# LabVIEW™

## Version 5.0

These upgrade notes describe the process of upgrading LabVIEW for Windows, Macintosh, and UNIX to version 5.0.

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

# About These Upgrade Notes

This document is historical. It records all of the features, compatibility issues, changes, and bug fixes from versions prior to LabVIEW 3.0.1 until LabVIEW 5.0. Upgrade issues pertain to LabVIEW 5.0 only.

## For More Information

For more information about features, refer to the *LabVIEW User Manual* and the *G Programming Reference Manual*. LabVIEW also offers extensive online documentation, which you can access by choosing **Help**»**Online Reference...**.

# Contents


Upgrade Issues ..... 3
- Converting VIs ..... 3
- Upgrading Applibs and Toolkits ..... 4
- Upgrading Previous Versions of LabVIEW ..... 5
  - Upgrading from LabVIEW 4.*x* ..... 5
  - Upgrading from LabVIEW 3.*x* ..... 6
  - Upgrading from LabVIEW 2.*x* ..... 7
  - Upgrading from Versions prior to 2.*x* ..... 7

LabVIEW Features ..... 7
- New Features in LabVIEW 5.0 ..... 7
  - Multithreading ..... 7
  - ActiveX Automation ..... 9




*321780A-01*  *January 1998*

Instrumentation ..... 10
Data Acquisition ..... 11
Translation Tools ..... 12
General Interface ..... 13
New Features in LabVIEW 4.1 ..... 18
Compatibility and Added Support ..... 18
Data Acquisition ..... 19
General Interface ..... 21
Performance ..... 22
Printing ..... 22
New Features in LabVIEW 4.0 ..... 23
Block Diagram Features ..... 23
Compatibility and Added Support ..... 27
Data Acquisition Changes ..... 27
Front Panel Features ..... 28
General Interface Changes ..... 29
Instrumentation (GPIB/VISA/Serial/VXI) Changes ..... 32
Menu Changes ..... 33
Palettes ..... 36
Performance Changes ..... 37
New Features in LabVIEW 3.1 ..... 38
Block Diagram Features ..... 38
Compatibility and Added Support ..... 39
Data Acquisition Changes ..... 42
Front Panel Features ..... 42
General Interface Changes ..... 44
Instrumentation (GPIB/Serial/VXI) Changes ..... 47
Menu Changes ..... 48
Performance ..... 48
Printing Features ..... 48
New Features in LabVIEW 3.0.1 ..... 49
Compatibility Issues ..... 50
Compatibility between Versions 4.1 and 5.0 ..... 50
Compatibility between Versions 4.0 and 4.1 ..... 50
Compatibility between Versions 3.1 and 4.1 ..... 50
Compatibility between Versions 3.0.1 and 3.1 ..... 51
Changes and Bug Fixes ..... 53
LabVIEW 5.0 ..... 53
LabVIEW 4.1 ..... 56
LabVIEW 4.0.1 ..... 61
LabVIEW 4.0 ..... 65
LabVIEW 3.1 ..... 70
LabVIEW 3.0.1 ..... 72

# Upgrade Issues

If you are upgrading from LabVIEW 4.*x*, review the following sections: *Converting VIs*, *Upgrading Applibs and Toolkits*, and *Upgrading from LabVIEW 4.x*.

If you are upgrading a version of LabVIEW prior to 4.*x*, read *Converting VIs*, *Upgrading Applibs and Toolkits*, and *Upgrading Previous Versions of LabVIEW* from your version through version 4.*x*.

## Converting VIs

Upgrading LabVIEW is an automated process. When you open a VI created in a previous version, LabVIEW automatically converts and compiles the VI.

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

You can estimate the amount of memory required to convert VIs by totalling the amount of memory your VIs and all of their subVIs occupy on disk. If these VIs are in VI libraries, add approximately 30 percent of the VI library size because the VIs are compressed. The conversion process might require at least that much memory and an additional 3 MB of memory to run LabVIEW.

If your computer does not have enough memory to convert your VIs all at once, 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 then can progress gradually to the higher levels of the hierarchy. You also can choose **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 with the **Help**»**About LabVIEW...** option, which summarizes the amount of memory you have used.

**(Macintosh)** Before converting VIs, increase the memory allocated to LabVIEW from the Finder by selecting the LabVIEW icon, then choosing **Windows»Show VI Info...** from the menu.

## Upgrading Applibs and Toolkits

Most existing toolkits function with LabVIEW 5.0 without problems. However, you need to move the VIs so they appear in the menus. LabVIEW 5.0 is compatible with toolkits designed for 3.0, with the following exceptions.

You must upgrade the following toolkits for compatibility with LabVIEW 5.0:

- **LabVIEW Application Builder**—You must upgrade to LabVIEW Application Builder 5.0. This upgrade is free to existing users of the LabVIEW Application Builder. If you have the Professional Development System, the new version of the application builder libraries is included in the installation.
- **Professional G Developers Toolkit**—If you have the Professional G Developers Toolkit 4.1, you must upgrade to version 5.0. This upgrade is free to existing users of the Professional G Developers Toolkit. If you have the Professional Development System, the new version of the Professional G Developers Toolkit is included in the installation.
- **LabVIEW Test Executive**—If you use LabVIEW Test Executive 5.0 or earlier, you must upgrade to LabVIEW Test Executive 5.1. This upgrade is free to existing users of LabVIEW Test Executive 5.0.

With minor exceptions, you can use the previous version of the following toolkits with LabVIEW 5.0:

- **Picture Control Toolkit for G**—You can use the Picture Control Toolkit 1.0 with LabVIEW 5.0 with the exception of the Draw 1-bit Pixmap VI. You can download an updated version of this VI from the National Instruments FTP site (`ftp.natinst.com`). The Picture Control Toolkit is being updated to include the fix mentioned above, and the upgrade is free to existing users.
- **Internet Developers Toolkit for G**—You can use the Internet Developers Toolkit 4.1 with LabVIEW 5.0, but you must delete `printvi.llb`, located in the `user.lib\internet\image` directory. The Internet Developers Toolkit is being updated to version 5.0 to include this fix, and this upgrade is free to existing users.

The following toolkits do not install VIs in a location that causes them to appear in the palettes. These toolkits are being updated to version 5.0. You can use the existing toolkits by moving VIs to `vi.lib\addons` or `user.lib`. Alternatively, you can choose **Edit»Edit Control and Function Palettes** and add them to the palette of your choice.

- Picture Control Toolkit 1.0
- Statistical Process Control Toolkit 1.0
- Proportional-Integral-Derivative Toolkit 1.0

# Upgrading Previous Versions of LabVIEW

The following sections describe upgrade issues specific to different versions of LabVIEW.

## Upgrading from LabVIEW 4.*x*

This section describes changes in the Boolean data format and the VI Control VIs since LabVIEW 4.*x*.

### Converting Boolean Data to and from LabVIEW 4.*x*

The format in which LabVIEW 5.0 stores Boolean data has changed from the format of Boolean data in LabVIEW 4.*x*. LabVIEW 4.*x* stores Boolean data in two bytes unless the data is in an array, in which case LabVIEW 4.*x* stores each Boolean element in a single bit. LabVIEW 5.0 stores a Boolean value in a single byte, regardless of whether it is in an array. This change enables more block diagram functions to support arrays of Booleans and makes the behavior of these arrays more consistent with the behavior of arrays of numbers. The new Boolean data format affects data manipulation in code interface nodes (CINs), but LabVIEW5.0 provides compatibility for existing CINs.

When you open a datalog file created in a previous version of LabVIEW, LabVIEW 5.0 prompts you to convert the file to the LabVIEW 5.0 format. If you choose to convert it, LabVIEW replaces the datalog file with data converted to the new format. If you choose not to convert the file, LabVIEW 5.0 returns an error and does not open the file.

If you write binary data including one or more Booleans to a file in LabVIEW 4.*x*, its format is different than if you write the same data in LabVIEW 5.0. LabVIEW 5.0 provides a mechanism for reading binary data written in LabVIEW 4.*x* and writing binary data that LabVIEW 4.*x* can read. Five functions, Write File, Read File, Type Cast, Flatten To String, and Unflatten From String, have a **Convert 4.x Data** pop-up menu option. If you select this option, the function treats binary data as if it were written for LabVIEW 4.*x*. To produce data formatted for LabVIEW 4.*x*, use the Write File, Flatten to String, or Type Cast function. To read data formatted for LabVIEW 4.*x*, use the Read File, Unflatten From String, or Type Cast function. When you select the **Convert 4.x Data** option, LabVIEW 5.0 draws a red `4.x` on the function to indicate that it is converting data to or from LabVIEW 4.*x* format. To stop this conversion of data, deselect the **Convert 4.x Data** pop-up menu option.

If you have several data files with Boolean values, you can create a VI that opens these files, then writes the data to a new data file that LabVIEW 5.0 recognizes.

In LabVIEW 5.0, when you load a VI last saved in a previous version of LabVIEW, LabVIEW 5.0 automatically sets the **Convert 4.x Data** attribute on the Write File, Read File, Type Cast, Flatten To String, and Unflatten From String functions. Any VIs created in a previous version of LabVIEW continue to function as before. When you decide that your VIs need to use the new LabVIEW 5.0 Boolean data format, deselect the **Convert 4.x Data** attribute on each of the functions listed above. Typically, if your VIs do not need to manipulate files containing Boolean data written in a previous version of LabVIEW or send or receive data containing Booleans to or from VIs running in a previous version of LabVIEW, you should use the new LabVIEW 5.0 Boolean data format. Support for the previous Boolean data format might be discontinued in future versions of LabVIEW.

### VI Control VIs

The VI Control VIs (`vi.lib\utility\victl.llb`) have been removed from the default menu palette set and now exist as compatibility VIs. Their functionality has been subsumed by the new VI Server functions (Open VI Reference, Call By Reference, Property Node, and Invoke Node). You can find these functions in the new **Functions»Application Control** palette.

Some of the error codes passed from the VI Control VIs have changed in LabVIEW 5.0. In previous versions of LabVIEW, the VI Control VIs passed the error codes 7 and 1000. The VI Control VIs in LabVIEW 5.0 pass the correct error codes, 1004 and 1003, respectively. If a VI built in a previous version of LabVIEW checks for these specific error codes, you need to make modifications so the VI works in LabVIEW 5.0.

## Upgrading from LabVIEW 3.*x*

Only a few compatibility issues exist between LabVIEW 3.*x* and LabVIEW 5.0, so upgrading should not be difficult.

**(Sun)** National Instruments no longer maintains LabVIEW on Solaris 2.2 or SunOS 4.1.2. LabVIEW might work with these older operating systems, but National Instruments has not performed extensive testing on them. National Instruments recommends that you use Solaris 2.4 or later or SunOS 4.1.3 (Solaris 1.1).

