LabWindows/CVI™ Release Notes

Version 5.5

These release notes contain system requirements, installation instructions, new features, and updated information to help you begin using LabWindows/CVI 5.5.

Contents

LabWindows/CVI Installation ... 3
Minimum System Requirements ... 3
Installing LabWindows/CVI ... 4
Before Installation ... 4
Running the Installation ... 4
After Installation ... 5
What's New and Different in LabWindows/CVI 5.5 ... 6
Overview of New Features ... 6
New Sample Programs ... 6
ActiveX Samples ... 6
Application Samples ... 7
Custom Control Samples ... 7
DataSocket and OPC through DataSocket Samples ... 7
Dynamic Link Library (DLL) Samples ... 7
Multithreading Samples ... 7
Programmer's Toolbox Samples ... 7
Report Generation Samples ... 7
Serial Communication Samples ... 7
User Interface Samples ... 8
Windows SDK Samples ... 8
Source Code Control Integration ... 8
ActiveX Automation Controller ... 8
Instrument Drivers ... 8
Support for Events ... 8
Support For Multithreading ... 8
Dual Interface Support ... 9
ActiveX Automation Controller Wizard ... 9
Per-Object Property Get and Set Functions ... 9

New ActiveX Automation Library Functions 9
User Interface Enhancements 10
Table Control 10
UIR Localization Utility 10
UIConv Utility 10
Multiple Monitor Support 11
Localized Decimal Symbol Support 11
Graph Legend Custom Control 11
Combo Box Custom Control 11
Other User Interface Enhancements 11
New User Interface Library Functions 11
New User Interface Library Attributes 13
Automation Server Enhancements 14
New Methods 14
Compiler/Linker/Debugger/Build Changes and Enhancements 15
Debuggable DLLs 15
Multithreaded Debugging 15
DataTips For Variables and Expressions 15
Visual C++ 6.0 Import Libraries 15
Borland C++ Builder 4.0 15
Build States When Unloading and Reloading a Project 16
Debug and Release Configurations 16
Run-Time Checking in Debuggable DLLs 16
User Program Windows 16
Debug Output Window and Run-Time Error Window 16
Standard I/O Window 17
Go To Definition Command 17
Hidden Environment Windows 17
Change in Behavior of Ctrl-F12 17
Run State Change Callbacks 17
Instrument Driver Support Only 18
Target Settings Dialog Box 18
Threads Dialog Box 18
Changes to Run Options Dialog Box 18
Changes to Compiler Options Dialog Box 18
Changes to Environment Dialog Box 18
Memory Display Window 18
Force Include of SDK Import Libraries 19
Pop-up Menu in Build Error Window 19
Build Utility 19
Set Breakpoints While Program Is Running 19
Update Windows when Suspending 19
Function Panel Changes and Enhancements 19
Function Panel Format 19
Utility Library Changes and Enhancements 20
Multithreaded Utilities 20

LoadExternalModule in LabWindows/CVI 20
Loading Source Files Using LoadExternalModule 21
New Utility Library Functions 21
New DataSocket Library 23
New RS-232 Library Functions.. 24
New Advanced Analysis Functions.. 25
Toolslib Changes and Enhancements 25
Toolslib Instrument Drivers Multithreaded Safety.................. 25
New Toolbox Functions 25
Additional Enhancements... 28
Report Generation... 28
Console Applications Support.. 28
Run-Time Engine and Windows Shutdown 28
Updated Windows SDK 29
Formatting and I/O Library Fmt/FmtOut/FmtFile Functions... 29
Changes to the LabWindows/CVI User Manual.................................... 29
Chapter 3—Project Window... 29
Build Menu 29
Tools Menu.. 29
Help Menu 30
Options Menu 31
Windows Menu.. 31
Changes to the LabWindows/CVI Programmer Reference Manual....... 32
Chapter 4—Creating and Distributing Standalone
Executables and DLLs 32

LabWindows/CVI Installation

The following sections include information on system requirements and installation of LabWindows/CVI 5.5.

Minimum System Requirements

To run LabWindows/CVI, you must have the following:

- Personal computer using a Pentium 90 or higher microprocessor
- Windows 2000, Windows NT 4.0 Service Pack 3, Windows 98, or Windows 95
- 800 by 600 resolution (or higher) video adapter
- Minimum of 16 MB of RAM
- 50 MB free hard disk space
- Microsoft-compatible mouse

Installing LabWindows/CVI

National Instruments suggests that you install the complete LabWindows/CVI 5.5 program to take full advantage of all LabWindows/CVI 5.5 capabilities. If you choose to install with options, select the options you want and follow the onscreen directions. If necessary, you can run the setup program again and install additional files.

Before Installation

- Install Internet Explore 4.0.1 or later in order to access the Windows Software Developers Kit (SDK) Help. You can download Internet Explorer from Microsoft's web site at http://www.microsoft.com.
- Choose the **Custom Installation** option to install the complete Windows SDK from the CD-ROM version of the LabWindows/CVI Full Development System (FDS). Some Windows SDK header and import library files are included when you install the LabWindows/CVI Base Package. The Windows SDK help file is available only with the LabWindows/CVI FDS.
- Be aware that the LabWindows/CVI installation includes the LabWindows/CVI 5.5 Run-time Engine for Windows. This version overwrites previous versions of the run-time engine. Thus, LabWindows/CVI 4.*x*, 5.0, and 5.01-generated executables and DLLs use the new LabWindows/CVI 5.5 Run-time Engine. In addition, the LabWindows/CVI 4.*x*, 5.0, and 5.0.1 **Create Distribution Kit** feature includes, in generated distribution kits, the LabWindows/CVI 5.5 Run-time Engine instead of the LabWindows/CVI 4.*x*, 5.0, and 5.0.1 Run-time Engine.
- Use TestStand 1.0.1 if you want to use LabWindows/CVI 5.5 with TestStand. TestStand 1.0.1 is a free update to TestStand 1.0. Go to `http://www.ni.com/teststand` for instructions and more information on TestStand 1.0.1.

Running the Installation

If you have a previous version of LabWindows/CVI, be sure to install this version in a different directory or uninstall the older version before installing this version. Perform the following steps to install LabWindows/CVI:

1. Insert the Measurement Studio Installation CD into the CD-ROM drive.
2. After a few seconds, a dialog box appears. If the dialog box does not appear, click on Windows **Start»Run** and type `d:\setup.exe` in the **Open** text box, where *`d`* is your CD-ROM drive.
3. When the Measurement Studio Installation CD dialog box appears, click on the **View Readme** option to the right of the **Install LabWindows/CVI** option.
4. After reading and closing the **Readme** file, choose **Install LabWindows/CVI** from the Measurement Studio Installation CD dialog box and follow the onscreen instructions.

