup option Scale to Fit would not always scale objects correctly.
- **(Windows)** The Editable attribute for scales on graphs or slide controls did not work correctly.
- **(Windows)** When doing serial communication over a long period of time (several hours or days), LabVIEW might either crash or halt.

- **(Windows)** Serial function Send Break returned a 16385 error even though it worked correctly.
- **(Windows)** When importing a bitmap image from CorelDraw, LabVIEW did not display it correctly.
- **(Windows)** Objects that were set to blink did not retain the blink attribute if the VI was minimized.
- **(Windows)** Some front panel objects were not set to their initialized value when converted from LabWindows CVI Function Panels using the Convert CVI FP File option.
- **(Windows)** Selecting the User Name option sometimes caused LabVIEW to crash if the user had not logged onto a network.
- **(Macintosh)** In both 4.0 and 4.0.1, LabVIEW for Power Macintosh does a significant amount of work to try to efficiently manage memory so that it doesn't become fragmented. When loading VIs, this extra effort can add significantly to load times in some cases. If you want to speed up load time, you can add the following configuration line to your LabVIEW Preferences file (which is a text file in the System folder).

```
macExtremeCompacting:FALSE
```

Notice that while this preference will improve load time, memory will not be used as efficiently. Also, these preferences can be used in both the development system and in applications built with the Application Builder Libraries (application preferences are stored in a file with the same name as the application plus the word Preferences).

- **(Macintosh)** On the Macintosh, we recommend that for any GPIB communication you use repeat addressing because some instruments will not work correctly without it. To ensure that readdressing is on, either check the Repeat Addressing checkbox in the NI-488 control panel or call GPIB Initialization in your VI.
- **(Macintosh)** The Seconds to Date/Time function returned the incorrect number of days in the year for leap years.
- **(Macintosh)** Trying to create a new file on the desktop with native file dialogs caused LabVIEW to crash.
- **(Macintosh)** Using Native File Dialogs in LabVIEW on a Macintosh IIcx caused the machine to crash. With Native File Dialogs turned off in LabVIEW, LabVIEW still uses native dialogs for printing, which also caused the machine to crash.

- **(Macintosh)** Selecting Unopened SubVIs from the Project menu when working with a large number of subVIs caused LabVIEW to crash.
- **(Macintosh)** The Project Stationery file Read Me-SC8 in cintools was modified to correct references to THINK C to say Symantec C and put in new references to v 8.0.5 project model.
- **(Macintosh)** The cintools file for MPW cinmake was modified to take the new parameters “-SC” to use the MPW SC (68K) compiler and “-MrC” to use the MPW MrC (PowerPC) compiler.
- **(Macintosh)** New stationery files for Metrowerks CodeWarrior 9 for the 68K and Power Macintosh have been added to the cintools folder.
- **(Macintosh)** A new project model that works with Symantec C/C++ v8.0 has been added to the cintools folder.
- **(Macintosh)** The file `:cintools:MPW:generic.make.mrc` is a new file that handles MPW MrC PowerMac specific compiler options.
- **(Macintosh)** The file `:cintools:MPW:generic.make.sc` is a new file that handles MPW SC 68K compiler specific compiler options.
- **(UNIX)** The accuracy of the timer functions, such as “Wait (ms)”, is platform-dependent. However, at the expense of CPU usage, LabVIEW can offer more accuracy. There is a new preference option which can appear in your “.labviewrc” file:

```
labview.accurateTimer: True
```

If this option is on, LabVIEW will try to use all the CPU time available when any VI is running, even if the VIs are just waiting for timer functions or for I/O to complete. When this option is off, LabVIEW relinquishes CPU time when waiting for I/O or timers and no other VIs are active. Note that if no VIs are running, LabVIEW will always relinquish CPU time to other processes.
- **(UNIX)** On HP-UX, using From Decimal, From Octal, or From Hexadecimal caused LabVIEW to report the error “code could not be generated correctly”.
- **(UNIX)** On a Sun when using Open Windows with the Open Look Window Manager you might encounter problems when trying to wire objects on the block diagram.
- **(UNIX)** Using the function Wait for RQS on the Sun or HP-UX would crash LabVIEW.

- **(UNIX)** LabVIEW for the Sun crashed if you tried to import an unsupported image file type. Currently the only supported format is XWD.
- **(UNIX)** Calling the Format Into String function on the Sun with the first argument being NaN caused LabVIEW to crash.
- **(UNIX)** LabVIEW sometimes crashed if a new value was added to a unsigned word enum in a cluster.
- **(UNIX)** The OR Array Elements function on HP-UX would not always produce the correct result on arrays larger than 65535. This would not happen on the Sun platform.

LabVIEW Application Builder Libraries

LabVIEW Application Builder Libraries Version 4.0.1 corrects the following problems:

- Run-time pop-up options were not available in a built application.
- Under Windows the menu option for data logging was missing in a built application.
- Using the keyboard to select components of clusters did not work in a built application.
- Editing the cursor colors caused the application to crash in a built application.
- The Autoscale option in the pop-up menu for a graph did not work correctly in a built application.
- In a built application, the online help description for a control was not displayed if the control had a keyboard navigation shortcut.
- On Windows, installers built with the Create Distribution Kit would not create the correct program group name.
- On Windows, uninstalling a distribution built with Create Distribution Kit under Windows 3.1 would give a script error.
- On Windows, canceling the options of Load Setup and Save Setup in the Create Distribution Kit would not work correctly.

Compatibility Issues between Versions 4.0 and 4.1

The MPW script `cinmake (:cintools:MPW:cinmake)` has been modified to have its default behavior to be to use the MrC compiler to compile the cin for the PowerPC platform. A new command line option (`-C`) was put in to allow users to specify that the CIN should be compiled using the older C compiler for the 68K platform.

Compatibility Issues between Versions 3.1 and 4.1

LabVIEW 4.1 contains almost no compatibility problems and should be easy to upgrade from LabVIEW 3.x, with the following exceptions.

Call Instrument VI Compatibility

As described in the LabVIEW 4.1 Release Notes, National Instruments has changed the Call Instrument VI to improve performance and add additional error checking. However, this improvement could cause some compatibility problems.

In previous versions of LabVIEW, if you did not wire any specific outputs for the Call Instrument VI, it returned all outputs. 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 the outputs of a VI consumes considerable system time and memory, especially with the VIs that contain large numbers of indicators. In LabVIEW 4.0 and later, the Call Instrument VI defaults to returning only the outputs that you request. If you have not requested any specific outputs and want all outputs returned, wire a True Boolean to the **Return All Outputs** input of the VI.

Analysis VI Changes That Cause Incompatibilities

The Linear Equations VI is now Solve Linear Equations. If you have diagrams that refer to the Linear Equations VI, they will be broken when loaded into LabVIEW 4.0 or later. To correct them, pop up on the subVI calls to the previous VI and replace them with calls to the new VI. You must also rewire the **Known Vector** input.

To use the General LS Linear Fit VI, you must pop up on calls to the subVI and select **Relink to SubVI**.

To use the Inverse Matrix VI, you must pop up on calls to the subVI and select **Relink to SubVI**. You must also rewire the **Input Matrix** input.

To use the Determinant VI, you must pop up on calls to the subVI and select **Relink to SubVI**. You must also rewire the **Input Matrix** input.

Example VI Changes That Cause Incompatibilities

Many of the example VIs have changed. If you use any example VIs as subVIs, as you would use the VIs in `vi.lib`, you first need to save a copy of the version 3.1 example VIs in another directory to maintain their functionality.

Differences between LabVIEW 3.1 and LabVIEW 4.0

This section details the changes between LabVIEW 3.1 and 4.0. To help you learn more about LabVIEW, Version 4.0 and later comes with extensive online documentation, which you can access by selecting the **Help»Online Reference...** option.

Problems Fixed in LabVIEW 4.0

Following is a list of some of the major problems in version 3.1 that are corrected in this version. Additional problems that could not easily be characterized or that were fairly cosmetic in nature are not listed.

- LabVIEW crashed when PostScript printing a diagram that contained nodes with the **Show»Terminals** option enabled.
- Locking a VI using the **Get Info...** option, while the VI was running caused LabVIEW to display an error message and quit after the VI completed execution.
- Replacing an intensity chart with a waveform chart caused LabVIEW to display an error message and quit.
- Setting the x axis on a chart to have a range of less than one could occasionally cause LabVIEW to display an error message and quit.
- Manipulating cursors of graphs inside of clusters caused LabVIEW to display an error message and quit.
- Coloring a Square or Round LED's foreground color transparent caused LabVIEW to display an error message and quit.
- **(Power Macintosh)** Changing units on a control or indicator could cause LabVIEW to corrupt memory.

