LabWindows/CVI Version 5.0.1 Release Notes for Solaris 2

Contents

Introduction to LabWindows/CVI 5.0.1 for Solaris 2

These release notes contain installation instructions, system requirements, new features, and updated information to help you begin using LabWindows/CVI 5.0.1 for Sun Solaris 2.

LabWindows/CVI Installation

Before working on your 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 five minutes.

Minimum System Requirements

To run LabWindows/CVI for Solaris 2 you must have the following:

- Sun SPARCstation
- At least 24 MB of RAM
- At least 32 MB of disk swap space
- At least 53 MB of free hard disk space prior to installation
- Solaris 2.4 or greater (SunOS 5.4 or greater)

☞ **Note** ***The dialog box that the* Edit Instrument Attributes *command opens requires a video adapter resolution of 800-by-600 or greater.***

Installing LabWindows/CVI

To install LabWindows/CVI for Solaris 2, you must be able to mount a CD-ROM drive on your computer. The drive may be physically attached to your computer, or it may be attached to another computer that permits you to mount the drive remotely. In either case, you need to have root privileges to mount the CD-ROM drive.

☞ **Note** ***You must purchase a multiple-user license from National Instruments to use the LabWindows/CVI Run-time Libraries for Solaris 2 over a network.***

1. Choose the installation directory. The installation program copies all LabWindows/CVI files from the CD-ROM into the directory you choose. This directory must have at least 53 MB of free space. All LabWindows/CVI users must have read permissions for this directory. If LabWindows/CVI is run over a network, all machines that run LabWindows/CVI must be able to read the installation directory. The following steps refer to the installation directory as *install_location*.
2. Mount the LabWindows/CVI CD in your CD-ROM drive. The following steps assume you mounted the CD-ROM in the directory `/cdrom`.
3. Log in as root or use the `su` command to get root privileges.
4. Run the installation program. The installation program prompts you to identify the installation directory you chose in step 1 of this procedure.

   If your CD-ROM drive was automatically mounted, use the following commands:

   ```
   mysun# cd /cdrom/cvi_501/solaris2
   mysun# ./INSTALL
   ```

   If instead you used the `mount` command to mount the CD-ROM drive, use the following commands:

   ```
   mysun# cd /cdrom/solaris2
   mysun# ./INSTALL
   ```

   When you install the Run-time Libraries, the installation routine places symbolic links to the LabWindows/CVI shared libraries in `/usr/lib`.

☞ **Note** ***LabWindows/CVI uses specific Solaris files and tools to compile and create executables. If you have not installed the following packages with your operating system, you cannot install LabWindows/CVI on your system.***

- `SUNWtoo` ***(programming tools)—This package contains utilities for software development including*** `ld`***,*** `ldd`***,*** `od`***, and*** `truss`***. LabWindows/CVI uses*** `ld` ***to create executables.***
- `SUNWhea` ***(Sun Solaris header files)—This package contains the Sun C header files for general software development. LabWindows/CVI uses the ANSI header files to compile programs.***

5. Run LabWindows/CVI and enter your product registration information and serial number when the prompt appears. This information is written to the LabWindows/CVI resource file; you can set it only once.
   a. Run LabWindows/CVI. You do not need root privileges to run LabWindows/CVI, but you must be running an X Windows server and you must have permission to connect to that server. In particular, you should run LabWindows/CVI using the `userid` that you used to start the X Windows server.

      ```
      mysun# install_location/cvi
      ```

b. Enter the requested product registration information and serial number. Your serial number is printed on the product label affixed to the top of your LabWindows/CVI package.

c. Quit LabWindows/CVI by selecting the **Exit** menu item from the **File** menu.

d. Change the permissions of the LabWindows/CVI resource file with the following command. You need root privileges to perform this step.

```
mysun# chmod 644 install_location/bin/cvi.rsc
```

Refer to the `readme.cvi` file for installation instructions, programming considerations, and changes that are too recent to be included in the printed LabWindows/CVI documentation.

☞ **Note** ***You will need the IVI Engine shared library in order to use IVI (Interchangeable Virtual Instruments) drivers on a system. The IVI Engine is installed as part of LabWindows/CVI for Solaris 2. The*** `misc/bin` ***directory of the LabWindows/CVI installation includes a script you can use to distribute the IVI Engine.***

What Does the Setup Program Install?

The setup program for LabWindows/CVI installs the LabWindows/CVI Libraries and related files on your hard disk in the subdirectories listed in the following table. The installation includes sample programs that illustrate many of the features of LabWindows/CVI 5.0.1. Table 1 shows the directories and their contents.

Table 1. LabWindows/CVI 5.0.1 Subdirectories

Directory Name	Contents
`bin`	LabWindows/CVI internal library and support files
`lib`	LabWindows/CVI libraries
`fonts`	Font files
`include`	Header files associated with each of the libraries
`instr`	Instrument drivers
`hyperhelp`	HyperHelp Viewer files
`samples`	Source code for sample projects
`toolslib`	Utility instrument drivers
`misc`	Miscellaneous scripts for distributing executables, libraries, and so on
`wizard`	IVI Wizard and template files
`tutorials`	Programs you use with the tutorial sessions of the *Getting Started with LabWindows/CVI* manual

The message file (msgrt*n*.txt, where *n* is the version number of the Run-time Engine) is a text file containing the error messages displayed by the Run-time Engine. The message file is in the LabWindows/CVI bin subdirectory. 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.