**(Concurrent PowerMAX)** National Instruments no longer maintains LabVIEW on Concurrent PowerMAX 3.*x*. LabVIEW 5.0 has been tested extensively only on Concurrent PowerMAX 4.1.

### Upgrading from LabVIEW 2.*x*

LabVIEW 5.0 does not support VIs created prior to version 3.0. If you are upgrading VIs from an earlier version, you first must upgrade to version 3.*x*, then convert your VIs to LabVIEW 5.0. For this upgrade, you need the LabVIEW Conversion Package from 2.*x* to 3.*x* for your operating system.

### Upgrading from Versions prior to 2.*x*

**(Macintosh Only)** If you are upgrading from LabVIEW 1.*x*, you first must upgrade your VIs to LabVIEW 2.0, then upgrade the VIs to LabVIEW 3.0, and finally upgrade them to LabVIEW 5.0. To convert VIs from LabVIEW 1.*x* to LabVIEW 2.2.1, you also need the LabVIEW 1.*x* Conversion Package. This package assists in the conversion process and documents the differences between the versions.

# LabVIEW Features

This section describes features and improvements made to each significant LabVIEW release. The features and improvements are divided into major categories for each version.

## New Features in LabVIEW 5.0

LabVIEW 5.0 includes new multithreading and ActiveX Automation functionality, instrumentation and data acquisition features, translation tools, a VI Server interface, and multi-step undo and redo, as well as other improvements. The following sections (*Multithreading*, *ActiveX Automation*, *Instrumentation*, *Data Acquisition*, *Translation Tools*, and *General Interface*) describe the features added and the significant changes made between versions 4.1 and 5.0.

To help you learn more about LabVIEW, version 5.0 offers extensive online documentation, which you can access by choosing **Help**»**Online Reference...**.

☞ **Note** ***LabVIEW is Year-2000 compliant. Because LabVIEW has never stored two-digit years, the change to 2000 does not affect any internal storage of dates.***

### Multithreading

LabVIEW 5.0 incorporates multithreading technology, enabling different parts of your application to run independently. This capability can resolve performance conflicts between user interface and data acquisition, for example. Using multithreading, you can take full advantage of computers with multiple processors.

In previous versions of LabVIEW, you can execute several VIs simultaneously and still respond to user input from the mouse or keyboard. To provide this capability, the execution system uses *cooperative multitasking*—each of the different activities processes one at a time and in turn. Although cooperative multitasking works well, processor time does not distribute evenly to each activity, and one activity can prevent the execution of others.

With multithreading in LabVIEW 5.0, the operating system can preempt a thread of execution to give processor time to another thread. Therefore, processor time is shared more evenly among the threads.

To take advantage of multithreading you must use LabVIEW 5.0 on an operating system that supports it: Windows 95/NT, Solaris 2, or Concurrent PowerMAX. On operating systems that do not support multithreading, LabVIEW continues to operate with cooperative multitasking.

You need no special programming to use multithreading. For advanced multithreaded programming, you can use utilities such as semaphores, synchronization, and message queues.

**Multithreaded Performance Profiler**—The performance profiler monitors multithreaded and multiprocessor applications. You access the performance profiler by choosing **Project»Show Profile Window**. If you are executing on a multiprocessor computer, the displayed timings add the time spent on each processor. If many parallel activities exist, the times approach the number of processors multiplied by the execution time.

**Multithreading and Call Library Node**—On a multithreaded operating system, you can make multiple calls to a dynamic link library (DLL) or a shared library simultaneously. The library must be reentrant or use other protection techniques before it is marked as reentrant.

**Multithreading and Code Interface Node**—By default, code interface nodes (CINs) written prior to LabVIEW 5.0 run in a single thread, the user interface thread. When you change a CIN to be reentrant (execute in multiple threads), more than one execution thread can call the CIN at the same time.

**Color of Code Interface and Call Library Function Nodes**—In LabVIEW 5.0, the color of a code interface node (CIN) or Call Library Function node on a block diagram changes depending on whether LabVIEW considers it reentrant. If LabVIEW considers a CIN or Call Library Function node reentrant, LabVIEW assigns it the current primitive color (the default is pale yellow). If a CIN or Call Library Function node is not considered reentrant, its color is orange. This color designation exists on all platforms, even if the platform itself is not threaded.

**Synchronous Display on Indicators**—With multithreading, the user interface is decoupled from the diagrams that create the data to display. As a result, the user interface might discard some data if it already has been replaced by new data to draw. If you need to see all the data on your front panel, you can choose **Synchronous Display** from the pop-up menu of front panel controls. With this option selected, the diagram waits for any previous data to be drawn before sending new data to the front panel.

**Synchronization VIs**—You can synchronize tasks executing in parallel by using the Synchronization VIs. You also can use these VIs to pass data between parallel tasks. You access the **Synchronization** palette by choosing **Functions»Advanced»Synchronization**. This palette consists of five subpalettes containing the Notification VIs, Queue VIs, Rendezvous VIs, Semaphore VIs, and Occurrence Functions.

## ActiveX Automation

**(Windows 95/NT)** LabVIEW 5.0 incorporates enhanced ActiveX (OLE) Automation functionality, including the ActiveX server, ActiveX Container, and an improved ActiveX client interface.

**ActiveX Automation Server**—Using LabVIEW as an ActiveX Automation server, other ActiveX-enabled applications (such as LabWindows/CVI, Excel, Visual Basic, and so on) can control LabVIEW. These applications can request properties and methods from LabVIEW and individual VIs to call VIs and pass them data, for example.

**ActiveX Front Panel Objects**—LabVIEW 5.0 front panels include a new control subpalette, the **ActiveX** subpalette, which includes two ActiveX objects: ActiveX Container and ActiveX Variant. With these new objects, you can take advantage of the ActiveX Container capability and enhance the interactions between LabVIEW and other applications.

- **ActiveX Container**—Use the ActiveX Container to embed ActiveX controls on LabVIEW front panels. You can display programmatic changes in the container on the front panel. For example, you can embed a web browser or a calendar.
- **ActiveX Variant Control and Indicator**—Use the ActiveX Variant to pass ActiveX Variant data into LabVIEW, which enhances ActiveX client functionality. Use this front panel object when ActiveX Variant data is converted to data that LabVIEW can display.

**ActiveX Client Automation**—LabVIEW 5.0 provides updated ActiveX Automation Client functions, which you can use to control other ActiveX-enabled applications and ActiveX controls. To implement LabVIEW as an ActiveX client, use the following new functions: Automation Open, Automation Close, Invoke Node, and Property Node.

**ole_lv5container.dll**—The ActiveX Container uses a DLL named `ole_lv5container.dll`, which is located in the `resource` directory. If you build an application that includes ActiveX controls and move it to another machine, you must install this file in the same directory as the built application or in the `System` directory.

**Data Format**—The compatibility VIs for the LabVIEW 4.*x* Automation functions require that you pass flattened data in the LabVIEW 4.*x* format. LabVIEW 5.0 loads your LabVIEW 4.*x* VIs and automatically selects the **Convert 4.x Data** option for the Flatten To String and Unflatten From String functions. For more information, see *Converting Boolean Data to and from LabVIEW 4.x* in the *Upgrading from LabVIEW 4.x* section earlier in this document.

## Instrumentation

**Solution Wizard for DAQ and Instrument I/O**—The Instrument Wizard has been added to the Solution Wizard for DAQ and Instrument I/O. The Instrument Wizard guides you through finding and installing instrument drivers for your computer-based instruments as well as GPIB, VXI, and serial instruments.

**Signal Generator by Duration VI**—The Signal Generator by Duration VI has been added to the **Analysis»Signal Generation** palette. This VI generates a signal with a shape given by the waveform type: sine, cosine, triangle, square, sawtooth, increasing ramp, or decreasing ramp.

**CVI Function Panel Converter Changes**—The improved CVI Function Panel Converter creates hierarchical text menus so you can find functions quickly. Two new options have been added to the CVI Function Panel Converter. These options are ON by default.

- **Map ViSession type to VISA Session RefNum**—This option specifies that instrument session numbers of type ViSession in the CVI Function Panel are converted to LabVIEW VISA RefNums in the resulting VI. Functions that contain the string `_init` in their name automatically register with the VISA refnum; functions that contain `_close` in their name automatically close the VISA refnum.
- **Create instr.lib menu mirroring CVI Class Hierarchy**—This option specifies that when converting a Function Panel file, a palette menu for the instrument is created in the **Instrument Drivers** menu. This menu is organized hierarchically according to the Function Panel Tree in the `.fp` file.

**Easy VISA VIs**—A new set of Easy VISA functions reduces the steps needed to write a VISA application.

**IVI Support**—LabVIEW 5.0 supports Intelligent Virtual Instrumentation (IVI) instrument drivers, which you can use to control instruments. IVI instrument drivers are DLL-based drivers developed in LabWindows/CVI that give production test users additional benefits, including instrument state catching for improved performance, simulation, multithread safety, and instrument attribute access.

## Data Acquisition

**(Windows 95/NT and Power Macintosh) NI-DAQ 6.0**—LabVIEW 5.0 incorporates NI-DAQ 6.0, which supports new DAQ hardware devices on Windows 95/NT and Power Macintosh and adds several new library VIs. NI-DAQ 6.0 only works on Power Macintoshes that have PCI slots. For more information, see the NI-DAQ `README` file.

**Signal Generator by Duration VI**—The Signal Generator by Duration VI has been added to the **Analysis»Signal Generation** palette. This VI generates a signal with a shape given by the waveform type: sine, cosine, triangle, square, sawtooth, increasing ramp, or decreasing ramp.

**(Windows 95/NT) NI-DAQ Remote Device Access Support**—LabVIEW 5.0 supports NI-DAQ Remote Device Access, which enables you to acquire data from DAQ boards in remote computers over a network. You do not need to change your DAQ applications to support this feature.

**(Windows 95/NT and Macintosh) New DAQ VIs**—The following DAQ VIs have been added to the **Data Acquisition»Calibration and Configuration** palette: DSA Calibrate, Get DAQ Channel Names, Get Channel Information, and Get Scale Information.

**(Windows and Macintosh) New DAQ Examples**—This release includes many new data acquisition examples, which are divided into categories to illustrate techniques, and more complete examples called *solutions*. The technique examples are based on the DAQ examples in previous releases, but LabVIEW 5.0 includes many new and improved examples for analog input, analog output, digital input/output, counter/timer, and more.

**(Windows and Macintosh) Solution Wizard for DAQ and Instrument I/O**—You can use the Solution Wizard to generate data acquisition VIs based on user-defined parameters. You specify solutions either from a list of common solutions or by defining the desired DAQ components. The number of these solutions has increased since LabVIEW 4.1. To find the example you need, run the Solution Wizard, which you access from the LabVIEW dialog box or from the **Project** menu. You can find these examples by browsing the folders and VI libraries in `examples\daq`. To locate your solution, run the DAQ Solution Wizard and choose **Solutions Gallery**. You can find these examples in VI libraries in the `examples\daq\solution` directory.