- Deleting the unit label of a control and immediately editing another label caused LabVIEW to display an error message and quit.
- If you dynamically called a Reentrant VI using the Call Instrument VI and then closed Call Instrument's front panel while it was still running, under some conditions the VI was not aborted; subsequently, LabVIEW could crash if you closed the caller.
- If you used the Call Instrument VI to dynamically call a VI that had datalogging enabled but did not have a datalog path set, LabVIEW displayed an error message and quit.
- The Call Instrument VI could not correctly return paths as arguments. If you attempted to manipulate the resulting path, LabVIEW displayed an error message and quit.
- Calling the Array to Spreadsheet function using the %g floating-point conversion for some values caused LabVIEW to display an error message and quit
- Calling the Array to Spreadsheet function on an array of enumerations caused LabVIEW to crash.
- **(Power Macintosh)** Calling Interleave 1D Array with Boolean arrays caused LabVIEW to crash.
- Compiling a VI with a large number of front panel objects occasionally caused LabVIEW to display an error message and quit.
- **(Power Macintosh)** Compiling a VI using the From Decimal or To Decimal functions with a floating-point number caused LabVIEW to display an error message and quit.
- **(Power Macintosh)** Compiling a VI that called the Scale by Power of 2 function with a floating-point number and a power that was an 8-bit integer caused LabVIEW to display an error message and quit.
- Pressing the <TAB> key inside of an indicator while in run mode could cause LabVIEW to crash.
- Tabbing through the fields in the Control Editor window to resize or position an object could occasionally cause LabVIEW to display an error message and quit.
- Changing the connector pane of a subVI that had database access enabled and then relinking to the subVI could cause LabVIEW to display an error message and quit.

- When dragging objects, the screen occasionally would erase the areas you dragged over instead of redrawing the areas.
- **(Power Macintosh)** Replacing a listbox with another listbox could cause LabVIEW to crash.

Incorrect Behavior Fixed in LabVIEW 4.0

- PostScript printing did not print special characters such as ü, ö, ß, and ä.
- PostScript printing of VIs in landscape mode with the **Print Hidden Frames** option enabled did not layout the frames correctly.
- PostScript printing graphs potentially had a problem where the plot line did not accurately match the plot on the screen.
- The Print window option ignored the **VI Setup...»Surround Panel with Border** option.
- **(Power Macintosh)** Logical Shift did not work with an array of unsigned 16-bit integers, if the shift was negative and was specified as a signed, 16-bit integer.
- When editing a table's data, if you changed a cell and then clicked in an empty cell, the previous cell's data was not necessarily committed.
- Charts did not replot data after the minimum or maximum y value was modified using an attribute node.
- If a chart was overlapped by its legend, it did not update as new data was passed to it.
- **(Power Macintosh)** Digital indicators with relative time returned -0.00 instead of 0.00.
- The Coherence Function returned data that was out of bounds if the number of points was not a power of 2.
- Comparing 2D Arrays of Booleans with more than one row did not work reliably.
- The From Decimal function with a floating-point default value behaved inconsistently; the data was parsed as a uInt32. However, if an error occurred the default value was truncated to an int32.
- The From Decimal, From Hexadecimal, and From Octal functions did not handle overflow correctly.
- The Formula Node incorrectly accepted the same name for both an input and an output, which produced inconsistent results for different types of operations.

- **(Power Macintosh)** Convert Units did not work correctly on conversions with additive parts, such as degrees C to degrees F.
- **(68K Macintosh)** Wait on Occurrence did not correctly support the “ignore previous” input.
- **(Power Macintosh)** Absolute Value did not convert -0 to 0.
- When you dynamically loaded the same VI from multiple VIs, under some conditions, releasing the VI would unload the VI from memory even though not all callers had released it.
- Using the File Write VI with Convert EOL set to TRUE did not handle carriage returns followed by newlines– the last character of string was lost or duplicated.
- **(Power Macintosh)** The Complex to Polar VI returned incorrect results for extended precision numbers.
- The TCP Read VI occasionally did not return a result if the data was transmitted in several packets.
- The **standard deviation** and **mean** outputs of the Normalize Matrix VI were switched.
- The Rational Interpolation VI returned an error code if two consecutive data points had the same y value.
- The Givens Function in the General LS Fit VI was incorrect.
- The CINMake script produced make files with rules that did not work.
- LabVIEW’s CIN header files were incompatible with recent Apple include files.
- If you ejected a disk from LabVIEW's file dialog box and the disk was named “a”, LabVIEW prompted you to reinsert the disk.
- If a user was in a login group that did not exist in /etc/group, the user could not open custom controls.
- In **VI Setup...**, the **Allow User To Close Window** option was ignored, such that users could close windows regardless of whether this option was set.
- In **VI Setup...**, the **Allow User To Resize Window** option was ignored, such that users could close windows regardless of whether this option was set.

New Features Since 3.1

In addition to bug fixes, LabVIEW 4.0 contains the following new features relative to version 3.1. If you are upgrading from a version earlier than 3.1, you should also refer to subsequent sections of these upgrade notes that discuss the differences between the version you are using and 4.0.

Menu Changes

- LabVIEW now contains a **Project** menu, which includes the following options—**Show VI Hierarchy**, **This VI's Callers**, **This VI's SubVIs**, **Unopened SubVIs**, **Unopened Type Defs**, **Find**, **Search Results**, **Find Next**, **Find Previous**, and **Show Profile Window**.
- The **Edit** menu now contains the following options—**Select Palette Set**, **Edit Control and Function Palettes**, and **Create SubVI from Selection**.
- The **Edit»Text** option no longer exists. You can now change the font type, style, size, color, and so on by using the **Font Ring** option located in the toolbar.
- The **Edit»Alignment** option no longer exists. You can now change the alignment by using the **Alignment Ring** option located in the toolbar. You can still use <command-a> on Sun and on HP-UX to access this option.
- The **Edit»Distribution** option no longer exists. You can now change distribution by using the **Distribute Ring** option located in the toolbar. You can still use <command-d> on Sun and on HP-UX to access this option.
- The **Windows** menu has new options for showing the **Controls**, **Functions**, and **Tools** palettes.
- The **File»Get Info...** option, has changed to **Windows»Show VI Info...**
- The **Help** menu has new options for locking online help and for choosing between a simple or complex help view.
- The **Show Help Window** option has changed to **Show Help**.

Controls and Functions Palette Changes

The **Controls** and **Functions** menus have moved from the menu bar and are now floating palettes. You can still use pop up clicking to access a temporary copy of these palettes. If you have closed the palettes and want to open them again, choose **Windows»Show Controls Palette** or **Windows»Show Functions Palette**.

These floating palettes contain the following advantages over the list menus used in previous LabVIEW versions.

- You can tailor the items in these graphical palettes to your needs using a menu editor. You access the menu editor by selecting **Edit»Edit Control & Function Palettes**. You can create your own palette views by customizing existing views to add new subpalettes, hide options, or move items from one palette to another.
- You can mix functions, VIs, and subpalettes into the same palette. For example, the **File I/O** palette contains the high-level, easy to use file VIs plus some of the more commonly used functions. The functions and VIs that are used less frequently are located in subpalettes within the **File I/O** palette.
- You can edit the placement of a VI into various palettes or subpalettes. For example, if you create a VI using trigonometric functions, you can place it into the **Trigonometric** subpalette for easy access. You can also edit the palettes so that the functions you use most frequently are located at the top-level of a palette, while the functions you use less frequently are located on a subpalette.
- You can convert any **Controls** or **Functions** subpalette into a floating palette so that you can drag multiple items out of the same palette. To convert a subpalette into a floating palette, select the thumb tack, located in the upper left corner of the subpalette.

The following list describes the changes to the **Functions** palette.

- The **Functions** palette has a new **Help** subpalette (**Advanced»Help**), which contains VIs that you can use to control the Help window and online help and to get status about online help.
- The **Functions»Structs & Constants** option has changed to **Functions»Structures**. Individual constants are now located in the palettes according to their functionality. For example, the string constant is now located in the **String** palette, the numeric constant is located in the **Numeric** palette, and so on.
- The **Functions»Utility»File VIs** have moved to **Functions»File I/O**.
- The **Functions»Utility»Error Handlers** VIs have moved to **Functions»Time & Dialog**.
- The **Functions»Network** option has changed to **Functions»Communication**.
- The **Functions»Arithmetic** option name has changed to **Functions»Numeric**. Boolean arithmetic functions are now located in a separate palette—**Functions»Boolean**.
- The **Functions»Trig & Log** option has moved to **Functions»Numeric»Trigonometric** or **Functions»Numeric»Logarithmic**.
- The **Functions»Conversion** option has moved to **Functions»Numeric»Conversion**.
- The **Functions»Array & Cluster** option has moved to **Functions»Array** or **Functions»Cluster**.
- The **Functions»Utility»HiQ** functions have moved to **Functions»Communication»HiQ**.
- The **Functions»Miscellaneous** functions are now located in the **Functions»Advanced** palette.
- The **Functions»Utility»System** VIs have moved to **Functions»Advanced»Memory**.
- The **Functions»Utility»VI Control** VIs have moved to **Functions»Advanced»VI Control**.
- The **Functions»VI...** option has changed to **Functions»Select a VI...**

General Interface Changes

Show Panel/Diagram Window Shortcut Key has Changed

The shortcut key for toggling between the front panel and block diagram windows has changed to <command-e> on Sun and Concurrent and on HP-UX. This key changed so that the new Find feature shortcut key could use the <command-f> on Sun and Concurrent and on HP-UX.

Tools are Located in a Floating Palette

The tools in the toolbar are now located on a floating **Tools** palette. This palette automatically opens when you launch LabVIEW. You can open a temporary copy of this palette by holding down the <Shift> key and popping up on the window. If you have closed this palette and want to open it again, choose **Windows»Show Tools Palette**.