The LabWindows/CVI 5.0.1 INSTALL script installs five separate packages. You may remove these packages in order to uninstall LabWindows/CVI. Table 2 shows the packages and what components they contain.

Table 2. LabWindows/CVI Installation Packages

Package Name	Component the Package Contains
NICcvi	LabWindows/CVI environment
NICcvirt	LabWindows/CVI Run-time Engine
NICinstr	Instrument driver support
NICividev	IVI Wizard and template files
NICivi	IVI Engine

☞ **Note** ***The*** pkgadd ***utility appends the*** .N ***tag to packages it installs if the package has previously been installed, where*** N ***is an integer greater than that of the previously installed package. For example, if you use*** pkgadd ***to install a newer version of the IVI Engine, the new package might be named*** NICivi.2**.**

Installing the VISA Library

You need the VISA Library in order to use the instrument driver standard accepted by the VXI*plug&play* Systems Alliance. VISA stands for Virtual Instrument Software Architecture.

Installing the VISA Library for Solaris 2

1. Log in as root or use the su command to get root privileges.
2. Place the LabWindows/CVI CD in your CD-ROM drive. Make sure you have mounted your CD-ROM.
3. Run the installation program.

```
mysun# cd /cdrom/visa/solaris2
mysun# ./INSTALL
```

☞ **Note** ***The preceding example assumes that the CD-ROM is mounted in a directory called*** /cdrom**. *Be sure to use the correct path name for your CD-ROM.***

Installing on a Network

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

What Is New and Different in LabWindows/CVI 5.0.1

This section includes information about changes and enhancements in LabWindows/CVI 5.0.1 that have been made since LabWindows/CVI 4.0.1.

New Features

- IVI (Interchangeable Virtual Instruments) driver wizard and support library
- Ability to scale panels and contents to different screen resolutions and when you resize the panels
- New advanced analysis functions for vector and matrix algebra

New Sample Programs

LabWindows/CVI 5.0.1 adds several new samples which are shown in Table 3. Remember that more involved, application-level samples are included in the `samples/apps` directory.

Table 3. New Sample Projects

Directory/Filename	Description
`samples/analysis/`	
`2dfft.prj`	Demonstrates using FFT to do a 2D FFT
`nlinfit.prj`	`NonLinearFit` using the Levenberg-Marquardt method
`parsevls.prj`	Demonstrates Parseval's theorem
`polyfit1.prj`	Demonstrates data conditioning for `PolyFit`
`transmit.prj`	Simulates a transmission and receiver system
`thd.prj`	Total Harmonic Distortion
`stat.prj`	Simple statistics example
`stability.prj`	Evaluates the stability of the system
`samples/apps/`	
`smithdem.prj`	Illustrates the use of a Smith Chart control

Table 3. New Sample Projects (Continued)

Directory/Filename	Description
samples/toolbox/	
ini.prj	Demonstrates the *.ini file instrument driver
list.prj	Simple example using the Toolbox List API
menudemo.prj	Example using Menu Utilities instrument driver
samples/userint/	
scaling.prj	Demonstrates UIR panel scaling

Changes to the LabWindows/CVI Development Environment

This section contains information on several enhancements to the LabWindows/CVI development environment.

Automatic Generation of Object Files

A new create-object icon, "O," appears in the middle column of the Project window. The icon only works with source (.c) files and indicates that you enabled the Compile into Object option. If this option is enabled when you compile the source file, LabWindows/CVI creates an object (*.o) file on disk that contains code that users cannot debug, instead of generating debuggable code in memory. To toggle this option, double-click below this icon, on the row where your source file appears.

When you open a project, LabWindows/CVI marks each source file with the "O" icon for compilation only if one of the following conditions applies:

- The object file does not exist on disk.
- The source file or any of the include files on which it depends has a date later than the date of the object file.

As you work, LabWindows/CVI marks source files for recompilation whenever they or any of their include files are modified, regardless of the state of the icon. When you select the **Build Project** or **Run Project** command, LabWindows/CVI compiles all source files marked for recompilation. For source files that you mark with the "O" icon, LabWindows/CVI also generates the corresponding object files.

You can use this feature when you do not want to recompile all your source files each time you load your project. You can also use this feature when you want to suppress debugging on a source file, because debugging will not available for files you mark with the "O" icon.

Breakpoints/Tags in Non-Project Files Are Saved in Project

In LabWindows/CVI you can set breakpoints and tags in source files that are either included in the project or not included in the project. In both cases, LabWindows/CVI version 5.0.1 saves the breakpoints/tags in the project file when you unload a project. Previous versions of LabWindows/CVI did not save breakpoints/tags that you set in non-project files.

