NATIONAL INSTRUMENTS®
*The Software is the Instrument*

# LabWindows®/CVI Version 4.0.1 Release Notes for Windows

## Welcome to LabWindows/CVI

These release notes contain the installation instructions, system requirements, new features, and updated information to help you begin using LabWindows/CVI, version 4.0.1.

## Contents



---

# LabWindows/CVI Installation for Windows

Before getting to work on your data acquisition and instrument control applications, you must install LabWindows/CVI on your computer. The LabWindows/CVI setup program does this for you in a process that lasts approximately ten minutes.

## Minimum System Requirements for Windows 95 and NT

To run LabWindows/CVI for Windows 95 and NT, you must have the following:

- Microsoft Windows 95, or Microsoft Window NT operating system, version 3.51 or later
- Personal computer using at least a 33 MHz 80386 or higher microprocessor
- VGA resolution (or higher) video adapter
- Minimum of 16 MB of memory
- 40 MB free hard disk space
- Microsoft-compatible mouse

## Minimum System Requirements for Windows 3.1

To run LabWindows/CVI for Windows 3.1, you must have the following:

- MS-DOS, version 3.1 or later
- Microsoft Windows operating system, version 3.1 or later
- Personal computer using at least a 25 MHz 80386 or higher microprocessor (National Instruments recommends a 33 MHz 80486 or higher microprocessor)
- VGA resolution (or higher) video adapter
- Math coprocessor or one of the following coprocessor emulation programs
  - WEMM387.386 from WATCOM
  - Q387 from Quickware
- Minimum of 8 MB of memory

- 30 MB free hard disk space
- Microsoft-compatible mouse

# Installing LabWindows/CVI on a Computer

1. Make sure that your computer and monitor are turned on and that you have installed Microsoft Windows.

2. If installing from diskette, make backup copies of your LabWindows/CVI diskettes and store the originals in a safe place.

3. Close all open Windows applications, and leave the operating system in Windows.

4. Insert the installation CD into the CD-ROM Drive, or insert Disk 1 into drive A or drive B and close the drive door, if necessary.

5. For Windows 95 or Windows NT version 4.0 or later, choose the **Run** option from the Desktop Taskbar. For Windows 3.1 or Windows NT version 3.51, choose **Run** from the **File Menu** in the Program Manager.

6. Type `x:setup` (where `x` is the drive you are using) in the command line box and click on **OK**.

7. Follow the instructions that appear in the dialog boxes.

   **Note:** ***When installing LabWindows/CVI forWindow 95 or NT, you must choose your compatible compiler. If sometime later you want to change your choice, run the installation program again and select the option that allows you to choose a new compatible compiler. This method is much faster than changing your compiler choice by reinstalling LabWindows/CVI.***

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

# Installing the Run-Time Engine on a Computer

The Run-Time Engine is now installed during the LabWindows/CVI installation. Included with LabWindows/CVI is a separate Run-Time Engine distribution disk (or a directory on the CD-ROM) which has its own installation program so that you can make copies for distribution as permitted by your license agreement.

To install the Run-Time Engine from the Run-Time Engine distribution disk, follow the directions in the *Installing LabWindows/CVI on a Computer* section. Refer to Chapter 7, *Creating and Distributing Standalone Executables and DLLs*, of the *LabWindows/CVI Programmer Reference Manual* for a description of the LabWindows/CVI Run-Time Engine.

# Installing the VISA Library

You need the VISA Library to use the new instrument driver standard accepted by the VXIplug&play Systems Alliance. (VISA stands for Virtual Instrument Software Architecture.) You should install VISA if you plan to use GPIB or VXI instrument drivers. Run the setup program, `setup.exe`, that appears either on the VISA distribution disk that comes with LabWindows/CVI or in the `\VISA` directory on the CD-ROM.

# What Is Installed by the Setup Programs

The setup program for LabWindows/CVI installs the LabWindows/CVI development environment and related files. The full installation includes sample programs that illustrate many of the new features in LabWindows/CVI, and tutorial programs that you can use with the *Getting Started with LabWindows/CVI* manual. LabWindows/CVI and the associated files are installed on your hard disk in the subdirectories listed in Table 1.

Table 1. LabWindows/CVI Subdirectories

| Directory Name | Contents |
|---|---|
| `bin` | LabWindows/CVI Library files |
| `extlib` | LabWindows/CVI Import Library files for external compilers |
| `fonts` | Font files |
| `include` | #include files associated with libraries |
| `instr` | Instrument modules |
| `samples` | Source code for sample programs |
| `sdk` | Microsoft Windows SDK import library, header and utility files |
| `toolslib` | Utility instrument drivers and ECHO controls |
| `tutorial` | Programs you use with the *Getting Started with LabWindows/CVI* tutorial sessions |
| `vxd` | VxD and Device Driver templates and documentation files |

The setup program for the Run-Time Engine distribution disk installs two subdirectories within the installation directory as shown in Table 2.

Table 2. LabWindows/CVI Run-Time Engine Subdirectories

| Directory Name | Contents |
|---|---|
| `bin` | Message file (`msgrt`*`n`*`.txt`), and other system files |
| `fonts` | Font files supplied by NI |

The message file (msgrt*n*.txt where *n* is the version of the Run-Time Engine) is a text file containing the error messages displayed by the Run-Time Engine. You can translate the message file into other languages. Refer to the *Configuring the Run-Time Engine* section in Chapter 7, *Creating and Distributing Standalone Executables and DLLs*, of the *LabWindows/CVI Programmer Reference Manual* for information on translating the message file.

## Installing on a Network

Contact National Instruments for licensing information if you plan to run LabWindows/CVI on a network.

# What's New and Different in LabWindows/CVI 4.0.1

This document includes information about changes and enhancements in LabWindows/CVI 4.0.1 that have been made since LabWindows/CVI 3.1.

## Windows 95 and Windows NT Support

In LabWindows/CVI under Windows 95 and NT, you can now do the following.

- Load 32-bit DLLs, via the standard import library mechanism
- Create 32-bit DLLs and DLL import libraries
- Create library files as well as object files
- Call the LabWindows/CVI libraries from executables or DLLs created in any of the four compatible external compilers. These are Microsoft Visual C/C++, Borland C/C++, WATCOM C/C++, and Symantec C/C++.