After Installation

- **Choosing a Compatible Compiler**—When you install LabWindows/CVI, you must choose a compatible compiler. If you want to change your choice later, run the installation program again and select the option that allows you to choose a new compatible compiler.
- **Using NI-VISA and the IVI Engine**—The LabWindows/CVI installation includes NI-VISA and the IVI Engine. You must have the Virtual Instrument Software Architecture (VISA) Library to use the new instrument driver standard that the *VXIplug&play* Systems Alliance accepts. You must have the Interchangeable Virtual Instruments (IVI) Engine to use IVI drivers on a system.
- **Making Backup `.uir` Files**—If you have been using LabWindows/CVI 4.0.1 or earlier, you should make backup copies of your`.uir` files. Once `.uir` files have been saved in version 5.5, they are partially compatible with versions 4.0.1, 3.1 and 3.0.1. You can load them, but if you save them in the previous versions you lose the new 5.5 features. Once the`.uir` files have been saved in version 5.5, they are no longer compatible with version 3.0.
- **Using GPIB, VXI, DAQ, or VISA—**If you plan to use GPIB, VXI, DAQ, or VISA with LabWindows/CVI, you must have a compatible driver from National Instruments.
- **Installing LabWindows/CVI in a New Directory**—If you already installed the NI-DAQ software for a previous version of LabWindows/CVI and you are installing this version of LabWindows/CVI in a new directory, you must reinstall the CVI portion of the NI-DAQ software into the new CVI directory.
- **Installing NI-DAQ 6.6 and GPIB 1.6**—The National Instruments hardware and software drivers for NI-DAQ 6.6 amd GPIB 1.6 are included on the CD-ROM version of the LabWindows/CVI in the `\Drivers\` subdirectory.
- **Using GPIB**—If you plan to use GPIB with LabWindows/CVI, you must have a 488.2-compatible GPIB board from National Instruments. You also must have installed the NI-488.2 software for Windows 95 or Windows NT.
- **Using the Data Acquisition Library**—If you plan to use the Data Acquisition library with LabWindows/CVI, you must have a data acquisition board from National Instruments.
- **Installing NI-DAQ Software**—You must install the NI-DAQ software from the distribution diskettes for NI-DAQ version 4.9.0 or later. NI-DAQ 6.6 is included on the CD-ROM version of LabWindows/CVI.
- **Using the NI-VXI Library**—If you plan to use the NI-VXI library with LabWindows/CVI, you must have installed the NI-VXI Software Interface drivers. Please contact National Instruments for more information.
- **Converting LabWindows/DOS Programs**—If you want to convert LabWindows/DOS programs to run under LabWindows/CVI, read Chapter 12, *Converting LabWindows for DOS Applications,* in the *Getting Started with LabWindows/CVI* manual.

Note Refer to the `readme.cvi` file for installation instructions, programming considerations, and changes that are too recent to be included in the LabWindows/CVI 5.5 documentation, help files, and release notes.

What's New and Different in LabWindows/CVI 5.5

This document does not contain detailed information on all changes to LabWindows/CVI 5.5. Please refer to the manual set for additional information on changes to the program.

This section includes information about changes and enhancements in LabWindows/CVI 5.5 that have been made since LabWindows/CVI 5.0.

Overview of New Features

LabWindows/CVI 5.5 includes the following new features:

- Source Code Control Integration
- Ability to debug multithreaded programs
- Events and Dual Interface support in ActiveX Automation controllers
- Ability to debug DLLs with full run-time checking
- Table Control
- User Interface Localization Utility
- Support for multiple monitor systems
- Report generation
- Ability to create Console applications
- DataTips to view variables and expressions while debugging
- Compatibility with Borland C++ Builder 4.0
- Updated Windows SDK
- DataSocket Library for networking
- Additional functions or features added to RS-232, Utility, and Formatting and I/O Libraries
- Additional Hilbert Transform and error calculation functions

New Sample Programs

LabWindows/CVI 5.5 includes the following sample programs:

ActiveX Samples

`samples\activex\excel\excelevents97.prj`

`samples\activex\excel\excelevents2000.prj`

`samples\activex\matlab\matlabdemo.prj`

`samples\activex\outlook\outlookrep97.prj`

`samples\activex\outlook\outlookrep2000.prj`

`samples\activex\powerpoint\simple.prj`

`samples\activex\word\wordrpt.prj`

Application Samples

`samples\apps\finger\finger.prj`

`samples\apps\localui\intgraph.prj`

`samples\apps\tankdemo\tank.prj`

Custom Control Samples

`samples\userint\custctrl\combobox\combodemo.prj`

`samples\userint\custctrl\legend\lgdemo.prj`

DataSocket and OPC through DataSocket Samples

`samples\datasocket\*.*`

Dynamic Link Library (DLL) Samples

`samples\dll\gui\guidll.prj`

`samples\dll\gui\useguidll.prj`

Multithreading Samples

`samples\apps\daqmthread\daqMT.prj`

`samples\utility\threading\*.*`

Programmer’s Toolbox Samples

`samples\toolbox\msgdemo.prj`

`samples\toolbox\trayicon.prj`

Report Generation Samples

`samples\reportgen\nirsample.prj`

`samples\activex\word\wordrpt.prj`

Serial Communication Samples

`samples\rs232\commcallback.prj`

User Interface Samples

`samples\userint\colview.prj`

`samples\userint\gridview.prj`

`samples\userint\multimonitor.prj`

`samples\userint\splash.prj`

Windows SDK Samples

`samples\sdk\audio\cdplayer.prj`

Source Code Control Integration

You can integrate a third-party source code control provider with the LabWindows/CVI 5.5 environment, and then interface with your source code control provider from LabWindows/CVI 5.5. Select **Options»Source Code Control Options** in the Project window to configure LabWindows/CVI 5.5 to work with your provider. Use the commands in **Tools»Source Code Control** to perform common source code control actions. Errors and warnings generated by source code control commands appear in the Source Code Control Error window.

ActiveX Automation Controller

This section describes ActiveX Automation changes and enhancements in LabWindows/CVI.

Instrument Drivers

To support the new features in ActiveX Automation Controller instrument drivers, the drivers generated by LabWindows/CVI 5.5 differ from the drivers generated by LabWindows/CVI 5.0.1. If you need to regenerate an ActiveX Automation controller instrument driver, and want to minimize the changes to the driver functions, select **Per Server** in the Property Access Functions control in the ActiveX Automation Controller Wizard.

