LabVIEW™ Professional Development System

Version 5.0

These release notes introduce you to LabVIEW, describe the system requirements for the LabVIEW software, and contain installation instructions and updated documentation information. The LabVIEW 5.0 Professional Development System includes the LabVIEW Full Development System, the LabVIEW Application Builder, and the Professional G Developers Toolkit.

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.***

Contents

321888A-01 *January 1998*

How to Proceed

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

Read the *Required System Configuration* section, then follow the instructions in the *Installing LabVIEW* section of these release notes. After you have installed LabVIEW, see the *Where to Go from Here* section for information about getting started with LabVIEW. In addition, you should read the *Manual Clarifications and Additions* section before using LabVIEW 5.0.

Required System Configuration

This section describes the minimum system requirements needed to run LabVIEW 5.0.

Windows

LabVIEW 5.0 Professional Development System for Windows is distributed on CD. The LabVIEW CD contains versions for Windows 95 and Windows NT and includes the complete instrument driver library available when LabVIEW 5.0 released.

The *LabVIEW Online Tutorial* default configuration requires the LabVIEW 5.0 distribution CD to be in your CD-ROM drive. You also can install the *LabVIEW Online Tutorial* files on your hard drive. This installation requires approximately 40 MB of hard disk space.

The *LabVIEW Online Tutorial* and LabVIEW Help files contain 256-color graphics. Your video driver, configured through **Control Panels»Display**, must be configured for 256 colors. Minimum requirements to view the tutorial are 800 × 600 pixel resolution and the Microsoft Video for Windows driver. For viewing Help files, National Instruments recommends that you configure your video driver for 256 colors with 800 × 600 pixel resolution.

Note **(Windows NT)** ***If your display is in 16-bit mode, you might want to change it to 256 or 24-bit color for better performance when viewing the*** *LabVIEW Online Tutorial****.***

(Windows 95) LabVIEW 5.0 runs on any system that supports Windows 95. You should have a minimum of 16 MB of RAM, but National Instruments recommends 32 MB for this version to run effectively.

(Windows NT) LabVIEW 5.0 only runs on Windows NT 4.0 Service Pack 3 or later. You should have a minimum of 16 MB of RAM and a 486/DX processor, but National Instruments recommends 32 MB of RAM and a Pentium processor for this version to run effectively. LabVIEW for Windows NT only runs on Windows NT 80*x*86 computers. It does not run on other processors, such as the DEC Alpha, MIPS, or PowerPC, because LabVIEW uses 80386 instructions. DEC Alpha, MIPS, and PowerPC 80*x*86 emulators must emulate 80386 instructions to run LabVIEW.

☞ **Note** ***On Windows systems, you need approximately 115 MB of disk storage space.***

Macintosh

LabVIEW for Macintosh is distributed on a CD containing both 680*x*0 and Power Macintosh software. In addition to the LabVIEW application, the CD also contains an instrument driver directory that includes all the LabVIEW instrument drivers available when LabVIEW 5.0 released.

LabVIEW for the 680*x*0-based Macintosh requires a math coprocessor. You cannot use a 68000-based computer, such as the Macintosh SE. If you are using a Macintosh IIsi or a Macintosh LC, you might not have a math coprocessor in your computer. Also, computers using the 680LC40 processor (Centris 610s and some low-end Centris 650s) do not have the math coprocessor functions, and therefore cannot run LabVIEW.

LabVIEW requires a minimum of 6 MB of available RAM, in addition to the RAM requirements for your system software and any other applications that you want to run simultaneously. National Instruments recommends that you have at least 8 MB of RAM. You might need more memory, depending on the size of the application you design in LabVIEW and the amount of data that your application manipulates.

(Power Macintosh) LabVIEW requires 24 MB of RAM and at least 130 MB of hard disk space.

(680*x*0-Based Macintosh) LabVIEW requires at least 130 MB of hard disk space.

LabVIEW requires System 7 or 8 and cannot run on System 6 or any previous versions of the Macintosh operating system.

For more accurate timing, install the Apple QuickTime extension. When you use QuickTime, timing accuracy should increase from 16.6 ms resolution to approximately 1 ms resolution. System response varies depending on background applications, other extensions, networking activity, and disk caching.

☞ **Note** ***On Macintosh systems, you need approximately 100 MB of disk storage space for a minimal installation of LabVIEW or 150 MB for a full installation.***

UNIX

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

LabVIEW for Sun runs on SPARCstations with SunOS 4.1.3 (Solaris 1.1) or later and Solaris 2.4 or later. LabVIEW for HP-UX runs on Hewlett-Packard Model 9000 Series 700 computers with HP-UX 9.0.5 or later. LabVIEW for Concurrent PowerMAX runs on PowerMAX version 4.1 or later.

The workstation should have 32 MB of RAM with 32 MB or more of swap space storage for LabVIEW to run effectively. In addition, the workstation must have a minimum of 95 MB of disk storage space if you want to install the entire LabVIEW package. To save space, install only the VIs you plan to use.

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

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

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

Operating System Patches on Sun

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

- 100458-*xx*: Setitimer sometimes fails to deliver SIGALRM

You do not need any other patches to run LabVIEW on Solaris 2.4 or later.

Operating System Patches on Concurrent PowerMAX

You need at least the following patches to run LabVIEW on Concurrent PowerMAX 4.1. The base patches are odd numbers only.

patch	base-001	Base System Patch 001
patch	base-003	Base System Patch 003
patch	base-005	Base System Patch 005
patch	base-007	Base System Patch 007
patch	base-009	Base System Patch 009

Installing LabVIEW

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

Windows

Complete the following steps to install LabVIEW for Windows.

1. **(Windows NT)** Log on to Windows NT as an administrator or as a user with administrator privileges.
2. Run *x*:\WIN95-NT\INSTALLER\SETUP, where *x* is the drive letter for your CD-ROM drive.
3. The installer gives you the option of performing a full installation or a minimal installation. If you do not have sufficient disk space (approximately 115 MB), choose the minimal installation and use your LabVIEW CD to access the remaining components.
4. After you choose an installation, follow the instructions that appear on your screen.
5. **(Windows 95)** To use ActiveX Automation, you need to install Microsoft DCOM 95.

   The LabVIEW installer automatically launches the installer for DCOM 95.
6. If you use data acquisition (DAQ) devices with LabVIEW, complete the following step. For more information about installation and configuration of DAQ drivers, refer to the *Data Acquisition and GPIB Installation Notes* section of this document.

   The LabVIEW installer calls the NI-DAQ installer, which installs the correct DAQ driver software.

7. The LabVIEW 5.0 CD contains all the latest versions of our GPIB drivers, located in the `drivers` directory. If you use GPIB devices with LabVIEW, complete the following step. For more information about installation and configuration of GPIB drivers, refer to the *Data Acquisition and GPIB Installation Notes* section of this document.

   The LabVIEW installer calls the National Instruments GPIB installer, which installs the GPIB drivers and modifies the registry.

8. LabVIEW completes the following steps.
   a. The LabVIEW installer installs the Advanced Analysis VIs automatically.
   b. The LabVIEW installer installs NI-VISA automatically.

After you have completed the installation, LabVIEW is ready to run. If you plan to use DAQ or GPIB devices with LabVIEW, you must restart your computer to load the new drivers.

By default, the *LabVIEW Online Tutorial* requires the LabVIEW 5.0 distribution CD to be in the CD drive. If you want to install all the *LabVIEW Online Tutorial* files on you hard drive, copy the files located in *x*`:\tutorial` to the LabVIEW directory on you hard drive, where *x* is the drive letter for your CD-ROM drive. Overwrite any files that already exist in that directory.

Note ***If you have installed LabVIEW on a server, new users might want to copy the*** `Activity` ***directory from the server to their local machine. You use the*** `Activity` ***directory to complete activities that illustrate basic LabVIEW concepts. You can find these activities in the*** *LabVIEW User Manual* ***and the*** *LabVIEW Online Reference****, which you can access by selecting*** **Help»Online Reference….**

Macintosh

Complete the following steps to install LabVIEW for Macintosh.

1. **(CD Installation)** If you are installing LabVIEW on a Power Macintosh, double-click the LabVIEW Installer icon in the Power Macintosh directory. If you are installing from a CD on a 680*x*0-based Macintosh, double-click the LabVIEW Installer icon in the 680*x*0 Macintosh directory.
2. The installer gives you the option of performing a full installation or a minimal installation. If you do not have sufficient disk space (approximately 150 MB), choose the minimal installation and use your LabVIEW CD to access the remaining components.
3. After you click the **Install** button, you are prompted to select a destination directory. Click the **New Folder** button to create a LabVIEW directory, then click the **Install In Current Folder** button to install the LabVIEW files in that directory.

After you have completed the installation, LabVIEW is ready to run. If you plan to use DAQ or GPIB devices with LabVIEW, you must restart your computer to load the new drivers.

UNIX

Complete the following steps to install LabVIEW for UNIX.

Solaris 1

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

   ```
   mkdir /cdrom
   ```
3. Mount the LabVIEW CD and type the following command:

   ```
   mount -rt hsfs /dev/sr0 /cdrom
   ```
4. To change to the installation directory, type

   ```
   cd /cdrom/solaris1
   ```
5. To run the installation script, type the following command:

   ```
   ./INSTALL
   ```
6. Follow the instructions on your screen.

Solaris 2

1. To enable superuser privileges, type `su root` and enter the root password.
2. Insert the LabVIEW CD. On Solaris 2.*x*, the CD automatically mounts as soon as the CD is inserted into the drive. If this feature is disabled on your workstation, you must mount the CD by typing the following command:

   ```
   mount -o ro -F hsfs /dev/dsk/c0t6d0s2 /cdrom
   ```
3. See the `README` file in `/cdrom/cdrom0/solaris2` or `/cdrom/solaris2` for instructions on custom installation or other additional information.
4. If your CD was mounted automatically, type the following command:

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

   If you used the mount command in step 2, type the following command:

   ```
   pkgadd -d /cdrom/solaris2
   ```
5. Follow the instructions on your screen.

HP-UX

1. To enable superuser privileges, type `su root` and enter the root password.
2. Mount the LabVIEW CD on the `/cdrom` directory with the SAM system administration utility.
3. To change to the installation directory, type the following command:

   `cd /cdrom/HP-UX`
4. To run the installation script, type the following command:

   `./INSTALL`
5. Follow the instructions on your screen.

PowerMAX

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

   `mkdir /opt/lv50`
3. To change to the new directory, type the following command:

   `cd /opt/lv50`
4. Extract the files from the tape by typing the following command:

   `tar xv`
5. To run the installation script, type the following command:

   `./INSTALL`
6. Follow the instructions on your screen.

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

☞ **Note** ***The*** *LabVIEW User Manual* ***and the*** *LabVIEW Online Reference****, which you can access by selecting*** **Help»Online Reference…,** ***provide activities that illustrate basic LabVIEW concepts. If you want to complete these activities, copy the*** `Activity` ***directory from the*** `LabVIEW` ***directory to your home directory.***

☞ **Note** ***The LabVIEW documentation set, including the*** *Code Interface Reference Manual* ***and the*** *VXI VI Reference Manual****, is available in Portable Document Format (PDF) on the LabVIEW CD in the*** `manuals` ***directory. You can copy this directory or selected PDF files to the*** `LabVIEW\manuals` ***directory on your hard drive. You must have Adobe Acrobat Reader 3.0 or later installed to view these files.***

☞ **Note** ***If you are upgrading from a previous version of LabVIEW, read the*** *LabVIEW Upgrade Notes.* ***If you have one of the add-on packages, such as the LabVIEW Test Executive or the Picture Control Toolkit, consider installing those files at this time.***

Data Acquisition and GPIB Installation Notes

All National Instruments GPIB interfaces and DAQ devices come with the drivers and other software you need to use them. LabVIEW also comes with the drivers and other software you need to use National Instruments hardware. While the drivers included with LabVIEW are the same NI-488.2 and NI-DAQ drivers National Instruments includes with its GPIB and DAQ hardware, the version numbers might differ. Always use the driver with the higher version number. You can determine which version of NI-DAQ you are using with LabVIEW by running the Get Device Information VI. If you already own a device with a version of NI-DAQ earlier than 5.0, do not install that driver.