**(Windows 95/NT and Power Macintosh) DAQ Channel Wizard**—For LabVIEW 5.0, the DAQ Channel Wizard has been updated to run on Power Macintoshes with PCI slots running NI-DAQ 6.0, in addition to Windows 95/NT. The DAQ Channel Wizard simplifies the configuration of analog input channels, analog output channels, and digital lines on your DAQ device, which include DAQ plug-in boards, stand-alone DAQ products, and SCXI modules. The DAQ Channel Wizard helps you define the physical quantities you are measuring on each DAQ Hardware channel by querying for information about the physical quantity being measured, the sensor being used, and the associated DAQ hardware. As you configure channels in the DAQ Channel Wizard, you assign each channel configuration a unique name that you use when addressing your channels in LabVIEW. The channel configurations you define are saved in a file that instructs the NI-DAQ driver how to scale and process each DAQ channel by its name.

## Translation Tools

New tools included in LabVIEW 5.0 simplify the translation of text in the user interface of VIs into different languages.

**Switching between Languages**—In conjunction with the VI server, you programmatically can switch between two languages, such as English and Japanese. For more information about the VI Server, see the *VI Server Capabilities* item in the following section, *General Interface*.

**Importing and Exporting VI Strings**—The VI string export and import tool writes all the localizable strings contained in the front panel of a VI to the VI string file, a tagged text file. You can localize the following strings: VI name and description, object caption labels, free labels, default data (string, table, path, and array default data), and private data (list-box item names, graph plot names, graph cursor names, and table, row, and column headers).

**Editing VI Window Titles**—In previous versions, the VI window title is the same as the VI file name. Now you can customize the VI window title so it is different from and more descriptive than the VI file name. This feature is important for localized VIs because you easily can translate the VI window title to the local language; file system naming constraints do not govern the window title and the VI still is recognized by the VIs that call it.

**Period and Comma Decimal Separators**—In LabVIEW 5.0, you can control the formatting of the decimal point precisely. This feature is useful, for example, in countries in which a comma is used for the decimal point when formatting strings for instruments that require a period decimal point.

In previous versions of LabVIEW, you can choose to use the decimal separator or period (.) of the system from the **Front Panel** section in the

**Edit»Preferences…** dialog box. When previous versions of LabVIEW use the system decimal separator and the system uses a comma (,), the comma is not recognized as a decimal separator by the instruments when numbers are converted to strings for instrument communication.

In LabVIEW 5.0, you can force a period as a decimal separator when converting numbers to strings, and vice versa, using the following functions: To Engineering, To Fractional, To Exponential, From Exponential/Fract/Eng, Format Into String, and Scan From String.

**Front Panel Caption Labels**—Front panel objects can have caption labels. The caption does not affect the name of the object and you can use it as a more descriptive name of the object. You also can show, hide, and change the caption programmatically with attribute nodes. Changes to the caption do not cause the VI or its callers to be recompiled, so you can localize captions without affecting the code of the VI or that of its callers.

**Format Date/Time String Function**—You can display the date and time in a format you specify.

## General Interface

**Undo and Redo**—LabVIEW 5.0 incorporates multi-step undo and redo, which simplify the correction of mistakes made while editing. You can undo an action immediately after you have performed it, and once you undo an action you can redo it. Set the number of actions that you can undo or redo in **Preferences»Block Diagram»Maximum undo steps per VI**.

**VI Server Capabilities**—LabVIEW 5.0 now exports many of its capabilities to other applications through a new set of features collectively referred to as the *VI Server*. You can use the VI Server interface to control VIs and the LabVIEW application programmatically. You can load VIs dynamically, get and set attributes of those VIs, print them, save them, and so on. This feature easily automates routine tasks that previously were done manually.

You also can execute the server across a heterogeneous network. With the VI Server, you can set properties and invoke operations on applications written in LabVIEW and on VIs located on the local computer or anywhere on your TCP/IP network. With new diagram functions, you can access these capabilities within LabVIEW. These capabilities subsume the functionality of the VI Control VIs, which have been removed from `vi.lib\utility\victl.llb`.

**Printing and Exporting VI Documentation to an RTF or HTML File**—LabVIEW 5.0 simplifies web and help publishing for your VI documentation. In previous versions of LabVIEW, you are limited to printing your control and VI descriptions to a printer or a text file.

With LabVIEW 5.0, you can print or export these descriptions to formatted file formats such as Rich Text Format (RTF) and Hypertext Markup Language (HTML). You can import RTF files into most document-publishing software or use them as the source for Help files. You use the HTML format for online documents, particularly those you intend to publish on the World Wide Web. The Print to RTF/HTML feature supports graphics in uncompressed graphics interchange format (GIF).

**Custom Menus**—You can create custom menus for applications built with LabVIEW 5.0. VIs can override existing LabVIEW menus and install and respond to their own menu items. For every VI you build, you can customize menus in two steps: creating the menus and responding to menu selections. The custom menus are installed only when the VI is running.

**Case Structure Enhancements**—The enhanced Case structure in LabVIEW 5.0 simplifies programming for common situations, such as state machines and handling menu selections. You now can set any frame as the default case and map multiple cases to a single frame. Additionally, you can wire strings directly to the case selector with no need for parsing.

Case structures perform a certain action based on a particular value, which is called the *selector*. With Case structures in LabVIEW 5.0, you can specify ranges of selector values, include negative integers as selectors, specify a default case (or action), sort cases based on the first selector value, and convert the input selector to the selector values listed in the Case structure. The enhancements made to the LabVIEW 5.0 Case structure do not break your existing VIs. The new options for non-Boolean cases are **Rearrange Cases...** and **Make This Case Default**.

**Password-Protected VIs**—You now can protect VIs with a password, which prevents viewing and editing inadvertently. By keeping the block diagram, this feature allows platform and version changes.

**Configuration File VIs**—The Configuration File VIs provide tools for reading from and writing to a platform-independent configuration file similar in format to a Windows initialization (`.ini`) file.

**Dragging and Dropping VI Icons**—LabVIEW 5.0 simplifies the creation of VI icons. By selecting an image file and dropping it onto the VI icon in the upper-right corner of a front panel, a 32-by-32 version of the image replaces the existing icon.

You can drag a VI icon from the icon pane in the upper-right corner to a block diagram to instantly create a subVI call. By pressing <Shift> while dragging the VI icon, you automatically wire the non-default values of the controls as constants for the subVI.

If the subVI already appears in a block diagram, pressing <Shift> while dragging onto the existing call updates the attached constants. A control at its default value discards the constant attached to the subVI, and an input wired to anything other than a constant is unaffected.

When you press <Shift> while double-clicking a subVI icon to open the subVI front panel, LabVIEW loads the values of the constants wired to the subVI into the front panel controls. All unwired controls retain the default values.

You also can use the drag-and-drop technique for global variables and custom controls. Additionally, you can drag a VI icon into a VI refnum on a front panel control to load VIs into memory dynamically, which is part of the VI Server functionality.

**Call Library Parameter Type Enhancements**—LabVIEW 5.0 includes one new return type and one new parameter type. The new string return option enables functions to return a C or Pascal string. You can use the **Any LabVIEW type** input parameter to pass arbitrary G data types to DLLs. Neither enhancement is available for Windows 3.1.

**Max&Min and In Range Functions Enhancements**—LabVIEW 5.0 includes two enhanced Comparison functions: Max & Min and In Range. In previous versions of LabVIEW, the Max & Min and In Range functions only compare arrays and clusters as a group of elements (aggregates). In LabVIEW 5.0, the Max & Min and In Range functions also can compare individual elements of arrays and clusters.

**Boolean Data Storage Format**—LabVIEW 5.0 stores Boolean data in a single byte, regardless of whether it is an array. For more information, see *Converting Boolean Data to and from LabVIEW 4.x* in the *Upgrading from LabVIEW 4.x* section earlier in this document.

**Macintosh Open Transport Support**—LabVIEW 5.0 supports Open Transport on Power Macintosh machines. Open Transport is a PowerPC-native networking driver.

**New Preferences Options**—LabVIEW 5.0 adds the following two options in the **Miscellaneous** view of the **Edit»Preferences…** dialog box.

- **Automatically close VISA sessions**—Use this option to specify that VISA sessions, like file refnums, close automatically when the top-level VI goes idle. The default is ON, which closes VISA sessions automatically.
- **Treat read-only VI as locked**—Using this option, you can choose whether to treat read-only VIs as locked. You cannot edit locked VIs, but you can re-compile and execute them. By default the option is not selected so that read-only VIs appear normally. However, you cannot

save the VI to the same location (the read-only file) unless you change the file permissions outside LabVIEW. This behavior is consistent with the behavior in previous versions of LabVIEW. When using the VI Server, the read-only status of files is ignored except when saving. This option is designed primarily to support the source-code control of the Professional G Developers Toolkit.

**Reorder Commands**—The reorder commands (**Move Forward**, **Move Backward**, **Move to Front**, **Move to Back**) have moved from the **Edit** menu to the Reorder ring on the far-right corner of the toolbar.

**(Windows 3.1) lvdevice.dll Change**—`lvdevice.dll` has changed to `resource\lvdev5.dll`.

**Offscreen Updates Default Value**—The default value for offscreen updates is now ON instead of OFF.

**Icon and Text Palettes**—You can display the **Controls** or **Functions** palette using icons, text, or a combination of both. For more information about icon and text palettes, see the *Manual Clarifications and Additions* section of the *LabVIEW Release Notes*.

**Icon Editor**—In the Icon Editor, you can select an area of an icon for moving, copying, or deleting. You also can use the **Edit** menu to cut, copy, and paste images from and to the icon. When you paste an image and a portion of the icon is selected, the image is resized to fit into the selection. The **Undo** button has been removed from the Icon Editor, but you can undo an action by choosing **Edit»Undo** or <Ctrl-Z>.

**Seconds to Date/Time Function Change for Windows 95/NT**—In Windows 95/NT, the value of the DST element of the **date time rec** cluster returned by the Seconds to Date/Time function is not set correctly. Instead of returning whether the **date time rec** cluster has been modified for daylight saving time, it returns whether your current system settings are set to account for daylight saving time.

**Save Command**—The **Save** command, which you use by choosing **File»Save**, now is enabled always, regardless of whether you have made changes.

**Disabling Items in Ring Controls**—You can disable items in ring controls while you are editing or running a VI.

**Create Constant, Control, and Indicator Improvements**—When you create a constant, control, or indicator for a VI by popping up on a subVI terminal, LabVIEW 5.0 copies the appropriate control or indicator from the subVI, which preserves its appearance, description, and so on. If the terminal corresponds to a type definition, you create a type definition

constant, control, or indicator. If the type definition is changed later, the control, indicator, or constant is updated.