Clarification about Activate Panels When Resuming

The **Activate Panels When Resuming** option in the **Run** menu of the Source window causes LabWindows/CVI to activate your most recently active panel when you resume execution. However, the panel must be selected at the time you suspend execution. If a LabWindows/CVI window, such as a Source window, is active at the time you suspend execution, LabWindows/CVI does not reactivate the window or a panel.

More specifically, if you suspend execution by selecting the **Break Execution** option from the **Run** menu in a Source window, LabWindows/CVI does not reactivate the window or a panel when you resume execution.

Revised Print Dialog Boxes

For more information about printing graphics and text under Solaris 2 using the reorganized Print dialog box, refer to the *Revised Print Dialog Boxes* section in the *Changes to the User Interface Library* section later in this document.

New Options for Source File Printing

New options for printing source files include the ability to insert the date, time, and filename at the top of each page, and the ability to number pages. The new options on the dialog box correspond to the new text printing attributes in the User Interface Library. Refer to the *Application of Attributes to Text Printing* and *New Print Attributes* sections in the *Changes to the User Interface Library* section later in this document.

Edit Menu

The **Find** and **Replace** commands of the **Edit** menu of the Function Tree Editor window and the Function Panel Editor window open dialog boxes similar to the Find and Replace dialog boxes in the Source window. You use these dialog boxes to search and replace text in the entire function panel (`.fp`).

Code Menu

In panels for functions that set or get attribute values, the **Select Attribute Constant** command in the **Code** menu replaces the **Select UIR Constant** command. The User Interface Library, the VISA Library, and IVI drivers contain such functions, for example, `GetCtrlAttribute`, `SetCtrlAttribute`, `GetPanelAttribute`, and `SetPanelAttribute` in the User Interface Library. The panel for each of these functions

contains an Attribute ring control and a corresponding Value input control. When you use either of these two controls, the **Select Attribute Constant** command appears in the **Code** menu. The action of the command differs based on whether you use the Attribute or Value control.

New Tools Menu

This section describes the commands in the new **Tools** menu of the Project, Source, Function Panel, Function Tree Editor, and Function Panel Editor windows.

Create IVI Instrument Driver

To create interchangeable virtual instrument (IVI) drivers for controlling an instrument, you can use the **Create IVI Instrument Driver** command and the IVI wizard to create the required source file, include file, and function panel file. You can base the new instrument driver on one of the following:

- An existing driver for a similar instrument
- The core IVI driver template
- A template for a particular type of instrument

The IVI wizard copies the template or existing driver files and replaces all instances of the original instrument prefix with the prefix you select for your new driver.

Refer to the *LabWindows/CVI Instrument Driver Developers Guide* for more information on the benefits and implementation of IVI drivers.

Edit Instrument Attributes

Use the **Edit Instrument Attributes** command to add, delete, or edit attributes for an IVI driver. You can use this command only if the file in the Source window has the same path and base filename as an instrument driver function panel (`.fp`) file and its associated `.sub` file. The command is useful only if the instrument driver files were generated using the **Create IVI Instrument Driver** command.

This command analyzes the instrument driver files to find all the attributes the driver uses. It then opens a dialog box that displays the attributes and information about them. In the dialog box, you can add or delete attributes, modify their properties, and enter help text for them. When you apply the changes, the command modifies the source, include, and function panel files for the instrument driver.

If you use this command while the text cursor is located above the defined constant name or callback function name for one for the attributes, the dialog box appears with that attribute selected in the list box.

Refer to the *LabWindows/CVI Instrument Driver Developers Guide* for more information on IVI drivers.

Edit Function Tree

Use the **Edit Function Tree** command to display the Function Tree window for the function panel (`.fp`) file associated with the file in the Source window. The function panel file must have the same path and base filename as the file in the Source window.

Edit Function Panel

Use the **Edit Function Panel** command to display the Function Panel Editor window for a function defined in an instrument driver source file. You can use this command only if the file in the Source window has the same path and base filename as an instrument driver function panel (`.fp`) file. You must locate the text cursor over the name of a function that has a function panel in the `.fp` file.

Generate Source for Instrument Driver Functions

LabWindows/CVI 5.0.1 includes two new menu items for generating source for instrument driver functions: **Generate Source for Function Panel** in the Function Panel Editor window and **Generate New Source for Function Tree** in the Function Tree Editor window.

Go To Definition

Use the **Go To Definition** command to display the function in the source (`.c`) file associated with the specified function in the Function Panel Editor window. The function panel file must have the same path and base filename as the file in the Source window.

Go To Declaration

Use the **Go To Declaration** command to display the function prototype in the header (`.h`) file associated with the specified function in the Function Panel Editor window. The function panel file must have the same path and base filename as the file in the Source window.

Enable Auto Replace

Use the **Enable Auto Replace** command to enable automatic updating of all definitions and declarations for function names when LabWindows/CVI updates the instrument prefix or the function name in the Function Tree Editor window or Function Panel Editor window.

Generate IVI C++ Wrapper

Use the **Generate IVI C++ Wrapper** command to generate a C++ wrapper for an IVI driver. Refer to the *LabWindows/CVI Instrument Driver Developers Guide* for more information on IVI drivers.