You must configure your DAQ hardware before you can use the LabVIEW DAQ VIs. To configure your DAQ hardware, you need to run the NI-DAQ Configuration Utility. For more information about how to configure your DAQ hardware with the NI-DAQ Configuration Utility, refer to the Help files installed with LabVIEW.

Windows

When you install LabVIEW, the installer places the application and most of the related files in a directory you specify. The default name of this directory is `LABVIEW`. If you install DAQ or GPIB VIs, the installer places additional files, described in the following sections.

Use the NI-DAQ Configuration Utility, which you launch by clicking the **Start** button and choosing **Programs»LabVIEW»NI-DAQ Configuration Utility**.

For information about how to configure your particular DAQ device, refer to the NI-DAQ Configuration Help file. To view this Help file, click the **Start** button and select **Programs»LabVIEW»NI-DAQ Configuration Help**. Follow the links for configuring the devices you have.

You can find further information about the NI-DAQ driver in the NI-DAQ Read Me File. To view this file, click the **Start** button and select **Programs»LabVIEW»NI-DAQ Read Me File**.

Macintosh

The LabVIEW installation program installs two control panels and an extension in your system folder. `NI-GPIB` contains the driver code that communicates with your GPIB devices. `NI-DAQ` contains driver code that communicates with your DAQ devices. The `NI-DMA/DSP` extension contains DSP and DMA drivers used by DAQ, GPIB, and DSP drivers.

GPIB and VXI Installation Notes for Sun

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

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

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

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

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

Where to Go from Here

The following resources can help get started with LabVIEW 5.0 quickly.

If you are a new LabVIEW user, see the *LabVIEW QuickStart Guide* to learn how to begin programming with LabVIEW. The exercises in this manual guide you through the basic steps of building and debugging a LabVIEW application.

(Windows) For an introduction to the LabVIEW environment, complete the exercises in the *LabVIEW Online Tutorial*. Launch the tutorial by clicking **LabVIEW Tutorial** in the LabVIEW dialog box.

☞ **Note** ***To access the LabVIEW launch dialog box, either launch LabVIEW or close all open LabVIEW VIs if you already are running LabVIEW.***

If you are using the Application Builder Library, see the *Using the Application Builder Libraries* section. Refer to the *Source Code Control»Compare Files* section for information about source code control.

Examples and Solutions for Your LabVIEW Programs

(Windows and Macintosh) If you are using data acquisition (DAQ) or instrument I/O and want to find examples or generate solutions for your LabVIEW programs, launch the DAQ Solution Wizard by clicking **Solution Wizard** in the LabVIEW dialog box. For more information about the Solution Wizard, see the Chapter 3, *Data Acquisition*, and Chapter 4, *Instrumentation*, of the *LabVIEW QuickStart Guide*.

(Windows) To find any other type of example, open the Search Examples Help file by clicking **Search Examples** in the LabVIEW dialog box.

(Macintosh and UNIX) The `examples` directory contains a VI named `readme.vi`. With this VI, you can find the available examples. When you select a VI, you can see the documentation that was entered for that VI by choosing **Window»Show VI Info...**. To open a VI, choose **File»Open...**.

Additional Resources

☞ **Note** ***The LabVIEW documentation set, including the*** *Code Interface Reference Manual* ***and the*** *VXI VI Reference Manual****, is available in Portable Document Format (PDF) on the LabVIEW CD in the*** `manuals` ***directory. You can copy this directory or selected PDF files to the*** `LabVIEW\manuals` ***directory on your hard drive. You must have Adobe Acrobat Reader 3.0 or later installed to view these files.***

(Windows and Macintosh) If you need to perform data acquisition, read the *LabVIEW Data Acquisition Basics Manual,* which contains important information about using the DAQ VIs and examples you can find in LabVIEW. For reference information about a particular DAQ VI, refer to the *LabVIEW Function and VI Reference Manual* and the *LabVIEW Online Reference*, which you can access by selecting **Help»Online Reference…**. The *New Features in LabVIEW 5.0* section in the *LabVIEW Upgrade Notes* also contains information about new features and changes in DAQ VIs.

The DAQ examples folder contains a VI library named `RUN_ME.LLB` that has a Getting Started example VI for analog input, analog output, digital I/O, and counters. The `RUN_ME.LLB` examples provide an excellent starting place for data acquisition.

Difference between LabVIEW for Windows 95 and Windows NT

Low-Level Register I/O

LabVIEW for Windows 95 has a set of VIs named In Port and Out Port that you can use to read or write hardware registers. Windows NT applications cannot manipulate hardware directly. If you need to communicate with a hardware device in Windows NT, you must write a Windows NT driver.

Common LabVIEW Launch Errors on UNIX

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

Error Message/Description	Probable Cause/Solution

`Xlib: connection to :0.0 refused by server`
`client is not authorized to connect to server`
`internal error during connection authorization check`
	Probable Cause—Trying to run LabVIEW as a user who does not have permission to open a window on the display server. Typically seen after running the `su` command to temporarily become a different user, such as root (superuser).
	Solution—Exit the `su` command and launch LabVIEW as the login user.

`"Executable version doesn't match resource file"`	**Probable Cause**—Version of LabVIEW executable does not match version of `labview.rsc`. **Solution**—Verify that the `appResFilePath` parameter in the configuration file correctly sets the path to the `labview.rsc` file.

Configuring LabVIEW Windows on UNIX

This section describes procedures for configuring LabVIEW windows on UNIX operating systems.

Configuring LabVIEW with the Tab Window Manager

If you use the Tab Window Manger (`twm`), you can change environment settings so that `twm` interacts better with LabVIEW. With `twm`, you cannot close the floating palette menus in LabVIEW if these windows do not have title bars. To correct this problem, add the following line to your `.twmrc` file in your home directory:

```
DecorateTransients
```

This line adds title bars to the floating windows so you can close them.

Configuring LabVIEW with the HP VUE Window Manager

If you use the HP VUE Window Manager (`vuewm`), you can change environment settings so that `vuewm` interacts better with LabVIEW. By default, `vuewm` does not incorporate the window position requests of an application. This behavior causes LabVIEW windows, such as the Panel, Diagram, Help, and file dialog windows, to not appear in consistent locations on your screen. To change the `vuewm` behavior, use the `xrdb` command to set two `vuewm` settings:

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

To add the two entries, you also can edit the following files manually:

```
$HOME/.vue/sessions/home/vue.resources
$HOME/.vue/sessions/current/vue.resources
```

Configuring LabVIEW with the Motif Window Manager

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

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

To add the two entries, you also can edit the following file manually:

`$HOME/.Xdefaults`

Notice to Sun SPARCstation 5 Users

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

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

To determine whether your SPARCstation 5 is affected, perform the following steps.

Caution ***Following these steps temporarily interrupts the operation of your computer, so you should warn anyone who might be using your computer remotely.***

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

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

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

   In the first case, select `n` to go to new command mode, where you see an `ok` prompt. If you already have an `ok` prompt, skip to step 3.
3. At the `ok` prompt, type

   ```
   module-info
   ```

   You then see information similar to the following lines:

   ```
   CPU FMI,MB86904 Rev. 2.5 : 70.0 MHz
   SBus (Divide By 3)       : 23.3 MHz
   ```
4. Type `go` to exit the monitor and resume operation of your system.

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

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

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

Notice to HP-UX Users

By default, HP workstations limit the size of a process such as LabVIEW to 64 MB. You can change this setting by adjusting a kernel configuration parameter that limits the amount of data a process can use. To edit this parameter, enable superuser privileges by typing `su root` and entering the root password. Use the SAM system administration utility to view the list of kernel configuration parameters. Change the value of the maximum data segment size in bytes parameter to a larger value. If you need to rebuild the kernel and reboot for changes to take effect, the SAM utility guides you through this process.

Notice to Concurrent PowerMAX Users

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

Although you can select the hardware flow control option for the LabVIEW Serial VIs, serial-line hardware flow control is not supported by the operating system and therefore does not work.

Manual Clarifications and Additions

This section contains information that was not included in the LabVIEW documentation as well as corrections to the LabVIEW manuals. For information about new features in this version of LabVIEW, see the *LabVIEW Upgrade Notes*.

Multithreading

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.

ActiveX

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. In the LabVIEW documentation, references to `ole_container.dll` should be `ole_lv5container.dll`.

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.

Instrumentation

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.

General Interface Features

Icon and Text Palettes

You can display palettes in three modes: Standard, All Icons, or All Text. Choose the palette display mode in the **Edit»Select Palette Set»Display Style** submenu.

In All Icons mode, the **Controls** and **Functions** palettes contain icons that represent options such as controls, indicators, functions, and VIs. These icon palettes are similar to the icon palettes in previous versions of LabVIEW.

In All Text mode, you pop up to access the **Controls** or **Functions** palette. These text palettes contain the names of options. Items in text palettes are organized in the same order as in the icon palettes when you read the icon palette row by row, left to right. Empty spaces in the icon palettes are skipped. Unlike icon palettes, you cannot tack down text palettes or subpalettes. In All Text mode, LabVIEW displays text palettes in the **Project** menu and the Find dialog box. In Standard or All Icons mode, LabVIEW displays these palettes as icon palettes.

Standard is the default mode. In Standard mode, all palettes default to icon palettes, but you can edit individual palettes to display them as text palettes.

When you edit a palette by selecting **Edit»Edit Control & Function Palettes…**, LabVIEW displays the palettes in All Icons mode. You cannot edit palettes in the other modes because they do not contain as much information (icon palettes have both icons and two-dimensional layout, while text palettes do not). In the editor, you can specify the mode for any palette by popping up and selecting either **Icons** or **Text** from the **Standard Menu View** submenu. The mode you select affects only the menu you are editing.

☞ **Note** ***In LabVIEW 5.0, you pop up on palettes or the icon of a submenu so you have a method for changing the settings of the top-level* Controls *and* Functions *palettes.***

File Manager Tool

The File Manager tool, which you access by choosing **Project» File Manager…**, simplifies copying, renaming, and deleting files within VI libraries (LLBs). You also can use this tool to create new LLBs and directories and convert LLBs to and from directories, an important step if you need to manage your VIs with the Source Code Control tools.

To avoid performing a file operation on a VI already in memory, close all VIs that might be affected before using this tool.

In the File Manager dialog box, shown in the following figure, you can view two locations (directory or LLB) simultaneously. When you select a file, you can copy, rename, or delete it using the corresponding buttons between the two lists. Click **New…** to create a new directory or LLB.