**Scan String for Tokens Function**—Use this function to scan an input string starting at a specified offset and returning the next token found. A *token* is a substring of the input string, which is surrounded by specified delimiters or matches an element in the operators array. Typically, tokens represent individual keywords, numeric values, or operators found when parsing a configuration file or other text-based data format.

**New Modes for the File Dialog Function**—The File Dialog Function displays a file dialog box so you can select an existing file or directory or select the location for a new file or directory. In LabVIEW 5.0, the following three new **select modes** have been added so you can open any file stored in VI libraries (LLBs): select an existing file in an LLB, select a new file in an LLB, and select an existing or new file in an LLB.

**TCP/IP Functions**—The following TCP/IP VIs are now functions in LabVIEW 5.0: TCP Open Connection, TCP Create Listener, TCP Wait on Listener, TCP Write, TCP Read, and TCP Close Connection.

TCP Read now has four operating modes. A TCP Read with zero timeout in standard (default) mode now reports a timeout error if no bytes are present.

**Advanced Palette Changes**—New VIs for VI Server functionality and programmatic control of window menu bars have been added to the **Advanced** palette. The VI Control VIs have been replaced by the VI Server functions, which contain the functionality of the VI Control VIs. The following palettes and functions have been moved from the **Advanced** palette to the **Application Control** palette: **Printing** palette, **Help** palette, Quit LabVIEW function, Exit function, and Call Chain function.

**File Manager Tool**—LabVIEW 5.0 includes the File Manager Tool, which you can access by choosing **Project»File Manager**. Use this tool to organize files in VI libraries (LLBs) and directories. See the *Manual Clarifications and Additions* section of the *LabVIEW Release Notes* for more information.

# New Features in LabVIEW 4.1

This section describes the features added between 4.0 and 4.1.

## Compatibility and Added Support

**New OLE Automation VIs for Communication with HiQ**—New OLE Automation VIs have been included for communication with the Windows version of HiQ. HiQ is a National Instruments application for interactive analysis, 3-dimensional data visualization, and automated report generation with LabVIEW. Using the new OLE automation interface link between LabVIEW 4.1 and HiQ 3.1, you can generate technical reports directly from LabVIEW. Design your report template interactively using HiQ, then automatically generate your publication-quality technical reports directly from LabVIEW using the new HiQ VIs. Your LabVIEW data, analysis, and graphs are organized into HiQ Notebooks, which you can print directly from LabVIEW. New VIs include the following: Set Data (Text, Script, Integer, Real, Complex, Vector, Matrix), Get Data (Text, Script, Integer, Real, Complex, Vector, Matrix), Launch HiQ, Save Notebook, Open Notebook, Print Notebook, Run Script, Close Notebook, and Exit HiQ.

**Support for Template VIs and Controls**—You can save commonly used VIs and controls as templates. To create a template VI, save a VI with a `*.vit` extension (or `*.ctt` extension for typedefs). When you close the VI, LabVIEW prompts you to save the file. To modify a template, open it, make your changes, and then save over the `.vit` (or `.ctt`) file that you originally created.

**(Macintosh)** 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.

**Better Support for Toolkit Files**—Files installed in `vi.lib\addons` automatically appear at the top level of the **Controls** 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, National Instruments recommends placing them in `user.lib` or adding them to a custom palette set.

**Changes to MPW CIN Tools for Macintosh**—The MPW script cinmake (`:cintools:MPW:cinmake`) has been modified so its default behavior uses the MrC compiler to compile the cin for the PowerPC platform. A new command line option (`-C`) enables users to specify that the CIN should be compiled using the older C compiler for the 68K platform.

## Data Acquisition

**NI-DAQ 5.0 for Windows**—NI-DAQ version 5.0 adds complete support for Windows NT 4.0, new DAQ hardware devices, and four new library VIs. Also, several DAQ hardware devices have been made obsolete in NI-DAQ version 5.0. You can find details in the NI-DAQ 5.0 `readme.txt` file that installs in your NI-DAQ for Windows directory when you install NI-DAQ version 5.0.

**DAQ Solution Wizard for Windows**—The DAQ Solution Wizard is a tool that lets you generate data acquisition VIs based on user-defined parameters. Solutions can be specified either from a list of common solutions or by defining the desired DAQ components. You can run, modify, or save generated solutions to your hard drive. The DAQ Solution Wizard is accessible from the Startup dialog box and from the **File** menu.

**New DAQ Examples for Windows and Macintosh**—This release includes a number of new data acquisition examples, which are broken down to 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, DAQ Occurrences for asynchronous acquisition, and other techniques. The analog output examples include digital triggering, external clocking, generating data from a file, and other techniques. The simultaneous analog input/output examples include buffered acquisition and generation examples for E-series boards (such as the PCI-MIO-16E-1), as well as legacy boards (such as the AT-MIO-16 or NB-MIO-16).

The solution examples show how you can integrate data acquisition into your application. They include examples for 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; and applications for transducers such as thermocouples and strain gauges.

**(Windows)** To find the example you want, run the DAQ Solution Wizard (accessible from the LabVIEW dialog box or from the **File** menu) and choose **Custom DAQ Applications**. You can find these examples by browsing the folders and VI libraries in `examples\daq`. To locate your solution, run the DAQ Solution Wizard and choose the **Solutions Gallery**. These examples are in VI libraries in the `examples\daq\solution` directory.

**DAQ Channel Wizard for Windows**—The DAQ Channel Wizard is a Windows application you can use to configure the analog input channels on your DAQ device, which include DAQ plug-in boards, stand-alone DAQ products, and SCXI modules. The DAQ Channel Wizard helps you define the physical quantities you are measuring on each DAQ Hardware channel by querying for information about the physical quantity being measured, the sensor being used, and the associated DAQ hardware. As you configure channels in the DAQ Channel Wizard, you give each channel configuration a unique name that is used when addressing your channels in LabVIEW. The channel configurations you define are saved in a file that instructs the NI-DAQ Driver how to scale and process each DAQ channel by its name. You can simplify the programming required to measure your signal by using the DAQ Channel Wizard to configure your channels.

**Using DAQ Channel Wizard with LabVIEW**—If you use the DAQ Channel Wizard to configure your analog input channels, you can address your channels by name in LabVIEW. The Analog Input VIs have a **channels** parameter you can use to specify the channel names from which the VIs read. The **channels** parameter can be an array of strings or, as with the Easy VIs, a scalar string control. If you have a channels array, you can use one channel entry per array element, specify the entire list in a single element, or use any combination of these two. If you enter multiple channel names in the **channels** parameter, all of the channels in the list must be measured by the same DAQ Device. For example, if you have configured channels with names of temperature and pressure, both of which are measured by the same DAQ Device, you can specify a list of channels in a single element by separating them with commas (for example, `pressure`, `temperature`). When specifying channel names, spelling and spaces are important, but case is not.

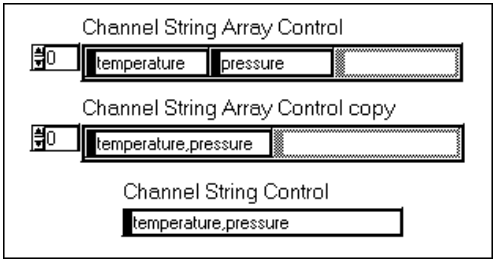


**Figure 1.** Channel String Controls

You do not need to wire the **device**, **input limits**, or **input config** input parameters when using channel names. LabVIEW configures your hardware to make the measurement in terms of your channel configuration. Unless you need to overwrite your channel name configuration, do not wire these inputs; allow LabVIEW to set it up for you. (Notice that when using channel names, the **input limits**, **hysteresis**, and **level** parameters are

relative to the physical quantity units of the channel, not the hardware units.) In addition, LabVIEW calibrates your hardware, accounts for cold-junction compensation, and scales your readings for you. The data returned is scaled to the physical quantity units defined in the channel configuration, as shown in Figure 2.

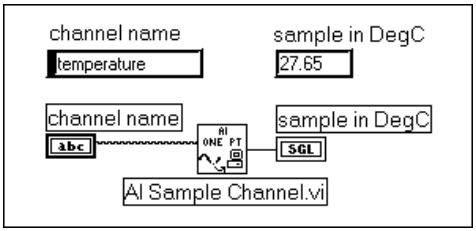


**Figure 2.** Channel Name Usage

## General Interface

**Startup Screen**—When you launch LabVIEW in Windows, you are greeted with a navigation dialog box that contains introductory material, common commands, and Quick Tips. This new dialog box helps new users learn LabVIEW more quickly. If you prefer to bypass the navigation dialog box, disable it by checking the checkbox at the bottom of the dialog box. To reenable it, use the Preferences dialog box.

When all VIs are closed, a similar dialog box appears. If you click the **Small Dialog** button within this dialog box, you switch to a simple version of the dialog box that contains only the **New**, **Open**, and **Exit** buttons (similar to previous versions).

**Online Tutorial**—The *LabVIEW Online Tutorial* introduces various aspects of the LabVIEW environment interactively. We recommend that new LabVIEW users work through this tutorial before they build their first VI. The following concepts are covered in this tutorial: virtual instruments (VIs), parts of a VI, dataflow execution, subVIs, hierarchy, pop-up menus, data type colors, and controls and indicators. The tutorial is included only with the CD distribution of LabVIEW for Windows.

**Examples Browser for Windows**—The Search Examples Help file contains hyperlinks to all LabVIEW examples. This Help file divides the LabVIEW examples into categories and provides a brief description about the functionality of each example. You can access the Search Examples Help file from the LabVIEW dialog box or the **Help** menu.

**Basic Palette View**—The new basic control and functions palette view provides a simplified set of palettes, including only the most commonly used functions. Because it emphasizes only the functions you might need

for beginner applications, this palette might be useful for those who are learning LabVIEW.

**(Windows)** This palette is used as part of the *LabVIEW QuickStart Guide*. To switch to the basic palette set, choose **Edit»Select Palette Set»basic**.

## Performance

**Load Memory Usage Option (PowerMac)**—This preference 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. This preference is on by default and presented as `Less memory fragmentation/Slower loading`.

This feature was implemented because of the Modern Memory Manager, which allows memory to become fragmented more easily even though it is faster for most operations.

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 might otherwise. It can dramatically speed up loading for some applications, especially those with many reentrant VIs.

## Printing

**Print Quality Improvements for Windows**—LabVIEW supports three modes for printing: Standard, Bitmap, and PostScript. PostScript printing gives the best results but is only an option if you have a PostScript printer. Standard PC printing now does a much better job adapting to the differences between screen and printer fonts and should eliminate most clipping and text overruns. You can get an even better match between what you see on the screen and on the printer by using True Type fonts instead of Bitmap fonts. The predefined fonts (Application, System, and Dialog) are usually Bitmap fonts, but you can redefine them using the Fonts view of the Preferences dialog box.