User-Defined Entries in Tools Menu

You can install your own entries in the **Tools** menu. Each entry invokes an executable with optional command line arguments. Use the **Tools Menu Options** command from the **Options** menu of the Project window to add your own entries to the **Tools** menu.

Options Menu

The new **Tools Menu Options** command in the **Options** menu of the Project window allows you to add your own menu items to the **Tools** menu that appears in the Project, Source, Function Panel, Function Tree Editor, and Function Panel Editor windows. Each entry consists of a menu item name and an associated command line to execute. Each command line consists of a program name and optional arguments. When you execute an item from the **Tools** menu, LabWindows/CVI calls a system function to start another process, passing the command line as a parameter.

Context Menus

You can access a context menu in the Function Tree Editor window by clicking on and holding the right mouse button. The context menu contains the most commonly used menu commands from the Function Tree Editor window menu bar. The Function Tree Editor window now includes the following context menu items:

- Edit Node
- Edit Function Panel Window
- Edit Help
- Generate Source For Function Node
- Go To Declaration
- Go To Definition

The Source window now includes the following new context menu items:

- Edit Function Tree
- Edit Function Panel
- Edit Instrument Attributes

New Configuration Option

The DST (daylight savings time) rules string allows you to specify the periods of the year in which daylight savings time is in effect. The time and date functions in the LabWindows/CVI ANSI C Library use this information. In previous versions of LabWindows/CVI, you had to modify the `cvimsgs.txt` or `msgsrt4.txt` files in the `cvi/bin` directory in order to specify a daylight savings time period.

For information on the format of the DST rules string and how the time and date functions use it, refer to the *Time and Date Functions* section in Chapter 1, *ANSI C Library*, of the *LabWindows/CVI Standard Libraries Reference Manual*.

For instructions on setting the configuration options, refer to the *How to Set the Configuration Options* section in Chapter 1, *Configuring LabWindows/CVI*, of the *LabWindows/CVI User Manual*.

Changes to the User Interface Library

This section contains information on the various changes in the User Interface Library, including the following:

- Updated print dialog boxes
- Two new attributes keep track of the current printer
- Several new attributes apply to printing text files and buffers
- A new attribute allows scaling of panels to different screen resolutions
- New attributes allow the scaling of panel contents when you resize the panels
- Several new functions
- Several new error codes

☞ **Note** ***Be sure to read the Notice of Change to Text Format (.tui) Files section later in this document.***

Change to Default Value of SetSleepPolicy

You can use the `SetSleepPolicy` function to adjust the CPU resources that LabWindows/CVI normally uses. This function sets the degree 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 need to make an adjustment, try the different settings and choose the setting that performs best.

Beginning with LabWindows/CVI 5.0.1, the default value for the User Interface Library's `SetSleepPolicy` is `VAL_SLEEP_MORE`. In earlier versions of LabWindows/CVI, the default sleep policy is `VAL_SLEEP_NONE`.

Table 4. Sleep Settings for LabWindows/CVI

Setting	Description
`VAL_SLEEP_NONE`	Never sleep
`VAL_SLEEP_SOME`	Sleep for a small period
`VAL_SLEEP_MORE`	Sleep for a longer period (default)

Clarifications and Corrections to the LabWindows/CVI User Interface Reference Manual

The manual incorrectly states that values for ATTR_XOFFSET and ATTR_YOFFSET are expressed in inches. In fact, they are expressed in tenths of millimeters.

The top-level panel callback receives the EVENT_CLOSE message when the user executes the **Close** command from the **System** menu.

When you are using the new function SetBitmapData on a bitmap that already has a mask, you can either supply the same mask array, supply a new mask array, or pass REMOVE_TRANSPARENCY_MASK to remove the mask. When you use REMOVE_TRANSPARENCY_MASK, the color values of the pixels that were transparent under the old mask are *unpredictable*. You can set the value of these pixels in the **bits** parameter.

The following text is incorrect in the *Using Callback Functions to Respond to User Interface Events* section of Chapter 3, *Programming with the User Interface Library*, in the *LabWindows/CVI User Interface Reference Manual*: "A panel callback receives the EVENT_SIZE and EVENT_MOVE messages when the user resizes or moves the panel. The panel callback does not receive these messages when you programmatically resize or move a panel." The correct text follows:

> LabWindows/CVI posts the events EVENT_PANEL_MOVE and EVENT_PANEL_SIZE when you programmatically move or resize a panel. The panel callback receives the events when the application processes events.

The manual incorrectly defines the return value description for InsertSeparator. The returned integer parameter should be defined as follows:

MenuItemID — Returns the ID that LabWindows/CVI uses to specify this menu item in subsequent function calls. Negative values indicate that an error occurred. Refer to Appendix A, *Error Conditions*, for error codes.

Revised Print Dialog Boxes

For programmatic printing and for printing from the **File** menu, a common, unified print dialog box allows you to perform the following tasks:

- Select a printer.
- Specify printing options that are specific to LabWindows/CVI. Different sets of options appear for graphics printing as opposed to text printing.
- For graphics printing, select **Properties** from the main print dialog box to open a printer-specific dialog and specify additional printing options such as page orientation.