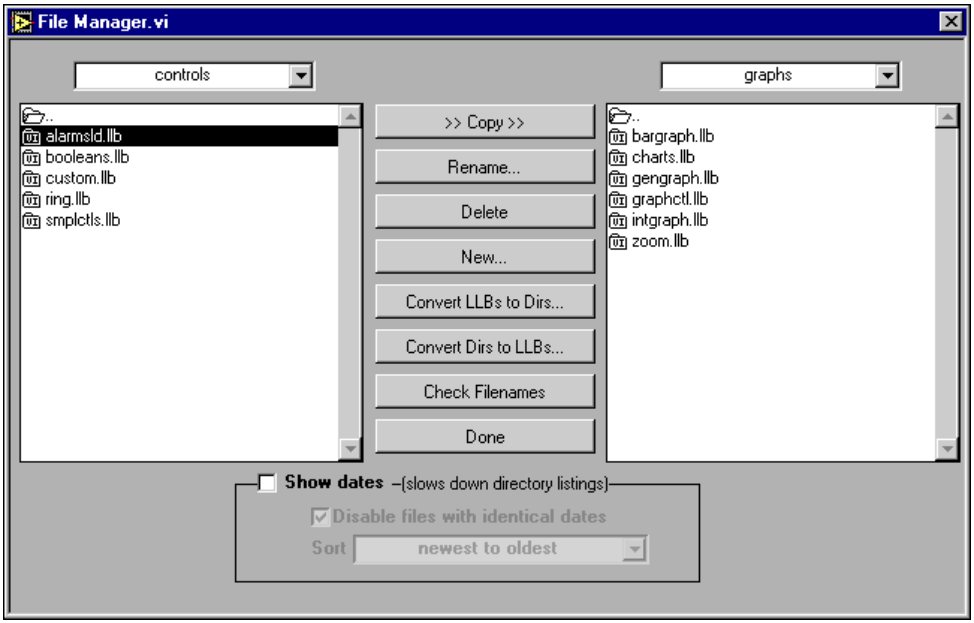


Figure 1. File Manager Tool Dialog Box

If you select an LLB, you can click **Convert LLBs to Dirs** to convert it to a directory. If you select a directory and click this button, the tool scans for all LLBs within that directory and gives you the option to convert them to directories. The new directory is created in the same location as the original LLB.

If you assign the new directory a name that differs from that of the original LLB, LabVIEW searches for the files that were within the LLB when calling a VI (even when the name is the same minus the `.llb` extension). When you convert an LLB to a directory, you have the option to back up the LLB (the `.llb` extension changes to `.llx`).

To convert a directory to an LLB, select a directory and click **Convert Dirs to LLBs**.

Click **Check Filenames** to scan a directory or VI library for platform-dependent filenames. The tool scans all filenames for invalid characters (`:`, `\`, `/`, `?`, `*`, `<`, `>`, or `|`) and verifies filenames to be 31 characters or less (a limitation on the Macintosh). The **Check Filenames** option also scans files within LLBs. These files are portable, even if their names contain characters that are invalid on some platforms. By scanning within LLBs, this tool helps you detect potential problems if you move your files out of VI libraries.

Use the **Show dates** option at the bottom of the dialog box to display file modification dates next to each file. You can choose to sort the files alphabetically or by date and disable files with the same name and date in both directory listings. Use this technique when comparing two directories to determine whether any files have changed.

Other General Interface Features

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.

Print to RTF/HTML Feature—The Print to RTF/HTML feature can export graphics in uncompressed graphics interchange format (GIF).

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.

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. In the Professional Developers Version, the option is selected by default. This setting specifies that if the source-code control uses file permissions to lock checked-in files, you cannot edit the files until you check them out. 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

Execution System Selection—The default preferred execution system for a VI is **same as caller**. This setting allows the VI to run in the same execution system in which caller is running when the subVI call to the VI is made. The **same as caller** setting has the lowest run-time overhead. When you set a VI to **same as caller** and you run it at the top level, it runs in the **standard** execution system at its selected priority.

Icon Editor—The **Undo** button has been removed from the Icon Editor, but you can undo an action by selecting **Edit»Undo** or `<Ctrl-Z>`.

Seconds to Date/Time Function Change for Windows—
In Windows, 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.

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

Support for Template VIs and Controls

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

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

(Macintosh) You also can use the **Stationery Pad** checkbox of the Get Info dialog box in the Finder to change a VI to a template.

Adding VIs to the Project and Help Menus

You can add VIs to the **Project** or **Help** menus by placing them inside the `Project` or `Help` directories in the `LabVIEW` directory. You can use this technique to provide quick access to VIs that act as tools in your system. National Instruments uses this feature to make the Tech Support VIs accessible from the **Help** menu. Also, if you have the Application Builder libraries installed, you can see a **Create Distribution Kit** option in the **Project** menu.

Any VI placed at the top level of the `Project` or `Help` directory is appended directly to the corresponding menu. If you create a subdirectory, a submenu is appended. If you place a library (`.llb`) inside of one of these directories, only VIs that are marked as top level are appended to the menu bar.

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 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 the 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

Allocation of Threads on Concurrent PowerMAX and Solaris 2

On Concurrent PowerMAX and Solaris 2, LabVIEW allocates threads in the following two ways.

- If LabVIEW has permission to increase its Light Weight Process (LWP) priorities from the default, it binds all created threads to LWPs.
  - Profiling is very accurate because each thread is bound to a LWP and the kernel monitors the execution timing of LWPs.
  - The LabVIEW priority system is reflected in the way the kernel runs LWPs. Higher-priority execution threads (LWPs) take over the system, not allowing lower-priority system tasks to execute.
  - Switching between threads might require more time because LabVIEW runs through the system scheduler.
- If LabVIEW cannot increase its LWP priorities from the default, it creates a LWP per thread, but leaves the threads and LWPs unbound so the created threads have a pool of LWPs on which to run. The typical user does not have permission to raise LWP priorities. If LabVIEW threads are not bound to LWPs:
  - Profiling strictly uses wall-clock time. Threads might switch LWPs dynamically without kernel knowledge, so LabVIEW cannot use LWP timing statistics.
  - The LabVIEW priority system only has an effect internal to LabVIEW. The system treats all of the LabVIEW LWPs as another process to schedule at the same priority as any other task in the system.
  - Context switching between threads might be faster because it does not involve the kernel scheduler.

The About LabVIEW dialog box, which you can view by choosing **Help»About LabVIEW…**, indicates how LabVIEW currently allocates threads.

Clarifications to the LabVIEW User Manual

The following clarifications pertain to the *LabVIEW User Manual*:

- In Chapter 2, *Creating VIs*, the text and an illustration refer to an **Undo** button in the Icon Editor. The **Undo** button has been removed, but you can undo an action by selecting **Edit»Undo** or <Ctrl-Z>.
- In Activity 6-3, *String Subsets and Number Extraction*, the block diagram shows the From Exponential/Fract/Eng function. The block diagram should show the Scan From String function, as described in the text.
- In Activity 15-3, *Calculate Harmonic Distortion*, the pathname for library that includes the THD Example VI should be `examples\analysis\measure\measxmpl.ll`.
- **(Windows 95)** In the *Using NetDDE* section of Chapter 23, *Using DDE*, the manual refers to `REGEDIT` and `REDEGIT` executables. The correct name is `REGEDIT`.
- In Chapter 25, *Program-to-Program Communication*, the *PPC Client Example* section refers to the PPC Open Connection, PPC Open Session, PPC Close Session, and PPC Close Connection VIs. These should be the PPC Open Port, PPC Start Session, PPC End Session, and PPC Close Port VIs, respectively. The *PPC Server Example* section refers to the PPC Close Session VI, which should be the PPC End Session VI.

VISA Error Codes

The following table lists numeric VISA error codes that were not included in the printed documentation.

Error Code	Error Name	Description
-1073807333	VI_ERROR_INV_DEGREE	Specified degree is invalid.
-1073807328	VI_ERROR_INV_LOCK_TYPE	The specified type of lock is not supported by this resource.
-1073807327	VI_ERROR_INV_ACCESS_KEY	The access key to the specified resource is invalid.
-1073807312	VI_ERROR_ABORT	User abort occurred during transfer.
-1073807301	VI_ERROR_QUEUE_ERROR	Unable to queue the asynchronous operation.
-1073807254	VI_ERROR_ASRL_PARITY	A parity error occurred during transfer.
-1073807253	VI_ERROR_ASRL_FRAMING	A framing error occurred during transfer.

Error Code	Error Name	Description
-1073807252	VI_ERROR_ASRL_OVERRUN	An overrun error occurred during transfer. A character was not read from the hardware before the next character arrived.
-1073807240	VI_ERROR_INV_PARAMETER	The value of some parameter (which parameter is not known) is invalid.
-1073807229	VI_ERROR_INV_LENGTH	Invalid length specified.
-1073807204	VI_ERROR_SESN_NLOCKED	The current session did not have a lock on the resource.
1073676290	VI_SUCCESS_EVENT_EN	Specified event is already enabled for at least one of the specified mechanisms.
1073676291	VI_SUCCESS_EVENT_DIS	Specified event is already disabled for at least one of the specified mechanisms.
1073676292	VI_SUCCESS_QUEUE_EMPTY	Operation completed successfully, but queue was already empty.
1073676293	VI_SUCCESS_TERM_CHAR	The specified termination character was read.
1073676294	VI_SUCCESS_MAX_CNT	The number of bytes transferred is equal to the input count.
1073676416	VI_SUCCESS_QUEUE_NEMPTY	Wait terminated successfully on receipt of an event notification. There is at least one more event occurrence of the type specified by inEventType available for this session.
1073676420	VI_WARN_NSUP_ATTR_STATE	Although the specified state of the attribute is valid, it is not supported by this resource implementation.
1073676421	VI_WARN_UNKNOWN_STATUS	The status code passed to the operation could not be interpreted.
1073676424	VI_WARN_NSUP_BUF	The specified I/O buffer is not supported.
1073676441	VI_SUCCESS_NESTED_SHARED	Operation completed successfully, and this session has nested shared locks.
1073676442	VI_SUCCESS_NESTED_EXCLUSIVE	Operation completed successfully, and this session has nested exclusive locks.
1073676443	VI_SUCCESS_SYNC	Operation completed successfully, but the operation was actually synchronous rather than asynchronous.
-1073807278	VI_ERROR_INV_WIDTH	Invalid access width specified.
-1073807275	VI_ERROR_NSUP_VAR_WIDTH	Cannot support source and destination widths that are different.
-1073807248	VI_ERROR_NSUP_ALIGN_OFFSET	The specified offset is not properly aligned for the access width of the operation.
-1073807247	VI_ERROR_USER_BUF	A specified user buffer is not valid or cannot be accessed for the required size.

Error Code	Error Name	Description
-1073807246	VI_ERROR_RSRC_BUSY	The resource is valid, but VISA cannot currently access it.
1073676407	VI_WARN_CONFIG_NLOADED	The specified configuration either does not exist or could not be loaded. VISA-specified defaults will be used.
1073676413	VI_SUCCESS_DEV_NPRESENT	Session opened successfully, but the device at the specified address is not responding.
1073676418	VI_WARN_NULL_OBJECT	The specified object reference is uninitialized.
-1073807313	VI_ERROR_NENABLED	You must be enabled for events of the specified type in order to receive them.
-1073807202	VI_ERROR_LIBRARY_NFOUND	A code library required by VISA could not be located or loaded.
-1073807271	VI_ERROR_RESP_PENDING	A previous response is still pending, causing a multiple query error.
-1073807263	VI_ERROR_NSYS_CNTLR	The interface associated with this session is not the system controller.
1073676440	VI_SUCCESS_NCHAIN	Event handled successfully. Do not invoke any other handlers on this session for this event.

Using the Application Builder Libraries

For more information about LabVIEW features refer to Chapter 27, *Managing Your Applications*, in the *G Programming Reference Manual*. This chapter contains tips for managing the source files of multiple developers and describes how to use the VI History item.

Required System Configuration

Applications you create with the Application Builder Libraries have the same approximate requirements as the development system. Memory requirements depend on the size of your application. Typically, applications require about the same amount of memory it takes to run your VIs in the development system.