**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 view of the Preference dialog box. To set margins for a single VI, use the Execution view of the VI Setup dialog box, which you can access from the connector pane of the front panel. Settings for a single VI override the global setting. You can specify margins arbitrarily small, but they might 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** prints 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** prints 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** retrieves a panel image programmatically in a format that you can pass to one of the VIs in the 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.

# New Features in LabVIEW 4.0

This section describes the features added between 3.1 and 4.0.

## 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.

**Probe Tool**—You can create a probe on a wire by using the Probe tool.

**Breakpoint Tool**—You can set a breakpoint on VIs, nodes, wires, and diagrams to pause execution so that you can probe the wires.

**Copying Front Panel Terminals to New VIs**—You can 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.

**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.

**Wiring Stubs and Tip Strips Help With Wiring**—When you move over a node, a wire stub that shows the data type of that node and a tip strip with

the name of that input or output appears. This feature might help you wire the correct information to the terminal.

**Default Labels for Functions and VIs**—When you choose **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 choosing **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 and modify online help.

**Sequence and Case Structures Indicate Range**—LabVIEW displays the range of frames at the top of the Sequence and Case structures.

**Array and Cluster Constants**—You can use array and cluster constants to create constant data on the block diagram.

**Concatenate String Function Supports Arrays of Strings**—You can 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.

**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.

**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 execute calls more quickly.

- The flattened data types of the **requested outputs** input previously were 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 run-time 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 VIs containing 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 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 front panel. The VI must be in memory, but its front panel does not have to be open. 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 includes only the visible part of the front panel and does not include the window title bar, scrollbars, toolbar, or 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. So you can edit the VI or look at its block diagram, set it to FALSE if you are opening a VI that you do not plan to run immediately.

**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.

**Spreadsheet Functions Support More Formats**—With the updated spreadsheet functions, you can specify a delimiter (other than a tab) in your spreadsheet, which eliminates the need to parse your spreadsheet if you have specified a comma or other character as a spreadsheet delimiter.

**Object Linking and Embedding (OLE) VIs on Windows**—LabVIEW includes OLE VIs, which serve as automated clients.

**Call Library Function**—Use this function to communicate with Code Fragment Libraries on the Macintosh.

**Compilers Supported for CINs and DLLs**—National Instruments has tested Symantec C for Windows 95/NT and has successfully built CINs and DLLs that you can call from LabVIEW. After performing additional testing for other Windows 95/NT compilers, specifically Borland C and Watcom C, National Instruments will make compatibility information available through standard support routes, including its FTP site `ftp.natinst.com`.

Following is a summary of LabVIEW support for compilers for creating CINs and DLLs on different operating systems.

**(Windows 3.1)** You can only use CINs in LabVIEW created with the Watcom C compiler. You should also be able to call 16-bit DLLs created with almost any compiler, including Watcom C, Visual C version 1.5, and Microsoft C.

**(Window 95/NT)** You can now use CINs in LabVIEW created with the following compilers:

- Visual C (version 2.0 or later).
- Symantec C (version 7.0 or later)—Notice that external subroutines are not supported for Symantec C.
- Watcom C for Windows 3.1—LabVIEW has backward compatibility with CINs created using Watcom C for Windows 3.1. However, those CINs cannot call DLLs and system functions because the technique Watcom uses on Windows 3.1 does not work on Windows 95/NT.

You should also be able to call Win32 DLLs created with almost any compiler, including Visual C (version 2.0 or later), Symantec C (version 7.0 or later), and Borland C (version 4.0 or later). Notice that DLLs on Windows 95/NT can call LabVIEW functions described in the *LabVIEW Code Interface Reference Manual*. Currently, LabVIEW only has libraries that you can use with Visual C and Symantec C to call LabVIEW functions from DLLs.

**(Macintosh)** LabVIEW now supports the Symantec C++ (8.0) and Metrowerks C/C++ 68K compilers for the Macintosh and includes project templates that make it easier to create CIN projects with the correct settings.

## Compatibility and Added Support

**Conversion Improvements of Special Macintosh Characters to Other Platforms**—LabVIEW contains improved mapping features for the Macintosh Roman character set so 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 supports QuickDrawGX printing dialog boxes. With a QuickDrawGX dialog box, you can redirect print jobs and use new features provided by printing extensions.

**Code Interface Node Changes**—The following CIN data type names have changed to reduce incompatibilities with system header files: `boolean` to `Bool32`, `Ptr` to `UPtr`, and `Handle` to `UHandle`.

## Data Acquisition Changes

The Intermediate VIs 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**, and **DAQ Utilities** has changed to **Signal Conditioning**.

The following new VIs have been added to the **Data Acquisition** palette: SCXI Cal Constants VI (located in **Calibration and Configuration**), 1200 Calibrate VI (located in **Calibration and Configuration**), and 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.

## Front Panel Features

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

**Setting Arbitrary Marker Spacing for Scales**—You can specify an arbitrary distribution of markers so that you place inner markers on an exact point of the scale.

**Changing Marker Location**—In addition to typing in a scale marker for slides, ramps, knobs, or graphs, you can drag a tick mark next to a marker to move the marker location.

**Setting Color in Color Ramps or Intensity Graphs**—Use this option after creating a marker to pop up on a tick mark or marker text and 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, and set plot width and plot fill.

**Setting Line Width for Graph Plots**—With the Line Width option, you can make a line thicker than the default one pixel or change the line to a hairline width if your printer supports hairline printing.

**Selecting Text and Cells in Tables**—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**—Platform independent versions of radio buttons and check marks allow you to create VIs with a similar look, regardless of the platform on which your VI runs.

**Programmatic Blinking, Sizing, and Positioning**—You can programmatically specify certain controls to blink, resize, or reposition.

**Enhanced Metafile Picture Support for Windows**—You can load the new metafile format, called an enhanced metafile, from a file or clipboard without LabVIEW converting them to bitmaps. Unless you are using an enhanced metafile, LabVIEW converts all other metafiles to bitmaps.

**New and Improved VI Examples**—The following VIs are new or revised examples: FileWrite VI (`labview\examples\cin`), FileRead VI (`labview\examples\cin`), and HP34401A Application Example VI

(`labview\examples\instr\hp34401a.llb`). For additional information about individual VIs, choose **Windows**»**Show VI Info...** and read the descriptions included with LabVIEW.

The `labview\examples` directory contains the 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.

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. Because these VIs are structured as templates, you can open the VI, modify the data type for that queue, and save it 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.

## General Interface Changes

**Show Panel/Diagram Window Shortcut Key Change**—The shortcut key for toggling between the front panel and block diagram windows has changed on Sun and Concurrent and on HP-UX for the new Find feature shortcut key. Table 1 lists the new shortcut keys.

**Table 1.** Shortcut Key Changes

| Platform | Toggle Panel and Diagram | Find |
|---|---|---|
| Windows | <Ctrl-e> | <Ctrl-f> |
| UNIX (Sun and Concurrent) | <meta-e> | <meta-f> |
| UNIX (HP-UX) | <alt-e> | <alt-f> |
| Macintosh | <command-e> | <command-f> |

**Tools Are Located in a Floating Palette**—The tools in the toolbar are located on a floating **Tools** palette. This palette automatically opens when you launch LabVIEW. By holding down the <Shift> key and popping up on the window, you can open a temporary copy of this palette. 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**—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, a VI automatically returns to edit mode so you can debug it. To manually change modes, choose **Operate**»**Change to Run Mode** or **Change to Edit Mode**.

**Technical Support VI**—LabVIEW 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 needed to solve 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.

**(Macintosh) DAQ Navigator VI**—LabVIEW includes a DAQ Navigator VI 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 displays references you can use for further research.

**Find 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 all VIs in memory.

**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 VI callers, and so on. In addition, you can search for a particular label or VI in the hierarchy.

**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.

**Creating and Wiring Controls, Constants, and Indicators**—You can 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 or Controls from Constants on Windows and Macintosh**—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 with Windows or Macintosh, you can drag information from the file system or other applications to LabVIEW front panels and block diagrams.

**Tip Strips for Icons, Buttons, and Objects**—The palette icons and the buttons in the toolbar contain tip strips explaining the function of the icon or button.

**Adding VIs to the Project and Help Menus**—You can 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 feature for 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 choose 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.

**(Windows)** Because Windows does not support mixed case or long file names, you can place a file with the same name as the directory plus `.txt` extension next to the directory to provide clearer names for these subdirectory menus. If you place a VI library inside one of the directories, LabVIEW places only VIs that are marked as top level on the menu bar.

**Help Window Displays More Information and Toggles between Views**—The Help window is 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 the **Simple/Complex Diagram Help** button in the Help window or choose **Help**»**Simple Diagram Help** to toggle between views.

**Online Help Enhancements**—You can access online help through the LabVIEW Help Window by clicking on the Online Help button. You also can define your own links to online help documents.

**History Improvements**—The History window includes VI revision number information, remembers the name of last person who logged in, and shows or hides empty history entries.

**System File Dialog Boxes Work with LabVIEW on Windows and Macintosh**—Previously, LabVIEW could use only 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.

**Universal Naming Convention (UNC) Support for Windows**—With UNC filenames, you can specify the location of a file or directory in a networked environment.

**Macintosh Preferences Location and Format**—LabVIEW now stores preferences in a text file format rather than a binary file format. You can now position the preference file next to LabVIEW or your application.

**Separate Default Fonts for Front Panel/Block Diagram**—You can 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.

**LabWindows/CVI Panel Converter on Windows and UNIX**—Using this feature, you can call code created using LabWindows/CVI.

**Update VXI*plug&play* Drivers Menu Option on Windows and UNIX**—VXI*plug&play* installs files in a special `vxipnp` directory on your system. If LabVIEW detects this directory, it adds the **File**»**Update VXIplug&play Drivers** menu option to the **File** menu, which makes it easy to import these drivers into the LabVIEW **Functions** palette. When you choose **File**»**Update VXIplug&play Drivers**, LabVIEW scans the `vxipnp` directory to find new instrument drivers. LabVIEW looks for LabWindows/CVI Function Panels and VI Libraries and displays lists of the new files that it found, asking which function panels you want to convert and which LLBs you want to copy to the `instr.lib` directory. For every function panel you select, LabVIEW converts the function panel just as if you selected **File**»**Convert FP File** from the menu.

## 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 to work with the VISA functions. These functions should have better performance and take up less memory then 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.

**GPIBDRV Support File is No Longer Needed**—By converting GPIB VIs to functions, LabVIEW eliminates 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.

VISA now includes the following functions:

- VISA Assert Trigger
- VISA Clear
- VISA Close
- VISA Find Resource
- VISA Lock
- VISA Open
- **(Windows)** VISA Write (functions asynchronously)
- **(Windows)**VISA Read (functions asynchronously)
- VISA Read STB
- VISA Status Description
- VISA Unlock
- VISA Write
- VISA Read
- 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

**Instrument Handle Parameters are Now VISA Session Refnums**—The **instr handle in** input is called the **VISA session** input and the **instr handle out** output is called the **dup VISA session** output. Both of these parameters have changed to refnums.