New Scroll Tool

Using the scroll tool, you can scroll in the open window.

New Object Pop-Up Menu Tool

You can use this tool to automatically open the pop-up menu of an object.

Edit and Run Mode Changes Automatically

The reorganized toolbar virtually eliminates the need to switch between edit and run modes. VIs automatically open in edit mode, until you run them. After running, VIs automatically return to edit mode so that you can debug them. To manually change modes, you can select the **Operate»Change to Run Mode** or **Change to Edit Mode** options.

Technical Support VI

LabVIEW now includes a Technical Support VI, which is located on the **Help** menu. This VI opens a series of dialog boxes that prompt you for information that is useful for solving technical support issues. After entering the information, the VI saves the information to a file that you can e-mail or fax to National Instruments.

DAQ Navigator VI

LabVIEW now includes a DAQ Navigator VI, which is located on the **Help** menu. This VI opens a series of dialog boxes that prompt you about the type of DAQ application you want to create. The VI then highlights and opens useful examples that illustrate how you might create your application, and also displays references to LabVIEW manuals that you can use for further research.

Find Option

With this option, you can search for objects or text, including functions, VIs, type definitions, labels, and so on. You can limit your search scope to a single VI, a set of VIs, or to include all VIs in memory. For more information on the **Find** option, see Chapter 4, *Creating SubVIs*, in the *LabVIEW User Manual*.

Hierarchy Window

When using the Hierarchy window, you can use options on the toolbar or the **View** menu to highlight VI connections, show or hide subVIs, expand or collapse the VI's callers, and so on. In addition, you can also search for a particular label or VI in the hierarchy. For a more detailed description of the Hierarchy window options, see Chapter 4, *Creating SubVIs*, in the *LabVIEW User Manual*.

Profile Option

With this option, you can time the execution of your VIs and then provide an interactive display for analyzing this information. Profiling statistics are grouped into four categories—basic statistics, timing statistics, timing details, and memory usage. For more information on the **Profile** option, see Chapter 27, *Performance Issues*, in the *LabVIEW User Manual*.

Creating and Wiring Controls, Constants, and Indicators

You can now pop up on a control, function, or VI input or output to create and wire the correct control, constant, or indicator type. With this feature, you no longer have to manually place a control, constant, or indicator on the window and then wire it to the appropriate object.

Creating Constants from Controls/Controls from Constants

You can copy or drag a front panel control to a block diagram to automatically create a corresponding constant. You can also copy or drag a block diagram constant to a front panel to create a corresponding control.

Drag and Drop Support for VIs, Text, and Pictures

Using drag and drop, you can drag information from the file system or other applications to LabVIEW's front panels and block diagrams. For more information on this feature, see Chapter 9, *Introduction to Front Panel Objects*, in the *LabVIEW User Manual*.

Tip Strips for Icons, Buttons, and Objects

The palette icons as well as the buttons in the toolbar now contain tip strips, which explain the function of an icon or button.

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. You might use this to provide quick access to VIs that act as tools in your system. National Instruments uses this feature to make the Tech Support and the DAQ Navigator VIs accessible from the **Help** menu. Also, if you have the Application Builder Libraries installed, you can click on the **Create Distribution Kit** option in the **Project** menu.

Any VI placed at the top-level of the project or help directory is directly placed on the corresponding menu. If you create a subdirectory, LabVIEW appends a submenu.

Help Window Displays More Information and Toggles Between Views

The Help window is now resizable and contains scrollbars so that you can view information without reducing its content to an arbitrary size. In addition, the Help window supports a simple/complex view for functions and VIs with large numbers of inputs and outputs. You can click on the Simple/Complex Diagram Help button in the Help window or select **Help»Simple Diagram Help** to toggle between views.

Online Help Enhancements

You can now access online help through LabVIEW's Help Window by clicking on the Online Help button. In addition, you can now define your own links to online help documents. For more information on creating your own help files, see Chapter 25, *Managing Your Applications*, in the *LabVIEW User Manual*.

History Improvements

The History window now includes VI revision number information, remembers the name of last person who logged in, and can show or hide empty history entries.

System File Dialog Boxes Work With LabVIEW

Previously, LabVIEW could only use its own file dialog box for opening or saving VIs. Now, you can use the file dialog boxes that come with your operating system to perform these operations. For information on setting the file dialog box, see Chapter 1, *Introduction to LabVIEW*, in the *LabVIEW Tutorial Manual*.

Preferences Location and Format

LabVIEW now stores preferences in a text file format rather than a binary file format. In addition, you can now position the preference file next to LabVIEW or your application.

Separate Default Fonts for Front Panel/Block Diagram

You can now set a font for a particular window by specifying a font from the **Font Dialog** option, located in the font ring, and then selecting either the **FP Default** or **BD Default** checkbox.

Conversion Improvements of Special Macintosh Characters to Other Platforms

LabVIEW now contains improved mapping features for the Macintosh Roman character set so that you can port VIs between platforms. However, LabVIEW cannot change characters in strings because they might represent binary data.

QuickDrawGX Printing Support for Macintosh

LabVIEW now supports QuickDrawGX's printing dialog boxes. With QuickDrawGX's dialog box, you can redirect print jobs as well as use new features provided by printing extensions.

Front Panel Features

Setting Absolute Time

You can now specify a time and date for a numeric, scale, knob or graph by selecting the Time & Date option from the **Format & Precision...** pop-up menu.

Setting Arbitrary Marker Spacing for Scales

With this feature, you can specify an arbitrary distribution of markers so that you place inner markers on an exact point of the scale. For more information on this topic, see the *Changing Scale Marker* section, in Chapter 10, *Numeric Controls and Indicators*, of the *LabVIEW User Manual*.

Changing Marker Location

In addition to typing in a scale marker for slides, ramps, knobs, or graphs, you can now drag a tick mark next to a marker to move the marker's location. For more information on this topic, see Chapter 10, *Numeric Controls and Indicators*, in the *LabVIEW User Manual*.

Setting Color in Color Ramps or Intensity Graphs

You can use this option after creating a marker to pop up on a tick mark or marker text and can then change the color using the color palette that appears.

Creating Plot and Fill Styles for Graphs

With the Common Plots, Bar Plots, and Fill Baseline options, you can create histogram graphs, bar plots, and so on, as well as to set plot width and plot fill. For more information on this topic, see Chapter 16, *Graph and Chart Controls and Indicators*, in the *LabVIEW User Manual*.

Setting Line Width for Graph Plots

With the Line Width option, you can make a line thicker than the default 1 pixel or into a hairline width, if your printer supports hairline printing.

Selecting Text and Cells in Tables

You now use the Operating tool to select text and data, while you use the Labeling tool to enter text. <Shift>-clicking extends data selection.

Hiding Overlap Shadows at Run-Time

When editing VIs, overlapping objects are highlighted with a shadow beneath the top-most object. At run-time, LabVIEW hides this shadow.

Platform Independent Controls

LabVIEW now features platform independent versions of radio buttons and check marks so that you can create VIs with a similar look, regardless of what platform your VI is running on.

Programmatic Blinking, Sizing, and Positioning

With this feature, you can specify that controls should blink, be resized, or repositioned.



Note:

Unless you are using an enhanced metafile, LabVIEW converts all other metafiles to bitmaps.

Block Diagram Features

Execution Stepping

You can use the step into, step over, and step out buttons to give you more control when debugging VIs. For more information on execution stepping, see Chapter 5, *Executing and Debugging VIs*, in the *LabVIEW User Manual*.

Probe Tool

You can now create a probe on a wire by using the Probe tool. For more information on the Probe tool, see Chapter 5, *Executing and Debugging VIs*, in the *LabVIEW User Manual*.

Breakpoint Tool

You can now set a breakpoint on VIs, nodes, wires, and diagrams to pause execution so that you can probe the wires. For more information on the Breakpoint tool, see Chapter 5, *Executing and Debugging VIs*, in the *LabVIEW User Manual*.

Copying Front Panel Terminals to New VIs

You can now copy or drag block diagrams, including diagram terminals, attribute nodes, and local variables that refer to a front panel control to a new VI without rewiring any connections. For more information on this topic, see Chapter 3, *Editing VIs*, in the *LabVIEW User Manual*.

Creating SubVIs from a VI Selection

With this feature, you can convert sections of your block diagram into a subVI, as long as doing so would not change the behavior of the VI. LabVIEW replaces the selected portion of the block diagram with the subVI and automatically wires the subVI. For more information on this feature, see Chapter 4, *Creating SubVIs*, in the *LabVIEW User Manual*.

Wiring Stubs and Tip Strips Help With Wiring

When you move over a node, a wire stub, which shows the data type of that node, as well as a tip strip, with the name of that input or output, appears. This feature should help you in wiring the correct information to the terminal.

Default Labels for Functions and VIs

When you select **Show»Label** from a function or VI pop-up menu, LabVIEW automatically displays a label with the function or VI default name inside.

Improved Local and Global Variable Interface

Local and global variables now resemble front panel terminals. Popping up on a terminal or control and selecting **Create»Local Variable** or **Global Variable** creates a corresponding local or global variable for that terminal or control.