LabVIEW applications use a directory for storing temporary files. Some of the temporary files are large and it is best if several megabytes of disk space are available for this temporary directory. The default temporary directory is `\tmp`. You can change the temporary directory by selecting **Edit»Preferences...**.

It is best if the workstation has 32 MB of RAM, with 32 MB or more of swap space storage. It is possible for the Application Builder to run on less than 24 MB of RAM, but performance suffers.

Standard Features

LabVIEW applications feature a simplified user interface that permits only the operation of VIs. The menus do not contain editing options. For example, the **Save** command and the **Functions** and **Controls** palettes are not present.

Menus display items related to VI operation. Because you cannot edit the VI, pop-up menus are short—displaying the same items the development system displays when a VI is running.

(Windows and UNIX) Users access a pop-up menu by clicking a control or indicator with the right mouse button.

(Macintosh) Users access a pop-up menu by holding down the command key and clicking a control or indicator.

The items available to the user include the following:

- Operate controls and change their values.
- Interact with strip chart and graph indicators.
- Change the scale limits.
- Set controls, indicators, and array elements to default values.
- Use the pop-up menu of a control or indicator to cut, copy, or paste data from a control or indicator to another control.
- Use the pop-up menu of a control or indicator to view the description of the item, and perform additional run-time operations, such as showing the control palette of the graph.
- Use any execution palette button the developer has not disabled.
- Log and print the front panel.
- View the **Show VI Info...** information for a VI.
- Use the Help window to see descriptions of controls and indicators.

Customizable Features

When you build a run-time application, you can customize the following items:

- Do you want to embed a VI library in the application?

  If you choose to embed a VI library in the application, the library and a LabVIEW run-time engine become a single file. When you launch the file, it automatically opens all top-level VIs in the library. If you do not embed a VI library, when you launch the application you can use it to open any VI, assuming the VI was saved with a development system for that platform.

  By embedding a VI library, you can create a complete stand-alone application, one that prevents the user, or customer, from accessing the source VIs—even if the user has the development system. The advantage to not embedding a VI library is you can use the same run-time engine for multiple sets of VIs. This reduces the disk space usage for a customer running multiple VIs.

  Additionally, you can use a combination of these two solutions. If you embed VIs within a library, they can still call subVIs outside of the application. You might want to do this when you have a set of VIs common to two applications, a set of VIs that require upgrading after the user receives the application, or a large number of VIs to call (to keep the base size of the application down). One disadvantage of not embedding every VI is the subVIs can be used in another development system, because the users can view the diagrams.

- **(Windows)** Do you want the application to be an ActiveX server?

  If you enable the **ActiveX Server Support** option, your application responds to requests from ActiveX clients. The functionality of the ActiveX server in your application is a subset of the LabVIEW ActiveX server. When you build an application `myapp.exe`, an ActiveX type library `myapp.tlb` is also created along with the executable. The type library defines a createable class, *Application*, and a dispatch class, *Virtual Instrument*, and exports the properties and methods for these classes. You can find the Help for these properties and methods in `lvcomm.hlp` in the LabVIEW `Help` directory. When you distribute the application make sure the type library and the help file are located with the executable.

  When you assign the name of the application to the **ProgID Prefix** your application is uniquely identified in the system registry. Once you build the application, you should run it at least once to enable registry with the system. After the application is registered, ActiveX clients access the server objects using ProgIDs. For example, if you specify the **ProgID Prefix** as `myapp`, clients instantiate an application object using the progID `myapp.application`.

- Do you want a customized **About...** dialog box?

  When you build an application, you can supply an About VI that runs when the user selects **Help**»**About…** from the **Help** menu. When you do not supply an About VI, the application has a basic, default About dialog box. For more information, refer to the *Create and Save an About VI* section in this document.

How to Build an Application

This section describes how to build an application. If you want your top-level VIs to run when opened, select **Run When Opened** from the VI Setup dialog of the top-level VI. If you want all of your VIs embedded within the application, save your application VIs into a single VI library by selecting **File»Save with Options**.

If you click the **Application Distribution** item, your program prompts you to select the VI library or directory where you want to save the hierarchy. Enter the name of a new library that you want to use to build the application. This selection automatically saves the VIs without their diagrams and includes any external subroutines and run-time menus referenced by the VIs in the VI library.

Next, choose the **Project**»**Build Application...** item to build the application. If you choose to embed a VI Library, and no VIs are marked as Top-Level, the **Build Application...** item brings up the **Edit VI Library** dialog. The VIs you mark with the Top-Level option open when the application is launched. The following paragraphs describe these steps in greater detail.

Besides saving all the VIs necessary for your application in a VI library, you can also save a VI as an About VI, so users can view information about your application (such as the full name, version number, company name, copyright information, and so on). More information on the About VI is in the next section of this document, *Create and Save an About VI*.

Create and Save an About VI (Optional)

Most applications have an About dialog box that displays information about the application and the user or company that designed it. You can create an About VI that LabVIEW executes when a user selects **Help**»**About…**. You can have only one About VI per application, and you can only have one if you embed a VI Library. If you do not supply an About VI, LabVIEW displays a default dialog box like the one displayed in the LabVIEW development system.

To create your own About VI, create a VI and save it so that its name begins with the word *About* (the first letter must be capitalized, with the subsequent letters in lowercase). When the application is launched it looks for a VI beginning with the word *About*.

If the user selects the **About…** menu item and an About VI is installed, that VI runs. When it finishes execution, LabVIEW closes it automatically.

The About VI you create can share subVIs with your application VIs. However, your About VI cannot be a subVI in an application VI because the About VI cannot run while an application VI is running.

☞ **Note** ***Your About VI must contain a message indicating that your application was created using LabVIEW from National Instruments. Please read the*** *Distribution Rights* ***section in the*** *LabVIEW Software License Agreement* ***for the copyright notice you must use to legally distribute your applications.***

Save VIs that Use VI Server Functions

VI Server functions dynamically load into applications; therefore, they are not part of the hierarchy of the top-level VIs and require a different procedure for saving. If you build an application that contains VI Server functions, use one of the following methods for saving:

- The easiest method is to save all dynamically loaded VIs into a single library:
  - Build a dummy VI whose diagram contains the top-level application VI and all the dynamic top-level VIs. When you save this VI with the options for Application Distribution, it saves all the necessary VIs as well. Delete the dummy VI from the application library at this point, if you choose. The paths to the dynamic VIs might change and it might be necessary to modify your code to account for this change.
  - Select **File»Save With Options...»Application Distribution...** and save all dynamically loaded VIs to a new library, as you normally do. Then, for each VI that is dynamically loaded, repeat this process, but save to the same library as your main application. This includes the Hierarchies of your dynamically called applications.

  From a file management standpoint, this method can create a large executable file. From a memory standpoint, you still benefit from dynamic loading because the entire executable file does not load into memory.

- The second method is to embed only your main application within the executable and dynamically call external applications as separate files (VIs and/or LLBs). If you use VI Server functions, you then must specify the path to the dynamically called applications by using the function Current VIs Path, followed by a Strip Path.

Use the Build Application Item

Select the **Project»Build Application...** menu item to create an executable.

If you want to embed a VI library, click the **embed** menu item. A dialog box appears prompting you to select which VI library you want to use to build an application. Select the VI library you created in the first step of this section.

In addition, on the Windows platform, you can choose whether the application acts as an ActiveX server.

When you finish making selections, click **OK**. If you embed a VI library and no VIs in the library are marked with the **Top Level** item, LabVIEW displays the **Edit VI Library...** dialog box so you can select which VIs open at launch time. Use the **Top Level** item to mark the VIs that you want to open when you launch the application. Most applications consist of a single top-level VI that calls other VIs. However, you can create applications that consist of multiple top-level VIs that open when you launch an application.

After the prompt, enter a name and destination for the application. The build process can take a few minutes for very large applications.

Application Building Example

Complete the following steps to explore application building using your LabVIEW development system:

1. Open the Sample VI, located in `examples\appbuild.llb`. This VI calls some Analysis VIs, including the Histogram VI, which call external subroutines. Items in the **VI Setup…** dialog box are currently configured to hide a number of the attributes of the window.
2. Run the Sample VI to see its behavior. When you are finished, click the **STOP** button. Do not click the **QUIT** button unless you want to quit LabVIEW.
3. Choose **File»Save with Options...»Application Distribution»Save**. When prompted, enter the name `sample.llb` and then click **Select**.

4. Examine the About Sample VI, which is also in `examples\appbuild.llb`. This VI serves as the About dialog box for this application. When this VI executes, it acts similarly to a dialog box, in that it prevents you from interacting with other windows while it is open.
5. In the **edit** mode, select **VI Setup...** in the pop-up menu of the icon pane of the Sample VI. Then select the **Run When Opened**»**OK**.
6. Save a copy of the About Sample VI into `examples\sample.llb` and click **OK**.
7. Now you can build the application. Select **File»Build Application…**. Click the **Embed VI Library** button and choose `sample.llb` and click **OK**. LabVIEW then prompts you to mark which VIs you want opened when the application launches. Choose the Sample VI from the displayed list, and select **Top-Level**»**OK**. A dialog box prompts you for a destination and name for the application you want to build. Move upward in the file hierarchy to the top level of your LabVIEW directory, and name the application **(Windows)** `sample.exe` or **(Macintosh** and **UNIX)** `sample`.

☞ **Note** **(Windows and UNIX)** ***If you do not build the application at the top-level of the LabVIEW directory, you must place several files in the same directory as the application. These files communicate with hardware. See the*** *Additional Files Required by Applications* ***section of this document for a list of the files to be stored with your application.***

8. Quit LabVIEW, and run the **(Windows)** `sample.exe` or **(Macintosh and UNIX)** `sample` application. It launches and then automatically opens and runs the Sample VI. Look at the menu items that are now available. Select **(Windows, Macintosh, and UNIX) Help»About…** or **(Macintosh) Apple»About…**. When you finish, click the **QUIT** button on the front panel of the application.
9. In practice, you might want to completely remove the **STOP** button from the front panel. If the top-level VI stops, the application does not automatically quit, because that might not be the desired behavior. It is best to structure most applications so that the **Abort** item is disabled on the top-level VI, and the VI has a **Quit** item that calls the Quit LabVIEW function, located in **Functions**»**Advanced**.

Distributing Your Applications

The following sections describe some relevant issues concerning the distribution of LabVIEW applications.

Additional Files Required by Applications

It might be necessary to distribute additional files with your application. If required for execution or other operations, these files must be placed in the same directory as the application.

If your application uses serial port or data acquisition functionality include the `serpdrv` or `daqdrv` files. If your application uses a GPIB or data acquisition board the user must install the hardware drivers that come with their boards.

(Windows) If you created an ActiveX type library for your application, the type library and help file must be included.

(Windows) 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.

(Windows) LabVIEW requires two system DLLs that might not be installed on all systems. These are `msvcrt` and `mfc42.dll`. The file `msvcrt.dll` is always required and `mfc42.dll` is required only if the application uses the ActiveX container object. If they are not already present, these files are placed in your Windows 95 `system` directory (`system32` directory on the NT platform) by the LabVIEW installer. You must make sure these files exist on the computer which runs the application. They can be placed in either the Windows 95 `system` directory (`system32` directory on the NT platform) or in the same directory as the application. If you put them in the Windows directory take care not to overwrite newer versions of these DLLs that might have been installed by other applications.

(Macintosh) If you are building an application for the Power Macintosh that uses analysis routines or Program to Program Communication (PPC) VIs, put a copy of the `Shared Libraries` folder in the folder that contains your application.