- Create object files, library files, and DLL import libraries that can be used in the compatible external compilers
- Load object files, library files, and DLL import libraries created in any of the compatible external compilers.
- Call Windows SDK functions
- Have a run-time stack of up to 1,000,000 bytes
- Run on a machine without a floating point coprocessor

## Using DLLs in Windows 95 and NT

### Loading 32-bit DLLs under Windows 95 and NT

Under Windows 95 and NT, LabWindows/CVI can load 32-bit DLLs. Because the environment is 32-bit, special glue code is no longer needed. LabWindows/CVI links to DLLs using the standard 32-bit DLL import libraries that you generate when you create 32-bit DLLs in any of the compatible compilers or in LabWindows/CVI.

The DLL import library contains the file name of the DLL. LabWindows/CVI uses the standard Windows DLL search algorithm to find the DLL. For this reason, DLL path (`.pth`) files do not work under Windows 95 and NT.

LabWindows/CVI for Windows 95 and NT does not load 16-bit DLLs. If you want to do so, you must obtain a 32-to-16-bit thunking DLL and a 32-bit DLL import library.

### Generating an Import Library

If you do not have a DLL import library or the one you have contains references not exported by the DLL, you can generate an import library in LabWindows/CVI from a header file.

### Default Unloading/Reloading Policy

In LabWindows/CVI for Windows 95 and NT, DLLs are, by default, unloaded after each execution of a user program in the development environment. You can change the default behavior by turning off the **Unload DLLs After Each Run** option in the **Run Options** dialog box.

## Compatibility with External Compilers

LabWindows/CVI for Windows 95 and NT can be compatible at the object code level with any of the four compatible external compilers (Microsoft Visual C/C++, Borland C/C++, WATCOM C/C++, and Symantec C/C++). Because these compilers are not compatible with each other at the object code level, LabWindows/CVI can be compatible with only one external compiler at a time.

Object, library, or DLL import library files loaded in LabWindows/CVI must have been created either in the selected compatible compiler, or in LabWindows/CVI under the same compatibility choice.

## Using the LabWindows/CVI libraries in External Compilers

Under Windows 95 and NT, you can use the LabWindows/CVI libraries in any of the four compatible external compilers. You can create executables and DLLs that call the LabWindows/CVI libraries. All of the libraries are contained in DLLs.

### Multithreaded Safe LabWindows/CVI Libraries

You can use the following libraries in more than one thread at a time:

- Analysis and Advanced Analysis
- GPIB (if you are using a native 32-bit driver)
- VXI
- VISA
- RS-232
- Data Acquisition
- Easy I/O for DAQ
- Formatting and I/O  (except for the Standard I/O Window)
- ANSI C  (except for the Standard I/O Window)

## Creating Executables in LabWindows/CVI

You can create true 32-bit Windows executables in LabWindows/CVI for Windows 95 and NT. Under Windows 95 and NT, the LabWindows/CVI run-time libraries come in DLL form. The same DLLs are used by standalone executables created in LabWindows/CVI and executables created in external compilers. If more than one program is run at a time, only one copy of each DLL  is loaded.

You can also add Standard Windows version information to executables when you create them.

### Creating DLLs in LabWindows/CVI

In LabWindows/CVI for Windows 95 and NT, you can create 32-bit DLLs. Along with each DLL, LabWindows/CVI creates a DLL import library for your compatible compiler. You can choose to create DLL import libraries compatible with all four compatible external compilers.

You can also add Standard Windows version information to DLLs when you create them.

## Customizing an Import Library

If you need to perform some special processing in your DLL import library, you can customize it by using LabWindows/CVI to generate a source file version of the import library.

## Automatic Inclusion of Type Library Resource for Visual Basic

When creating a DLL that has an associated function panel, you have the option to create a Type Library resource automatically and include it in the DLL. When you use this option, Visual Basic users can call the DLL without having to use a header file containing `Declare` statements for the DLL functions.

## Calling Windows SDK Functions in LabWindows/CVI

You can call Windows SDK Functions in LabWindows/CVI for Windows 95 and NT. You can install on-line help for the Windows SDK functions with the CD-ROM installation.

## Run-Time Stack Size

Under LabWindows/CVI for Windows 95, and NT, the maximum stack size is 1,000,000 bytes. The default size and minimum size is 100,000. Under LabWindows/CVI for Windows 3.1, your run-time stack is limited to a maximum of 16,384 bytes.

## No Floating Point Coprocessor Required for Windows 95/NT

No floating point coprocessor or emulator is required to run LabWindows/CVI for Windows 95 and NT or to use the DLLs containing the LabWindows/CVI libraries for Windows 95 and NT.

## New Predefined Macros

New predefined macros have been added for Windows 95 and NT. Refer to Chapter 2, *Compiler/Linker Issues for Windows 95/NT*, in the *LabWindows/CVI Programmer Reference Manual* for more information.

# General Compiler/Linker Enhancements

## Maximum Nesting of Include Files

The maximum nesting of `#include` statements has been increased from 8 to 32.

## C++-Style Comment Markers

You can use double slashes (//) to begin a comment. The comment continues until the end of the line.

## Duplicate Typedefs

The LabWindows/CVI compiler no longer reports an error on multiple definitions of the same `typedef` identifier, as long as the definitions are identical.

## Structure Packing Pragma

The `pack pragma` now works under Windows 3.1 and Windows 95/NT. You can use it to specify the maximum alignment factor for elements within a structure.

## Non-Project-Specific User-Defined Include Paths

The **Include Paths** dialog box now has two lists, one for include paths specific to the project, and one not specific to the project.

## VXIplug&play Include Directory

When you install VXI*plug&play* instrument drivers, the include files for the drivers are placed in a specific VXI*plug&play* include directory. LabWindows/CVI now searches that directory for include files.

## New Compiler/Linker/Run-Time Errors and Warnings

New Compiler/Linker/Run-Time Errors and Warnings have been added to LabWindows/CVI. Refer to Appendix A, *Errors and Warnings*, in *the LabWindows/CVI Programmer Reference Manual*.

## Calling Convention for Exported Functions (Windows 95/NT Only)

LabWindows/CVI for Windows 95/NT support the `stdcall` and the `cdecl` (or WATCOM stack-based) calling convention for exported DLL functions.

## Exporting DLL Functions and Variables (Windows 95/NT Only)

You can use one of two methods for exporting functions and variables when creating a DLL in LabWindows/CVI.

- **Include File Method—**You can identify symbols to export via include files that contain the declarations of the symbols you want to export.
- **Export Qualifier Method—**You can mark each function and variable you want to export with an export qualifier.

## Program Entry Points (Windows 95 and NT only)

In Windows 95 and NT, you can use `WinMain` instead of `main` as the entry point to your program.

# LabWindows/CVI Development Environment

## Project Window Changes

### Auto Save Project

The **Auto Save Project** command replaces the **Read Only** command in the **File** menu of the Project Window.