You can use SetPrintAttribute to programmatically set all the options that are specific to LabWindows/CVI.

Interaction between Print Dialog Boxes and Programmatic Attributes

LabWindows/CVI 5.0.1 modifies the way the print dialog boxes interact with attributes that you use in a program.

Previously, if you selected a different printer in the dialog box, the User Interface Library did not retain the name of the selected printer. Every time the print dialog box appeared, the printer selection was reset to the current system printer. Now, the library stores the name of the printer you select in a new attribute, ATTR_PRINTER_NAME. Whenever the print dialog box appears, LabWindows/CVI sets the printer selection to the current value of ATTR_PRINTER_NAME. If ATTR_PRINTER_NAME is NULL, the empty string, or the name of a printer that is not currently known by the operating system, the next call to a printing function uses the current system printer and stores its name as the ATTR_PRINTER_NAME value.

Revised Constant Names

The constant names of the attributes and values in Table 5 have changed, but the actual values remain the same. The obsolete constant names remain in the userint.h include file, so you do not have to change your source code.

Table 5. Changes to Constant Names for Attributes and Values

Obsolete Name	Replacement Name
ATTR_PRINT_AREA_HEIGHT	ATTR_PAPER_HEIGHT
ATTR_PRINT_AREA_WIDTH	ATTR_PAPER_WIDTH
For ATTR_PRINT_AREA_HEIGHT and ATTR_PRINT_AREA_WIDTH, VAL_USE_ENTIRE_PAPER is obsolete.	VAL_USE_PRINTER_DEFAULT
For ATTR_XOFFSET and ATTR_YOFFSET, VAL_CENTER_ON_PAPER is obsolete.	VAL_USE_PRINTER_DEFAULT
For ATTR_XRESOLUTION and ATTR_YRESOLUTION, VAL_USE_PRINTER_SETTING is obsolete.	VAL_USE_PRINTER_DEFAULT

Application of Attributes to Text Printing

The following existing attributes now apply to text printing and to graphics printing:

- ATTR_DUPLEX
- ATTR_EJECT_AFTER
- ATTR_NUMCOPIES

New Print Attributes

The following new print attributes apply to text and graphics printing.

Table 6. New Text and Graphics Print Attributes

Name	Description
ATTR_PRINTER_NAME	Currently selected printer
ATTR_PRINTER_NAME_LENGTH	Number of characters in currently selected printer
ATTR_BITMAP_PRINTING	On Windows platforms, specifies whether to use Bitmap or GDI printing. Sun versions always uses Bitmap printing.
ATTR_SYSTEM_PRINT_DIALOG_ONLY	Specifies to display only the native printer-specific dialog box; does not show the LabWindows/CVI print dialog box

The following new print attributes apply only to printing text files and buffers, including source code.

Table 7. New Text-Only Print Attributes

Name	Description
ATTR_CHARS_PER_LINE	Number of characters per line (default is 90)
ATTR_LINES_PER_PAGE	Number of lines per page (default is 77)
ATTR_SHOW_DATE	Display current date on first line of each page
ATTR_SHOW_FILE_NAME	Display filename on first line of each page
ATTR_SHOW_LINE_NUMBERS	Display line numbers
ATTR_SHOW_PAGE_NUMBERS	Display page numbers
ATTR_SHOW_TIME	Display current time on first line of each page
ATTR_TAB_INTERVAL	Number of spaces represented by a <Tab> character

New Panel Attributes

LabWindows/CVI 5.0.1 adds the following panel attributes.

Table 8. New Panel Attributes

Name	Description
ATTR_SCALE_CONTENTS_ON_RESIZE	LabWindows/CVI scales panel contents when panel is resized.
ATTR_MIN_HEIGHT_FOR_SCALING	Smallest panel height for which LabWindows/CVI allows scaling
ATTR_MIN_WIDTH_FOR_SCALING	Smallest panel width for which LabWindows/CVI allows scaling
ATTR_RESOLUTION_ADJUSTMENT	Panel scaled when displayed on different screen resolutions (read-only)
ATTR_HAS_TASKBAR_BUTTON	Panel has its own taskbar button on Windows 95/NT.

☞ **Note** ***When you use*** ATTR_SCALE_CONTENTS_ON_RESIZE ***and*** ATTR_RESOLUTION_ADJUSTMENT ***to scale panels and controls, use TrueType fonts for optimal results.***

New Control Attributes

LabWindows/CVI 5.0.1 adds the following control attributes.

Table 9. New Control Attributes

Name	Description
ATTR_AUTO_SIZING	Specifies whether LabWindows/CVI resizes command button when text is changed; valid values: VAL_ALWAYS_AUTO_SIZE VAL_GROW_ONLY VAL_NEVER_AUTO_SIZE VAL_SHRINK_ONLY
ATTR_PLOT_AREA_LEFT	Offset in pixels of the left edge of the plot area from the left edge of the graph control
ATTR_PLOT_AREA_TOP	Offset in pixels of the top of the plot area from the top of the graph control

New Plot Attribute

LabWindows/CVI 5.0.1 adds the following plot attribute:

`ATTR_PLOT_THICKNESS` Thickness of the plot line, in pixels; applies only when `ATTR_LINE_STYLE` is `VAL_SOLID`; if `ATTR_PLOT_STYLE` is `ATTR_FAT_LINE` or `ATTR_FAT_STEP`, LabWindows/CVI draws the plot with three times the thickness specified in this attribute.