Support for Events

Some ActiveX Automation server objects generate events in addition to providing methods for you to call from an ActiveX Automation client program. LabWindows/CVI 5.5 generates code in ActiveX Automation controller instrument drivers that allows you to receive events. LabWindows/CVI 5.5 generates callback registration functions that allow you to specify a callback that is called when the server generates a specific event.

Support For Multithreading

You can now safely use an ActiveX Automation handle across threads, regardless of the threading model of the threads, if you pass TRUE in the Support Multithreading parameter in the object creation functions.

Dual Interface Support

Some ActiveX Automation servers support an alternate method of accessing functions and properties through dual interfaces. LabWindows/CVI 5.5 can generate ActiveX Automation Controller instrument drivers that use dual interfaces to access the server more efficiently. You can select this option when you use the ActiveX Automation Controller Wizard.

ActiveX Automation Controller Wizard

You can use this wizard to create an instrument driver to control an ActiveX Automation Server. The ActiveX Automation Controller Wizard includes support for ActiveX Automation Events as well as dual interfaces. The ActiveX Automation Controller Wizard is located in the **Tools** menu.

Per-Object Property Get and Set Functions

The ActiveX Automation Controller generated in LabWindows/CVI 5.0.1 contained a single set of property get and set functions for each ActiveX Automation Controller. In LabWindows/CVI 5.5, you can choose to generate property get and set functions for each object in the controller. If you choose to generate per-object property get and set functions, you must use the appropriate property get and set functions to access properties for a particular object.

New ActiveX Automation Library Functions

Free BSTR	`CA_FreeBSTR`
Fill Error Info	`CA_FillErrorInfo`
Init ActiveX Thread Style For Current Thread	`CA_InitActiveXThreadStyleForCurrentThread`
Set Support For Multithreading	`CA_SetSupportForMultithreading`
Get Support For Multithreading	`CA_GetSupportForMultithreading`
Get Active Object By Class ID	`CA_GetActiveObjectByClassIdEx`
Get Active Object By Prog ID	`CA_GetActiveObjectByProgIdEx`
Create Object By Class ID	`CA_CreateObjectByClassIdEx`
Create Object By Prog ID	`CA_CreateObjectByProgIdEx`
Load Object From File	`CA_LoadObjectFromFileEx`
Load Object From File By Cls ID	`CA_LoadObjectFromFileByClassIdEx`
Load Object From File By Prog ID	`CA_LoadObjectFromFileByProgIdEx`

Create ObjHandle from Interface	`CA_CreateObjHandleFromInterface`
Register Event Callback	`CA_RegisterEventCallback`
Unregister Event Callback	`CA_UnregisterEventCallback`
Unregister All Event Callbacks	`CA_UnregisterAllEventCallback`
Enable Events For Server Object	`CA_EnableEventsForServerObject`
Get Event Callback	`CA_GetEventCallback`
Safe Array to 1D Array	`CA_SafeArrayTo1DArrayEx`
Safe Array to 2D Array	`CA_SafeArrayTo2DArrayEx`
Safe Array to 1D Array Buffer	`CA_SafeArrayTo1DArrayBufEx`
Safe Array to 2D Array Buffer	`CA_SafeArrayTo2DarrayBufEx`
Duplicate ObjHandle	`CA_DuplicateObjHandle`

User Interface Enhancements

This section describes the user interface enhancements in LabWindows/CVI.

Table Control

You can create a table control in your graphical user interface. A table control can contain string, numeric, and picture cells. The user interface library contains functions to create and manipulate a table control. Refer to LabWindows/CVI Online help for more information on using a table control.

UIR Localization Utility

You can use the Localization utility to easily translate your user interface resource files. To access this utility, choose **Tools»User Interface Localizer**. You can then use the `LoadLocalizedPanel` and `LoadLocalizedMenu` functions—located in the `CVI\toolslib\localui\localui.fp` function panel file—to load the translated panels and menu bars.

UIConv Utility

UIConv is a utility that reads the contents of `.tui` files and generates code to build the user interface programmatically, eliminating the need to distribute a separate `.uir` file with your LabWindows/CVI 5.5 application. To access this utility, choose the Utility Programs submenu of the Measurement Studio Group in the Start menu of the Windows taskbar. A `.tui` file is a text version of a `.uir` file. To generate a `.tui` file, open a `.uir` file in the User Interface Editor and then choose **Options»Save in Text Format**.

Multiple Monitor Support

CVI supports multiple monitors. Features in the environment such as windows, dialog boxes, and menus behave in an appropriate manner on a multiple monitor system. You also can use the functions in the user interface library to iterate through all the monitors, get monitor attributes such as coordinates and color depth, and set the default monitor for your application. Windows 98 and Windows 2000 support multiple monitors.

Localized Decimal Symbol Support

You can configure the User Interface Editor to use the operating system's decimal symbol as follows. Access the Editor Preferences dialog box in the **Options»Preference** in the User Interface Editor and select the **Use localized decimal symbol** checkbox. To make your program use the system's decimal symbol, make the following function call:

```
SetSystemAttribute(ATTR_USE_LOCALIZED_DECIMAL_SYMBOL,
VAL_USE_SYSTEM_SETTING)
```

Graph Legend Custom Control

The sample program `samples\userint\custctrl\lg_demo.prj` contains a graph legend control instrument driver.

Combo Box Custom Control

The sample program `samples\userint\custctrl\combobox.prj` contains a combo box control instrument driver.

Other User Interface Enhancements

- The IDs for message controls and other indicator controls now remain stable when you edit them in the User Interface Editor.
- The number of lines in a Listbox or Textbox is now limited only by available memory.
- The number of Graph cursors is now limited only by available memory.
- The Edit Tab Order dialog box is now sized to display the entire panel, if the dialog box fits on your screen.

Additional attributes and functions are listed in the following sections.

New User Interface Library Functions

This section lists the new User Interface Library functions in LabWindows/CVI 5.5.

