Moving from BASIC to C with LabWindows®/CVI

John Pasquarette

Introduction

The instrumentation industry has historically used the BASIC language for automating test and measurement systems. Most experienced instrument programmers have some history of developing systems using HP-BASIC, HT-BASIC, or QuickBASIC. Today, more and more instrumentation programmers are looking to the more powerful and flexible C language to build their systems. Indeed, the professional software community has standardized on C and C++ for the majority of today's large-scale software products such as spreadsheets, word processors, and graphics packages. Although the benefits are numerous, the transition from BASIC to C can be daunting to the long-time BASIC programmer. This application note focuses on how instrumentation programmers can simplify this transition using LabWindows/CVI, an interactive C programming environment. It outlines the benefits for instrumentation system developers of moving from BASIC to C using LabWindows/CVI and describes how some common BASIC programming tasks translate to LabWindows/CVI.

More Choices with C

As a standard, widely-used programming language, C offers the largest collection of software tools and libraries to help with your programming tasks. If you are using C, you probably can find a C library to accelerate your programming efforts. Indeed, a whole industry of products and utilities designed for C programmers has evolved, including compilers, debuggers, source code control tools, documentation aids, and libraries for almost every known operation. Many of these tools are also portable across platforms. C is generally the first available language on new platforms or operating systems because it is so standardized. For example, the Microsoft SDK (a C programming environment) was introduced with Windows 3.0 as the first tool available for building Windows applications. The growing set of supporting development and maintenance tools and the ability to quickly adapt to new operating systems and platforms makes C a choice with guaranteed long-term stability and flexibility for instrumentation systems.
An Interactive Approach to C Programming

Most users would agree that C is a more powerful language, and that C offers more development tools and long-term stability than BASIC. However, the real question is how can BASIC users move to C without battling a long and arduous learning curve. The solution is LabWindows/CVI, an interactive C programming environment that combines the power and flexibility of ANSI C with the ease of developing in BASIC. The single biggest advantage of using BASIC is its interactive, interpreted programming approach. BASIC users have enjoyed the ability to quickly test their code by simply entering individual lines of code, or sections of code, and running them interactively. Although lacking the performance and flexibility of compiled C, these interpreted BASIC environments offer a straightforward approach to program development.

LabWindows/CVI provides an interactive C environment where users can experiment with sections of code while they develop, for a faster, more productive programming effort. With LabWindows/CVI, you can develop C programs without being hindered with tedious compiling and linking details. You simply enter code and select Run Project. The built-in ANSI C compiler compiles and links your project and runs the program. In addition, LabWindows/CVI has an Interactive Execution Window where you can execute single lines of code or sections of code to test their operation. You can even set breakpoints interactively in your source code. When program execution reaches a breakpoint, you can interactively execute library functions, call functions in your own source code, or view program variables, and then continue with program execution. With LabWindows/CVI, you get the power of compiled C with a development approach very similar to interpreted BASIC.

Another key element in LabWindows/CVI that makes C programming easier is the function panel. A function panel is a graphical representation of a library function in LabWindows/CVI. Each library function has a function panel. With function panels, you can interactively build function calls and run them to learn how they operate. In addition, function panels have built-in help information to teach you about each function and its parameters. Once you have built and run the function call to see how it operates, you can automatically paste the function call into your source code. These automatic code-generation and execution tools make C programming in LabWindows/CVI straightforward. To learn more about the interactive programming tools in LabWindows/CVI, refer to Application Note 52, Interactive Programming for Instrumentation.

Figure 1 shows a function panel that you use for interactive code-generation and execution. By entering values in each control on the function panel, you automatically build the function call at the bottom of the window. By selecting Run Function Panel from the Code menu, you can
execute the function call. **Insert Function Call** pastes the line of code from the bottom of the function panel into your source code.

![LabWindows/CVI Function Panel](image)

**Figure 1. LabWindows/CVI Function Panel to Interactively Build and Run a Function Call**

**Improvements for Instrumentation Programming**

So far, you have learned that with LabWindows/CVI you can program in C using a method that is similar to the method used for programming in BASIC. What about improvements over using BASIC? Because LabWindows/CVI is designed specifically for developing instrumentation systems, it offers the following advantages that help you save time and money over programming in traditional BASIC environments.