### Print

A **Print** command now appears in the **File** menu in the Project window. It brings up a selectable list of all of the files in the project that are printable. You can checkmark the files you want to print.

### Most Recently Closed Files

The **File** menu in all windows now contains two lists.

- A list of the four most recently closed files (other than project files)
- A list of the four most recently closed project files

### Target

Use the **Target** command in the **Build** menu to specify whether you want to create an executable, a DLL, or a static library.

### Instrument Driver Support Only

With the **Instrument Driver Support Only** command in the **Build** menu, you can create a DLL or executable that contains one or more instrument drivers but does not require the full LabWindows/CVI run-time engine DLL. A DLL or executable created in this way uses `instrsup.dll`, which is much smaller.

### Create Distribution Kit Enhancements

The **Create Distribution Kit** command now includes an uninstall program on the distribution disk.

The **Replace Existing Files** ring control now has a **Check Version** option.

The following new options now appear in the Create Distribution Kit dialog box.

- **Distribute Objects and Libraries for All Compilers—**This checkbox appears in LabWindows/CVI for Windows 95 and NT to help you distribute object files, static libraries, and DLL import libraries for all of the compatible external compilers.
- **Install Low-Level Support Driver**—This option appears in Windows 95 and NT so you can choose whether to install the LabWindows/CVI low-level support driver on the end-user's computer. This driver is required by some Utility Library functions.
- **Advanced**—You can use this option to specify a executable file to run after the end-user installation is complete, customize the installed program group name and installation name, and select a custom installation script file.

### Easy I/O for DAQ added to Library Menu

In LabWindows/CVI version 3.1, Easy I/O for DAQ was distributed as an instrument driver. The Easy I/O for DAQ is now found in the **Library** menu. The *LabWindows/CVI Standard Libraries Reference Manual* now contains the Easy I/O Library function reference.

### Minimize All (Windows 95 only)

Use the **Minimize All** command in the **Window** menu to hide all of the LabWindows/CVI windows.

### CloseAll

Use the **Close All** command in the **Window** menu to close all of the LabWindows/CVI windows, excluding the Project window and any User Interface Library panels displayed by a program.

### Compiler Options

The **Compiler Options** dialog box from the **Options** menu under Windows 95 and NT now indicates the compatible compiler.

### Compiler Defines

The **Compiler Defines** dialog box from the **Options** menu now contains a list of the macro names and values that are predefined by LabWindows/CVI.

### Include Paths

The **Include Paths** dialog box from the **Options** menu now has two lists of paths in which to search for include files. The top list is saved with the project file. The bottom list is saved from one LabWindows/CVI session to another on the same machine, regardless of project.

## Source Window Changes

### Notification of External Modification

After a file in a source window has been modified externally, a dialog box appears where you can update the Source window from the file on disk, overwrite the file on disk with the contents of the source window, or do nothing.

### Backspace to Beginning of Word

The **Backspace to Beginning of Word** command is available only from the keyboard, using the <Ctrl-Shift-Backspace> key combination.

### Context Menus

You can bring up a context menu in the Source window by pressing the right mouse button. The context menu contains a selection of the most commonly used menu commands from the Source window menu bar. The selection of commands is different depending on whether the mouse is over the text editing area or over the line number or line icon area.

### Recall Panel

The algorithm which recognizes the function name in source code text has been improved. **Recall Panel** now recognizes a function name even if it is not followed by a parameter list. **Recall Panel** can now find a function call that is embedded within an expression or that is split among several lines.

### Find Function Panel

The **Find Function Panel** command now can find the panel for a function based on a substring appearing anywhere in its name. If more than one function is found, a dialog box appears with a list of the functions. <Shift-Ctrl-P> is now the short-cut key.

### Terminate Execution Shortcut Key for Windows 95/NT

The short-cut key for terminating execution of a suspended program or suspending a running program is <Ctrl-F12> on Windows 95 and NT.

### Activate Panels When Resuming

You can now use **Activate Panels when Resuming** in the **Run** menu to choose whether the user interface panels in your programs are reactivated every time you resume execution during debugging.

### Colors and Syntax Coloring

In the **Options** menu, the Color dialog box contains new color types for syntax coloring. LabWindows/CVI color codes the various types of tokens in your source and include files.

Under Windows 95 and Windows NT 4.0 or later, the Color dialog box has a new checkbox, **Use System Colors**. When this option is enabled, colors are automatically assigned based on the system colors defined in the Appearance tab in the Windows Display Properties dialog box.

The **Syntax Coloring** command toggles syntax coloring. Use the **User Defined Tokens for Coloring** command to specify tokens to be colored differently in the source window.

## Function Panel Window Changes

### Select Variable

The new **Select Variable** command gives you a list of previously used variables or expressions having data types that are compatible with the currently active function panel control. The command is enabled only when the currently active function panel control is one that accepts text entry.

## Changes to the Function Tree and Function Panel Editors

### Maximum Number of Levels Increased to Eight

The new maximum number of levels in a function tree is 8, up from 4 in previous versions.

### Create DLL Project (Windows 95/NT Only)

The **Create DLL Project** command is new in the **Options** menu of the Function Tree Editor window for Windows 95 and NT. The **Create DLL Project** command creates a LabWindows/CVI project (`.prj`) file you can use to create a dynamic link library (`.dll`) from the program file associated with the function panel (`.fp`) file.

### VXIplug&play Style

The **VXIplug&play Style** command is new in the **Options** menu in the Function Tree Editor window for Windows 95 and NT. It works in conjunction with the **Create DLL Project** command. It sets various options so that the DLL you create from your instrument driver can be VXI*plug&play* compatible, including the installation program.

### Numeric Control Supports Additional Data Types

You can now use the following data types in a numeric control.

```
int
short
char
unsigned int
unsigned short
unsigned char
double
float
```

## Searching for DLLs Associated with .fp Files

Starting in LabWindows/CVI version 4.0 for Windows 3.1, if there is no instrument driver program file (`.lib`, `.obj`, `.c`, `.pth`, or `.dll`) in the same directory as the `.fp` file, LabWindows/CVI looks for a DLL with the same base name as the `.fp` file using the standard Windows search algorithm.

## Handling Hardware Interrupts under Windows 95 and NT

In Windows 3.1, you can handle hardware interrupts in a DLL. In Windows 95, you must handle hardware interrupts in a VxD. In Windows NT, you must handle hardware interrupts in a kernel mode driver. VxDs and kernel mode drivers cannot be created in LabWindows/CVI. Instead, you must create them in Microsoft Visual C/C++, and you must have the Microsoft Device Driver Developer Kit (DDK).