Table Control

Insert Table Rows	`InsertTableRows`
Insert Table Columns	`InsertTableColumns`
Delete Table Rows	`DeleteTableRows`
Delete Table Columns	`DeleteTableColumns`
Get Number of Table Rows	`GetNumTableRows`
Get Number of Table Columns	`GetNumTableColumns`
Get Active Table Cell	`GetActiveTableCell`
Set Active Table Cell	`SetActiveTableCell`
Get Table Selection	`GetTableSelection`
Set Table Selection	`SetTableSelection`
Get Table Row Attribute	`GetTableRowAttribute`
Set Table Row Attribute	`SetTableRowAttribute`
Get Table Column Attribute	`GetTableColumnAttribute`
Set Table Column Attribute	`SetTableColumnAttribute`
Get Table Cell Attribute	`GetTableCellAttribute`
Set Table Cell Attribute	`SetTableCellAttribute`
Set Table Cell Range Attribute	`SetTableCellRangeAttribute`
Get Table Cell From Value	`GetTableCellFromValue`
Get Table Row From Label	`GetTableRowFromLabel`
Get Table Column From Label	`GetTableColumnFromLabel`
Get Table Cell From Point	`GetTableCellFromPoint`
Get Table Cell Range Rect	`GetTableCellRangeRect`
Get Table Cell Value	`GetTableCellVal`
Set Table Cell Value	`SetTableCellVal`
Get Table Cell Range Values	`GetTableCellRangeVals`
Set Table Cell Range Values	`SetTableCellRangeVals`
Fill Table Cell Range	`FillTableCellRange`

Get Table Cell Value Length	`GetTableCellValLength`
Free Table Value Strings	`FreeTableValStrings`
Sort Table Cells	`SortTableCells`
New Control Menu Item	`NewCtrlMenuItem`
Discard Control Menu Item	`DiscardCtrlMenuItem`
New Control Menu Separator	`NewCtrlMenuSeparator`
Hide Built-In Control Menu Item	`HideBuiltInCtrlMenuItem`
Show Built-In Control Menu Item	`ShowBuiltInCtrlMenuItem`
Get Table Values from Clipboard	`ClipboardGetTableVals`
Put Table Values on Clipboard	`ClipboardPutTableVals`

Multiple Monitor Support

Get Monitor Attribute	`GetMonitorAttribute`
Get Monitor From Point	`GetMonitorFromPoint`
Get Monitor From Rect	`GetMonitorFromRect`
Get Monitor From Panel	`GetMonitorFromPanel`

Other User Interface Functions

Delete Text Box Lines	`DeleteTextBoxLines`
Add to File Popup Dir History	`AddToFilePopupDirHistory`
Clear File Popup Dir History	`ClearFilePopupDirHistory`
Get Typeface Length from Font	`GetFontTypefaceNameLength`
Get Typeface Name from Font	`GetFontTypefaceName`

New User Interface Library Attributes

Rings

`ATTR_DISABLE_CHECK_MARK`

Graphs

```
ATTR_ACTUAL_XDIVISIONS
ATTR_ACTUAL_YDIVISIONS
ATTR_ACTUAL_XPRECISION
ATTR_ACTUAL_YPRECISION
ATTR_XPADDING
ATTR_YPADDING
ATTR_XLOOSE_FIT_AUTOSCALING
ATTR_YLOOSE_FIT_AUTOSCALING
ATTR_XLOOSE_FIT_AUTOSCALING_UNIT
ATTR_YLOOSE_FIT_AUTOSCALING_UNIT
ATTR_CURSOR_ENABLED
```

Numerics, Slides, Tables

```
ATTR_PADDING
```

Menus

```
ATTR_BOLD
ATTR_NEXT_ITEM_ID
ATTR_NEXT_MENU_ID
ATTR_FIRST_ITEM_ID
ATTR_FIRST_MENU_ID
```

Panel

```
ATTR_OWNER_THREAD_ID
```

System

```
ATTR_USE_LOCALIZED_DECIMAL_SYMBOL
ATTR_LOCALIZED_DECIMAL_SYMBOL
ATTR_DEFAULT_MONITOR
ATTR_PRIMARY_MONITOR
ATTR_NUM_MONITORS
ATTR_FIRST_MONITOR
ATTR_DISABLE_PROG_PANEL_SIZE_EVENTS
```

Automation Server Enhancements

This section describes the Automation server enhancements in LabWindows/CVI 5.5.

New Methods

The following functions have been added to the Automation server.

`GenWinHelpFileFromFPFile`

`GoToDefinition`

`SetActiveConfiguration`

```
SetDllExportHeaderFiles
GetDebuggeeProcessId
DiffTextFiles
SetProjectDebugTargetPath
SetProjectDebuggingLevel
SetBreakOnFirstChanceExceptions
SetInteractiveWindowMemorySize
ForceSourceFilesIntoInteractiveWindow
SaveBuildErrorWindowContents
SetProjectVersionInfo
GetProjectVersionInfo
```

Compiler/Linker/Debugger/Build Changes and Enhancements

This section describes the compiler/linker/debugger changes and enhancements in LabWindows/CVI 5.5.

Debuggable DLLs

Debuggable DLLs built with LabWindows/CVI 5.0 or LabWindows/CVI 5.0.1 are not debuggable with LabWindows/CVI 5.5. You must rebuild your debuggable DLLs with LabWindows/CVI 5.5 if you want to debug those DLLs with LabWindows/CVI 5.5. It is not necessary to rebuild non-debuggable DLLs that you built with LabWindows/CVI 5.0 or LabWindows/CVI 5.0.1.

Multithreaded Debugging

You now can create and debug multithreaded programs in the LabWindows/CVI environment.

DataTips For Variables and Expressions

To view variable values, you can place the mouse cursor over the variable name while execution is suspended. You can also use DataTips to view the values of highlighted expressions.

Visual C++ 6.0 Import Libraries

LabWindows/CVI 5.5 supports the default format of import libraries that Visual C++ 6.0 generates.

Borland C++ Builder 4.0

LabWindows/CVI 5.5 is now compatible with Borland C++ Builder 4.0.

Build States When Unloading and Reloading a Project

LabWindows/CVI 5.5 saves the compiled state of your projects in files on disk. This means that you do not need to rebuild the project each time you load the project in LabWindows/CVI. After you have built the project, LabWindows/CVI 5.5 compiles only the files that have changed since the last time you built the project.

Debug and Release Configurations

LabWindows/CVI 5.5 supports two build configurations for your project. Use **Build»Configuration** in the Project window to set the active configuration. When you debug your projects, set the active configuration to the **Debug** configuration. In the **Debug** configuration, LabWindows/CVI 5.5 uses the **Debugging Level** option in the Build Options dialog box to determine how much debugging information to generate for the debug executable or DLL. When you want to create a DLL or executable that is suitable for distribution, set the active configuration to the **Release** configuration. In the **Release** configuration, LabWindows/CVI 5.5 does not generate any debugging information into the executable or DLL. Executables and DLLs that you build in the **Release** configuration are smaller and execute faster. You can specify different filenames for **Debug** and **Release** executables and DLLs.