Help Functions Control Help Window and Online Help

LabVIEW now contains help functions (**Functions»Advanced»Help**), which you can use to modify or check the status of the Help window as well as to modify online help. For more information on these functions, see **Online Reference»Function and VI Reference»Advanced Functions**.

Sequence and Case Structures Indicate Range

LabVIEW now displays the range of frames at the top of the Sequence and Case structures.

Array and Cluster Constants

You can now use array and cluster constants to create constant data on the block diagram.

Concatenate String Function Supports Arrays of Strings

You can now wire 1D arrays of strings as inputs to the Concatenate Strings function. The output consists of a single string containing the concatenation of array strings.

Scan from String and Format into String Functions

These functions can convert multiple values simultaneously, and access a dialog box that you can use to automatically create and wire format strings. For more information, see **Online Reference»Function and VI Reference»String Functions**.

Compound Arithmetic Function Uses Resizable Inputs

With this function, you can add, multiply, AND, or OR multiple values. In addition, you can invert inputs and outputs for this function. For more information on this topic, see **Online Reference»Function and VI Reference»Numeric Functions**.

Required, Recommended, and Optional Inputs and Outputs for VIs

With this feature, you can determine whether an input or output for a VI is required, recommended, or optional.

VI Control Changes

The Call Instrument VI, located in **Advanced»VI Control**, includes several new enhancements that allow it to perform more error checking and to execute calls more quickly.

- The flattened data types of the **requested outputs** input was previously ignored. The Call Instrument VI now checks the requested types against the types of the subVI you are calling and returns an error if they are incompatible.
- In previous versions of LabVIEW, if you did not wire any specific outputs for the Call Instrument VI, it returned all outputs. 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 the outputs of a VI consumes considerable system time and memory, especially with the VIs that contain large numbers of indicators. In LabVIEW 4.0, the Call Instrument VI defaults to returning only the outputs that you request. If you do not request any 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 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 front 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 example, 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.

File I/O Functions Use Error I/O

All File I/O functions now use error I/O clusters, which behave just as the File I/O Utility VI error clusters did in LabVIEW 3.1. For more information on error I/O, see **Online Reference»Function and VI Reference»File I/O Functions**.

Spreadsheet Functions Support More Formats

With the updated spreadsheet functions, you can specify a delimiter, other than a tab, in your spreadsheet. This eliminates the need to parse your spreadsheet if you have specified a comma, or some other character, as a spreadsheet delimiter.

Call Library Function

You can now use this function to communicate with Code Fragment Libraries.

Compilers Supported for CINs and DLLs

LabVIEW now supports the Symantec C++ (8.0) and Metrowerks C/C++ 68K compilers for the Macintosh as well as including project templates that make it easier to create CIN projects with the correct settings. See Chapter 24, *Calling Code From Other Languages*, in the *LabVIEW User Manual* for more information.

Performance Changes

Decreased Memory Fragmentation

Previously, LabVIEW for Power Macintosh had problems with memory fragmentation. LabVIEW now uses better memory management to reduce memory fragmentation in large applications. These changes benefit all platforms, but they particularly reduce memory fragmentation on the Power Macintosh.

Code Interface Node Changes

The following CIN data type names have changed to reduce incompatibilities with system header files.

- `boolean` changed to `BOOL32`
- `PTR` changed to `UPtr`
- `Handle` changed to `UHandle`

Instrumentation (GPIB/VISA/Serial/VXI) Changes

GPIB VIs are now Functions

LabVIEW 4.0 converts the GPIB interface from VIs to functions, which use error I/O, so that they can work in conjunction with the VISA functions. These functions should have better performance and should take up less memory than corresponding VIs. The GPIB Read and the GPIB Write functions have a **Do I/O Async** option in their pop-up menus that you can use to perform error I/O asynchronously. For more information on GPIB functions, see the *LabVIEW Instrument I/O VI Reference Manual* or the *Instrument I/O* section of the *LabVIEW Function and VI Reference Manual*.

GPIBDRV Support File is No Longer Needed

By converting GPIB VIs to functions, LabVIEW eliminated the need for the GPIBDRV file.

VISA Transition VIs are now Functions

LabVIEW converts calls to the VISA Transition VIs from 3.1 into calls to the new VISA functions. The VISA functions include a VISA refnum for connection to multiple VISA functions and a VISA attribute node for reading or writing attributes for a given instrument. For more information on VISA functions, see Chapter 5, *VISA Library Reference*, in the *LabVIEW Instrument I/O VI Reference Manual* or the *Instrument I/O* section of the *LabVIEW Function and VI Reference Manual*.

VISA now includes the following functions:

- VISA Assert Trigger
- VISA Clear
- VISA Close
- VISA Find Resource
- VISA Lock
- VISA Open
- VISA Read
- VISA Read STB
- VISA Status Description
- VISA Unlock

- VISA Write
- VISA Disable Event
- VISA Discard Events
- VISA Enable Event
- VISA Wait On Event
- VISA In8 / In16 / In32
- VISA Memory Allocation
- VISA Memory Free
- VISA Move In8 / Move In16 / Move In32
- VISA Move Out8 / Move Out16 / Move Out32
- VISA Out8 / Out16 / Out32
- VISA Map Address
- VISA Peek8 / Peek16 / Peek32
- VISA Poke8 / Poke16 / Poke32
- VISA Unmap Address
- VISA Attribute Node—For more information on attribute node descriptions, see Chapter 5, *VISA Library Reference*, in the *LabVIEW Instrument I/O VI Reference Manual* or the *Instrument I/O* section of the *LabVIEW Function and VI Reference Manual*.

Instrument Handle Parameters are Now VISA Session Refnums

The **instr handle in** input is now called the **VISA session** input and the **instr handle out** output is now called the **dup VISA session** output. Both of these parameters have changed to refnums, as well.

Data Acquisition Changes

The Intermediate VIs now appear in two groups—Intermediate VIs and Intermediate Utility VIs. Categories of VIs were moved into the appropriate subpalette rather than having a separate palette for all Easy, Intermediate, and Advanced VIs. Now, Easy I/O VIs are located on the first row of the **DAQ** subpalette, Intermediate VIs are located on the second row, and Intermediate Utility and Advanced VIs are located in subpalettes.

The following palette names have changed:

- **Digital Input and Output** has changed to **Digital I/O**.
- **Calibration and Config** has changed to **Calibration and Configuration**.
- **DAQ Utilities** has changed to **Signal Conditioning**.

The following new VIs have been added to the **Data Acquisition** palette:

- The SCXI Cal Constants VI (located in **Calibration and Configuration**)
- The 1200 Calibrate VI (located in **Calibration and Configuration**)
- The SCXI Temperature Scan VI (located in **Signal Conditioning**)

The Thermocouple Conversion VIs (Convert Thermocouple Reading VI, Convert Thermocouple Buffer VI, Volts to Temperature VI, and Temperature to Volts VI) all have N-type thermocouple support.

In addition, the Scaling Constant Tuner VI, which was located in the **Calibration and Configuration** palette, is now located in the **Signal Conditioning** palette.

Communication Palette Changes

The **Communication** palette used to be named the **Networking** menu. To accommodate this change, all example VIs are now located in `labview\examples\comm`.

The AppleEvent VIs have been separated into three categories—general AppleEvent VIs are located on the top-level of the **AppleEvent** palette; LabVIEW Specific Apple Events VIs are located in their own subpalette; and Low Level Apple Events VIs are located in their own subpalette. The **AppleEvent** palette also contains the Get Target ID VI and the PPC Browser VI.

Analysis Palette Changes

The following palette names have changed:

- **Regression** has changed to **Curve Fitting**.
- **Signal Processing** has changed to **Digital Signal Processing**.
- **Statistics** has changed to **Probability and Statistics**.
- **Array and Numeric** has changed to **Array Operations** and **Additional Numerical Methods**.

In addition, LabVIEW now supports complex matrices and vectors, Eigenvalues and Eigenvectors, singular value decomposition and other factorizations, real and complex matrix arithmetic, and real and complex matrix characterization.

The following VIs have been added to the Analysis library:

- Peak Detector
- Complex Polynomial Roots
- Sample Variance
- IIR Cascade Filter with I.C.
- Complex A x B
- Complex Determinant
- Complex Outer Product
- Cholesky Factorization
- Complex Cholesky Factorization
- Complex Conjugate Transpose Matrix
- Eigenvalues and Vectors
- Complex Eigenvalues and Vectors
- Complex Inverse Matrix
- Complex LU Factorization
- Complex Matrix Condition Number
- Complex Matrix Norm
- Complex Matrix Rank
- Complex Matrix Trace
- Complex PseudoInverse Matrix
- QR Factorization
- Complex QR Factorization
- SVD Factorization
- Complex SVD Factorization
- Create Special Matrix
- Create Special Complex Matrix
- Solve Linear Equations
- Solve Complex Linear Equations
- Test Positive Definite
- Test Complex Positive Definite

The following VIs have been updated for this release.