Distribution Rights

Refer to the *LabVIEW Software License Agreement* in your software package for information on the distribution rights for your platform.

Packaging Your Files for Distribution

A LabVIEW application can be large. A core run-time system is smaller than the development version because it does not require the compiler or the editor. In addition, the run-time system saves the VIs without diagrams, which usually account for at least half of the size of your VIs. Even so, most run-time applications do not fit on a single floppy disk. To distribute your applications, you can put them on a larger capacity medium, such as a CD or magnetic tape, or you can compress the applications and create an installer for them.

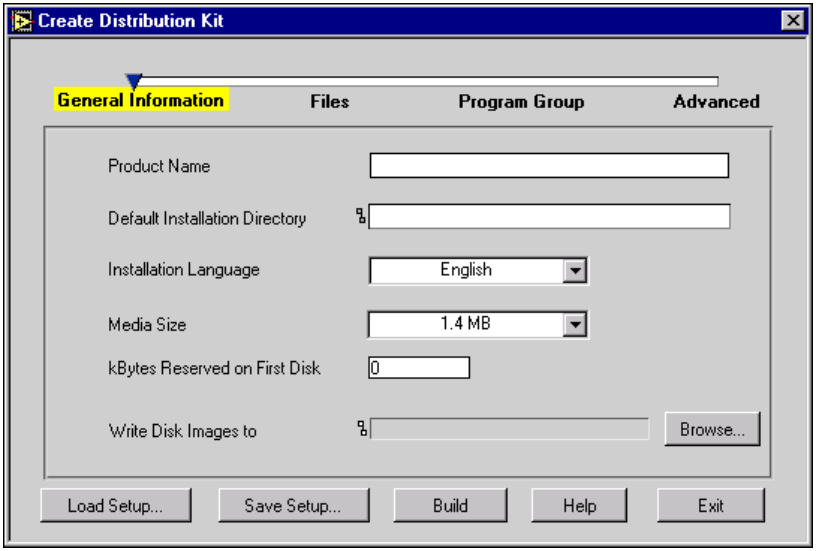
(Windows) The Windows version of the Application Builder Libraries adds a **Create Distribution Kit** item to the **Project** menu. This command makes it simple to create professional installers for your LabVIEW applications.

(Macintosh) There are two shareware compression utilities, Compact Pro and Stuffit, widely available from online services. You can use these compression utilities to create self-extracting archives, which decompress themselves when you double-click them.

(UNIX) You can use the UNIX `tar` command to group the application files into a single file for distribution.

Create Distribution Kit

(Windows) When you select Project»Create Distribution Kit the following dialog box appears.



To create an installer, enter the information in this dialog box, then click the **Build** button at the bottom. This command compresses the files, builds an installer, and writes the disk images to the hard drive path specified in **Write Disk Images to**. You can run the installer image from that directory or copy it to floppies or other distribution media.

If you are not sure of the meaning of a field or button, click the **Help** button at the bottom of the dialog to bring up the Help window in LabVIEW. You can then put your cursor over any field or button, and a description of the field or button appears in the Help window.

You can use the **Save Setup** item to save the information that you enter in this dialog box. Subsequently, you can use the **Load Setup** item to retrieve information.

The **Create Distribution Kit** item has four dialog boxes: **General Information**, **Files**, **Program Group**, and **Advanced**. To switch between these dialog boxes, click the name of the box you want or drag the slider.

General Information

You use the General Information dialog box to describe appearance and default items of the installer, and to specify a location on your drive to write the installer images.

The **Product Name** item is used as part of a banner at the top of the installer dialog and in various prompts for the user.

The **Default Installation Directory** item is used in the installer as the default location for installed files. The user can change the path in the installer program.

The **Installation Language** item lets you select the language that is used for messages in the resulting installer. You can select from Danish, Dutch, English, Finnish, French, German, Italian, Japanese, Norwegian, Portuguese, Spanish, and Swedish.

The **Media Size** ring item lets you specify how the file is to be segmented—for 720 KB, 1.2 MB, or 1.4 MB floppies. Even if you plan to distribute the files by CD, it is necessary to segment them. However, if you want to run the installer from a CD or from your drive, you can place all of the files in the same directory and run the setup program from that directory.

The **kBytes Reserved on First Disk** item lets you reserve space on the first disk. You might reserve space on the first disk if you want to put a readme file on the first floppy.

The **Write Disk Images to** item lets you specify where to write the disk images on your computer. The **Create Distribution Kit** item creates a setup program in that directory as well as files named `data.001`, `data.002`, and so on. If you plan to put the disk images on floppy, it is best to copy the `setup` and `data.001` files to the first floppy, copy the `data.002` file to the second floppy, and so on.

Files

You use the Files dialog box to specify the files that you want to install. Specify the files as a set of file groups. A file group can contain one or more files from several possible source locations on your computer. The application installs every file in a group to the same directory and shares the same overwrite setting. For example, you can choose whether the files overwrite existing files of the same name without a prompt, with a prompt, only if the file being installed is newer, or never overwrite.

Press the **Add Group** button to define a group. You are prompted for a name for the group. This name is never seen by the user; it is only used in the **Create Distribution Kit** item to discriminate between groups. You are then prompted to select files. You can add as many files as you want. Click the **Add File** button to add a single file, or click **Add Directory Contents** to add every file in a directory, not including subdirectories.

Once you define a group, it shows up in the **File Groups** listbox item. When you select a group, you can specify the following settings for that group.

- The **Group Destination** item lets you specify whether you want the files in the selected group to install in the `Installation` directory or in the `Windows` directory. If you want to install the files in a subdirectory of the `Installation` directory or `Windows` directory, specify a relative path in the **Relative Path** item.
- The **Replace Existing Files** item lets you specify what happens if a file of the same name as one of the source files is found in the destination directory. You can choose to always replace the file, ask the user, replace the file automatically if the source file is newer than the destination file, or never replace the file.

You can also edit the contents of a group using **Edit Group** or delete the contents using **Delete Group**.

Program Group

You can use the Program Group dialog box to create a program group for your installation. Under Windows NT, a program group appears as an icon/window in the program manager. Under Windows 95, a program group shows up in **Start»Programs** as a submenu. The program group

contains icons for the files you select, serving as shortcuts that make it easy to start the program regardless of its location on your computer.

Regardless of whether you plan to add items to a program group, it is best to specify a program group name in the **Program Group** field. Even if you don't choose to install any files in a program group, an uninstall icon is added to the group under Windows NT (under Windows 95, you use the **Add/Remove Programs** control panel to uninstall programs).

Click **Add File** to add a file to the program group. A dialog box appears showing you the currently defined file groups. If you select a file group, you can then select a file within that group.

Advanced

The Advanced dialog box contains items that let you customize the behavior of the installer. These items are intended for fairly isolated applications, so typically it is not necessary to use them.

Select the **Use Custom Script** item if you want to change to look of the installer. The LabVIEW and LabWindows Create Distribution Kit installers are based upon licensed components from Helpful Programs Incorporated (HPI). **Create Distribution Kit** uses a file called `template.inf`, which is written in the HPI InstallIt scripting language and describes the dialogs that appear during the installation. To customize the look of the installer, make a copy of this file, make the necessary modifications, and then specify the location of the modified script using this item.

However, because the template script is written in a fairly flexible but somewhat complex scripting language, using this item is not recommended. National Instruments has some technical information available concerning a few common modifications you might make to an installer script (many of them are LabWindows/CVI technical notes, because both LabWindows/CVI and LabVIEW use the same licensed components for the **Create Distribution Kit** items). Call National Instruments to order these technical notes. If you plan to customize the look of the installer beyond the scope of those very simple modifications, it is necessary to buy the InstallIt package from HPI in order to get documentation and support for questions related to the scripting language.

Select the **Run Executable After Installation** item if you want to run a program after the installation completes. Additionally, you can use this item to run a program that finishes the installation. For example, you might write a DOS batch program or a C program that modifies a `.ini` file or a `registry` file. Install the file as part of your installation and then run it afterwards to make the necessary modifications. The file that you run must be one of the files that you install.

If you choose to run an executable after the installation completes, you can use the **Command Line Arguments** to specify arguments passed to the program. In addition to specifying standard arguments, you can embed any of the following items in the command line argument string:

`%dest`	The application installation directory chosen by the user
`%src`	The directory that contains `setup.exe`
`%group`	The installation program group name
`%name`	The installation name

If any of these options are present at installation time, they are replaced with the proper values before the arguments are passed to the executable.

Additional Notes for Building Applications

The following sections contain some additional information that might be useful in setting up your applications.

Setting Preferences

Your application has a Preferences dialog box. If you make changes, preference information for your application are written to **(Windows and Macintosh)** a preferences file or **(UNIX)** the `.labviewrc` file. The format for this information is the same as for LabVIEW; for more information see Chapter 7, *Customizing Your Environment*, in the *G Programming Reference Manual*.

(Windows) The only difference is the file name, which is the application name instead of `labview`. For example, if your application is named `simple.exe`, the preferences are stored in `simple.ini`. Remember to include your preference file (`.ini`) with your application.

(Macintosh) The only difference is the file name, which is the application name instead of `labview`. For example, if your application is named `Simple`, the preferences are stored in `Simple Preferences`. Remember to include your preference file with your application.

(UNIX) To create a preferences file for your stand-alone application, you must create a `.labviewrc` file on the target machine and place it inside your home directory. Inside this file you create preferences using the following format: `<application file name>.<preference>`. For example, the `.labviewrc` file for LabVIEW has its preference set as `labview.<preference>`. An easy way to create a new executable

specific preference is to set your preferences in LabVIEW and then place a copy of the `.labviewrc` file in your home directory of the target computer. All the user has to do at this point is change all the `labview` prefixes to the executable file name. The preferences file name, however, is still called `.labviewrc`.

Using the VI Setup... Option to Limit VI Options

When you design an application, consider which user options you might want. For example, with the LabVIEW Development System, it is convenient to have an **Abort** button so users can easily test and halt VIs. Because this button aborts the program immediately—sometimes in the middle of input and output of sensitive data—you probably do not want the user to halt the VI by this method. Instead, use a front panel control to stop the program synchronously.

You can use the **VI Setup** item in your VIs to make execution operations—such as the **Abort** button—available to the user. If the VI is to be used as a subVI, you can use the **SubVI Node Setup...** command to specialize operation of the subVI, such as configuring its front panel to open when the subVI is called.

You might want to disable the following three items: **Abort**, **Close Box**, and **Free Run**. You also might want to hide all of the buttons, and then configure the top-level VI to run automatically when the VI is loaded. If you disable the Abort button, verify that your program does not accidentally run in an infinite loop.

You can disable the run-time pop-up menu for your VIs, so that users cannot set controls to default values or turn on autoscaling in graphs.

Additionally, you can customize your panels. For example, you can hide scrollbars or make specific VIs function like dialog boxes. When you choose **VI Setup...**»**Dialog Box**, the panel is modal, which means the user cannot interact with other panels while the panel is active.

Providing Help Information

As a VI developer, it is best to document your VIs for other users, including all information necessary to load and operate the VIs. The regular LabVIEW documentation set is copyrighted material. Do not ship this documentation with the applications you create.

To create online help, you might want to enter information in the description field of the **Show VI Info...** dialog box for each panel. You might also want to place information in the Description dialog box for controls and indicators. When the user opens the Help window and

moves the cursor over indicators, the Help window displays description information.

Professional Development Tools

The LabVIEW Professional Development system includes tools that also are distributed separately as part of the Professional G Developers toolkit. These tools are described in the *Professional G Developers Toolkit Reference Manual*, which is included with the Professional Development system.