Run-Time Checking in Debuggable DLLs

You now can build debuggable DLLs with full run-time checking. To include run-time checking in your DLL, set the **Debugging Level** option in the Build Options dialog box to **Standard** or **Extended** and set the active configuration to **Debug**.

User Program Windows

User program windows are not accessible when a program is suspended. LabWindows/CVI 5.5 creates a separate process to execute your program even when you are debugging the program in the LabWindows/CVI environment. Therefore, you cannot access the windows of your program, including the Standard I/O window, while your program is suspended in the debugger. Furthermore, while your program is suspended, the windows of your programs do not redraw if another window covers them and then you uncover them. Another option is to use the Utility library `DebugPrintf` function to send debug output to the Debug Output Window (accessible from the environment Window menu).

Debug Output Window and Run-Time Error Window

The environment now has a Debug Output window that contains the output of the Utility Library `DebugPrintf` function as well as the Windows SDK `OutputDebugString` function. This window is useful for debug strings output by your program. Unlike the Standard I/O window, this window is accessible while your program is suspended. You can choose to have the Debug Output window come to the front whenever text is added to the window by checking the **Bring Debug Output Window to Front Whenever Modified** option in the Environment Options dialog box.

You can now send output to the Run-time Error Window by using the Utility Library `ErrorPrintf` function. The advantage of using `ErrorPrintf` instead of `DebugPrintf` is that the Run-time Error Window saves the source code location of the output of each `ErrorPrintf` call. This allows you to click on the error string in the Run-time Error Window to go to the position of the source code that generated that error.

Standard I/O Window

The Standard I/O window is not accessible when a program is suspended. Because LabWindows/CVI 5.5 creates a separate process to execute your program even when it is debugging the program from the environment, the Standard I/O window is now a window owned by your program. You cannot access this window while your program is suspended in the debugger. If you want to view debugging output from your program, use the Utility Library `ErrorPrintf` function to send debug output to the Run-time Errors window. Alternatively, you can enable the **Use Console Window for Standard I/O While Debugging** option in the Environment dialog box to use a Windows console window instead of the Standard I/O window while debugging. In contrast to the LabWindows/CVI Standard I/O window, you can access a Windows console window even when your program is suspended in the debugger.

Go To Definition Command

The **Go To Definition** command now works on preprocessor macros. It previously worked only on function and variable declarations.

Hidden Environment Windows

Using the Windows taskbar menu, you can display hidden LabWindows/CVI 5.5 environment windows. If you enable the **Hide Windows** option from **Options»Run Options** and begin debugging your program, you can right-click on the LabWindows/CVI icon in the Windows taskbar and select **Display Windows** to show all of the LabWindows/CVI 5.5 environment windows.

Change in Behavior of Ctrl-F12

In previous versions of LabWindows/CVI, you could suspend a running program by pressing `Ctrl-F12` even when a LabWindows/CVI environment window, such as the project window or a source window, was NOT active. In CVI 5.5, `Ctrl-F12` will only work if a LabWindows/CVI environment window is the active window. If none of the LabWindows/CVI environment is visible while you are running, you can right-click on the Windows taskbar item for LabWindows/CVI and choose **Show Windows** to make the LabWindows/CVI environment windows visible.

Run State Change Callbacks

LabWindows/CVI 5.5 does not call Run State Change Callbacks when your program is suspended or resumed. Such calls are not necessary because the program runs in a separate process and the entire process is suspended when the program encounters a breakpoint, watchpoint, or other condition that suspends your program. It is no longer necessary for you to use Run State Change Callbacks to disable callbacks or otherwise suspend activity in your libraries.

Instrument Driver Support Only

The **Instrument Driver Support Only** menu item has been converted to a checkbox on the Target Settings dialog box.

Target Settings Dialog Box

The Target Settings command in the **Build** menu displays the Target Settings dialog box where you can configure the output of your build. This dialog box contains all of the items that previously appeared in the Create Standalone Executable, Create Dynamic Link Library, and Create Static Library dialog boxes.

Threads Dialog Box

Run»Threads displays the Threads dialog box where you can view the current threads in your program while execution is suspended.

Changes to Run Options Dialog Box

The Run Options dialog box has a new option called **Break on First Chance Exceptions**. Some of the options previously on the Run Options dialog box now appear in the Build Options dialog box.

Changes to Compiler Options Dialog Box

The Compiler Options dialog box is now called the Build Options dialog box. Some of the options previously on the Run Options dialog box now appear in the Build Options dialog box.

Changes to Environment Dialog Box

The Environment dialog box, which you access in the **Options** menu, has new options to go to source after inserting code from the function panel, check foreground lockout setting on startup, hide data tool tips, force loaded instrument drivers' source into interactive window, and force project source files into interactive window.

Memory Display Window

Use the new Memory Display window to view the contents of arbitrary memory locations in your programs while your program is suspended.

Force Include of SDK Import Libraries

If you use the Add To Executable dialog box to force a Windows SDK import library into your project, your executable or DLL might fail to load. The Windows SDK import libraries that come with LabWindows/CVI 5.5 contain functions that are not present on all versions of Windows. Therefore, forcing an entire import library into your executable or DLL might cause it to fail to load or start because it is referencing a function that is not available in the DLL on your operating system.

Pop-up Menu in Build Error Window

You can right click in the Build Error Window to bring up a popup menu that allows you to copy one or all build errors to the clipboard, save the build errors to a file, or go to the source position for a build error.

Build Utility

The Build utility (`samples\activex\cvi\build.exe`) now allows you to set the version number and string for the target file and save the build errors to a file.

Set Breakpoints While Program Is Running

You can now set breakpoints while your program is running. You do not have to suspend the program to set breakpoints.

Drop Files into the Project Window

You can add files to the project by dragging files from the Microsoft Windows shell and dropping them on the project window.

Update Windows when Suspending

When the **Update Windows when Suspending** option in the Run Options dialog box is checked, the system redraws the panels in the program being debugged, if necessary, before suspending due to a breakpoint, watchpoint, or a Step command. This is useful when you are stepping through code that calls User Interface Library functions that set attributes or values but do not refresh the panel or control immediately.

Function Panel Changes and Enhancements

This section describes the function panel changes and enhancements in LabWindows/CVI 5.5.

Function Panel Format

You now have the option to create a function panel in a new format. Use **Options»FP File Format** in the Function Tree editor to select the format for your function panel file. Refer to the Table below for a comparison of the two formats.