- A x B
- Determinant
- Outer Product
- Inverse Matrix
- LU Factorization
- Matrix Condition Number
- Matrix Norm
- Matrix Rank
- Trace
- PseudoInverse Matrix
- Cross Power
- Network Functions (avg)
- Cross Power Spectrum
- General LS Linear Fit

For information about individual VIs, see the *LabVIEW Analysis VI Reference Manual* or the *Analysis VI* section of the *LabVIEW Function and VI Reference Manual*.

Analysis VI Examples

The following VIs are located in the `labview\examples\analysis` directory.

- 2D FFT
- Interpolation Solver
- Linear Algebra Calculator
- Norm of a Matrix and Fixpoint
- Heat Equation Example
- Shortest Paths Example
- Simulation of Tomography
- Linear Differential Equation Example
- Peak Detection Example
- Financial Forecasting
- Linear Combinations

- Predicting Cost
- Regression Solver
- Real Roots Example
- Stability of Systems
- Confidence Interval Example
- Statistics Solver

New and Improved VI Examples

The following list describes new or revised examples for you to examine. For additional information about individual VIs, select **Windows>Show VI Info...** and read the descriptions included with LabVIEW.

- The FileWrite VI, located in `labview\examples\cin`
- The FileRead VI, located in `labview\examples\cin`
- The HP34401A Application Example VI, located in `labview\examples\instr\hp34401a.llb`

The `labview\examples` directory contains a new VI library—`demos.llb`, which includes VIs illustrating how various markets use LabVIEW. You can use these VIs to give you ideas for front panel arrangements and display techniques. The block diagrams for these VIs use simulated data and do not necessarily offer the best design techniques.

LabVIEW also contains a new set of VIs, located in `labview\general\queue.llb`, that you can use for managing a fixed size queue within LabVIEW. These VIs are structured as templates that you can open, modify the data type for that queue, and save for use within your own applications. These VIs also illustrate the use of occurrences. For example, if the queue is full, the Enqueue VI waits until space is available, and the Dequeue VI waits if there is not an item currently in the queue.

Changes to LabVIEW Prior to Version 4.0

Problems Fixed in 3.1

The following list describes some of the problems in 3.0.1 that were corrected in LabVIEW 3.1.

- In version 3.0.1, changing the representation on a control from the block diagram terminal pop-up menu caused LabVIEW to crash if the control had certain data range settings.
- <Shift>-clicking on the increment or decrement buttons of numerics sometimes caused LabVIEW to crash when using some data types and controls.
- Editing the scale markers for a gauge while the gauge was updating sometimes caused LabVIEW to crash.
- Selecting Stack Plots on a chart sometimes caused a crash or memory corruption.
- After changing the panel order of a VI for which you had logged data, an attempt to retrieve data sometimes caused crashes.
- LabVIEW crashed if you locked a cursor to a plot and later emptied the graph.
- **Data Range** settings for min, max, and increment did not work correctly for some data values and data types.
- String controls and indicators could not manage more than 32,768 pixels horizontally or vertically.
- When you disabled a table control, you could still scroll it.
- In some cases, LabVIEW could not work with True Color graphic displays (more than 256 colors).
- When you hid the run button specifically (not just the whole execution palette/toolbar), you could not run the VI at all, even using the Run When Opened setting. The Run When Opened option now works if the run button is hidden.
- LabVIEW displayed an error message if you had a graph inside of a cluster. This also caused problems with conversions from previous versions of LabVIEW.
- Editing arrays inside of clusters in an outer array caused LabVIEW to display an error message.

- Clicking in a probe indicator and then pressing the tab key caused LabVIEW to display an error message.
- Highly parallel, asynchronous diagrams caused LabVIEW to display an error message when you compiled the VI.
- Some diagram descriptions for VIs that were converted from 2.2.1 to 3.0 were converted incorrectly. When viewed in 3.0, LabVIEW crashed.
- The Spreadsheet String to Array function could not manage tables containing some rows with more columns than others.
- The Flush File function did not work correctly.
- The Unflatten from String function caused LabVIEW to corrupt memory if you tried to unflatten a path from data that was not really a path.
- The Formula Node returned an incorrect value when you took a negative number to an odd power, if the power was a constant. This would work if the power was a variable.
- If you wired an input of unit radians to the Sin+Cos function, the output unit type was also radians. The outputs should not have a unit.
- Modes 2 and 3 of the GPIB Write VI, which set the EOI line in addition to sending an EOS character, were incorrect. The Case structure in which the modes were formerly chosen output a decimal 2; the Case structure now outputs a hexadecimal 100.
- In previous versions, you could use the Serial Port Init VI to select an unsigned 16-bit integer for the buffer size, when it should be limited to a signed 16-bit integer. Using a buffer size greater than 32K could cause memory corruption.
- The AESend VI could not handle more than 999 characters for the data to be transferred.
- The AESend Print Document VI did not print the specified VI in version 3.0.1.
- Loop with timing functions (Wait ms or Wait until Next ms Multiple) gave up time to other applications if nothing else was processing.
- If you ran a DAQ counter VI, then quit and relaunched LabVIEW, you would receive error message -10460 `interfaceInteractionErr` until you restarted or ran the LabVIEW 2.2.1 DAQ Board Reset VI.

- When saving a VI to a VI library, you sometimes corrupted the library if the disk filled up during the copy.
- The Pulse Parameters VI could not handle negative-going pulses.
- The Quick Scale 1D and Quick Scale 2D VIs could not correctly compute the largest absolute value of the input array.

New Features in 3.1

In addition to bug fixes, LabVIEW 3.1 added the following new features relative to version 3.0.1. If you need to upgrade from a version earlier than 3.0.1, you should also refer to subsequent sections of these upgrade notes that discuss the differences between the version you are using and 3.1.

Power Macintosh Compatibility

LabVIEW for the Power Macintosh is a native application, meaning that it is compiled for the Power PC. The Power Macintosh version of LabVIEW compiles VIs to Power PC object code, and works with CINs created using Power PC-based compilers.

Power Macintosh systems frequently deliver a significant performance improvement over 680x0-based Macintosh's. In some cases, this increase in performance speed can be as much as four times faster than a comparable Macintosh system, especially in graphics-intensive applications. In other areas, performance may be approximately the same as on the 680x0 Macintosh. Future versions of LabVIEW for the Power Macintosh should continue to gain better performance as our compiler is optimized for the Power PC.

Plug-In Board Compatibility

Some of our plug-in boards have problems in Power Macintosh systems. The NuBus slots in the Power Macintosh systems are not 100% compatible with the 680x0-based NuBus slots. This incompatibility does not affect all plug-in cards. National Instruments has new revisions of all of its boards that were affected by this NuBus problem. Following is a list of all of our Macintosh boards and the shipping date of a revision that is compatible with the Power Macintosh. If you have a board that is not compatible, National Instruments can modify or upgrade it for you

Power Macintosh Hardware Compatibility

| Board | Revision of Compatible Version | Shipping Date |
|---------------|---------------------------------------|------------------------------|
| NB-TIO-10 | all revisions work | not applicable |
| NB-DMA2800 | D2 and higher | Nov 1, 1994 |
| NB-DSP230x | D2 and higher | Nov 1, 1994 |
| NB-A2000 | all revisions work | not applicable |
| NB-GPIB/TNT | D2 and higher | March 21, 1994 (see Note) |
| NB-GPIB-P/TNT | A1 and higher | March 21, 1994 (see Note) |
| NB-DIO-24 | higher than D2 | April 1, 1994 |
| NB-DIO-32F | higher than D3 | April 1, 1994 |
| NB-PRL | higher than A3 | April 1, 1994 |
| NB-MIO-16 | higher than F1 | May 1, 1994 |
| NB-MIO-16X | higher than C9 | May 1, 1994 |
| Lab-NB | higher than C6 | May 1, 1994 |
| NB-DIO-96 | higher than A1 | May 1, 1994 |
| NB-A2150 | higher than B2 | May 1, 1994 |
| NB-A2100 | higher than D | May 1, 1994 |
| NB-DMA-8G | higher than D7 | May 1, 1994 |
| NB-AO-6 | higher than C2 | May 1, 1994 |
| NB-MXI | higher than B3 | May 15, 1994 (see Note) |

**Note:**

Almost all VXI applications work with all revisions of the NB-MXI. Applications that transfer data to VXI devices and simultaneously accept transfers from bus master VXI devices into the Power Macintosh local memory do not work. This is a deadlock condition. NB-MXI boards that are revision B4 and higher can handle this case.

Previous and current versions of the Turbo488/NAT4882-based NB-GPIB boards do not work in the Power Macintosh, nor will they in the future; we recommend that you upgrade to a TNT4882C-based board.

If you upgraded your Macintosh to a Power Macintosh by replacing the motherboard, you essentially have a new Power Macintosh, and you must use a new or upgraded board. You can, however, upgrade some Quadra computers by adding the PDS slot upgrade board. Although this combination is not fully tested, your existing National Instruments boards should continue to work correctly.

Please contact your regional National Instruments sales or service representative or an applications engineer for additional information or to arrange for an upgrade.

Power Macintosh CINs

CINs compiled for the Motorola 680x0 (68K) Macintoshes do not run in LabVIEW for the Power Macintosh. They must be recompiled for the Power Macintosh using one of the available compilers for the Power Macintosh. Most CINs should recompile without modification.