New System Attribute

The `ATTR_RESOLUTION_ADJUSTMENT` system attribute specifies to what extent LabWindows/CVI scales panels and their contents when they appear on screens with differing resolutions.

Discussion of Resolution Adjustment

When a panel is displayed on a screen with a different resolution than the screen on which you edited the panel, the panel might appear too large or too small. The Edit Panel dialog box in the User Interface Editor contains an option to scale the panel to the resolution of the screen. You can choose to make no adjustment or an adjustment of up to 100 percent. LabWindows/CVI saves the value you assign for this option for each panel in the `.uir` file. LabWindows/CVI scales the panel and its contents when your program calls `LoadPanel` or `LoadPanelEx`.

You can use the `ATTR_RESOLUTION_ADJUSTMENT` system attribute to override the settings in the `.uir` file. To override the setting for a panel, call `SetSystemAttribute` to set the `ATTR_RESOLUTION_ADJUSTMENT` attribute before you call `LoadPanel` or `LoadPanelEx`. After calling `LoadPanel` or `LoadPanelEx`, you can call `GetPanelAttribute` with `ATTR_RESOLUTION_ADJUSTMENT` to obtain the setting that LabWindows/CVI saved in the `.uir` file.

User Interface Editor Changes

Changes in the User Interface Editor accommodate the new panel and system attributes. The Other Attributes dialog box that you can invoke from the Edit Panel dialog box now has controls for the following items:

- Scale Contents on Resize
- Minimum Height for Rescaling
- Minimum Width for Rescaling
- Resolution Adjustment

Except for the Initial Editor Background Color control, all controls that were previously located in the Color Preferences section of the User Interface Editor Preferences dialog box moved to a new section called Preferences for New Panels. The Preferences for New Panels section also includes a new Resolution Adjustment control.

A new Preferences for New Controls section lets you specify the default test styles for controls and labels that you create.

A **Default** command button in the Editor Color Preferences section allows you to return to the original background color of the editor.

Changes to Existing Functions

For `GetPrintAttribute` and `SetPrintAttribute`, the last parameter is the attribute value. In previous versions of LabWindows/CVI, this value is an integer. Now the parameter is an argument of variable type. This change does not require any modifications to your existing source code.

`LoadPanel`, `LoadPanelEx`, `LoadMenuBar`, and `LoadMenuBarEx` now work on `.tui` files and on `.uir` files. Whereas `.uir` files are in binary format and load quickly, `.tui` files are in text format and load slowly. You can save a `.tui` file by using the **Save in Text Format** command in the **Options** menu of the User Interface Editor. Because text format files load slowly, National Instruments recommends that you continue to use `.uir` files. The fact that LabWindows/CVI can load `.tui` files can be useful, because you can write wizards to generate user interface files that you can load programmatically.

Details on Loading Panels and Menubars from .tui Files

When you call a `.tui` file with `LoadPanel` or `LoadPanelEx`, the panel resource ID parameter must be the header number of the `.tui` file section in which the panel is defined. For example, if the section header for the desired panel is `[Panel003]`, pass `3` as the panel resource ID.

`LoadPanel` or `LoadPanelEx` loads all the controls in the `.tui` file that has section headers in the form `[Panel`*`NNN`*`_Control`*`YYY`*`]`, where *`NNN`* is the panel resource ID and *`YYY`* is `001` or greater. The control numbers must start at `001` and must be consecutive. To pass a control ID to other User Interface Library functions, pass *`YYY`* + 1. For instance, if the section header is `[Panel003_Control001]` for a control on which you want to set an attribute value, pass `2` as the control ID parameter to `SetCtrlAttribute`.

When you call a `.tui` file with `LoadMenuBar` or `LoadMenuBarEx`, the menu bar ID parameter must be the header number of the `.tui` file section in which the panel is defined. For example, if the section header for the desired menu bar is `[MenuBar003]`, pass `3` as the menu bar ID.

`LoadMenuBar` or `LoadMenuBarEx` loads all the menus and menu items in the `.tui` file that has section headers in the form `[MenuBar`*`NNN`*`_...]`, where *`NNN`* is the menu bar ID passed to the function. The menu ID or menu item ID that you pass to User Interface Library functions is based on a depth-first traversal of all the items in the menu tree, starting at 2. For submenu items, the submenu itself has an ID that is one greater than the item ID of the submenu entry in the parent menu.

If you save a .tui file in the User Interface Editor in LabWindows/CVI 5.0.1 or later and you have an up-to-date include (.h) file that the User Interface Editor generates, you can use the panel, control, menu bar, menu, submenu, and menu item constants in the include file as parameters to User Interface Library functions.

Notice of Change to Text Format (.tui) Files

The order in which panel and menu bar sections are written has changed to allow you to use user interface include file constants in conjunction with .tui files. In addition, the version number has changed from 101 to 102.