## Menu Changes

**Project Menu**—The **Project** menu 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**.

**Edit Menu**—The **Edit** menu now contains the following options: **Select Palette Set**, **Edit Control and Function Palettes**, and **Create SubVI from Selection**.

**Edit**»**Text** no longer exists. You can change the font type, style, size, color, and so on by using the Font Ring option located in the toolbar.

**Edit»Alignment** no longer exists. You can change the alignment by using the Alignment Ring option located in the toolbar. You can still use the shortcut keys listed in Table 2 to access this option.

**Table 2.** Alignment Shortcut Keys

| Platform | On HP-UX | On Sun |
|---|---|---|
| Windows | <Ctrl-a> | <Ctrl-a> |
| Macintosh | <command-a> | <command-a> |
| UNIX | <Alt-a> | <meta-a> |

**Edit»Distribution** no longer exists. You can change distribution by using the Distribute Ring option located in the toolbar. You can still use the shortcut keys listed in Table 3 to access this option.

**Table 3.** Distribution Shortcut Keys

| Platform | On HP-UX | On Sun |
|---|---|---|
| Windows | <Ctrl-d> | <Ctrl-d> |
| Macintosh | <command-d> | <command-d> |
| UNIX | <Alt-d> | <meta-d> |

**Windows Menu**—The **Windows** menu has new options for showing the **Controls**, **Functions**, and **Tools** palettes. **File**»**Get Info...** has changed to **Windows**»**Show VI Info...**.

**Help Menu**—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 floating palettes. You still can pop up 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**.

Floating palettes have the following advantages over 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 choosing **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 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 and 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 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...**.

## Palettes

**Communication Palette Changes**—Because the **Networking** menu is now the **Communication** palette, all example VIs are now located in `labview\examples\comm`.

**(Windows)** The DDE client VIs are now located on the top level of the DDE palette. The server VIs have been moved to the DDE Server subpalette.

The **Communication** palette now includes Object Linking and Embedding (OLE) Automation VIs. Example VIs are located in `examples\comm\OLE-xxx.llb`. The following OLE VIs are located in the **OLE** palette: Create Automation Refnum, Execute Method, Get Property, List Methods or Properties, List Objects in Type Library, Release Refnum, and Set Property.

**(Macintosh)** 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** to **Curve Fitting**, **Signal Processing** to **Digital Signal Processing**, **Statistics** to **Probability and Statistics**, and **Array and Numeric** 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, Create Special Matrix, Complex SVD Factorization, Create Special Complex Matrix, Solve Linear Equations, Solve Complex Linear Equations, Test Positive Definite, and 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, and General LS Linear Fit.

**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, and Statistics Solver.

## Performance Changes

**Decreased 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 Windows 3.1, Solaris2, and Power Macintosh.

**Faster HP-UX Arithmetic (UNIX)**—LabVIEW now features improved arithmetic performance for basic mathematic operations, such as the add and subtract functions.

# New Features in LabVIEW 3.1

This section describes the features added between 3.0.1 and 3.1.

## Block Diagram Features

**Call Library Function in Windows and UNIX**—Use the Call Library Function node from the **Miscellaneous** palette of the **Functions** menu to call a Windows DLL function or a UNIX Shared Library function directly without writing a code interface node (CIN).

**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. Previously created CINs do not need to be recompiled, and their source files work with the new version. Using the **Create .c File** option simplifies the creation of new CINs because you only have to fill the contents of the `CINRun` routine for many CINs.

Solaris 2.*x* CINs use a shared library format, which makes it easier to link to standard libraries. All CINs written with LabVIEW 3.0.*x* should continue to work without recompiling. 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 might 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 when you want to listen for a connection without using a timeout, and you want to cancel the listen when some other condition becomes true (for example, when the user presses a button).

**UDP Networking Capability**—LabVIEW works with the User Datagram Protocol (UDP), one of the protocols in the TCP/IP suite. UDP is generally used in applications that need 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.

**File Utility VI Enhancements**—Using the File Utility VIs, you can 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 count value now defaults 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.

**Probe Enhancements**—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 choose 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 choosing the **Find Wire** option.

## Compatibility and Added Support

**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.

A Power Macintosh system frequently delivers a significant performance improvement over a 680*x*0-based Macintosh. In some cases, the increased performance speed can be four times faster than a comparable Macintosh

system, especially in graphics-intensive applications. In other areas, performance might be approximately the same as on the 680*x*0 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 680*x*0-based NuBus slots. This incompatibility does not affect all plug-in cards.

National Instruments has new revisions of all of its boards affected by this NuBus problem. Table 4 lists Macintosh boards and the shipping date of a compatible Power Macintosh revision. If you have a board that is not compatible, National Instruments can modify or upgrade it for you.

**Table 4.** Power Macintosh Hardware Compatibility

| Board | Revision of Compatible Version | Shipping Date |
|---|---|---|
| NB-TIO-10 | All revisions work | NA |
| NB-DMA2800 | D2 and higher | Nov 1, 1994 |
| NB-DSP230x | D2 and higher | Nov 1, 1994 |
| NB-A2000 | All revisions work | NA |
| 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 |

**Table 4.** Power Macintosh Hardware Compatibility (Continued)

| Board | Revision of Compatible Version | Shipping Date |
|---|---|---|
| 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, revision B4 and higher, 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 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.

**Power Macintosh CINs**—CINs compiled for the Motorola 680*x*0 (68K) Macintoshes do not run in LabVIEW for the Power Macintosh. These CINs must be recompiled for the Power Macintosh using one of the available compilers for the Power Macintosh. Most CINs should recompile without modification. LabVIEW 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 might 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 concerning building both the Power Macintosh and 68K versions of a CIN in the same directory. Because the naming conventions for object files and `.lsb` 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 similar to Symantec THINK C. Symantec has not released a version of THINK C for the development of Power Macintosh code at this time.

## Data Acquisition Changes

**PC Data Acquisition Boards**—LabVIEW now works with the new E-series boards.

**New Counter VIs for Windows and Macintosh**—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 for Windows and Macintosh**—In the CTR Mode Config and CTR Control VIs, you cannot change the **count direction** from its default setting.

**Definitions of Low-Level CTR Pulse Config Parameters for Windows and Macintosh**—The parameters named period one and period two are now phase 1 and phase 2, and their definitions are different.

**Renaming of the NI-DAQ DLL**—The NI-DAQ DLL name has changed from `ATWDAQ.DLL` (or `MCWDAQ.DLL` on Micro Channel computers) to `NIDAQ.DLL`. The `DAQDRV` file in your `LABVIEW` directory, which tells LabVIEW the name of the NI-DAQ DLL, has been updated to reflect the new name. The old DLL (`ATWDAQ.DLL` or `MCWDAQ.DLL`) is deleted from your windows or system directory when you install LabVIEW 3.1.

## Front Panel Features

**Type Definitions**—You can change all copies of a type definition to ordinary controls. First choose **Save As...** for the type definition. Change it to an ordinary custom control by deselecting the type definition boxes, and then choose **Apply Changes** from the **File** menu, which changes all copies of the type definition to ordinary controls on all front panels currently in memory.

**List-Box Controls**—New list-box controls are available from the **List & Ring** palette of the **Controls** menu. Use the list box to present the user with

a scrollable list of options, similar to the list you see in the File dialog box. Use the Single Selection Listbox if you want to limit the number of selections to one and the Multiple Selection Listbox to allow more than one selection.

**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 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, toggle the button. You also can use this dialog box to designate specific controls to skip when the user tabs from control to control.

**String Enhancements**—The string control can 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 and 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 display string data in normal fashion, as backslash codes in place of nonprintable characters, in a password style where the control displays an asterisk symbol (*) for each character you enter into it, or as Hex values.

**Table Font Capability**—You can change the font of text within a table using the **Text** menu.

**Graph and Chart Enhancements**—Several options on 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.

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. 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.

**Scale Enhancements**—In LabVIEW 3.0.*x*, scale 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 whether 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.

**Picture Control Mouse Attributes for Windows and Macintosh**—If you have the Picture Control toolkit, you can use a new mouse location attribute to read the location of the mouse when the pointer 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 or menu key) are pressed. Use the Help window with the attribute node for the picture control to understand the new mouse attribute 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 might be problematic for VIs you do not want to change accidently, such as those delivered with LabVIEW. For this reason, National Instruments suggests you protect files in the `EXAMPLES` and `VI.LIB` directories and your own finished VIs.

**Enclose Existing Diagrams in Structures**—When you create a structure (While Loop, For Loop, Sequence structure, or Case structure) by choosing it from the **Functions** menu or the **Functions** pop-up menu, 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 accidently 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 choosing a structure from the **Functions** menu, if you click in 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 choosing an option from the structure pop-up menu. With While 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 all frames and cases are deleted, a dialog box appears when you select this operation to inform 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 contains 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 if 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 track changes made 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 choosing **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 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 might 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 on how the File Dialog is used. For instance, when you choose **Open**, the option to view only VIs and controls is 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 then can 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. Use this feature when you discover that the top-level VI is broken, but the error that caused it to break is actually in a subVI.

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. (To change memory allocation on a Macintosh, use the Finder **Get Info...** option.) Windows users can also select whether LabVIEW should use the default timer resolution (55 milliseconds) or one millisecond resolution. The disk space checking preferences are now part of this page.

☞ **Note** ***LabVIEW no longer stores Preference information in the*** `.Xdefaults` ***file on UNIX systems. This information is now stored in the*** `.labviewrc` ***file. You also can specify the file that LabVIEW uses for Preference information using a command-line option.***

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

With the **Printing** view, you can choose between Standard Printing (enables each platform to convert drawing commands to printer

commands), bitmap printing, and PostScript printing. If you select PostScript printing, you get options to select Level 1 or Level 2 PostScript and to enable color/grayscale printing. To use PostScript printing, you must have a PostScript printer.

You can use the **History** view to choose the default history settings for a given VI. You can also use it to determine the user name for the history window.

The **Miscellaneous** view 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 specify whether all function keys should be assignable to controls 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 also can select whether LabVIEW should use the U.S. standard for decimal points (a period) or the localized standard (usually a comma).

**(UNIX) Sun Bitmap Scaling Enhancements**—In previous versions of LabVIEW, bitmaps could not be scaled for display; if you tried to make a bitmap smaller by shrinking it, the image was clipped. In addition to causing problems for bitmaps on panels, this made it difficult to use the print preview option because it could not accurately create a small image of your printout. In version 3.1, LabVIEW correctly scales bitmaps as necessary for display.