LabVIEW also does not currently work with fat binaries (a format that includes multiple executables in one file, in this case both 68K and Power Macintosh executables).

Some CIN code that calls Macintosh OS or Toolbox functions may require source code changes. Any code that passes a function pointer to a Mac OS or Toolbox function must be modified to pass a Routine Descriptor (see Apple's *Inside Macintosh* chapter on the Mixed Mode Manager, available in the Macintosh on RISC SDK from APDA). Also, if you use any 68K assembly language in your CIN, it must be ported to either C or Power PC assembly language.

The LabVIEW tools for building Power Macintosh CINs are similar to the tools that are available for 68K CIN development. Some development issues arise, though, concerning building both the Power Macintosh and 68K versions of a CIN in the same directory. Because

the naming conventions for object files and .l**s**b files are the same, it is important to make sure that one version of these files does not replace the other. These issues are dealt with in different ways, depending on your development environment.

Currently, there are only two development environments available for the Power Macintosh: Apple Macintosh on RISC SDK, which runs in the Macintosh Programmer's Workshop (MPW) environment, and Metrowerks CodeWarrior, an integrated development environment very similar to Symantec THINK C. Symantec has not released a version of THINK C for the development of Power Macintosh code at this time.

See the *LabVIEW Code Interface Reference Manual* for information on creating CINs using MPW or Metrowerks CodeWarrior.

Menu Changes

The **Show Help Window** and **About LabVIEW...** options have been moved to a new, **Help** menu. The **Help** menu also has an **Online Reference...** option, which contains extensive online documentation.

The **Windows** menu has new options for showing the History window for a particular VI (described in the *VI History Option* section that follows) and showing the Error List window (described in the following *Improved Error Window* section). There is a new tile option that you can use to tile a VI front panel and diagram window with the panel at the top and the diagram at the bottom.

The **Controls** menu has a new **List & Ring** option that contains the existing ring controls as well as the new list controls.

The **File** menu contains new printing options, **Print Documentation...** and **Print Window...**, which replace the **Page Layout**, **Print Preview**, and **Print** options in version 3.0.1. You can preview a printout with the **Print Documentation...** dialog box. See Chapter 7, *Printing and Documentation*, in your *LabVIEW User Manual* for information on these new options.

General Interface Changes

VIs Open in Edit Mode by Default

When you open a VI in the development system, the VI is automatically placed in edit mode. If you prefer to have VIs open in run mode, you can change the default mode using an option in the **Miscellaneous** page of the Preferences dialog box. Although this change is useful for your VIs under development, it may be problematic for VIs you do not want to change accidentally, such as those delivered with LabVIEW. For this reason, we suggest you protect files in the `examples` and `vi.lib` folders plus your own finished VIs.

Enclose Existing Diagrams in Structures

When you create a structure (While Loop, For Loop, Sequence structure, or Case structure) by selecting it from the **Functions** menu or the **Functions** menu popup, the structure is not immediately created. Instead, you can set the location and size of the structure by clicking and dragging a rectangular area on the diagram. By dragging out this rectangle, you set the size of the new structure. In addition, any objects in that rectangular area are moved into the structure. Wires that cross the boundary are not broken; instead, tunnels are created as necessary. When placing part of an existing loop within a new structure, be careful not to accidentally highlight the loop terminals with the rectangle of the new structure. If you encompass one of the terminals, the entire existing loop is placed within the new structure.

After selecting a structure from the **Functions** menu, if you click on a diagram without dragging out a region, the structure appears on the diagram at its default size with nothing inside.

Remove Structures without Losing Contents

You can remove a structure without losing the contents by selecting an option from the structure's pop-up menu. In the case of While Loops and For Loops, the contents of the loop are copied to the underlying diagram. Any wires that were connected by tunnels are automatically connected together.

In the case of a Sequence or Case structure, removing the structure only preserves the visible frame or case. All other frames or cases are deleted. Because of this action, a dialog box appears when you select this operation, informing you that hidden frames or cases will be lost.

The dialog box also has options you can use to cancel or continue with the operation.

Cancel Loading a VI

If you load a VI that takes more than a few seconds to open, LabVIEW displays a status dialog box. This status dialog describes the subVIs that are being loaded, and has an option you can use to cancel the loading process.

Replace a VI for Another with the Same Name

In previous versions of LabVIEW, if you popped up on a subVI icon and tried to replace the subVI with a different subVI, you could not choose a VI with the same name as the original name.

Now, when you try to replace a subVI, you are asked whether you want to substitute all calls to the original subVI with calls to the new subVI. The replace operation applies to all open VIs and their subVIs that reference the original subVI.

VI History Option

The VI history option is a feature intended to help developers keep track of changes that they make to a VI. It is not a method for comparing two VIs to detect differences. Developers can use this option to record changes and assign version numbers to VIs.

Some options related to VI history include the **User Name** login option, **VI Setup** changes, **Preference** dialog changes, and the **Show History** option.

Enhanced Save with Options Dialog

A reorganization of the Save with Options dialog box (accessed by selecting **Save with Options...** from the **File** menu) now makes it simpler to save an entire hierarchy of VIs for distribution. With the new features you can selectively save an entire hierarchy, minus the VIs in `vi.lib`, and save all external subroutines that are referenced by VIs in your hierarchy.

Top-Level VIs at Beginning of the List in the File Dialog

When top-level VIs are distributed together with their subVIs in one library, it is difficult for the user to determine which VIs to open without going through the entire list and reading all the VI names.

To make the top-level VIs easier to identify, the file list in the File Dialog box now shows VIs that are marked *top level* (using the **Edit VI Library** dialog box) at the beginning of the list, followed by a separator and then the other VIs in the library.

Filtering Options in File Dialog

The File Dialog box contains two new controls—a filter ring and a pattern string.

The filter ring contains a list of filters that the user may select to control the files listed in the File Dialog. This list always includes the **View All** and **Custom Pattern** items. It also includes additional filter options, depending upon how the File Dialog is used. For instance, if you select **Open**, options to view only VIs and controls are added to this filter ring.

When **View All** is selected, all files are visible. When **Custom Pattern** is selected, a pattern string becomes visible. The user can then enter a pattern-matching expression such as `*.txt`.

Improved Error Window

You can use the **Show Error List** option in the **Windows** menu to bring up the Error List window at any time. This window now contains a list of VIs with errors or warnings. Selecting a name from this list displays the information for the selected VI. This is useful in situations where you discover that the top-level VI is broken, but the error that caused it to break is actually in a subVI.

In addition, errors are now sorted into distinct lists (Front Panel Errors, Block Diagram Errors) that make it easier to navigate through the list of error messages.

Preferences Option Changes

You can use the **Performance & Disk** page of the Preferences dialog box to control the interaction between LabVIEW and background applications, and to designate VI memory usage. The disk space checking preferences are now part of this page. See the *Configuring LabVIEW* section of Chapter 8, *Customizing Your LabVIEW Environment*, in your *LabVIEW User Manual* for details.

Using the **Fonts** page, you can change the fonts that LabVIEW uses for the application, system, and dialog fonts. See the *Configuring*

LabVIEW section of Chapter 8, *Customizing Your LabVIEW Environment*, in your *LabVIEW User Manual* for details.

With the **Printing** page, you can choose between Standard Printing (enables the Macintosh to convert drawing commands to printer commands) and PostScript printing. If you select PostScript printing, you get options to select PostScript level 1 or level 2, and to enable color/grayscale printing. To use PostScript printing, you must have a PostScript printer. See Chapter 7, *Printing and Documentation*, in your *LabVIEW User Manual* for details.

You can use the **History** page to choose the default history settings for a given VI. You can also use it to determine the user name for the history window. See Chapter 25, *Managing Your Applications*, in your *LabVIEW User Manual* for details.

The **Miscellaneous** page has several new options. You can select whether the Error List information box should contain warnings in addition to errors. You can select whether it should list warnings for objects that are not available in the LabVIEW Student Edition (the Student Edition is sold only through school bookstores to students). You can also specify whether all function keys should be assignable to controls (see the *Key Navigation Dialog* section that follows), or whether some should map to their standard meaning (F1 through F4 usually have the functions cut, copy, paste and clear, and F10 usually performs a specific action). You can set the VIs to open in run mode or in edit mode. You can also select whether LabVIEW should use the U.S. standard for decimal points (a period), or the localized standard (usually a comma). See the *Configuring LabVIEW* section of Chapter 8, *Customizing Your LabVIEW Environment*, in your *LabVIEW User Manual* for details.

You cannot use the Preferences dialog box to change the amount of memory allocated to LabVIEW. To change memory allocation, use the Finder **Get Info...** option.

Front Panel Features

Type Definitions

You can now easily change all copies of a type definition to ordinary controls. To do so, first select **Save As...** for the type definition. Change it to an ordinary custom control by deselecting the type definition boxes, and then select **Apply Changes** from the **File** menu. This

changes all copies of the type definition to ordinary controls on all front panels currently in memory.

Listbox Controls

New listbox controls are available from the **List & Ring** palette of the **Controls** menu. You can use the listbox to present the user a scrollable list of options, similar to the list you see in the File Dialog box. You use the Single Selection Listbox if you want to limit the number of selections to one. You use the Multiple Selection Listbox if you want to allow more than one selection. See Chapter 14, *List and Ring Controls and Indicators*, in your *LabVIEW User Manual* for information on how to use these new controls.