This section is a supplement to *Professional G Developers Toolkit Reference Manual* and provides information on the enhancements to the Professional Developers tools in LabVIEW 5.0.

VI Comparison

You can use the VI comparison tools to manage different versions of VIs as you develop large applications. When you make changes to a VI, you might want to compare the VI to an older version to verify the changes you have made. Also, when multiple users are working on the same VIs, you might need to view two versions of the same files to merge changes made by different users.

LabVIEW provides several tools for comparing VIs, which are described in the following sections.

- **Compare Hierarchy**—compares two different versions of the same hierarchy of VIs
- **Compare VIs**—compares two VIs
- **Source Code Control»Compare Files**—compares the local versions of files with the versions under Source Code Control

Compare Hierarchies

You can use the **Compare Hierarchy** command to compare two hierarchies of VIs. Any file with the same name in both hierarchies is compared. After the comparison completes, **Compare Hierarchy** displays a summary of the differences. You can select a set of VIs that have differences and visually compare them using the Compare VIs tool.

To compare two VI hierarchies, select **Project»Compare VI Hierarchies…**. Use the Compare VI Hierarchies dialog box, shown in Figure 2, to select two hierarchies to compare.

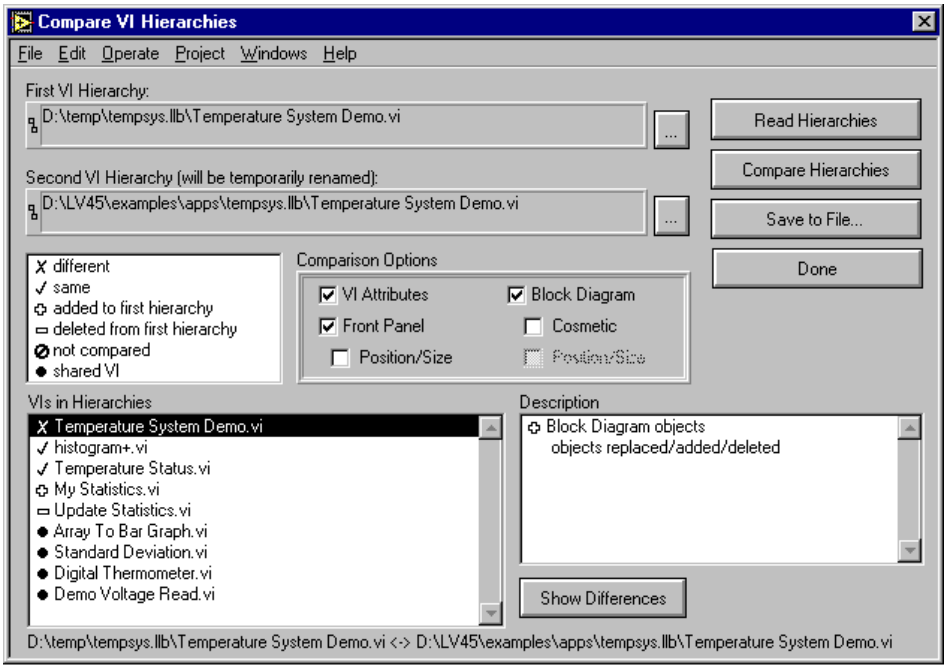


Figure 2. Compare VI Hierarchies

Select two VI hierarchies to compare by entering the path to the top-level VI. Use the **…** button to open a file dialog box to select a VI from the file system. You then can use the **Compare Hierarchies** button to compare VIs in the hierarchies. VIs with the same name but different paths are compared and categorized as being either the same or different. VIs with the same name and path are considered shared VIs. All other VIs are categorized as either added or deleted from the first VI hierarchy. The VIs and a symbol indicating how they are categorized are displayed in the **VIs in Hierarchies** listbox. As you select different VIs in the listbox, a more detailed description appears in the **Description** listbox. To view the differences on the screen, double-click an item in the listbox or use the **Show Differences** button.

The **Read Hierarchies** button categorizes the VIs in the hierarchy. VIs that need to be compared are categorized as `not compared`. You can then selectively compare individual VIs by double-clicking an item or using the **Show Differences** button.

To abort comparisons of large VI hierarchies, use the **Operate»Stop** menu item or the <Ctrl-.> (Control and the period) key combination.

Comparison Options

You selectively can find differences in the front panel and block diagram of a VI as well as VI attributes such as settings in VI Setup. For the block diagram, you can choose to ignore cosmetic differences. A *cosmetic change* is any change that does not affect the execution of the block diagram. Furthermore, you can choose to ignore the position and size changes in front panel and block diagram objects. Note that position and size changes include movement of an object from front to back and vice versa.

Show Differences

Differences are shown by tiling the front panels and block diagrams of the two VIs in the **Differences** window.

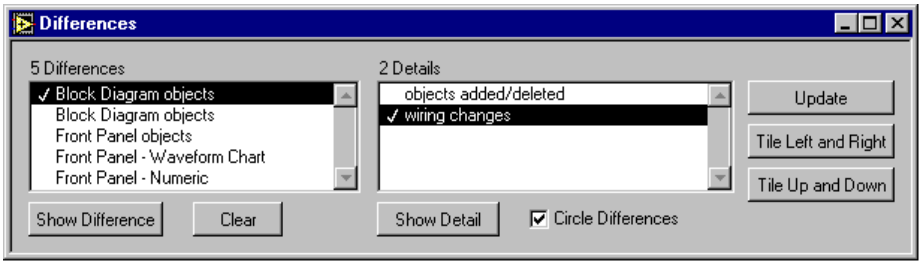


Figure 3. Differences Window

The **Differences** window includes a list of differences and details of the selected difference. To highlight a difference, double-click an item in the differences list, or select an item and click the **Show Difference** button. To highlight a detail, either double-click an item in the details list or select an item and click the **Show Detail** button. A check mark indicates the items you selected. You also can use the **Tile Left and Right** and the **Tile Up and Down** buttons to tile the windows of the two VIs you are comparing. Use the **Clear** button to clear the differences list. If you have made edits after a comparison, some of the differences might be stale; therefore, use the **Update** button to again compare the two VIs and update the differences list. You also can show the **Differences** window by selecting **Project»Show Differences**.

When you highlight a difference, objects that are part of the difference are selected. The **Circle Differences** checkbox option allows you to draw a red circle around the object(s) that has changed. An example of a block diagram difference is show below.

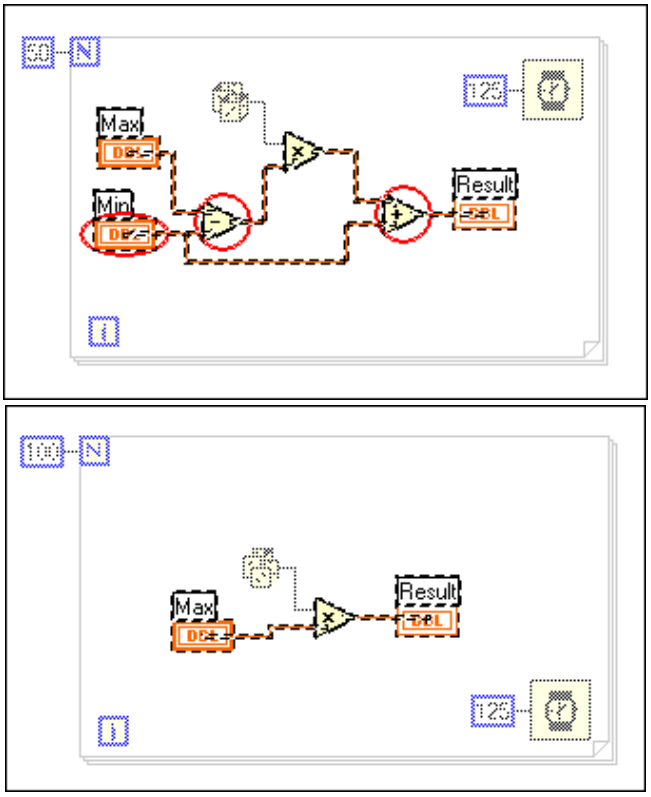


Figure 4. Block Diagram Difference

In Figure 4, the circled objects are inserted into the diagram. Objects selected but not circled are not differences. They are selected as *anchor objects* to provide reference points for the difference. Objects that appear dimmed are not part of the difference.

Compare VIs (Graphical Differencing)

When developing applications you might have multiple versions of the same VIs. The Compare VIs feature, also called Graphical Differencing, helps you track changes in your application by comparing multiple versions of a VI. This becomes especially important as your project grows and involves more developers.

You can use the **Project»Compare VIs** command to graphically compare two VIs. You can select options to control the types of differences you want to detect and view. For example, you can filter out cosmetic changes such as objects being moved or resized. When you compare the VIs, a dialog displays a summary of the differences. If you select an item from the

summary, **Compare VIs** displays and highlights the differences between the two VIs.

To compare two VIs, select **Project»Compare VIs…**. Use the following dialog box to select two VIs to compare.

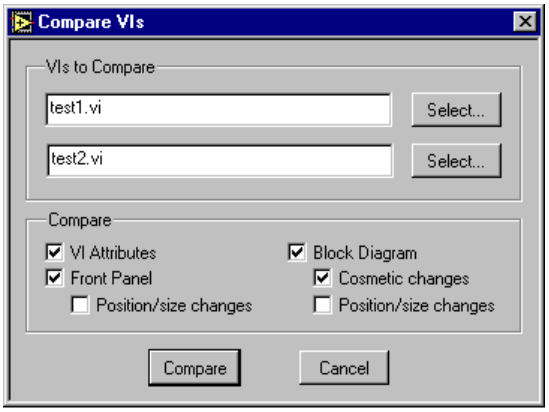


Figure 5. Compare VIs Dialog Box

The **Select…** button opens a dialog box to select a VI by name. You only can select VIs that are already loaded into memory.

Comparing very large VIs can be lengthy. You can cancel the comparison of two VIs through the **Comparison Progress** dialog box, shown in Figure 6.

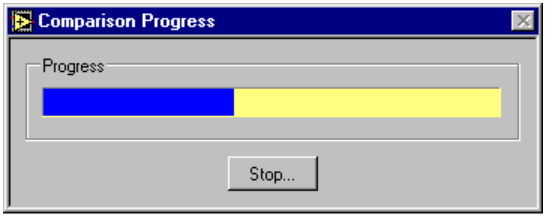


Figure 6. Comparison Progress Dialog Box

The progress bar indicates the steps in the comparison algorithm, not the number of differences left to find. When the comparison is complete, the front panels and block diagrams of the two VIs are displayed in the **Differences** window, as shown in Figure 3.

Comparison Issues

Because LabVIEW cannot load two VIs with the same name, you must rename your VIs to compare them. When the **Compare VI Hierarchies** tool compares two VIs, the first VI loads as is, and the second VI is moved to the temporary directory with a `cmp.` prefix. When using the **Compare VIs** tool you must rename the appropriate VI to load the two VIs into memory.

However, renaming the VIs does not affect the name of subVIs. Because the **Compare VI Hierarchies** tool does not rename the subVIs, *the renamed VI will link to the subVIs loaded by the first VI.*

Source Code Control»Compare Files

You can use the **Source Code Control»Compare Files…** command to compare files from projects under Source Code Control (SCC) with the local versions of those files.

To compare VIs under SCC, select **Project»Source Code Control»Compare Files…**. You can use the SCC Compare Files dialog box, shown in Figure 7, to compare the project files on the master directory with the project files on the local directory.