## 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 map the VISA Transition Library directly to native VISA calls.

## 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 and showing the Error List window. You can use the new tile option 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.

## Performance

**Expandable Memory Manager for Windows and UNIX**—In previous versions, LabVIEW allocated a single large block of memory (with a size set by the user) from which all subsequent memory was allocated. If this amount of memory was insufficient, LabVIEW could not allocate more memory. LabVIEW 3.1 no longer allocates a large block of memory at launch time. When LabVIEW needs more memory, it attempts to allocate it from the available memory in your system. The About LabVIEW dialog box displays the amount of memory being used for the VIs that are loaded. It does not include the base amount of memory that LabVIEW uses.

**Saving VIs is Much Faster**—Saving VIs with version 3.1 is considerably faster than with previous versions. This increase is most noticeable when saving large VIs or when saving VIs into large libraries.

**Memory Monitor VI**—The new Memory Monitor VI makes it easier to determine how your VIs use memory. You can access the Memory Monitor VI from `MEMMON.LLB` in the `EXAMPLES` directory.

## Printing Features

**PostScript Printing**—If you have a PostScript printer, LabVIEW can print PostScript. PostScript printouts can more accurately reproduce the screen image. PostScript facilitates the printing of high-resolution graphs and reproduces patterns and linestyles more accurately. On UNIX systems,

PostScript printing quality is significantly better than bitmap printing on the Sun.

You can use the Preferences dialog box to choose PostScript printing. LabVIEW works with Level 1 PostScript and Level 2 PostScript. Notice that Level 2 PostScript works with color. If you select PostScript printing, then you have the option to select Level 2 PostScript as well as color printing.

**Print Options**—The printing options in LabVIEW have been significantly reorganized and a number of new capabilities have been added. 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 and a list of the subVIs of a VI, including the icon, name, and path of each subVI. 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.

## New Features in LabVIEW 3.0.1

LabVIEW 3.0.1 introduced a few new features. Two new options were added to the Preferences dialog box. The **Performance & Disk** page has 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 might 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 makes it easier to differentiate between two wires that cross and a wire that branches.

**(Windows)** You can move the Preference file to a directory other than the LabVIEW directory. This capability makes it easier to use LabVIEW in a networked environment.

**(Macintosh)** 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.

# Compatibility Issues

This section describes the compatibility issues between different LabVIEW versions.

## Compatibility between Versions 4.1 and 5.0

**Compatibility VIs for New Server Functionality**—LabVIEW now can act as a server, so you have expanded control over VIs. You can control VIs across a TCP/IP network and, on Windows, the ActiveX interface. LabVIEW 5.0 includes Compatibility VIs for the VI Control VIs that exist in previous versions. For information about how to implement the functionality from the VI Control VIs using the new server functions, open each VI Control VI and analyze the implementation of the VI Server feature. You can copy this code to your new LabVIEW applications.

**Compatibility VIs for ActiveX Functions**—In LabVIEW 5.0, the ActiveX functionality has expanded. The functions are more generic because LabVIEW now can act as an ActiveX server as well as a client. Compatibility VIs are provided for the ActiveX functions that exist in previous versions. For more information about the new ActiveX functionality, refer to the *ActiveX Automation* section in the *New Features in LabVIEW 5.0* section earlier in this document.

## Compatibility between Versions 4.0 and 4.1

**(UNIX)** National Instruments no longer includes support for Data Acquisition boards on SUN. If you have used these boards with a previous version of LabVIEW, you can convert the driver files and VIs to LabVIEW 4.1 and your boards will continue to work.

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

## Compatibility between Versions 3.1 and 4.1

**Call Instrument VI Compatibility**—National Instruments has changed the Call Instrument VI to improve performance and add additional error checking. However, this improvement might 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 run-time 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 the Solve Linear Equations VI. 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 choose **Relink to SubVI**.

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

**Example VI Changes That Cause Incompatibilities**—Many example VIs have changed. If you use any example VIs as subVIs, as you might 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.

## Compatibility 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**, that you can use to control parameter weighting and an output array indicator, **Covariance**, that you can use to test the *goodness-of-fit* and the validity of the model function. You need to pop up on calls to the subVI and choose **Relink to SubVI** to use the new version.

IIR Filter VIs (Butterworth Filter, Chebyshev Filter, Inverse Chebyshev Filter, Elliptic Filter, and Bessel Filter VIs) now use cascade filter design techniques for improved stability. Neither the connector panes for each of these VIs or the filtering functionality have changed.

The IIR coefficient VIs (Butterworth Coefficients, Chebyshev Coefficients, Inverse Chebyshev Coefficients, Elliptic Coefficients, and Bessel Coefficients VIs) 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 VI, the Cascade->Direct Coefficients VI, 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` directory, 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 new attributes 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 might need to change VIs that call these VIs to set the counts correctly.

**(Windows and Macintosh) 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 different 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 previously were named period one and period two; however, which period was the delay phase and which period 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/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 the level it was at when the counter stopped.

These changes might require changes to your pulse-generating VIs. These alterations break a VI only if it uses Bundle by Name or Unbundle by Name to access the older 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 might 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 and Bug Fixes

This section lists the major changes and fixes in all major LabVIEW releases. Problems that are difficult to characterize or cosmetic in nature are not listed.

## LabVIEW 5.0

The following list describes major changes in LabVIEW 5.0.

- If you run code interface nodes (CINs) and call library nodes from previous versions of LabVIEW in LabVIEW 5.0, the CINs and call library nodes run in the user interface thread. Most VIs included with LabVIEW that use CINs, such as the Analysis VIs, run using multiple threads if the operating system supports multithreading.

- Priority categories for parallel tasks have changed.
- With LabVIEW 5.0, you no longer need to provide default subroutines for CINAbort, CINDispose, CINInit, CINLoad, CINSave, CINUnload, or the new CINProperties function.

  If a user supplies those functions, those versions are used, but otherwise, they link to default functions with the return values described in the following table.

**Table 5.** Return Values

| Function Name | Return Value |
|---|---|
| CINAbort | `noErr` |
| CINDispose | `noErr` |
| CINInit | `noErr` |
| CINLoad | `noErr` |
| CINSave | `noErr` |
| CINUnload | `noErr` |
| CINProperties | `mgNotSupported` |

- In LabVIEW 5.0, the storage of Booleans and Boolean arrays has changed. The storage format change affects the Read File, Type Cast, and Unflatten String functions. LabVIEW 5.0 stores Boolean data in a single byte, regardless of whether it is an array. For more information, see *Converting Boolean Data to and from LabVIEW 4.x* in the *Upgrading from LabVIEW 4.x* section earlier in this document.

The following list describes bug fixes in LabVIEW 5.0.

- Wiring an enumeration to the string to number functions (From Decimal, for example) no longer causes LabVIEW to report an error message and quit.
- When converting CVI Function Panels to LabVIEW, if the Function Panel file contains errors, LabVIEW no longer reports an error message and quits.
- **(Sun)** In the *LabVIEW Online Reference*, if you choose **Find**, the Help system no longer hangs. If you cancel the dialog box, the Help system no longer crashes.
- Knobs with hidden digital displays that are set to a logarithmic scale no longer crash if you use the arrow keys to increment or decrement them.

The following list describes general improvements in LabVIEW 5.0.

- If an indexing output tunnel on a loop is not wired on the outside of the loop, LabVIEW no longer allocates storage for the array.
- The To Hexadecimal function, which previously returned eight characters, now returns only four characters for 16-bit integer and unsigned 16-bit integer data and two characters for 8-bit integer and unsigned 8-bit integer data.
- **(HP-UX)** You now can wire an array with more than 16 elements to the Boolean To (0,1) function.
- LabVIEW 4.1 introduced a new behavior for type coercion of integers to enumerations. LabVIEW 5.0 converts signed integers according to new rules. Unsigned integers are converted with the standard LabVIEW rules. In particular, signed negative numbers now coerce to the first item in the enumeration.
- **(UNIX)** Various problems with the Help window and other windows moving up or down the screen have been corrected.
- You now can use Sort 1D Array for paths and pictures.
- **(UNIX)** If you launch LabVIEW with the –display option, LabVIEW now correctly passes the DISPLAY environment variable to a web browser when it launches a menu command from the **Internet Links** menu. LabVIEW also corrects this behavior for commands called from the System Exec VI.
- You now can assign Formula Node inputs without a corresponding output terminal.
- Unsigned numbers wired to the selector of a Case structure now coerce as if the selector also is unsigned.
- You now can use underscore in variable names in the Formula Node.
- When the Call Library Function cannot find its corresponding DLL or shared library, LabVIEW 5.0 reports errors that better describe the problem.
- String controls and indicators in hexadecimal display mode now print with a correct font size.
- If you attempt to read a large number of items in a byte stream file, LabVIEW no longer reports memory full.
- If you open the pop-up menu of a subVI and choose **Create Control** on an input or output that is a typedef in the subVI, LabVIEW now creates a copy of the typedef.
- **(Windows 95/NT)** You now can convert Julian dates correctly with the Date/Time To Seconds function.
- If the preference file cannot be edited (for example, it is on a read-only server or a CD-ROM), you now can change palette menu sets.

- You now can use Array Max Min with an array that begins with a NaN value.
- You now can change the serial port buffer size with VISA.
- **(Macintosh)** You now can drag and drop strings and paths without problems.
- You now can use To Upper Case and To Lower Case with languages other than English on all platforms, including UNIX.
- LabVIEW 5.0 corrects problems with hidden list boxes that draw in previous versions.
- When printing, all frames of a sequence now print.
- Extremely large printouts (for example, diagrams with hundreds of frames of a sequence) now print correctly.
- **(UNIX)** LabVIEW 5.0 now saves printer page orientation correctly.
- LabVIEW 5.0 corrects various page layout problems when printing.

## LabVIEW 4.1

The following list describes 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 might cause LabVIEW to crash.
- **(Windows)** LabVIEW might 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 reported 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 might display an error message and quit.
- Pasting diagram objects onto a new front panel might 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 might 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 might draw incorrectly or cause LabVIEW to display an error message and quit.
- When scrolling a diagram while doing cluster ordering, LabVIEW might display an error message and quit.
- If you configured a Boolean typedef to blink, LabVIEW displayed 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 might 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 might 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.

The following list describes 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 two data points might print incorrectly when using PostScript printing.
- We now print grayscale (instead of black and white) printouts on black and 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 might be set to 0, 1, 16, or 17.
- Traditional GPIB Read and 488.2 Receive, Find Listeners and Read Response Message might 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 might report the `Unable to queue asynchronous operation` error erroneously.
- Wait on occurrence might 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.
- The editable attribute of a chart or graph scale did not prevent you from editing the scale.
- Hidden controls might draw.
- 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 two 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 might 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 did 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 might 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) ran 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 might 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 might not 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 might draw the last chart incorrectly.
- LabVIEW incorrectly allowed an array of a cluster of one number to be wired to a graph.
- The **Clear** command in the **Edit** menu did not work to delete objects.
- **(Windows)** In certain situations, Enums might 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)** LabVIEW now accounts for the Windows 95 task bar when resizing windows to make them tiled or full screen.
- A block diagram pop-up menu on a refnum might show the cluster tools menu incorrectly.