LabWindows/CVI includes source code template files for a VxD and a kernel mode driver. It also includes a sample main program to show you how to read and write registers on a board.

## Online Help

The online help for LabWindows/CVI now includes all printed documentation except for portions of *Getting Started with LabWindows/CVI.*

# User Interface Library Enhancements

## New Canvas Control

You can use a new User Interface control called a *canvas control* as an arbitrary drawing surface. You can draw text, shapes, and bitmap images. An offscreen bitmap is maintained so that you can restore the appearance of the canvas when the region is exposed. If you want to display images that are not rectangular or that have "holes" in them, you can use bitmaps that have a transparent background.

Each canvas has a pen which is used by the canvas drawing functions. Set the canvas pen attributes individually using `SetCtrlAttribute`.

Although you can call the drawing functions at any time, they are most efficient when called from within a batch drawing operation. A batch drawing operation consists of a call to `CanvasStartBatchDraw`, followed by one or more calls to the canvas drawing functions, followed by a call to `CanvasEndBatchDraw`.

Use the two structures, `Rect` and `Point`, to specify locations and areas in the canvas control. Many canvas control functions use these structures.

See Chapter 3, *Programming with the User Interface Library* in the *LabWindows/CVI User Interface Reference Manual* for more information.

## Native System User Interface

LabWindows/CVI uses the Windows 95 native system dialog boxes for the `FileSelectPopup`, `MultiFileSelectPopup`, and `DirSelectPopup` functions. Menus, buttons and checkboxes are drawn in the Windows 95 style.

## CodeBuilder Changes

### WinMain

The Generate All Code and Generate Main Function dialog boxes uses a new checkbox, **Generate WinMain() instead of main()**. Enable this option if you want to use `WinMain` instead of `main` for your main program.

### DLL Projects

If your project target is a DLL, neither `WinMain` or `main` is generated. Instead, LabWindows/CVI generates a `DLLMain` function. The bulk of the User Interface function calls, however, are generated in a function call `InitUIForDLL`. You must call `InitUIForDLL` in your DLL at the point you want to load and display panels.

### InitCVIRTE and CloseCVIRTE Functions

When you link your executable or DLL in an external compiler, you must include a call to the `InitCVIRTE` function in `WinMain`, `main`, or `DLLMain`. In a DLL, you must also include a call to `CloseCVIRTE`. See Chapter 3, *Linker/Compiler Issues for Windows 95/NT* in the *LabWindows/CVI Programmer Reference Manual* for more information.

CodeBuilder automatically generates the necessary calls to `InitCVIRTE` and `CloseCVIRTE` in your `WinMain`, `main`, or `DLLMain` function. It also automatically generates a `#include` statement for the `cvirte.h` file.

**Note:** ***In LabWindows/CVI version 4.0.1, the number of parameters to this function was increased from one to three. Executables and DLLs created using just one parameter will continue to work with the Run-time Engine for LabWindows/CVI version 4.0.1.***

### New Qualifier for Callback Functions

The `CVICALLBACK` macro should now precede the function name in the declarations and function headers for all user interface callbacks. This ensures that the compiler treats the functions as `cdecl` (or stack-based in WATCOM), even when the default calling convention is `stdcall`. `CVICALLBACK` is defined in `cvidefs.h`, which is included by `userint.h`. The header files generated by the User Interface Editor and the source code generated by CodeBuilder include the `CVICALLBACK` macro where necessary.

## New Graph and Strip Chart Features

Graphs and Strip Charts now have the following new features.

- Two Y Axis—Graphs now have a second Y axis on the right side of the graph.
- Arbitrary Text for Tick Labeling—You can now create your own strings for tick labels on graphs and strip charts.
- Axis Scaling—You can now scale the values shown on graph and strip chart axes.
- Reversing Polarity—You can now reverse the polarity of the graph and strip chart axes. (Graph the minimum value at the top rather than the maximum value.)
- Zooming and Panning—You can expand or contract the viewport around a particular point in graph controls.

## Image Bits Functions Superseded by New Functions

LabWindows now has a more general set of functions for handling bitmap images as independent objects.

## 24-Bit and 32-Bit Pixel Depth Supported in Image Bits Functions

You can now use a 24-bit and 32-bit pixel depth in the four picture control image bits functions.

## Windows Metafiles

You can now use Windows Metafiles (`.WMF`) with LabWindows/CVI for Windows 95/NT.

## New System Attribute Functions

The `SetSystemAttribute` and `GetSystemAttribute` functions set attributes that apply to the User Interface Library in general, rather than to particular instances of user interface objects.

```
ATTR_REPORT_LOAD_FAILURE
ATTR_ALLOW_MISSING_CALLBACKS
ATTR_SUPPRESS_EVENT_PROCESSING
ATTR_ALLOW_UNSAFE_TIMER_EVENTS
ATTR_TASKBAR_BUTTON_TEXT
ATTR_TASKBAR_BUTTON_VISIBLE
```

## New User Interface Functions

### Canvas Control Functions

```
CanvasDrawPoint
CanvasDrawLine
CanvasDrawLineTo
CanvasDrawRect
CanvasDimRect
CanvasDrawRoundedRect
CanvasDrawOval
CanvasDrawArc
CanvasDrawPoly
CanvasDrawText
CanvasDrawTextAtAPoint
CanvasDrawBitmap
CanvasScroll
CanvasInvertRect
CanvasClear
CanvasStartBatchDraw
CanvasEndBatchDraw
CanvasUpdate
CanvasSetClipRect
CanvasDefaultPen
CanvasSetPenPosition
CanvasGetPenPosition
CanvasGetPixel
MakeRect
MakePoint
```

RectSet
RectSetFromPoints
RectSetBottom
RectSetRight
RectSetCenter
RectOffset
RectMove
RectGrow
PointSet
RectBottom
RectRight
RectCenter
RectEqual
RectEmpty
RectContainsPoint
RectContainsRect
RectSameSize
RectUnion
RectIntersection
PointEqual
PointPinnedToRect

### Bitmap Functions

NewBitmap
GetBitmapFromFile
GetCtrlBitmap
GetCtrlDisplayBitmap
GetPanelDisplayBitmap
ClipboardGetBitmap
DiscardBitmap
SetCtrlBitmap
ClipboardPutBitmap
GetBitmapInfo
AllocBitmapData
GetBitmapData