Key Navigation Dialog

All front panel controls have a new **Key Navigation...** option. Use this option to associate a keyboard key combination with a given control. When you enter that key combination while in run mode, LabVIEW responds as though you clicked on that control. The associated control becomes the key focus. If the control is a text control, any existing text within that control is highlighted. If the control is a Boolean control, you toggle the button. You can also use this dialog box to cause a given control to be skipped when the user tabs from control to control. See Chapter 9, *Introduction to Front Panel Objects*, in your *LabVIEW User Manual* for information on this dialog box.

String Enhancements

The string control can now display very large strings (strings with more than 32K pixels). With previous versions, text was clipped to this boundary.

You can use the **Single Line** option in the string pop-up menu to limit a string to prevent the user from entering carriage returns or newline characters into a string control. This option does not prevent the control from displaying newlines or carriage returns if it gets them either from a diagram or from the user pasting data into the control.

There are now several different display options in the string pop-up menu. You can choose to display string data in one of the following ways:

- Display data in normal fashion
- Display backslash codes in place of nonprintable characters

- Display data in a password style, where the control displays an asterisk symbol (*) for each character you enter into it
- Display data as Hex values

See Chapter 12, *String Controls and Indicators*, of your *LabVIEW User Manual* for details on these new options.

Table Font Capability

You can now change the font of text within a table using the **Text** menu.

Graph and Chart Enhancements

Several options under the **X Scale** and **Y Scale** submenus have been combined into a new **Formatting...** dialog box. You can use this formatting dialog box to set a scale factor for scales. These scale factors determine the initial value and spacing between points on a waveform chart or graph, or along the scales of an intensity chart or graph. See Chapter 16, *Graph and Chart Controls and Indicators*, of your *LabVIEW User Manual* for information on this dialog box.

In addition, the graph palette has new options for panning (scrolling the display area of a graph) and zooming into and out of sections of the graph. See Chapter 16, *Graph and Chart Controls and Indicators*, of your *LabVIEW User Manual* for information on these palette options.

Several new attributes have been added for graphs.

- The Allow Drag attribute is now a part of the Cursor Info cluster. If this attribute is false, the cursor cannot be moved by dragging it.
- The Cursor Locked attribute in the Cursor Info cluster is now a U32; in the previous version it was a Boolean. As a numeric, it provides three lock options—unlocked, snap to nearest point on any plot, and locked to a specific plot (specified separately by the Cursor Plot attribute).
- A new Selected Cursors attribute returns an array of the currently selected cursors.
- X Flipped and Y Flipped are new attributes in the X Scale Info and Y Scale Info clusters, which are described in the *Scale Enhancements* section.

Scale Enhancements

In version 3.0.x of LabVIEW, a scale's orientation (whether a scale was left to right versus right to left, or top to bottom versus bottom to top) could be changed as a side effect of setting the minimum and maximum values from the scale attributes. In version 3.1, new X Flipped and Y Flipped attributes make control over the orientation of a scale more explicit.

With these new X Flipped and Y Flipped attributes, setting the scale minimum or maximum never causes the scale to flip. If you set the cluster containing the minimum, maximum, and increment, LabVIEW determines if the minimum is larger than the maximum and switches them if necessary (the minimum of the two values is used to set the minimum value of the scale). If you set either the minimum or the maximum value individually, the other value might change to keep it larger than the minimum or smaller than the maximum. See Chapter 21, *Attribute Nodes*, of your *LabVIEW User Manual* for information on the scale attributes.

Picture Control Mouse Attributes

If you have the Picture Control toolkit, you can use a new mouse location attribute to read the location of the mouse, if it is within the picture control display. You can also use this attribute to tell if the mouse button is down and if any modifier keys (such as the <shift> key, option key, or menu key) are pressed. Use the Help window with the attribute node for the picture control to understand the new mouse attribute options.

Diagram Features

Code Interface Node Changes

The **Create Header File** pop-up option has changed to a **Create .c File** option, which creates a CIN source file. See the *LabVIEW Code Interface Reference Manual* for information on the contents of CIN source files. Previously created CINs do not need to be recompiled, and their source files should work with the new version. Using the **Create .c File** option simplifies the creation of new CINs, because for many CINs you only have to fill in the contents of the CINRun routine.

When you create the source file, LabVIEW tries to use the names of the input wires for variable and data type names.

TCP Networking Enhancements

Using an input of the TCP Open Connection VI, you can choose the port to use for the connection. Some servers may only allow connections to clients that use port numbers within a specified range, where the range is dependent upon the server.

You can use the existing TCP Listen VI to wait for a TCP connection at a specified port. There are also two new VIs that you can use as alternative methods for listening for connections. With these new VIs, you can create a listener using TCP Create Listener, and use Wait on Listener to actually listen and accept new connections. TCP Create Listener returns a listener ID that you can pass to Wait on Listener. Wait on Listener returns the connection ID for any connection it opens. It also returns a copy of the same listener ID that was passed to it, which you might pass to a subsequent Wait or to TCP Close. When you are finished waiting for new connections, you can use TCP Close to close a listener. You may not read or write to a listener.

The TCP Listen VI performs both of these operations in one VI. The advantage of these new VIs is that you can cancel a listen operation by calling TCP Close. This is useful in the case where you want to listen for a connection without using a timeout, but you want to cancel the listen when some other condition becomes true (for example, when the user presses a button).

See the *LabVIEW Networking Reference Manual* for information on the changes to the TCP VIs.

UDP Networking Capability

LabVIEW now works with the User Datagram Protocol (UDP), one of the protocols in the TCP/IP suite. UDP is generally used in applications where you need the ability to broadcast information to multiple sites. UDP works at a lower level than TCP. As a result, UDP can be more efficient than TCP, but it may be more difficult to program, because UDP does not deliver the same level of reliable data transmission as TCP. Typically, UDP is used in applications where reliability is not critical. For example, an application might transmit informative data to a destination so frequently that a few lost segments of data are not problematic. See the *LabVIEW Networking Reference Manual* for information on the UDP VIs.

File Utility VI Enhancements

Using the File Utility VIs, you can now read entire files without having to input a count value that you know is larger than the file size. The method used varies with the VI. The default count value is now to read the entire file.

VI Control Utility VIs

The **VI Control** palette from the **Utility** submenu of the **Functions** menu contains VIs you can use to dynamically load, call, and close other VIs. When you call a VI dynamically, you can determine if you want the VI to open its panel when called (and close it if originally closed). You can also pass parameters to and from the dynamically loaded subVI. See **Online Reference»Function and VI Reference»VI Control Utility VIs** for information on these VIs.

Probe Enhancements

There are several new options to access and use with the probe tool—you can probe using an indicator that you select or you can have LabVIEW choose a probe for the wire. You can also use the **Find Probe** and **Find Wire** options to help you work with the probes you create.

When you pop up to create a probe, the pop-up menu has two options. You can select **Probe** to create a new probe using the default type of control for the wire, or you can select a control from the **Controls** submenu.

If you pop up on a wire for which you have created a probe window, the wire has a **Find Probe** option. If you select this option, the probe window appears in front of all other windows and is highlighted momentarily.

If you pop up on a control or indicator in the probe window, you can find the associated wire by selecting the **Find Wire** option.

See Chapter 5, *Executing and Debugging VIs*, of your *LabVIEW User Manual* for information on how to use these new probe features.

Printing Features

PostScript Printing

LabVIEW now works with PostScript printing. If you have a PostScript printer, you can take advantage of the following benefits:

- PostScript printouts can more accurately reproduce the screen image.
- PostScript facilitates the printing of high-resolution graphs.
- PostScript reproduces patterns and linestyles more accurately.

LabVIEW works with Level 1 PostScript and Level 2 PostScript. Notice that Level 2 PostScript works with color.

You can use the Preferences dialog box to choose PostScript printing. If you select PostScript printing, then you have the option to select Level 2 PostScript, as well as color printing.

See Chapter 7, *Printing and Documentation*, of your *LabVIEW User Manual* for information on printing and the Preferences dialog box.

Print Options

The printing options in LabVIEW have been significantly reorganized, with a number of new capabilities added. Following is a brief list of these new capabilities.

- You can choose to have a panel or diagram scaled to fit on a page, if possible, or select best fit for multiple pages.
- You can print control descriptions.
- You can print a list of a VI's subVIs, including each subVI's icon, name, and path.
- You can save the text information for a VI to a text file. This information includes the VI name, path, description, control information, and subVI information.
- The **VI Setup** option delivers more control over the way a panel prints during programmatic printing.

The **File** menu now includes the following two printing options: a **Print Window...** option (for fast printing) and a **Print Documentation...** option that incorporates the Layout Options and the Preview option. In addition, **VI Setup** now has options related to programmatic printing. See Chapter 7, *Printing and Documentation*,

of your *LabVIEW User Manual* for information on how to use these new options.

Performance Changes

Saving VIs is Much Faster

Due to a number of changes in LabVIEW, saving VIs with version 3.1 should be considerably faster than in previous versions. This increase in speed is most noticeable when saving large VIs, or when saving VIs into large libraries.