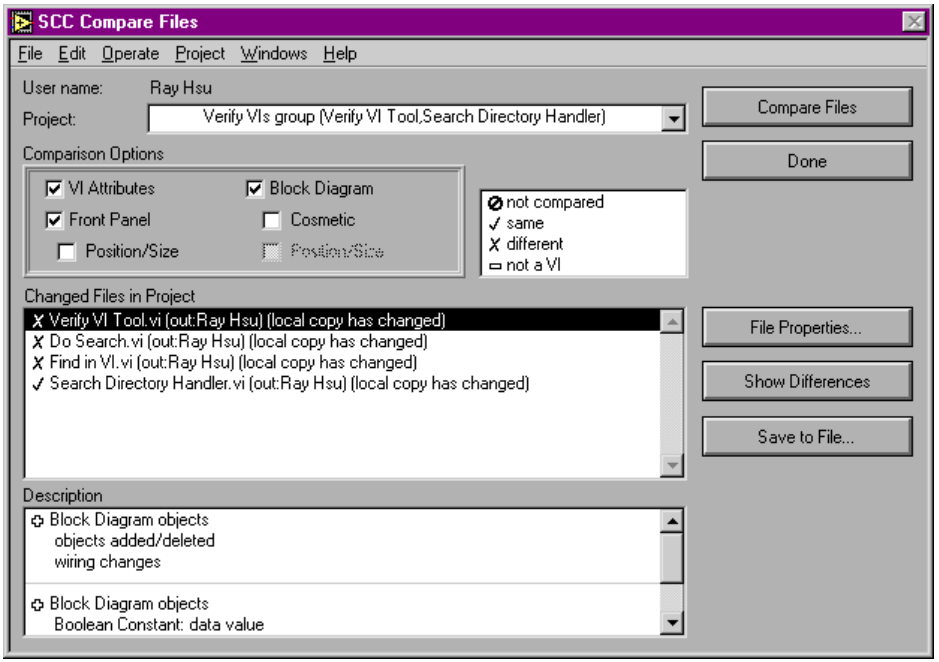


Figure 7. SCC Compare Files Dialog Box

After you select a project, the SCC Compare Files dialog box displays all changed files between the master copy and local copy. You then can use the **Compare Files** button to compare the changed files. Only VI files are compared. Other external files are not compared. This tool is similar to the **Project»Compare VI Hierarchies** tool. See the *Compare VIs (Graphical Differencing)* section for additional details.

Documentation Tools

☞ **Note** ***The* Project»Documentation Tool *described in this section replaces the Print Hierarchy tool described in the*** *Professional G Developers Toolkit Reference Manual.*

You can use the **File»Print Documentation** command to create documentation for a single VI, either by printing it directly to a printer or by creating source material for online help, web pages, or word processors.

You can use the **Project»Documentation Tool** command to create documentation for all of the VIs in your applications. The Documentation Tool dialog box is shown in Figure 8.

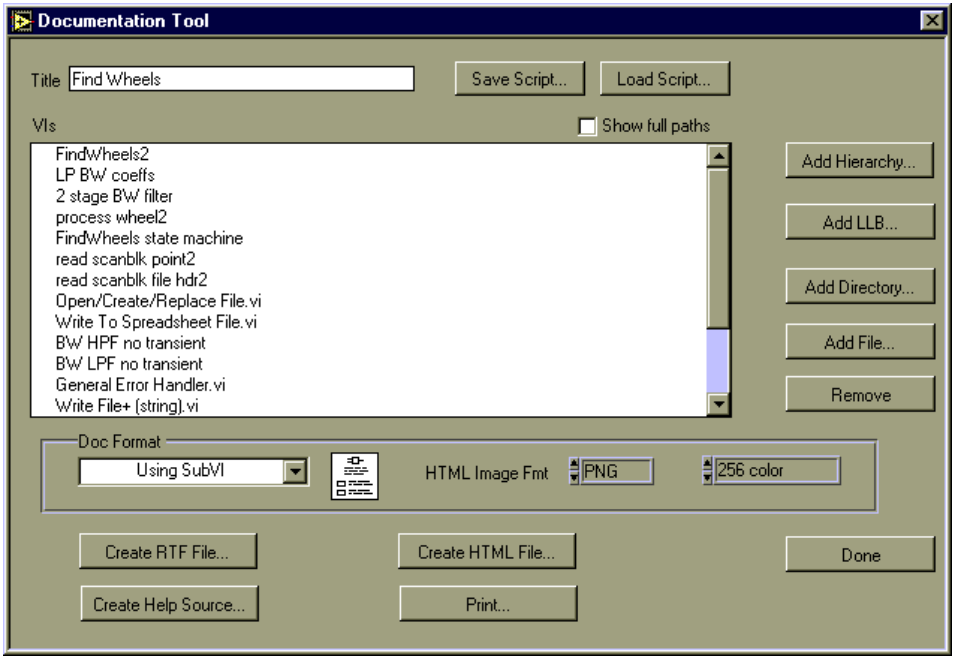


Figure 8. Documentation Tool Dialog Box

Use the **Add Hierarchy…**, **Add LLB…**, **Add Directory…**, and **Add File…** buttons on the right side of the dialog box to add VI hierarchies from memory, add all of the files in a directory or LLB, or add individual files from disk. The files are listed in the listbox in the dialog box after they are added. The order of the files controls the order that the VIs appear in the resulting documentation. You can reorder the VIs by selecting files from the list and dragging them to the desired location. If the paths are too long to view, you can deselect the **Show Full Paths** checkbox at the top of the dialog box.

When you have the files in the desired order, select one of the following buttons to create the documentation:

- **HTML**—With HTML, you can make your documentation accessible from the World Wide Web. Images are written to external files in `.PNG`, `.JPEG`, or `.GIF` format based upon the format you choose in the Documentation Tool dialog box. See the *Printing/Exporting Control and VI Descriptions to an RTF or HTML File* section in Chapter 5, *Printing and Documenting VIs*, in the *G Programming Reference Manual* for more information about these formats.
- **Rich Text Format (RTF)**—RTF is a standard format that is supported by a number of word processors, including Microsoft Word.
- **Online Help Source**—This produces an RTF file that is formatted for online help (`.hlp`) files. All images are written out as external bitmaps, and a help project and table of contents are created. These files can then be compiled using a help compiler. See the *Creating Your Own Help Files* section in Chapter 5, *Printing and Documenting VIs*, in the *G Programming Reference Manual* for information about compilers for different platforms.
- **Printer**—Select this item to print documentation to a printer.

You can use the **Save** button at the bottom of the dialog box to save the settings and list of files you create to a file. If you select the **Load** button, you can restore the settings from the file you created using the **Save** button.

Source Code Control Tools

The Source Code Control (SCC) tools make it easier to share work and coordinate access to LabVIEW files. They provide a complete system for controlling, accessing, and documenting the different revisions of VIs that make up your projects. These tools are described in detail in the *Professional G Developers Toolkit Reference Manual*. The following sections describe how to get started quickly with these tools and also describe several enhancements to these tools that are not described in the manual.

QuickStart Guide to Using the SCC Tools

Following is a brief summary of the steps for configuring and using the SCC tools. For more information on any of these steps, see the *Professional G Developers Toolkit Reference Manual*.

1. **Administrator sets up system**—At least one user in your workgroup should be selected to administer the SCC system. This user should select the Administration option when installing the LabVIEW Professional Developers version.

   The administrator is responsible for initializing the SCC system and setting up system-wide preferences. You do this from the **Project»Source Code Control»Administration** dialog box. When you set up the SCC system there are a couple of decisions you will need to make:

   – **Which storage system do you want to use?** The SCC tools can manage files or they can integrate with a third-party tool. See the *Support for Third-Party Tools* and the *Frequently Asked Questions about the SCC Tools* sections for help in deciding which system to use.

   – **Will users be developing on multiple platforms or a single platform?** This decision might help decide the storage system you will use, as well as how to configure that system. For multiplatform development, we recommend the built-in system. For single platform development you can use either the built-in system or one of the third party SCC tools that are supported.

   Once the system is up and running, there is typically not a lot of work involved unless you want to change access privileges for files or limit access for specific users.

   If you are using ClearCase, refer to the *ClearCase Installation Instructions* section.

2. **Users configure the SCC tools to access the system**—The main decision users must make is where they will have a *working directory.* The working directory is a directory on your local system where all VI development takes place. As you retrieve files from SCC, they are retrieved to this directory and its subdirectories. You cannot add files to SCC which are outside of this directory.

   Also, if you are using the Built-in system, it is important to understand the difference between the working directory, which is where you do your work, and the master directory, which is where SCC stores the latest versions of files. Users will not modify the master directory except when they check in files or create projects.

With ClearCase, your work directory is the same directory as the master directory. ClearCase uses a virtual file system model that allows multiple users to have a unique view and set of files in the same directory. See the *ClearCase Installation Instructions* section for detailed information.

3. **Users set up VIs for use with the SCC tools—**As mentioned above, all files that the user will work with must be in the working directory or one of its subdirectories. In addition, files that will be stored under SCC cannot be stored in LLBs. LLBs are not flexible enough for them to work well with an SCC system because they do not allow for easy access to individual files. You can use the **Project»File Manager** tool to convert your LLBs to directories of VIs.

   In converting LLBs to directories, you may find that you have some files with names that cannot be managed by the file system (i.e.: if it contains a separator character such as \, /, or :). Consequently, you might have to rename of some files and update references within your hierarchy.

   If you need to support Windows 3.1, see the *Frequently Asked Questions about the SCC Tools* section for suggestions on the best strategy for handling development.

4. **Create SCC projects—**A project in the SCC tools corresponds to a hierarchy of VIs. To create a project, first open the top-level VI of a hierarchy. Then select Project»Source Code Control»Project and click the New Project button. The subsequent dialog box lets you select your VI hierarchy, which can then be added to SCC.

   You can add other project-related files such as documentation and shared libraries by using the **Extra Files** button in this dialog box.

5. **Use SCC commands to access project files**—At this point, you can begin to work with files under SCC. All commands related to SCC are in the **Project»Source Code Control** menu. Following is a brief description of the remaining commands:

   – **Retrieve Files**—Use to copy files from SCC to your working directory.

   – **Checkout Files**—Use to checkout files to your working directory so that you can modify them.

   – **Checkin Files**—Use to copy a file from your working directory to SCC so that other users can access it.

   – **Compare Files**—Use to compare the files in your working directory with those under SCC. This command is described later in these release notes.

   – **Advanced**—Use to create reports, view the history of files under SCC, access previous versions of files, and delete files.

See the *Professional G Developers Toolkit Reference Manual* for more information about these commands. In addition, the following sections provided updated material and clarifications about using the SCC tools.

Support for Third-Party Tools

The Source Code Control (SCC) tools are designed to work with a variety of industry standard third-party tools for managing source code. You can either use the built-in system, Visual SourceSafe for Windows, or ClearCase for Solaris 2. Support for other third-party tools might be added in the future based upon demand.

The Built-in system and Visual SourceSafe are described in the *Professional G Developers Toolkit Reference Manual*. ClearCase for Solaris 2 is described in the following section.

ClearCase Support

ClearCase is a standard Source Code Control system on UNIX. If you have ClearCase installed, you can select ClearCase for source code control in the Administration dialog box. All Source Code Control tools work with the same interface as described in the *Professional G Developers Toolkit Reference Manual*.

ClearCase Installation Instructions

The Professional Developers version provides the interface for using ClearCase in the LabVIEW development environment. The administrator must create a ClearCase Versioned Object Base (VOB) and an associated storage space outside of LabVIEW before users can use source code control with ClearCase. When the administrator sets up the source code control system from inside LabVIEW, he or she must provide the VOB directory as the master directory. Because ClearCase uses a virtual file system, users also must use this VOB directory as their local directory.