- Increased productivity with instrument drivers
- Simplified GUI programming
- Compatibility with your current test programming methodology
- Protection against time-consuming memory errors
- Improved productivity with modern development tools

Each of these advantages, which LabWindows/CVI offers BASIC users, is discussed more thoroughly in the following paragraphs.
Increased Productivity with Instrument Drivers

Perhaps the most important feature of LabWindows/CVI for instrument control is its huge library of instrument drivers. An instrument driver is a custom library for controlling a particular GPIB or VXI instrument. Using instrument drivers can save tremendous amounts of development time because you no longer need to learn low-level ASCII command strings to control your instrument—the driver does that for you. The following list compares the steps required to transfer a waveform from a GPIB oscilloscope with and without an instrument driver.

- Steps required to transfer a waveform from a GPIB oscilloscope WITH an instrument driver:
  1. Load the instrument driver into LabWindows/CVI.
  2. Initialize the instrument.
  3. Find the ReadWaveform function in the instrument driver tree.
  4. Enter values in the function panel. (Check the online help if necessary.)
  5. Run the function panel. The waveform is returned to you in an array variable.

- Steps required to transfer a waveform from a GPIB oscilloscope WITHOUT an instrument driver:
  1. Look up the proper command in the instrument’s programming manual for transferring a waveform.
  2. Learn how to send strings back and forth over GPIB.
  3. Send the command to the scope over GPIB.
  4. Read the response from the scope (a string).
  5. Look up the format of the data returned in the programming manual.
  6. Parse the string to separate the datapoints from the header information.
  7. Use the header information to scale the datapoints to voltage values.

If you ever need to change instruments, you have to relearn the programming commands for the new instrument when you do not have an instrument driver. With instrument drivers, you do not have to learn all of the instrument programming details. Instrument drivers give you a high-level set of functions for communicating with the instrument. Instrument drivers act as a translator between your language and the instrument’s language.

For VXI users, LabWindows/CVI instrument drivers written for VXI instruments work on all three major interfaces for controlling the VXI bus: GPIB-VXI, MXI, and embedded VXI controller. Therefore, you can start developing your VXI system using a low-cost GPIB...
interface to control your chassis, and later move up to a high-performance MXI interface or embedded controller without changing your source code!

Instrument drivers have the same function panel interface as every other LabWindows/CVI library, so you can interactively build function calls and test your instrument using function panels before you begin writing test programs. And, each instrument driver is delivered in source code, so you can optimize the driver for your particular application, or convert the driver to work with other instruments quickly and easily. A sample instrument driver function panel is shown in Figure 2.

![Sample Instrument Driver Function Panel](image)

**Figure 2. Sample Instrument Driver Function Panel**

**Simplified GUI Programming**

Most BASIC programming environments have a graphics library for creating a user interface to display data to the end user. Outside of programming the instruments, instrumentation system developers probably spend the most time developing and manipulating user interface objects using these graphics libraries. User interface design and graphics manipulation can be a tedious, time-consuming task. LabWindows/CVI has a number of user interface tools that greatly simplify these tasks, saving you hours or even days of development time.

*With LabWindows/CVI, you do not have to develop any source code or call library functions to build your user interfaces.* LabWindows/CVI has an interactive graphical user interface (GUI)
editor for building your GUIs. To build a GUI, you simply select objects from the Create menu, shown in Figure 3, and place them on your screen.

![Create Menu](image)

Figure 3. Selecting Objects from the Create Menu to Build a GUI

The graphical objects (controls and indicators) in LabWindows/CVI are designed specifically for instrumentation systems. You can choose from knobs, meters, dials, gauges, graphs, strip charts, push buttons, pull-down menus, and thermometer controls to build GUIs in LabWindows/CVI. Each control and indicator in LabWindows/CVI has built-in functionality for displaying technical data in a manner familiar to scientists and engineers. For example, you can set options to make a graph control display the waveform with logarithmic scales and with engineering units on the graph axes. Or, you can set the format, precision, range, and numeric display of a knob control through a series of controls and dialog boxes. You access these editing tools through the
dialog box shown in Figure 4. By setting options for the controls in the User Interface Editor, you can eliminate the need for tedious GUI programming in your source code.