Table 1. Comparison of Function Panel Formats

Characteristic	Old Format	New Format
Maximum Instrument Prefix Length	8	32
Maximum Function Name Length	31	79
Maximum Class Name Length	31	79
Maximum Window Name Length	31	79
Max Type String Length	50	80
Default Text for Output Controls	Not Supported	Supported
Function Qualifiers	Not Supported	Supported
Set Precision on Numeric Controls	Not Supported	Supported

Converting Function Panel Help to Windows Help Files

The **Generate Windows Help** command now provides an option to automatically create a Windows Help file. Previously, this command only generated source files that you could compile with the Windows Help compiler to build the Windows Help file. LabWindows/CVI 5.5 runs the help compiler automatically and creates a `.hlp` file for you.

Utility Library Changes and Enhancements

This section describes the enhancements in the Utility Library.

Multithreaded Utilities

LabWindows/CVI 5.5 provides many utility functions to make multithreaded programming easier. You can create and manage threads with the new thread pool API. You also can protect your data in a multithreaded program with LabWindows/CVI's thread-safe queues, thread-local variables, and thread-safe variables. Refer to the Utility Library function panel documentation for more information. Also refer to the multithreading samples in `samples\utility\threading`.

LoadExternalModule in LabWindows/CVI

In LabWindows/CVI, programmers often use `LoadExternalModule` or `LoadExternalModuleEx` to load object files or libraries. During debugging in previous versions of LabWindows/CVI, including version 5.0.1, those object files or libraries could reference symbols in any of the following modules even if the modules were not statically referenced by the project:

- A loaded instrument driver

- Libraries in the **Library** menu
- Libraries in the project

However, in past versions of LabWindows/CVI, when you created a standalone executable, you had to force the required instrument and libraries into your executable.

Now, because LabWindows/CVI 5.5 creates an executable even for the debugging process, you must force the required instrument driver and libraries into your executable, even during the debugging process. In LabWindows/CVI 5.5, `LoadExternalModule` makes it easier to determine the object files and libraries you need to force into your project. `LoadExternalModule` automatically determines which instrument drivers and libraries need to be forced into your executable, and prompts you to force the instrument drivers and libraries into your project.

Loading Source Files Using LoadExternalModule

In LabWindows/CVI 5.0.1 and earlier, if you executed code from a function panel or from the Interactive window that used `LoadExternalModule` to load a source file, you were required to put the source file into the project that was currently loaded. In LabWindows/CVI 5.5, you choose one of the following options when you use `LoadExternalModule` from a function panel or the Interactive window.

- Add the source file to the project and enable the **Force Project Source Files into Interactive Window** option in the Environment dialog box.
- Load the source file as an instrument driver and enable the Force Loaded Instrument Drivers' Source into Interactive Window option in the Environment dialog box. To load the source file as an instrument driver, you must load its associated `.fp` file from the Instrument menu. If there is an `.obj` or `.lib` file on disk with the same base name, you must force the source file to be attached to the instrument by clicking on the **Attach and Edit Source** button in the Edit Instruments dialog box.

New Utility Library Functions

Multithreaded Support Functions

Thread Pool

Schedule Function	`CmtScheduleThreadPoolFunction`
Schedule Function Advanced	`CmtScheduleThreadPoolFunctionAdv`
Release Function ID	`CmtReleaseThreadPoolFunctionID`
Get Function Attribute	`CmtGetThreadPoolFunctionAttribute`
Wait For Function To Complete	`CmtWaitForThreadPoolFunctionCompletion`
Exit Thread	`CmtExitThreadPoolThread`

New Thread Pool	`CmtNewThreadPool`
Discard Thread Pool	`CmtDiscardThreadPool`
Pre-Allocate Threads	`CmtPreAllocThreadPoolThreads`
Terminate Thread	`CmtTerminateThreadPoolThread`
Get Attribute	`CmtGetThreadPoolAttribute`
Set Attribute	`CmtSetThreadPoolAttribute`
Install Callback	`CmtInstallThreadPoolCallback`
Uninstall Callback	`CmtUninstallThreadPoolCallback`
Thread Safe Queue	
New Queue	`CmtNewTSQ`
Discard Queue	`CmtDiscardTSQ`
Get Attribute	`CmtGetTSQAttribute`
Set Attribute	`CmtSetTSQAttribute`
Write Data	`CmtWriteTSQData`
Get Write Pointer	`CmtGetTSQWritePtr`
Release Write Pointer	`CmtReleaseTSQWritePtr`
Read Data	`CmtReadTSQData`
Get Read Pointer	`CmtGetTSQReadPtr`
Release Read Pointer	`CmtReleaseTSQReadPtr`
Flush Queue	`CmtFlushTSQ`
Install Callback	`CmtInstallTSQCallback`
Uninstall Callback	`CmtUninstallTSQCallback`
Thread Local Variable	
New Thread Local Variable	`CmtNewThreadLocalVar`
Discard Thread Local Variable	`CmtDiscardThreadLocalVar`
Get Thread Local Variable	`CmtGetThreadLocalVar`
Thread Safe Variable	
New Thread Safe Variable	`CmtNewTSV`

Discard Thread Safe Variable	`CmtDiscardTSV`
Get Thread Safe Variable Pointer	`CmtGetTSVPtr`
Release Thread Safe Variable Pointer	`CmtReleaseTSVPtr`
Set Thread Safe Variable	`CmtSetTSV`

Other Multithreading Utility Functions

Get Current Thread ID	`CmtGetCurrentThreadID`
Get Main Thread ID	`CmtGetMainThreadID`
Get Number Of Processors	`CmtGetNumProcessors`
Get Error Message	`CmtGetErrorMessage`

Other New Utility Functions

Get External Module Address Ext	`GetExternalModuleAddrEx`
Input Double Word From Port	`inpd`
Output Double Word To Port	`outpd`
Error Printf	`ErrorPrintf`
DebugPrintf	`DebugPrintf`
Is The Process Being Debugged By LabWindows/CVI?	`BeingDebuggedByCVI`
Set Break On First-Chance Exceptions	`SetBreakOnFirstChanceExceptions`
Get Break On First-Chance Exceptions	`GetBreakOnFirstChanceExceptions`

New DataSocket Library

DataSocket simplifies live data exchange between different applications on one computer or between computers connected through a network. DataSocket implements an easy-to-use, high-performance programming tool that is designed specifically for sharing and publishing live data in measurement and automation applications. You access the DataSocket Library in the LabWindows/CVI **Library** menu.