Memory Monitor VI

The new Memory Monitor VI (available from `memmon.llb` in the `examples` folder) makes it easier to determine how your VIs use memory. This VI is described in Chapter 27, *Performance Issues*, of your *LabVIEW User Manual*.

Instrumentation (GPIB/Serial/VXI) Changes

The GPIB, serial, and VXI VIs have all been moved to the **Instrument I/O** submenu of the **Functions** menu. This menu also contains the new VISA Transition Library, which is used by our new instrument drivers.

Instrument-driver developers and users can access the VISA Transition Library for an upgrade path to the VISA (Virtual Instrument Software Architecture) I/O library. VISA supplies a single interface library for controlling VXI, GPIB, RS-232, and other types of instruments. The VISA Transition Library works with the standard set of I/O routines used by LabVIEW instrument drivers.

The VISA Transition Library is a subset of the overall VISA feature set, providing the functionality needed by instrument drivers in an interface-independent fashion for GPIB, MXI, embedded VXI, and GPIB-VXI controllers. If a platform does not yet have the full VISA I/O library, the VISA Transition Library maps to the particular I/O library calls available on that platform. Platforms that have the full VISA I/O library simply map the VISA Transition Library directly to native VISA calls.

Data Acquisition Changes

New Counter VIs

Several new intermediate and high-level counter VIs have been added for use with Am9513 and DAQ-STC counter chips. (The DAQ-STC is used on the E-series boards.) These VIs are available from the **Easy I/O** and the **Counter** palettes in the **DAQ** submenu. The counter VIs that were in the **Counter** palette have been moved to the **Advanced** palette, except for the ICTR Control VI, which is used with the 8253 counter chip. Also, the DAQ subdirectory in the `examples` directory contains `counter.llb`, which has a number of examples that show how to use the new counter VIs.

Count Direction on Am9513-based Boards

In the CTR Mode Config and CTR Control VIs, you cannot change the **count direction** from its default setting. Refer to the *Conventions Used in This Manual* section in the *About This Manual* chapter of the *LabVIEW Data Acquisition VI Reference Manual* or the for a list of Am9513-based boards or of the *LabVIEW Function and VI Reference Manual*.

Definitions of Low-Level CTR Pulse Config Parameters

The parameters named period one and period two are now phase 1 and phase 2, and their definitions are different. See the complete description in the section entitled *Compatibility Issues Between Versions 3.0.1 and 3.1*.

Compatibility Issues between Versions 3.0.1 and 3.1

Analysis VI Changes That Cause Incompatibilities

The name of the Peak Detector VI is now Threshold Peak Detector. If you have diagrams that refer to the Peak Detector VI, they will be broken when loaded into LabVIEW 3.1. To fix them, pop up on the subVI calls to the previous VI and replace them with calls to the new VI.

The General LS Linear Fit VI has changed with the addition of an input array control, **Standard Deviation**, which you can use to control parameter weighting, and an output array indicator, **Covariance**, which

you can use to test the *goodness-of-fit* as well as the validity of the model function. You need to pop up on calls to the subVI and select **Relink to SubVI** to use the new version.

IIR Filter VIs (Butterworth Filter, Chebyshev Filter, Inverse Chebyshev Filter, Elliptic Filter, and Bessel Filter) now use cascade filter design techniques for improved stability. The connector panes for each of these VIs have not changed, nor has the filtering functionality.

The IIR coefficient VIs (Butterworth, Chebyshev, Inverse Chebyshev, Elliptic, and Bessel Coefficients) now return the new cascade-form coefficients. The output coefficients are returned in a cluster called IIR Filter Cluster. These five VIs are not compatible with the previous *direct form*. If you need direct-form coefficients, there is a new VI that converts cascade-form coefficients to the direct-form coefficients. This new VI is called Cascade->Direct Coefficients, and is located in the filters VI library. An example called Butterworth Direct Filter.vi located in fltrxmpl.llb of the analysis subdirectory of the examples folder demonstrates the use of this new VI.

The FIR Windowed Coefficients VI now returns symmetrical coefficients to guarantee that the FIR Windowed Filter phase responses are linear.

Pulse Parameters has been corrected to handle negative-going pulses. Additionally, several output control types were changed from integer to double (64-bit floating-point) for improved time parameter estimation (rise time, fall time, and slew rate).

Attribute Changes That Cause Incompatibilities

The following attributes either are new and increase the size of attribute clusters or have different data types or values:

- The Allow Drag attribute is now a part of the Cursor Info cluster. If this attribute is false, the cursor cannot be moved by dragging it.
- The Cursor Locked attribute in the Cursor Info cluster is now a U32; in the previous version it was a Boolean.

- X Flipped and Y Flipped are new attributes in the X Scale Info and Y Scale Info clusters.
- The String Control “\” Codes attribute is now a Display Style attribute with more options. This attribute was a Boolean in previous versions and is now a numeric. Diagrams using the old attribute will be broken because of the change in the data type.

File Utility VI Changes That Cause Incompatibilities

Some of the file utility VIs (Read Characters From File, Read Lines From File, Read From Spreadsheet File, Read From I16 File, and Read From SGL File) are now configured to read the entire file by default. You may need to change VIs that call these VIs to set the counts correctly.

Incompatibilities Created by Using Counters to Generate Pulses with Am9513 and DAQ-STC Devices

Prior to LabVIEW 3.1, the process used to set the shape of single or continuous pulses was confusing and differed from platform to platform. To alleviate this confusion and to help you make portable VIs, the following rules now apply to LabVIEW-generated pulses:

- All pulses consist of a delay phase called phase 1 followed by a pulse phase called phase 2. You can program these low-level parameters directly using the advanced CTR Pulse Config VI. (These parameters were previously named period one and period two; however, which was the delay phase and which was the pulse phase changed depending on the type of pulse.)
- The pulse polarity is the polarity of phase 2.
- The period of the pulse is the sum of the two phases, and the frequency is $1/\text{period}$.
- The duty cycle is phase 2 (pulse width) divided by the period.

Prior to LabVIEW 3.1, the delay phase of a continuous pulse train was set to one cycle of the timebase if you used frequency or period and duty-cycle parameters. As a result, the train appeared to start with the pulse phase rather than with the delay phase. This behavior has been corrected.

When you stop a pulse-generating counter in version 3.1 using the CTR Control VI with control code 0 (stop and reprogram), the counter output returns to the inactive level (that of phase 1). Previously it remained at whatever level it was at when the counter stopped.

These changes may require changes to your VIs that generate pulses. These alterations will break a VI only if it uses Bundle by Name or Unbundle by Name to access the old, low-level parameters named period one and period two.

Example VI Changes That Cause Incompatibilities

Many of the example VIs have changed. If you use any example VIs as subVIs, as you would use the VIs in `vi.lib` (a practice we do not recommend), you first need to save a copy of the version 3.0.1 example VIs in another directory to maintain their functionality.

Changes Introduced between Versions 3.0 and 3.0.1

This section details changes between LabVIEW 3.0 and 3.0.1. You only need to read this section if you are upgrading from 3.0 or earlier.

LabVIEW 3.0.1 was primarily a maintenance release with bug fixes and performance improvements. There was also a small number of enhancements.

Problems Fixed in 3.0.1

Following is a list of some of the major problems in 3.0 that were corrected in version 3.0.1.

- Several problems were corrected that lead to the error message `Wire Stretched and Broke`.
- The unit for radiation, `rem`, is now correctly defined.
- Reading and writing of Boolean arrays to bytestream files works correctly.
- The Elliptic Coefficients VI produces coefficients correctly.
- From Decimal works reliably.
- The Set EOF function works correctly.

- On original Macintosh II machines, some customers encountered an incompatibility with LabVIEW and NuBus boards that act as bus masters, which caused LabVIEW to crash. For example, this incompatibility occurred with the NB-DMA-2800 board and some graphics accelerator boards. This problem only affected Macintosh II computers. It did not affect Macintosh IIcx, IIsi, and so on. National Instruments fixed this incompatibility.
- LabVIEW had a performance problem in which asynchronous device drive calls used by the GPIB, serial VIs, and wait functions, gave other applications a chance to execute. This sometimes slowed performance and reduced the resolution of the Wait function. We changed this to improve runtime performance.
- A memory leak was corrected in the AE Send VI.

New Features Added in LabVIEW 3.0.1

In addition to bugs, LabVIEW 3.0.1 introduced a few new features. Two new options were added to the Preferences dialog box. The **Performance & Disk** page had a new option with which you could force LabVIEW to deallocate the memory of a VI after it completed execution. This improved memory usage in some applications because subVIs deallocate their memory immediately after executing. However, because LabVIEW must allocate and deallocate memory more frequently, using this option could also slow performance.

You can configure LabVIEW to show dots at wire junctions on diagrams using an option on the **Miscellaneous** page of the Preferences dialog box. This improvement could make it easier to differentiate between two wires that cross and a wire that branches.

LabVIEW 3.0.1 introduced a set of utility VIs for communicating with HiQ, a mathematics and numerical analysis program from National Instruments. These VIs are in the **Utility** submenu of the **Functions** menu in the **HiQ** palette. See **Online Reference»Function and VI Reference** for more information about VIs to link LabVIEW to HiQ.