If you use .tui files to find differences between versions of your .uir files and .tui files you generated in previous versions of LabWindows/CVI, use LabWindows/CVI 5.0.1 to create new baseline .tui files for all your .uir files.

New Functions

The following new functions in the User Interface Library appear in alphabetical order:

- GetScaledCtrlDisplayBitmap creates a bitmap object that contains a snapshot image of the current appearance of the specified control.
- GetScaledPanelDisplayBitmap creates a bitmap object that contains a snapshot image of the current appearance of the specified panel.
- GetTextBoxLineIndexFromOffset returns the zero-based index of the line on which the character at a specified byte offset is contained in a text box control.
- GetTextBoxLineOffset returns the zero-based index of the line on which the character at a specified byte offset is contained in a text box control.
- MakeApplicationActive activates your application and brings its topmost panel to the front.
- PostDeferredCallToThread has the same capabilities as PostDeferredCall on Solaris 2 platforms. On Windows 95/NT, it posts a call to a specific thread.
- SetBitmapData changes the image contents of an existing bitmap.
- SetPanelSize sets the height and width of the panel.

New Error Codes

Table 10 lists the new User Interface Library error codes.

Table 10. New User Interface Library Error Codes

Code	Error Message
-129	The specified operation can be performed only in the thread in which the top-level panel was created.
-130	The specified panel was not found in the .tui file.
-131	The specified menu bar was not found in the .tui file.
-132	The specified control style was not found in the .tui file.
-133	A tag or value is missing in the .tui file.

Changes to the RS-232 Library

This section contains information about changes to the RS-232 Library.

More COM Ports Allowed

The maximum valid value for the **COMPort** parameter to RS-232 Library functions increases from 32 to 1,000.

New Error Code

Table 11 shows the new RS-232 Library error code.

Table 11. New RS-232 Library Error Code

Code	Error Message
-1	Unable to allocate system resources

Changes to the TCP Library

This section contains information on enhancements to the TCP Library.

New TCP Library Functions

The following seven new functions in the TCP Library appear in alphabetical order:

- `GetHostTCPSocketHandle` obtains the system socket handle that corresponds to a TCP Library connection.
- `GetTCPHostAddr` obtains the IP address of the computer on which your program is running.
- `GetTCPHostName` obtains the name of the computer on which your program is running.
- `GetTCPPeerAddr` obtains the IP address of the computer on which a remote client or server is running.
- `GetTCPPeerName` obtains the name of the computer on which a remote client or server is running.
- `GetTCPSystemErrorString` obtains a system message that describes the error that caused a TCP Library function to fail.
- `SetTCPDisconnectMode` sets the method used to close the local conversation handle when a remote client or server terminates a connection.

Changes to Existing TCP Library Functions

For the `ServerTCPRead` and `ClientTCPRead` functions, the timeout parameter is now enforced. The functions now wait until data is available at the port, or until the specified interval expires. In previous versions of LabWindows/CVI, the function did not wait when no data was available at the port. This change might require modifications to your source code.

Changes to the Utility Library

This section contains information on changes and enhancements to the Utility Library.

Changes and Clarifications to Existing Utility Library Functions

For `Delay` and `SyncWait` under Solaris 2, the respective delay and interval parameters can now suspend the execution of the current thread while waiting for the specified interval to expire. This behavior prevents unnecessary use of CPU cycles. This change should not require any modifications to your source code. In previous versions of LabWindows/CVI, the functions waited without releasing control to other threads.

In previous versions of LabWindows/CVI, the `Timer` function returned an invalid result if your program ran for more than 49.71 days. At some point between 49.71 days and 99.42 days

after your first call to `Timer`, the function would return a value 4,294,967.297 seconds less than it should have been. Thereafter, the value would lose another 4,292,967.296 seconds for each 49.71-day period that passed. LabWindows/CVI 5.0.1 corrects this invalid result.

New IVI Library

The IVI (Interchangeable Virtual Instruments) Library gives developers a structured framework for creating VXI*plug&play* instrument drivers with advanced features such as state caching, simulation, and production and development modes. The library is supplemented by the IVI wizard, which automatically creates the skeleton of an IVI driver for you, including source code and function panels. The *LabWindows/CVI Instrument Driver Developers Guide* contains IVI Library function reference information and instructions on how to create IVI drivers.

Changes to the Advanced Analysis Library

This section contains information on enhancements to the Advanced Analysis Library.

New Advanced Analysis Library Functions

LabWindows/CVI 5.0.1 adds 40 new functions to the Advanced Analysis Library. Table 12 presents the new functions within their function tree classes.