Open	`DS_Open`
Discard Object Handle	`DS_DiscardObjHandle`
Get Data Type	`DS_GetDataType`
Get Data Value	`DS_GetDataValue`

Set Data Value	`DS_SetDataValue`
Update	`DS_Update`
Select URL	`DS_SelectURL`
Free Memory	`DS_FreeMemory`
Get Library Error String	`DS_GetLibraryErrorString`
Attributes	
Get Attribute Handle	`DS_GetAttrHandle`
Create Attribute Handle	`DS_CreateAttrHandle`
Delete Attribute	`DS_DeleteAttr`
Attribute Exists	`DS_AttrExists`
Get Attribute Names	`DS_GetAttrNames`
Get Attribute Type	`DS_GetAttrType`
Get Attribute Value	`DS_GetAttrValue`
Set Attribute Value	`DS_SetAttrValue`
Status	
Get Last Message	`DS_GetLastMessage`
Get Actual URL	`DS_GetActualURL`
Get Data Updated	`DS_GetDataUpdated`
Get Status Updated	`DS_GetStatusUpdated`
Get Status	`DS_GetStatus`
Get Last Error	`DS_GetLastError`
Local Server	
Control Local Server	`DS_ControlLocalServer`

New RS-232 Library Functions

Get COM Line Status	GetComLineStatus
Get System COM Handle	GetSystemComHandle

New Advanced Analysis Functions

Fast Hilbert Transform	FastHilbertTransform
Inverse Fast Hilbert Transform	InvFastHilbertTransform
Error Function	Erf
Complementary Error Function	Erfc

Toolslib Changes and Enhancements

This section describes the changes and enhancements in the Toolslib drivers.

Toolslib Instrument Drivers Multithreaded Safety

You can now use many of the Toolslib instrument drivers from multithreaded programs. But, some of the Toolslib instrument driver functions still require that you provide your own locking and they might also require that you do not use the same object from multiple threads. The Toolslib instrument drivers are in subdirectories in `LabWindows\CVI\toolslib`.

New Toolbox Functions

`cvi\toolslib\toolbox.fp` has several new functions. The new functions allow you to perform the following functions:

- Save control and panel images as `.bmp` files.
- Write applications that reside on the Windows taskbar system tray.
- Read and write values in the Registry.
- Get and set system configuration.
- Receive Windows messages sent to a LabWindows/CVI panel.
- Respond to files dropped on a panel.

The toolbox instrument driver is in `cvi\toolslib\toolbox.`

Saving Images as Bitmap Files

Save Bitmap To File	`SaveBitmapToFile`
Save Control Display to File	`SaveCtrlDisplayToFile`
Save Panel Display to File	`SavePanelDisplayToFile`

User Interface Utilities

Localize Number String	`LocalizeNumberString`
Delocalize Number String	`DelocalizeNumberString`

System Tray Icons

Install System Tray Icon	`InstallSysTrayIcon`
Set Tray Icon Attribute	`SetTrayIconAttr`
Get Tray Icon Attribute	`GetTrayIconAttr`
Remove System Tray Icon	`RemoveSysTrayIcon`

Pop-Up Menus

Attach Tray Icon Popup Menu	`AttachTrayIconMenu`
Set Popup Menu Attribute	`SetTrayIconMenuAttr`
Get Popup Menu Attribute	`GetTrayIconMenuAttr`
Add Tray Icon Popup Menu Item	`InsertTrayIconMenuItem`
Set Popup Menu Item Attribute	`SetTrayIconMenuItemAttr`
Get Popup Menu Item Attribute	`GetTrayIconMenuItemAttr`
Detach Tray Icon Popup Menu	`DetachTrayIconMenu`

Windows Registry

Write String to Registry	`RegWriteString`
Read String from Registry	`RegReadString`
Write ULong to Registry	`RegWriteULong`
Read ULong from Registry	`RegReadULong`
Write Binary to Registry	`RegWriteBinary`
Read Binary from Registry	`RegReadBinary`

System Configuration

Get Windows Version	`GetWinOSVersion`
Get Windows Directories	`GetWindowsDirs`
Get Computer Name	`GetCompName`

GUI

Get System Color	`GetWindowsColor`
Set ScreenSaver	`SetScreenSaver`
Get ScreenSaver	`GetScreenSaver`
Set Desktop Wallpaper	`SetWallpaper`
Get Desktop Wallpaper	`GetWallpaper`

Keyboard

Get Keyboard Preferences	`GetKeyboardPreferences`
Set Keyboard Preferences	`SetKeyboardPreferences`

Resources

Get Disk Space	`GetDiskSpace`
Compare UInt64Type to UInt	`UInt64TypeCompareUInt`
Get Memory Information	`GetMemoryInfo`

Windows Messaging

Install Win Message Callback	`InstallWinMsgCallback`
Set Callback Attribute	`SetMsgCallbackAttribute`
Get Callback Attribute	`GetMsgCallbackAttribute`
Remove Win Message Callback	`RemoveWinMsgCallback`

User Information

Get User Name	`GetCurrentUser`

Drag-And-Drop

Enable Drag-And-Drop	`EnableDragAndDrop`
Disable Drag-And-Drop	`DisableDragAndDrop`

Miscellaneous

Show HTML Help	`ShowHtmlHelp`

Additional Enhancements

The following sections describe additional enhancements to LabWindows/CVI 5.5.

Report Generation

CVI provides two instrument drivers to help you generate reports. You can easily create and print formatted text reports with the `toolslib\reportgen\NIReport.fp` instrument driver. You also can use the `samples\activex\word\wordreport.fp` instrument driver to create and manipulate reports in Microsoft Word.

Console Applications Support

A console application uses the host system's Standard I/O window to display standard output and to read standard input, instead of using the LabWindows/CVI Standard I/O window. You enable the creation of console applications by checking the **Create Console Application** checkbox in the Target Settings dialog box. You can use a console window for standard I/O, even when your application is not a console application, by passing `HOST_SYSTEM_STDIO` to the Utility Library `SetStdioPort` function.

Run-Time Engine and Windows Shutdown

Applications using the LabWindows/CVI Run-time Engine no longer interfere with the Windows shutdown process. In previous versions of the LabWindows/CVI Run-time Engine, if you attempt to shut down, restart, or logoff Windows while an application that uses the LabWindows/CVI Run-time Engine is running, the system might not shut down immediately and might always shut down regardless of the shutdown command issued. The LabWindows/CVI 5.5 Run-time Engine no longer interferes with the Windows shutdown process.