### Other New Functions

ClipboardGetText
ClipboardSetText

## New User Interface Attributes and Values

### Panel Attributes

ATTR_ACTIVE
ATTR_CONFORM_TO_SYSTEM
ATTR_FLOATING

### Font Values

VAL_MESSAGE_BOX_FONT
VAL_MESSAGE_BOX_META_FONT

### Menu and Menu Item Attributes

ATTR_DRAW_LIGHT_BEVEL
ATTR_DIMMER_CALLBACK

### Key Modifiers and Virtual Keys

VAL_UNDERLINE_MODIFIER

### Control Styles for ATTR_CTRL_STYLE

CTRL_CANVAS

### Control Attributes

For all controls

ATTR_OVERLAPPED

For list boxes

ATTR_ALLOW_ROOM_FOR_IMAGES
ATTR_HILITE_CURRENT_ITEM

For Graphs and Strip Charts

ATTR_INNER_LOG_MARKERS_VISIBLE
ATTR_XAXIS_GAIN
ATTR_XAXIS_OFFSET
ATTR_XUSE_LABEL_STRINGS
ATTR_YAXIS_GAIN
ATTR_YAXIS_OFFSET
ATTR_YAXIS_REVERSE
ATTR_YUSE_LABEL_STRINGS
ATTR_ACTIVE_YAXIS
ATTR_ENABLE_ZOOMING_AND_PAN
ATTR_XREVERSE
ATTR_CURSOR_YAXIS
ATTR_PLOT_ORIGIN
ATTR_PLOT_SNAPPABLE
ATTR_PLOT_YAXIS

Values associated with the ATTR_PLOT_ORIGIN attribute.

VAL_LOWER_LEFT
VAL_LOWER_LEFT
VAL_UPPER_LEFT
VAL_LOWER_CENTER
VAL_CENTER_CENTER
VAL_CENTER_CENTER

```
VAL_LOWER_RIGHT
VAL_CENTER_RIGHT
VAL_UPPER_RIGHT
```

For canvas controls

```
ATTR_PEN_WIDTH
ATTR_PEN_STYLE
ATTR_PEN_COLOR
ATTR_PEN_FILL_COLOR
ATTR_PEN_MODE
ATTR_PEN_PATTERN
ATTR_DRAW_POLICY
ATTR_OVERLAPPED_POLICY
ATTR_DRAW_POLICY
ATTR_OVERLAPPED_POLICY
ATTR_XCOORD_AT_ORIGIN
ATTR_XSCALING
ATTR_YCOORD_AT_ORIGIN
ATTR_YSCALING
```

Values associated with the ATTR_DRAW_POLICY attribute.

```
VAL_UPDATE_IMMEDIATELY
VAL_MARK_FOR_UPDATE
VAL_DIRECT_TO_SCREEN
```

Values associated with the ATTR_OVERLAPPED_POLICY attribute.

```
VAL_DEFER_DRAWING
VAL_DRAW_ON_TOP
```

Values associated with the ATTR_PEN_MODE attribute.

```
VAL_COPY_MODE
VAL_OR_MODE
VAL_XOR_MODE
VAL_AND_NOT_MODE
VAL_NOT_COPY_MODE
VAL_NOT_OR_MODE
VAL_NOT_XOR_MODE
VAL_AND_MODE
```

# Updates to the Standard Libraries

## Changes to the ANSI C Library and Low-Level I/O Functions

### errno Set by File I/O Functions

The `errno` global variable is now set to indicate specific error conditions by the ANSI C file I/O functions and the low-level I/O functions.

### New Low-Level I/O Function

The low-level I/O functions now include the `sopen` function. Use `sopen` for sharing files with other applications.

### New ANSI C Library Function

The ANSI C library now includes the `fdopen` function.

## Changes to the Formatting and I/O Library

### Improved File I/O Error Reporting

When a file I/O error occurs in a Formatting and I/O function, two new functions give you more specific error information.

```
GetFmtIOError
GetFmtIOErrorString
```

## Changes to the GPIB Library

### New iblock and ibunlock functions for GPIB-ENET

GPIB-ENET now has the `iblock` and `ibunlock` functions. The `iblock` function blocks other processes from accessing the specified GPIB-ENET board or device. You can release the lock by calling the `ibunlock` function with the same board or device descriptor.

In addition, you can now use the `IbaBlockIfLocked` constant when calling the `ibask` function. It enables and disables the blocking of a process when it attempts to use a locked board.

### Notification of SRQ and Other GPIB Events

Synchronous Callbacks

Under Windows 95 and NT, the conditions under which the `ibInstallCallback` function can be called have been expanded. You can specify the callback to be invoked on the occurrence of any board-level or device-level condition on which you can wait using the `ibwait` function.

Asynchronous Callbacks

Under Windows NT, or under Windows 95 with the native 32-bit driver, you can now install *asynchronous* callbacks. Install asynchronous callbacks with the `ibnotify` function `ibInstallCallback`.

### Multithreading

You can now call GPIB functions from more than one thread at the same time in an external compiler under Windows NT or Windows 95 with the native 32-bit driver. LabWindows/CVI contains the following new GPIB functions.

```
ThreadIbcnt
ThreadIbcntl
ThreadIberr
ThreadIbsta
```

**Note:** ***Refer to the LabWindows/CVI Standard Library Reference Manual for detailed information on the different levels of functionality depending on platform, GPIB board, and version of GPIB driver.***

## Changes to the RS-232 Library

Use the new `InstallComCallback` function to install a callback function for a particular COM port. The callback function is called whenever a specified event occurs on the COM port. Events occur when the transmit queue is empty, an ending character is received, or the inqueue is above a specified count.

## Changes to the DDE Library

Use the new `BroadcastDDEDataReady` function to direct a DDE server send data to all clients that have hot or warm links to a specified topic and item.

## Changes to the TCP Library

Use the new `DisconnectTCPClient` function to direct a TCP server terminate a connection with a client.

## Changes to the Utility Library

### New Utility Library Functions

The following new functions get the CVI version and platform.

```
CVILowLevelSupportDriverLoaded
GetCVIVersion
GetCurrentPlatform
```

The following function obtains the directory of the specified DLL module.

```
GetModuleDir
```

The following new functions controls the CVI run-time checking.

```
GetBreakOnLibraryErrors
GetBreakOnProtectionErrors
SetBreakOnLibraryErrors
SetBreakOnProtectionErrors
```

The following functions are new extended functions.

```
LoadExternalModuleEx
ReadFromPhysicalMemoryEx
WriteToPhysicalMemoryEx
```

The following new function determines if another application instance is running .

```
CheckForDuplicateAppInstance
```

The following new functions are needed in executables and DLLs created in external compilers.

```
InitCVIRTE
CloseCVIRTE
```

### Function Limitations for Windows NT

The following functions have some changes or limitations depending on the platform. Refer to the *LabWindows/CVI Standard Library Reference Manual* for specific information concerning the differences.

```
DisableTaskSwitching
LoadExternalModule
SetSystemDate
SetSystemTime
EnableInterrupts
DisableInterrupts
```

### Revised Error Codes

There are new error codes for the following Utility library functions. In some cases, the existing error codes were reduced in scope or changed in value.

```
CopyFile
DeleteDir
DeleteFile
GetDir
GetDrive
GetFileDate
GetFileSize
GetFileTime
GetFirstFile
MakeDir
RenameFile
SetFileDate
SetFileTime
```

## New Easy I/O for DAQ Library

Easy I/O for DAQ is now in the **Library** menu. This library was shipped as an instrument driver in previous versions of LabWindows/CVI.