- Using the Find feature on VIs with no block diagram might 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.

## LabVIEW 4.0.1

LabVIEW 4.0.1 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 did not close the file specified by the file refnum input.
- Sort 1D array did not work for an array of clusters if the cluster contained path or picture components.
- The Initialize Array and Resize Array functions produced 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 chose **Find** (<Ctrl-F>), the Find dialog box showed the password text.
- When you selected **Find»Locals** or **Find»Attribute Nodes** from a pop-up menu it might 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 might 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 crashed.
- If you passed a string parameter larger than 32K in size to Format Into String function, LabVIEW might give a memory full error, stop the VI, or crash.
- If you made more than one call to the Call Chain function, LabVIEW might 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 crashed.
- The Write File+[I16] utility function did not have the pos mode input wired to the Write File function in the block diagram. This caused the function to always overwrite data in a file.
- If you tried to use a type definition recursively, LabVIEW crashed.
- 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 might 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 might 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 might still allow their scales to be modified.
- **(Windows)** By default, when you copy a picture from LabVIEW on Windows 95/NT, LabVIEW exports a bitmap to the clipboard. With 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, on Windows 95, some programs import the metafile and lose track of the scale of the metafile (it becomes huge when it is imported). Second, on both Windows 95 and Windows NT, the bottom and right edges of the picture are lost in the resulting metafile.
- **(Windows)** The LabVIEW installation on Windows 3.1 did 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 on 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 did 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)** Choosing 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 does not 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)** National Instruments recommends 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)** Choosing **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 read `Symantec C` and put in new references to v8.0.5 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 always relinquishes 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 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 caused LabVIEW to crash.
- **(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 an unsigned word enum in a cluster.
- **(UNIX)** The OR Array Elements function on HP-UX might not produce the correct result on arrays larger than 65535. This behavior did not occur on the Sun platform.

## LabVIEW 4.0

The following list describes some of the major problems in version 3.1 that are corrected in LabVIEW 4.0.

- **(Windows)** Printing VIs with some printer drivers caused LabVIEW to display an error message and quit.
- **(Windows)** Using PostScript printing on a printer that did not support PostScript printing might cause LabVIEW to display an error message and quit.
- 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 might 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.
- **(Macintosh)** Coloring the foreground color of a Square or Round LED transparent caused LabVIEW to display an error message and quit.
- **(Power Macintosh)** Changing units on a control or indicator might 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 then closed the Call Instrument front panel while it was still running, under some conditions the VI was not aborted; subsequently, LabVIEW might 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.
- **(Windows NT)** Using the traditional Wait for RQS VI caused LabVIEW to crash in some situations.
- **(Windows)** Loading a VI containing a call to a DLL that used floating-point instructions and then loading the NI-DAQ DLL might cause a General Protection Fault.
- **(Macintosh)** 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 might cause LabVIEW to crash.
- Tabbing through the fields in the Control Editor window to resize or position an object might cause LabVIEW to display an error message and quit.
- **(Windows)** Using <Alt> + menu key, keyboard accelerators while the title screen was displayed might cause a General Protection Fault.
- Changing the connector pane of a subVI that had database access enabled and then relinking to the subVI might cause LabVIEW to display an error message and quit.
- **(Windows)** Scrolling through some sections of the help file might cause a page fault.
- **(Windows)** If the preferences file was read only and you tried to change preferences, LabVIEW crashed.
- **(Macintosh)** When dragging objects, the screen occasionally might erase the areas you dragged over instead of redrawing the areas.
- **(Windows)** Copying a picture from another application to LabVIEW occasionally caused LabVIEW to crash.

- **(Windows)** Restoring a minimized diagram that contained a selected object might cause LabVIEW to display an error message and quit.
- **(Power Macintosh)** Replacing a list box with another list box might cause LabVIEW to crash.
- **(Sun)** Opening the History windows of two VIs at the same time caused LabVIEW to crash.

The following list describes the incorrect behavior that has been 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.
- **(Windows)** When printing graphs with logarithmic scales, scaling the display might cause an overflow that resulted in the plot appearing upside down.
- **(Windows)** Changing page orientation from the Print dialog box did not work (it only worked from **Page Setup...**).
- **(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 the data in a table, if you changed a cell then clicked in an empty cell, the data in the previous cell 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.
- **(Windows 3.1)** Call Library Function might not find its DLL if the DLL was not in the search path and the VI calling the DLL was loaded from the **Functions** menu.
- 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.
- **(Windows)** The Formula Node incorrectly might return NaN for some results due to the floating-point stack being left in a bad state.
- **(Power Macintosh and HP-UX)** 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 and HP-UX)** Absolute Value did not convert –0 to 0.
- **(Windows)** The sin(x) function was not correctly exported in the Windows 3.1 Watcom CIN libraries.
- **(UNIX)** The Call Instrument VI returned an error if you tried to pass VI data whose flattened size was less than the minimum size of the structure, once alignment constraints were taken into consideration.
- When you dynamically loaded the same VI from multiple VIs, under some conditions, releasing the VI unloaded the VI from memory even though not all callers had released it.
- **(Windows)** If you used the Write File VI to write a string to a bytestream file and you specified write Header as TRUE, the header was written out in little-endian format. Data could not be read back using the Read File VI because the Read File VI correctly assumed the header should be in big-endian format.
- **(UNIX and Macintosh)** 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.
- **(Windows)** The DDE VIs did not return an error if a timeout occurred.
- **(Macintosh)** 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.

- **(Macintosh)** The CINMake script produced make files with rules that did not work.
- **(Macintosh)** LabVIEW CIN header files were incompatible with recent Apple include files.
- **(Windows 3.1)** LabVIEW could not open files inside of deep directory structures (files greater than a 60-character pathname).
- **(Windows 3.1)** LabVIEW restricted you to opening only 40 files at a given time; this limit has been changed to allow up to 256 files at once.
- **(Windows)** If the loading status dialog box came up while dynamically loading a subVI, LabVIEW essentially stalled unless the calling VI executed a wait function.
- **(Windows NT)** Attempting to read more than 4096 bytes from serial port returned an error code.
- **(Windows)** LabVIEW did not support saving to or launching from hard disks larger than 2 gigabytes.
- **(Windows)** Choosing the **Save Text Info** option from the **Print Documentation** menu, truncated multiline descriptions and history text.
- **(Windows)** Open Mode with mode 3 did not truncate the file.
- **(Macintosh)** If you ejected a disk from the LabVIEW file dialog box and the disk was named `a`, LabVIEW prompted you to reinsert the disk.
- **(UNIX)** Serial Port Break did not function.
- 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.
- **(HP-UX)** Opening help from the **Help** menu did not work unless you modified a script that launched the help program to point to the correct path.
- **(HP-UX)** Accumulating a 1D array of data that was double aligned (a double or a cluster containing a double) caused LabVIEW to overwrite the end of the array by 4 bytes.

# LabVIEW 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.
- **(Windows)** The Array to Spreadsheet String function crashed or corrupted memory for arrays with a small number of columns.
- 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 or 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 <Tab> caused LabVIEW to display an error message.
- Highly parallel, asynchronous diagrams caused LabVIEW to display an error message when you compiled the VI.

- **(Macintosh)** 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.
- **(Windows)** The Integer Quotient & Remainder function crashed if the top input was an unsigned 8-bit integer.
- The Spreadsheet String to Array function could not manage tables containing some rows with more columns than others.
- **(Macintosh)** 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.
- **(UNIX)** The To Decimal function returned a value of 0 for negative floating-point values.
- **(UNIX)** The Log(n+1) function for complex single-precision numbers returned incorrect values.
- The Formula Node returned an incorrect value when you took a negative number to an odd power, if the power was a constant. This worked 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.
- **(Windows)** Exiting LabVIEW without ending conversations while using networked DDE with multiple conversations established caused the computer to hang.
- **(UNIX)** The GPIB Serial Poll VI did not work correctly.
- 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.
- **(Windows and Macintosh)** 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 might cause memory corruption.
- **(Windows)** When writing data to a parallel port, you sometimes received a buffer overflow error.
- **(Macintosh)** The AESend VI could not handle more than 999 characters for the data to be transferred.
- **(Macintosh)** The AESend Print Document VI did not print the specified VI in version 3.0.1.
- **(Windows 3.1J)** On the Japanese version of Windows 3.1, the operating system crashed if you tried to display a NULL (ASCII code of 0) character.

- **(Windows)** Some printers might print with an all-black background.
- **(Macintosh)** Loop with timing functions (Wait ms or Wait until Next ms Multiple) gave up time to other applications if nothing else was processing.
- **(Macintosh)** If you ran a DAQ counter VI, then quit and relaunched LabVIEW, you received 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 might not compute the largest absolute value of the input array correctly.

## LabVIEW 3.0.1

This section details changes and bug fixes between LabVIEW 3.0 and LabVIEW 3.0.1.

☞ **Note** ***Since the LabVIEW 3.0 release, National Instruments has upgraded the Solaris 2 Sbus-GPIB driver and the GPIB-ENET firmware.***

- 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.
- **(Windows and Macintosh)** From Decimal works reliably.
- **(UNIX)** From Decimal, From Hexadecimal, and From Octal work reliably.
- **(Windows)** A problem that caused a crash during bitmap printing after several printouts and a problem that caused black printouts were both fixed.
- **(Windows)** A problem was corrected with the data acquisition driver file (`daqdrv`) that caused occasional crashes.
- **(Windows)** `daqcptrs.lsb` no longer causes page faults under some circumstances.
- **(Windows)** Data acquisition (DAQ) buffer reads are limited to 900 kilobytes; due to limitations in the compiler used to create LabVIEW, reads larger than this amount cause crashes.

- **(Windows)** The digital VIs can address the higher port addresses found on the AT-MIO-16D and the PC-DIO-96 boards.
- The Set EOF function works correctly.
- **(Windows)** The DDE Server VIs no longer have a memory leak. Also, the DDE Poke VI returns the correct error codes.
- **(Macintosh)** 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.
- **(Macintosh)** 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.
- **(Macintosh)** A memory leak was corrected in the AE Send VI.
- **(UNIX)** If you had a diagram iconified and you closed the front panel, LabVIEW might not close the associated diagram. Attempting to open the iconified window at this point might have unpredictable results, possibly including crashes. This problem has been corrected.
- **(UNIX)** In certain cases, LabVIEW used less memory to pass data from one frame of a sequence to another in sequence locals.
- **(UNIX)** LabVIEW makes better choices for connector panes.

321780A-01 Jan98