Updated Windows SDK

The Windows SDK has been updated to include the functions from the Windows 2000 Platform SDK. This includes all of the Windows 98 functions as well. Only a subset of the Windows SDK is included with the LabWindows/CVI beta versions. To use the SDK from your 4.*x* or 5.*x* version of LabWindows/CVI, perform the following steps in order:

1. Rename the LabWindows/CVI 5.5 `sdk directory (cvi55\sdk)` to `sdknew (cvi55\sdknew)`.
2. Copy the LabWindows/CVI 4.*x* or 5.*x* directory `(cvi50\sdk)` to `cvi55\sdk`.

Copy the `cvi55\sdknew` over `cvi55\sdk` replacing files that already exist.

Formatting and I/O Library Fmt/FmtOut/FmtFile Functions

The `Fmt`, `FmtFile`, and `FmtOut` functions now support the 'j' modifier, which allows you to specify characters that are inserted between each element when you use the repeat code to format multiple items.

Changes to the LabWindows/CVI User Manual

Chapter 3—Project Window

Build Menu

Add the following note to the end of the *Create Distribution Kit* section.

To copy a LabWindows/CVI generated distribution kit from a hard drive to individual floppy disks, place the following files on each disk.

Disk 1:

`setup.exe`

`projectname``.001`

(and any additional files for which space was allocated during build)

Disk 2:

`projectname``.002`

Disk 3:

`projectname``.003`

(If needed, continue the disks using the Disk *x* and *`projectname`*`.00`*`x`* formula.)

Tools Menu

Add the following two items to the Tools menu items:

User Interface Localizer—Use this utility to help you translate your user interface resource files (UIRs) into other languages. Refer to the help dialog box in the User Interface Localizer for more information on using this utility.

UI To Code Converter—Use this utility to convert your user interface files into code. This is useful when you want to distribute a single executable file without any UIR files. Refer to the documentation in the help dialog box in the UI to Code Converter for more information on using this utility.

Add the following paragraph to the end of the *Choose Server Panel* subsection in the *Create ActiveX Automation Controller* section.

Show All—Use this box if you want to see all ActiveX Automation Servers. If this checkbox is not checked, then the wizard shows only the ActiveX servers that are intended to be used through ActiveX Automation. The additional servers displayed when you check the **Show All** box might not function correctly when used through ActiveX Automation.

Delete the descriptions of the **Call Mechanism** and **Generate Per-Object Property Access Functions** from the *Configure Panel* subsection in the *Create ActiveX Automation Controller* section.

Add the following paragraph to the *Configure Panel* subsection in the *Create ActiveX Automation Controller* section.

Property Access Functions—Use this control to determine how property access functions are generated. Select the **Per Property** option to make the wizard generate property get/set functions for each property in the server. The **Per Property** option is available only if all interfaces in the server are dual interfaces. When this option is available, it provides the most efficient method of communicating with the server. Select the **Per Object** option to make the wizard generate property get/set functions for each object in the server. Select the **Per Server** option to make the wizard generate single **GetProperty** and **SetProperty** functions through which you get and set properties for all objects in the server.

Delete the words "of up to eight characters" from the *Instrument Prefix* subsection in the *Advanced Panel* section describing the Automation Controller Advanced Options dialog box.

Add the following sentence to the end of the description of the Automation Controller Advance Options dialog box in the *Advanced Panel* section: "Click on the **OK** button to accept the changes made in the dialog box and to close the dialog box."

Help Menu

Delete the *Search For Help On* subsection from the *Help Menu* section.

Options Menu

Add the following to the end of the *Run Options* section.

> **Update Windows When Suspending**—When you enable this checkbox, the system redraws the panels in the program being debugged, if necessary, before suspending due to a breakpoint, watchpoint, or a **Step** command. This is useful when you are stepping through code that calls User Interface Library functions that set attributes or values but do not refresh the panel or control immediately.

Add the following to the description to the end of the *Environment* section.

> **Bring Debug Output Window to Front Whenever Modified**—Check this box if you want the Debug Output window to be brought to the front whenever text is added to it.

Add the following to the description of the *Source Code Control Options* section.

> **Use Global Source Code Control Settings**—Choose this radio button if you want to use the source code control settings shared by all projects. When you choose this radio button, the controls on the dialog allow you to set the global source code control settings.
>
> **Use Project Source Code Control Settings**—Choose this radio button if you want to use source code control settings that are specific to the currently loaded project. When you choose this radio button, the controls on the dialog allow you to set the source code control settings that are specify to this project.

Windows Menu

Add the following to the end of the *Run-Time Errors* section.

> Note that you can use the utility library `DebugPrintf` function to send debug output to the Debug Output window.

Add the following paragraph to the end of the *Debug Output* section.

> This window is useful for debug strings output by your program. Unlike the Standard I/O window, this window is accessible while your program is suspended. You can choose to have the Debug Output window come to the front whenever text is added to the window by checking the **Bring Debug Output Window to Front Whenever Modified** option in the Environment Options dialog box.

Changes to the LabWindows/CVI Programmer Reference Manual

Chapter 4—Creating and Distributing Standalone Executables and DLLs

Add the following to the *Necessary Files for Running Executable Programs* section.

NIReports—If your LabWindows/CVI generated executable uses the ActiveX Automation library you must distribute the `CVIAUTO.DLL` located in the Windows system directory. When using the Create Distribution Kit, the DLL is automatically included in your distribution. This DLL requires Windows NT 4.0 Service Pack 3, Windows 95 with DCOM95 1.3, or Windows 98. The Create Distribution Kit features allows you to include DCOM95 in your distribution.

CVI OpenGL Control—If your LabWindows/CVI generated executable uses the CVIOGL 3D Plotting instrument driver you must include the `CVIOGL.DLL` file in your distribution kit. When distributing your executable on an earlier version of Windows 95 than OEM 2, you must also distribute the OpenGL files `OPENGL32.DLL` and `GLU32.DLL`.

Add the following to the *Introduction to the Run-Time Engine* section.

Using Third-Party Installation Packages—When distributing a LabWindows/CVI application, you must distribute the LabWindows/CVI Run-time Engine with your application. If you wish to incorporate the installation of the LabWindows/CVI Run-time Engine in a third-party installation package, you must either distribute the LabWindows/CVI Run-time Engine installation separately, or integrate the launching of the LabWindows/CVI Run-time Engine installation from your installer. A readme file and example InstallShield scripts in the `\cvi\bin\rteinst` subdirectory specify how to silently launch and install the LabWindows/CVI Run-time Engine.

DCOM95 required on Windows 95—If you are using the LabWindows/CVI ActiveX Automation library on Windows 95, you must install DCOM95. The Create Distribution Kit feature allows you to include DCOM95 in your distribution. The installer for DCOM95 is available in `cvi/redist/dcom95`.