The functions in the Easy I/O for DAQ Library make it easier to write simple DAQ programs than if you use the Data Acquisition Library. This library implements a subset of the functionality of the Data Acquisition Library, but it does not use the same functions as the Data Acquisition Library.

The following new features were added.

- Counter and timer functions
- Equivalent time sampling
- Ability to specify sampling delay between channels added to channel strings.
- Ability to specify per-channel limits added to channel strings.

The new functions include the following.

```
GetAILimitsOfChannel
CounterMeasureFrequency
CounterEventOrTimeConfig
ContinuousPulseGenConfig
DelayedPulseGenConfig
FrequencyDividerConfig
PulseWidthOrPeriodMeasConfig
CounterStart
CounterRead
```

```
CounterStop
ICounterControl
SetEasyIOMultitaskingMode
DIOPortConfig
DIOPortRead
DIOPortWrite
```

The following functions no longer exist.

```
AIStartTriggeredAcquisition
AIReadTriggeredAcquisition
AIClearTriggeredAcquisition
```

You can replace calls to `AIReadTriggeredAcquisition` with `AIAcquireTriggeredWaveform`. You can safely remove calls to `AIStartTriggeredAcquisition` and `AIClearTriggeredAcquisition`.

# Sample Program Modifications

All sample programs have been modified as follows.

- All source code `main` functions now call `InitCVIRTE` prior to calling any LabWindows/CVI functions.
- All source code callback function now use the `CVICALLBACK` qualifier.

The samples listed in Table 3, *New Sample Projects*, have been added.

Table 3. New Sample Projects

| File Name | Topic |
|---|---|
| samples\analysis\... | |
| convolve.prj | Convolve and Deconvolve functions |
| correlat.prj | Correlation function |
| interp.prj | PolyInterp, RatInterp and SpInterp functions |
| mode.prj | Histogram and Mode functions |
| nonlnfit.prj | NonLinearFit function |
| peak_est.prj | AutoPowerSpectrum and PowerFrequencyEstimate functions |
| phasedif.prj | Calculation of phase difference between two sine waves. |
| pulse.prj | PulseParam function |
| windowng.prj | Illustrates windowing functions |
| samples\apps\... | |
| life.prj | Canvas control's game of life |
| spectrum.prj | Frequency analysis program. |
| samples\sustctrl\easytab\... | |
| simpdemo.prj | Tab-control instrument driver |
| tabdemo.prj | Multiple tabs using Tab-control |
| samples\sustctrl\movectrl\... | |
| movedemo | Make controls movable/sizable |
| samples\easyio\... | |
| ai_acq.prj | Analog acquisition |
| ai_async.prj | Asynchronous timed analog acq. |
| ai_cmd.prj | demonstrates embed commands in the channel string |
| ai_ets.prj | Equivalent Time Sampling |
| ai_log.prj | Logging asynchronously data |
| ai_samp.prj | AISampleChannels function |

*continues*

Table 3. New Sample Projects (continued)

| File Name | Topic |
|---|---|
| ao_wave.prj | Async-timed analog waveform |
| ctr_evt.prj | Count TTL edges of signal |
| ctr_freq.prj | Measure the frequency of signal |
| ctr_perd.prj | Measure width/period of signal |
| ctr_puls.prj | Generate delayed TTL pulse train |
| ctri_pls.prj | Generate square waves or pulses |
| samples\easyio\simple\... | |
| ai^acq.prj | Timed analog acquisition |
| ai^async.prj | Async-timed analog acquisition |
| ai^log.prj | Logging asynchronously data |
| ai^samp.prj | Read analog input channels |
| ai^trig.prj | Acquire triggered data |
| samples\userint\... | |
| 2yaxis.prj | Left and right Y axes on graph |
| canvas.prj | Drawing on a canvas control |
| canvsbmk.prj | Compare canvas to graph control |
| clipbord.prj | Demo clipboard functions |
| drawpad.prj | Canvas drawing pad |
| piedemo.prj | Draw pie charts on canvas |
| timeaxis.prj | Arbitrary axis labels for a graph |
| samples\win32\... | |
| oneinst.prj | Allow one instance of a program |

See `samples.doc` for a complete listing of the sample programs.

# General Information

## Changes to the Data Acquisition Library

### Event Function Parameter Data Types Changed for Windows 95 and NT

Some parameters in the Data Acquisition event handling functions that are two bytes under Windows 3.1 have been increased in size to four bytes under Windows 95 and NT. The include file (`dataacq.h`) and the function panels now have typedefs so you can write source code that works on all three platforms.

The following table lists the typedefs and the intrinsic types under for the different platforms.

Table 4. Typedefs and Intrinsic Types for Different Platforms

| Typedef | Windows 3.1 | Windows 95/NT |
|---|---|---|
| `DAQEventHandle` | `short` | `int` |
| `DAQEventMsg` | `short` | `int` |
| `DAQEventWParam` | `unsigned short` | `unsigned int` |
| `DAQEventLParam` | `unsigned long` | `unsigned long` |

The following functions and typedefs have been affected by this change.

```
Config_Alarm_Deadband
Config_ATrig_Event_Message
Config_DAQ_Event_Message
Get_DAQ_Event
Peek_DAQ_Event
DAQEventCallbackPtr (typedef)
```

If you have written source code for Windows 3.1 that uses these functions and you want to use that source code under Windows 95 or NT, you must modify your source code. You must change the parameter declarations for all of your event callback functions to match the new callback function prototype. You must also use the new typedefs in the declarations of variables that are passed by reference to `Get_DAQ_Event` and `Peek_DAQ_Event`.

### Differences in Current NI-DAQ API for Windows NT

The following functions are not in the current NI-DAQ API for Windows NT. They will be added in a new release of the NI-DAQ DLL for Windows NT in the latter part of 1996.

```
SC_2040_Config
Calibrate_E_Series
Calibrate_1200
AI_Read_Scan
AI_Vread_Scan
```

```
AO_Change_Parameter
SCXI_Config_Filter
```

**NI-DAQ Error Codes**

The error codes returned by NI-DAQ version 4.9.0 and later are different than those returned by NI-DAQ version 4.8.5 and earlier. The `ConvertToNewNIDAQErrorCode` function converts old errors code into new error codes.