Table 12. New Advanced Analysis Functions

Class/Panel Name	Function Name
Vector & Matrix Algebra	
Real Matrices	
Create Special Matrix	`SpecialMatrix`
Determinant (General)	`GenDeterminant`
Invert Matrix (General)	`GenInvMatrix`
Solution of Linear Eqs (General)	`GenLinEqs`
Outer Product	`OuterProduct`
Rank	`MatrixRank`
Norm	`MatrixNorm`
Condition Number	`ConditionNumber`
Eigenvalues & Eigenvectors (Symmetric)	`SymEigenValueVector`
Eigenvalues & Eigenvectors (General)	`GenEigenValueVector`
Singular Values of a Matrix	`SVDS`
SVD Factorization	`SVD`
QR Factorization	`QR`
Cholesky Factorization	`Cholesky`
PseudoInverse Matrix	`PseudoInverse`
Test Positive Definiteness	`CheckPosDef`
Create Special Complex Matrix	`CxSpecialMatrix`
Complex Dot Product	`CxDotProduct`
Complex Transpose	`CxTranspose`

Table 12. New Advanced Analysis Functions (Continued)

Class/Panel Name	Function Name
Complex Determinant	`CxDeterminant`
Complex PseudoInverse Matrix	`CxPseudoInverse`
Complex Trace	`CxTrace`
Complex Invert Matrix	`CxGenInvMatrix`
Solution of Complex Linear Eqs	`CxGenLinEqs`
Complex Multiply Matrices	`CxMatrixMul`
Complex Outer Product	`CxOuterProduct`
Complex Rank	`CxMatrixRank`
Complex Norm	`CxMatrixNorm`
Complex Condition Number	`CxConditionNumber`
Complex Eigenvalues & Eigenvectors	`CxEigenValueVector`
Complex Singular Values	`CxSVDS`
Complex SVD Factorization	`CxSVD`
Complex QR Factorization	`CxQR`
Complex Cholesky Factorization	`CxCholesky`
Complex Test Positive Definite	`CxCheckPosDef`
Complex LU Factorization	`CxLU`
Additional Numerical Methods	
Complex Polynomial Roots	`CxPolyRoots`
Numeric Integration	`NumericIntegration`
Peak Detector	`PeakDetector`
Free Analysis Memory	`FreeAnalysisMem`

Changes to the Programmer Reference Manual

This section contains additions to the *LabWindows/CVI Programmer Reference Manual*.

Stack Size

After you install the LabWindows/CVI development environment, the default stack size for program execution increases to 250 KB.

Details of User Protection

The *Avoid Unassigned Dynamic Allocation in Function Parameters* section is new to the *Details of User Protection* discussion in Chapter 1, *LabWindows/CVI Compiler,* of the *LabWindows/CVI Programmer Reference Manual*.

Clarification about Modifying the DST Rules String

The LabWindows/CVI ANSI C Library uses the DST (daylight savings time) rules string to determine the period of each year in which daylight savings time is in effect. The *Time and Date Functions* section of Chapter 1, *ANSI C Library,* of the *LabWindows/CVI Standard Libraries Reference Manual* contains an improved discussion of the DST rules string and how you can modify the string.

General Information

This section contains programming information about the use of LabWindows/CVI under Solaris 2.

Incompatibilities between LabWindows/CVI, Sun Solaris, and ANSI C

Under the ANSI C standard, the programmer who implements the library chooses how certain functions behave. As a result, two implementations of a function can behave differently and still conform to the ANSI standard. This section outlines some of the differences between LabWindows/CVI (which uses the Sun Solaris C library), the ANSI standard, and the Solaris 2 operating system.

☞ **Note** ***None of these incompatibilities interfere with development of projects and standalone executables in LabWindows/CVI for Solaris 2.***

Differences between LabWindows/CVI and ANSI C

Under some versions of Sun Solaris 2, such as Solaris 2.5, the ANSI function `setlocale` does not work properly when running programs in the LabWindows/CVI development environment. LabWindows/CVI links programs in the development environment to the Sun Solaris static library `libc.a`, which contains a limited version of `setlocale`. In contrast, LabWindows/CVI links standalone executables to the shared library `libc.so`, which contains the fully functional version of `setlocale`.

Differences between LabWindows/CVI and Sun Solaris 2

The following differences exist between LabWindows/CVI and Sun Solaris 2:

- LabWindows/CVI does not support the `long long` data type used by some header files under Solaris 2. In LabWindows/CVI, you cannot use that data type or call any function that uses that data type.
- LabWindows/CVI implements the data type `long double` as an 8-byte object, in the same way that it implements `double`. Sun Solaris implements `long double` as a 16-byte object. As a result, any Sun Solaris function that uses `long double` does not work properly in LabWindows/CVI.
- The LabWindows/CVI implementation of the `printf` and `scanf` family of functions does not support the Sun Solaris implementation of `long double`.
- LabWindows/CVI does not support wide character constants (`wchar_t`) of the form `L'ab'`.
- The data types `jmp_buf` and `sigjmp_buf` defined in the header file `setjmp.h` are different for LabWindows/CVI and Sun Solaris. The LabWindows/CVI versions of these buffers are larger than the Sun Solaris versions because LabWindows/CVI stores additional debugging information in them. As a result, you must be careful when using `jmp_buf` and `sigjmp_buf` objects among multiple files. In particular, if you compile a

file in LabWindows/CVI with debugging enabled and the file uses `setjmp` or `longjmp`, your program must include the LabWindows/CVI `setjmp.h` to handle those functions correctly. The same is true for `sigjmp_buf`, `sigsetjmp`, and `siglongjmp`.

320688E-01 Dec98