In addition to providing the correct local directory, users must set up a ClearCase view, and make sure it is set every time they launch LabVIEW. To emulate the model of having local copies of files, users must set up unique labels to mark current working versions of files. When working under this model, users must edit their views to show labeled files before most recent versions.

Once the administrator and users configure their ClearCase systems, ClearCase is invisibly integrated into the LabVIEW source-code control interface.

Setup—Administrator

1. Install ClearCase if it is not already on your system. Refer to the ClearCase documentation for instructions.
2. If you want to create public VOBs, either set up a registry password or locate the existing registry password.
3. Create the VOB.
   a. Set your `umask`. For shared VOBs, the suggested `umask` setting is `2`.
   b. Create a mount point directory, such as `/vobs`.
   c. Create or choose a directory to store the VOB database, such as `/vobstore`.
   d. Type `cleartool mkvob -public -tag/vobs/vobname /vobstore/vobname.vbs`, where `vobs` is the mount point directory and `vobstore` is the directory where the VOB database is stored.
   e. Enter registry password (from step 2) if creating a public VOB.
   f. Enter comments, if any.
   g. Restore your `umask` to its original setting.
4. Mount the VOB. For example, type `cleartool mount /vobs/vobname`
5. Provide the name of the VOB (for example, `/vobs/vobname`) to users.
6. For ClearCase, you must specify that VIs are binary files and decide if you want to compress files. For each file extension that your VIs use, you must define an association of these extensions with their file type. For most LabVIEW applications, this includes VIs (.vi), controls (.ctl), and globals (.glb). If you use any other extensions, you must define file types for those as well.
   a. Choose a file type for VIs, such as `file`, `compressed_file`, or `binary_delta_file`. You also can define your own file type and type manager (see the ClearCase documentation). National Instruments recommends `compressed_file` because `binary_delta_file storage` does not work as well.
   b. You specify this association in a *magic file* in one of the paths in the *magic path*. See ClearCase documentation to determine where the magic path is.
   c. Somewhere in the search path for magic files, create a file with a `.magic` suffix.
   d. Add the line: `compressed_file : -name "*.vi" ;` Replace `compressed_file` with your chosen file type.

7. Set up a view. Instructions for setting up a view are in the *Setup—User* section.
8. Set the view, and launch LabVIEW.
9. Select **Project»Source Code Control»Administration**.
10. Select ClearCase as the source code control system.
11. The ClearCase configuration window opens.
12. Set VOB directory (e.g. `/vobs/vobname`).
13. Click **OK** in configuration window.
14. Click **OK** in administration window.

Additional Administrator Setup—Optional

You must complete these steps if you want to add VIs to ClearCase.

1. Complete local configuration (See the *Setup—User* section.)
2. Set your `umask` to `2` to give other users in your group access to any new directories you create.
3. Set a view and launch LabVIEW.
4. Copy files to the VOB directory. Use **Project»File Manager** or copy them using UNIX commands. The file manager tools is useful because it can help to convert LLBs to VIs. (LLBs are not supported by the SCC tools).
5. Open the top-level VI for a new project.
6. Create a new project, using **Project»Source Code Control»Project**.
7. Select **New Project**.
8. Select the top-level VI from the pop-up menu near the top of the dialog box. It will automatically assign a name for the project, although you can change it if you want to. Click **Save**.
9. The VI hierarchy and its subVIs are now in source code control. Repeat this process for additional hierarchies.

Setup—User

1. Find out the VOB directory name from the administrator(s). For example, `/vobs/vobname`.
2. Create a view.
   a. Set `umask` (typically `2` for shared files). See ClearCase documentation for more details.
   `cleartool mkview -tag` *`viewname viewstorage`*
   (for example, `cleartool mkview -tag george /viewstore/george.vws`)
   (each user should have his own view)

b. Restore `umask` to original value.

3. Set your view. For example, `cleartool setview` *viewname*.
4. Choose a label to mark current working versions of files. Your userid in all uppercase letters is a good choice. (Your label must be unique among all who will access a particular VOB).
5. Edit your view to show versions with this label.
   a. Edit your view. With your view set, execute `cleartool edcs`.
   b. Using the text editor (launched by cleartool), add the line: `element * LABELNAME`, replacing `LABELNAME` with your chosen label to the view configuration spec just after the line element `* CHECKEDOUT`.
6. Launch LabVIEW. You must set the view every time you launch LabVIEW.
7. Select **Project»Source Code Control»Local Configuration**.
8. Choose ClearCase as your source code control system.
9. Enter the VOB directory name given to you by the administrator in step 1.
10. Enter the label name from step 4.
11. Click **OK**.
12. Enter your local working directory, which must be the same as the VOB directory.
13. Click **OK**.
14. To freeze your working set of VIs so that changes made by other users do not affect your work until you retrieve them, select **Project»Source Code Control»Retrieve Files** and retrieve the files you want to freeze. When you select **Retrieve Files,** LabVIEW indicates if the files in your view are out of date and if retrieving files will update your view by showing a snapshot of the latest versions of those files.

Built-in System Configuration

If you select the built-in system from the Administrator dialog box, you have a set of options for deciding how files are locked for each user. Locking prevents users from accidentally modifying files without first checking them out. The Built-in Administration dialog box gives you the option of using file system locking or internal locking.

- **File System Locking**—For many sites, this is the appropriate selection. With File System Locking, as you check files in they are marked as read-only in the file system. As you check them out, they are changed to read-write. This works well for most developers.

- **Internal Locking**—With this option, as you check files in they are locked by LabVIEW. If you open the Show VI Info dialog box, you will see that the lock option is turned on. When you check files out they are automatically unlocked. This is the best choice for multiplatform development (i.e.: Windows, Macintosh, and/or UNIX development). When you bring a VI from another platform, it must be recompiled and saved. With internal locking, you can actually save the VI. The internal locking just prevents you from making edits to a VI. Note that the SCC tools can distinguish between real changes and simple modifications such as a recompile that occurs when loading the VI.

In addition to these locking options, you must select a master directory, which is where the Source Code Control tools store the files you add to SCC. Its important that all users have access to this directory. If you want users to be able to modify files, then they need to have read write access to this directory.

In addition, under Windows 95/NT, each user should connect to the volume containing this directory and map that volume as a Network drive (i.e.: as `D:\` rather than as a UNC path of the form `\\machine\volume`). UNC filename support varies slightly between Windows 95/NT, and consequently the LabVIEW SCC tools do not support UNC filenames.

Managing Files with the Same Name

The manual incorrectly indicates that you cannot manage files that have the same name under SCC. In general, you should be cautious when working with files with the same name because it is easy to accidentally link the wrong version of a file to a hierarchy. The SCC tools detect if you accidentally link the wrong version of a file when you edit a project using the **Project»Source Code Control»Project** dialog box. You have the choice to avoid changing the project by moving the existing file in SCC to the new location, or adding the new file as file with the same name in a different location.

SCC File Wizard

When you edit a project using **Project»Source Code Control»Project**, the SCC tools compare your hierarchy to the version that is already under SCC. If there are differences in the list of files, a File Wizard walks you through the process of updating SCC to reflect local changes.

This wizard first presents a dialog box summarizing the differences between the local version of your hierarchy and the version that is under SCC. This wizard detects whether one hierarchy has files that the other does not, and whether files are in different locations in the local hierarchy from the hierarchy under SCC. If you choose to edit the project, a series of

dialog boxes describe these differences in more detail and give you the option to change the project or ignore the difference. No changes are made unless you select **Save** in the final dialog box, which summarizes the new list of files.

Frequently Asked Questions about the SCC Tools

How do I decide between the Built-in system, SourceSafe, and ClearCase?

In many cases, the decision to use one of the third-party tools (SourceSafe for Win95/NT or ClearCase for Solaris 2) might be based on the fact that your company has standardized on one of these tools. If not, you must decide what strategy you want to use for managing your files.

Fortunately, the SCC user interface in LabVIEW is the same regardless of which storage system you select. The main difference between the three systems are cost, control capabilities, and platform support.

The built-in system is available on all platforms, and does not have any additional costs associated with it. It has most of the functionality of the other systems. One key difference is that it works by copying files to and from a shared network directory that all users in a development group can access. It currently does not provide tools for limiting user access to files. If you need to control the degree of access various users have, such as read-only and read-write, you can get some of this functionality using file system permissions. However, this is not as powerful as the control features found in third party packages.

SourceSafe for Windows 95/NT can be used transparently as part of the SCC tools. It provides a greater degree of user access control, allowing you to create accounts with passwords and to assign access levels on a per file basis. The principal disadvantages are that it is not currently supported by the SCC tools on either Macintosh or UNIX, and there is an additional licensing cost from Microsoft, the developers of SourceSafe.

ClearCase for Solaris 2 has many of the same advantages and disadvantages as SourceSafe when compared to the built-in system. It has a high degree of user access control, and works well when security is a primary issue. As with SourceSafe, it involves an additional licensing cost from Atria, the developers of ClearCase. While a version of ClearCase is available for Windows NT, it is currently not supported by the LabVIEW SCC tools.

How do I manage multiple hierarchies of VIs?

A project corresponds to a single VI hierarchy. This relationship makes it possible for the Project dialog box to scan your local hierarchy for changes and verify that it is consistent with the project under Source Code Control.

If you are using server functionality to dynamically call subVIs, your application may consist of multiple hierarchies. You can manage systems of multiple hierarchies by creating individual projects for each set of VIs that your top-level VIs call dynamically. You can then create a *project group* to make it easier to access files from any of the projects used by your application. A project group is a collection of projects. If you select a project group in the Checkout, Checkin, or Retrieve dialog boxes, you can work on any file that is in any of the VI hierarchies used by your application. You can create project groups by selecting **Source Code Control»Project**.

Why isn't Windows 3.1 supported? What if I need Windows 3.1 support?

Windows 3.1 is not supported because the SCC tools manage VIs as individual files and the Windows 3.1 8+3 character naming limitation makes this impractical. While LLBs make it possible to use longer names, they do not provide the level of transparent access to files that is needed for the LabVIEW SCC tools.

While you cannot use the SCC tools on Windows 3.1, National Instruments understands that customers need Windows 3.1 support. As a developer, you might need to deploy applications to customers that are using systems with Windows 3.1. In that case, we suggest you do your development on Windows 95 or NT. On those platforms you can use individual files for VIs. When you are ready to distribute to Windows 3.1 users, you can save copies of your VIs into LLBs. An easy way for to convert your files to or from LLBs is to use the File Manager dialog box, which has buttons to convert between directories and LLBs.

In addition to managing VIs, can I use the SCC tools to manage other project files such as Word documents, text files, CIN source files, and so on?

Yes. When you edit a project, you will see an **Extra Files** button. Use this button to add non-VI files to your project.

How do I access previous versions of files?

If you are using SourceSafe, you can access previous versions of files from the SourceSafe Explorer.

The **Project»Source Code Control»Advanced** dialog box has a button for accessing a file history. With the built-in system and with ClearCase, you can use this dialog box to retrieve previous versions of files. Note that the administrator can configure the number of previous versions of files to maintain, so you might not have access to all previous versions of a file.

You also can use this dialog box to label a version of a file with a short name such as BETA, making it easy to retrieve it later. Also, labeled versions are not automatically deleted based upon the administration option just described. Instead, you must delete them manually if you no longer want the files.

If you want to label multiple files or retrieve all files with the same label, you can use the System History option from the Advanced dialog box.

321888A-01 Jan98