The `GetNIDAQErrorString` function converts error codes returned by the Data Acquisition Library into meaningful error messages.

## Number of Parameters Changed for `InitCVIRTE`

In LabWindows/CVI version 4.0.1, the number of parameters to `InitCVIRTE` has increased from one to three. Executables and DLLs created using just one parameter will continue to work with the Run-time Engine for LabWindows/CVI version 4.0.1.

## LabWindows/CVI Using Large Amount of CPU Resources

If you require LabWindows/CVI to use fewer CPU resources so that other applications are given more processor time, you should use the function `SetSleepPolicy`. This function sets the degree to which your program "sleeps" when checking for events. The setting that is optimal for your program depends on the operating system you are using and the other applications you are running. If you think you may need to make an adjustment, try the different settings and observe the resulting behavior.

The different settings are:

```
VAL_SLEEP_NONE   -  Never be put to sleep (Default)
VAL_SLEEP_SOME   -  Be put to sleep a small period
VAL_SLEEP_MORE   -  Be put to sleep for a longer period
```

## Using LabWindows/CVI Utility Library in the Borland Compiler

When you are using the LabWindows/CVI libraries with the Borland compiler, the `utility.h` header file in the Borland `include` subdirectory might be erroneously referenced instead of the LabWindows/CVI `utility.h` header file. You can control the search order for the header files through one of the following methods.

- Place the LabWindows/CVI include directory before the Borland include directory in the Borland Project Options directory search paths.
- In your source code, reference the explicit path to the header file, such as, `#include "c:\cvi\include\utility.h"`
- Rename the Borland `utility.h` file.

## Command Line Argument Change for LabWindows/CVI Standalone Executables

Previous versions of LabWindows/CVI passed double-quote characters (") from the command line to the executable `main` function without any changes. In LabWindows/CVI version 4.0.1, the double-quote character now is interpreted as a literal qualifier, and is not passed to the `main` function.

If you want to pass the double-quote character to the executable, use the backslash double-quote character combination instead. For example, if the command line is

```
MYEXE.EXE "first parm"\"second parm\"
```

the `argv[]` parameter passed to the `main` function contains the following three strings.

```
first parm
"second
parm"
```

## Limitations with `LaunchExecutableEx` under Windows 95/NT

In LabWindows/CVI version 4.0.1 for Windows 95/NT, the `LaunchExecutableEx` function cannot maintain a usable handle to a launched a 16-bit Windows executable. Consequently, the `ExecutableHasTerminated` function always returns a value of 1 when given a handle of a launched 16-bit executable. Also, the `TerminateExecutable` function cannot terminate a previously launched 16-bit executable handle.

## Using NetDDE Under Windows

Please consult the LabWindows/CVI `readme.cvi` file for information on using NetDDE under Windows 3.1, Windows 95, and Windows NT.

## Changes and Additions to TOOLSLIB Instrument Drivers

The Programmer's Toolbox is a set of generally useful instrument drivers in the `cvi\toolslib\toolbox` directory.

### Changes to the INIFILE Instrument Driver

The following new functions treat backslashes as any other characters, which is useful when your want to use DOS or Windows pathnames as values in a `.ini` file.

```
Ini_PutRawString
Ini_GetPointerToRawString
Ini_GetRawStringCopy
Ini_GetRawStringIntoBuffer
```

Use the following new functions to write and read in-memory lists to and from locations other than disk files, such as memory mapped files or TCP/IP addresses.

```
Ini_WriteGeneric
Ini_ReadGeneric
```

The following function disables sorting of an in-memory list.

```
Ini_DisableSorting
```

The following function copies a section from one in-memory list to another.

```
Ini_CopySection
```

Use the following function to read only a subset of the sections in a `.ini` file.

```
Ini_SetSectionFilter
```

The INIFILE Instrument Driver has a new feature to prevent multiple items with the same name being written to an `.ini` file. By default, when you use one of the `Ini_Put` functions to add an item to an in-memory list, any existing items with the same name are deleted. This check can slow down the `Ini_Put` functions. Use the following function to disable or re-enable the duplicate checking.

```
Ini_SetDuplicateChecking
```

Use the following function to specify the delimiter tokens in the sections headings and tag/value pairs.

```
Ini_SetTokens
```

The following function stores a block of binary data (including ASCII NUL bytes) in the in-memory list. It converts data written to a `.ini` file to printable ASCII characters, which are in turn converted back to binary data when the file is read.

```
Ini_PutData
```

**New EASYTAB Instrument Driver**

You can use functions in the `easytab` instrument driver to display LabWindows/CVI User Interface panels in an overlapped manner with tabs, very similar to a Windows 95 tabbed dialog.

The following changes have been made since LabWindows/CVI 4.0.

- The `EasyTab_LoadPanels` function now has an additional parameter to support their use in Windows 95/NT DLLs.

- When you change to a different tab, the event EVENT_TAB_CHANGED is sent to the tab control's callback and to the callbacks for the panels associated with the current and previous active tabs. The callbacks are called with **eventData1** set to the panel handle associated with the currently active tab and **eventData2** set to the panel handle associated with the previously active tab (or 0 if no tab was previously active).

- The EVENT_COMMIT is no longer sent to the tab control's callback when the active tab changes. The EVENT_TAB_CHANGED event is sent instead.

**New Instrument Driver for Regular Expression Matching**

You can use the functions in the regexpr instrument driver to search text strings for patterns that match regular expressions.

## GPIB Instrument Driver Portability

If you want your instrument drivers to be portable, do not check for GPIB errors by checking for negative return values. Instead, check for the ERR bit in the return value. For example, the following statement is not portable because it checks for a negative return value.

```
if (ibwrt(bd[instrID], buf, cnt) <= 0)

      PREFIX_err = 230;
```

The following revised statement is portable because it checks for the ERR bit in the return value.

```
if (ibwrt(bd[instrID], buf, cnt) & ERR)

      PREFIX_err = 230;
```

Under Windows 3.*x*, the return value is a 16-bit value and is negative when the ERR bit is set. Under other systems, such as Windows 95/NT or UNIX, the return value is a 32-bit value and is never negative.

## Extra Lines Appearing on Printouts

If your printouts contain extra lines, the problem may be caused by the video driver generating incorrect data when converting a screen-oriented bitmap to a device independent bitmap. Try using a different video driver.

# Additions to LabWindows/CVI Documentation

## Additions to the User Interface Library Reference Manual

### Additions to Chapter 3, User Interface Library

#### New Attributes for SetSystemAttribute and GetSystemAttribute

| | | |
|---|---|---|
| ATTR_TASKBAR_BUTTON_VISIBLE | integer | 0 - Do not display a taskbar button (the default for DLLs).<br>1 - Display a taskbar button (the default for executables).<br>(See discussion below.) |
| ATTR_TASKBAR_BUTTON_TEXT | string | The text to display in the taskbar button. Only the first 80 characters are used. When calling GetSystemAttribute, you must pass a buffer with at least 81 bytes. |

Taskbar Buttons (Windows 95 and NT)

The LabWindows/CVI Run-time Engine automatically adds a button to the taskbar for executables created in LabWindows/CVI. The text in the taskbar button is the Application Title that you specify in the Create Standalone Executable dialog. If you leave the Application Title blank, the executable filename is used instead. The button is added to the taskbar before the `main` or `WinMain` function is called. You can change the button text with `ATTR_TASKBAR_BUTTON_TEXT`. You can hide the button with `ATTR_TASKBAR_BUTTON_VISIBLE`.

For executables that are created using external compilers and that directly call the LabWindows/CVI libraries, the LabWindows/CVI Run-time Engine adds a button to the taskbar when your program calls `InitCVIRTE`. To prevent this, make the following call before the call to `InitCVIRTE`.

```
SetSystemAttribute (ATTR_TASKBAR_BUTTON_VISIBLE, 0);
```

By default, the text in the taskbar button is the executable filename. You can specify a different text string in either of the following two ways.

- Create a string table resource (if you do not already have one) and add a string with ID number 65535, or

- Make the following call before the call to `InitCVIRTE`.

```
SetSystemAttribute (ATTR_TASKBAR_BUTTON_TEXT,
                    "<text>");
```

For executables that are created using external compilers, that do not directly call the LabWindows/CVI libraries, but that load DLLs which do call the LabWindows/CVI libraries, the LabWindows/CVI Run-time Engine does not by default add a button to the taskbar. You can add a button to the taskbar by making the following call from the DLL.

```
SetSystemAttribute (ATTR_TASKBAR_BUTTON_VISIBLE, 1);
```

By default, the text in the taskbar button is the executable filename. You can specify a different text string by making the following call.

```
SetSystemAttribute (ATTR_TASKBAR_BUTTON_TEXT,
                    "<text>");
```

**Note:** ***These attributes can be referenced on platforms other than Windows 95 and NT, but they have no effect.***

# Additions to the Standard Library Reference Manual

## Additions to Chapter 4, GPIB and GPIB-488.2 Libraries

### iblock

`int` **status** = **iblock** (`int` **boardDevice**);

### Purpose

This function blocks other processes from accessing the specified GPIB-ENET board or device. You can release the lock by calling the `ibunlock` function with the same board or device descriptor.

By default, processes return an ELCK (21) error when they attempt to use a board or device locked by another process. You can specify that processes block instead of returning an error by making the following function call:

```
ibconfig (boardOrDevice, IbcBlockIfLocked, 1);
```

There is no timeout on the process block.

Recommended Usage

In general, the iblock function should be used to gain critical access to a GPIB-ENET board or device when multiple processes might be accessing it at the same time. When the GPIB-ENET is locked, the GPIB driver guarantees that subsequent calls made to the GPIB-ENET are completed without interruption.

**Parameter List**

| | | | |
|---|---|---|---|
| Input | **boardDevice** | integer | A device descriptor returned by either `OpenDev`, `ibfind`, or `ibdev`. |

**Return Value**

| | | |
|---|---|---|
| **status** | integer | Indicates whether the function succeeded. |

---

**ibunlock**

`int` **status** = **ibunlock** (`int` **boardDevice**);

**Purpose**

Releases a lock on a GPIB-ENET board or device. See the help information for the `iblock` function.

In general, you should release your lock on a GPIB-ENET connection immediately after you make your critical access.

**Parameter List**

| | | | |
|---|---|---|---|
| Input | **boardDevice** | integer | A device descriptor returned by either `OpenDev`, `ibfind`, or `ibdev`. |

**Return Value**

| | | |
|---|---|---|
| **status** | integer | Indicates whether the function succeeded. |

---

## Additions to Chapter 8, Utility Library

### CheckForDuplicateAppInstance

`int` **status** = **CheckForDuplicateAppInstance** (`int` **activateOtherInstance**,
`int *`**thereIsAnotherInstance**);

### Purpose

For Windows 95 or NT, this function determines if another copy of the same executable is running, but only if the other copy has already called this function. You can specify an option to bring the other copy to the front.

For other platforms, this function always returns -1.

### Parameter List

| | | | |
|---|---|---|---|
| Input | **activateOtherInstance** | integer | Specifies whether the other application instance (if any) is brought to the front.<br>Valid Values:<br>`ACTIVATE_OTHER_INSTANCE`<br>`DO_NOT_ACTIVATE_OTHER_INSTANCE` |
| Output | **thereIsAnotherInstance** | integer | Is set to 1 if there is another instance of this executable. Otherwise, is set to 0. If an error code is returned by the function, this variable is always set to zero. |

### Return Value

| | | |
|---|---|---|
| **status** | integer | Indicates whether the function succeeded. |

### Return Codes

| | |
|---|---|
| 0 | Success |
| -1 | You are running on a platform other than Windows 95 or NT. |
| -2 | Could not allocate system resources to check for a duplicate application instance. |

# Corrections to LabWindows/CVI Documentation

## Corrections to the Standard Library Reference Manual

### Corrections to Chapter 8, Utility Library

*In the* `GetFirstFile` *function description, replace the last two paragraphs of the* **Purpose** *section with the following.*

If you use only the **normal** attribute, any file that is not read-only, not a system file, not hidden, and not a directory can match. Normal files can have the archive attribute on or off.

If you specify the **read-only** attribute, any file that is read-only can match, unless the file is a system (or hidden) file and you have not specified the **system** (or **hidden**) attribute.

If you specify the **system** attribute, any system file can match, unless the file is also a hidden file and you have not specified the **hidden** attribute. If you do **not** specify the **system** attribute, a system file cannot match regardless of its other attributes.

If you specify the **hidden** attribute, any hidden file can match, unless the file is also a system file and you have not specified the **system** attribute. If you do **not** specify the **hidden** attribute, a hidden file cannot match regardless of its other attributes.

If you use more than one attribute, the effect is additive. Any file that would match if you specified just one of the attributes can match regardless of the additional attributes you specify.