![Edit Dialog Box for a Knob Control](image)

Figure 4. Edit Dialog Box for a Knob Control

LabWindows/CVI simplifies event-driven programming for responding to your GUI. As PCs and operating systems become more advanced, their applications become more advanced. Today, Windows programs feature extensive GUIs for displaying data and controlling program execution. The programs controlling these GUIs are much different than the traditional DOS-based, sequential BASIC programs of the past. Understanding and developing Windows event-driven programs can be a complex and difficult undertaking for any programmer.

LabWindows/CVI has built-in event-handling capability that simplifies event-driven programming significantly. Each GUI object has its own section of C code (called a callback function) that is automatically executed whenever the control is operated. For example, you may have a push button called Acquire on your GUI that calls a function called AcquireData whenever the button is pushed. Therefore, your C programs are simply a collection of functions that are associated with each one of your GUI objects—LabWindows/CVI does the rest for you. You do not have to learn any of the low-level operating system details to develop extensive,
modern, event-driven programs. For more information about the GUI capabilities of LabWindows/CVI, refer to Application Note 54, *Windows GUI Techniques for Instrumentation*.

![Simplified Windows Programming Model](image)

**Figure 5. The LabWindows/CVI User Interface Programming Model**

**Compatibility with Your Current Test Programming Methodology**

LabWindows/CVI is an open development system, so there are many ways for you to combine, incorporate, or leverage off existing external programs from LabWindows/CVI. The following list outlines the most common methods for incorporating existing code or programs within your LabWindows/CVI applications:

- **Calling External Programs**—LabWindows/CVI has built-in hooks for running external programs. You can launch any DOS or Windows program from the LabWindows/CVI environment using the `system()` function in the ANSI C library or the `LaunchExecutable()` function. This is a great way for you to incorporate existing BASIC programs into a test executive built with LabWindows/CVI. *You can start programming with C in LabWindows/CVI without leaving behind your existing BASIC programs.*

- **ANSI C Source Code**—LabWindows/CVI is an ANSI C programming environment. Therefore, any source code that is ANSI C-compatible will run in LabWindows/CVI.

- **DLLs, Libraries, and Object Modules**—From LabWindows/CVI, you can call functions in a Windows 16-bit DLL (created with any compiler that can build a DLL). In addition, you can
call functions in 32-bit C libraries or object modules created from Borland, Symantec, or WATCOM 32-bit C compilers.

- DDE/TCP—LabWindows/CVI has two built-in interprocess communication libraries, the DDE Library (dynamic-data-exchange for Windows only) and the TCP Library. With these libraries, you can pass data to other applications or to other computers where the data can be stored, analyzed, or manipulated in some way.

### Protection Against Time-Consuming Memory Errors

One of the biggest problem areas for BASIC users who convert to C is memory management. In BASIC, string variables (character arrays) can change length dynamically at runtime. Therefore, if you try to write 20 elements into a 10-element length string, BASIC simply grows the string to fit the data into the variable. However, C does not automatically allocate strings or arrays. In most C programming environments, you will cause major problems when you try to write 20 elements of data into a 10-element string array. In C, the data will be written into memory, however, only the first 10 elements will be in the string array. The second 10 elements will be written into the adjacent memory space, overwriting the variables that are stored in the adjacent memory locations. Therefore, your program will exhibit unpredictable results, because variable values are changed randomly. These memory errors can be the most difficult and time-consuming to track down and fix because they only occur under certain conditions.

LabWindows/CVI has a built-in debugging feature called **user protection** which helps you avoid these memory errors. With user protection, LabWindows/CVI automatically checks all array and string operations to ensure that the variable sizes are large enough to handle the data you attempt to store within them. Therefore, when you attempt to overwrite variables in memory unknowingly, LabWindows/CVI stops, warns you of the problem, and highlights the particular array variable to which you are trying to write data. User protection also protects you against other common memory errors in C, such as indexing past the end of an array, passing parameters incorrectly to library functions, and responding to function return values when errors occur.

### Improved Productivity with Modern Development Tools

Most BASIC programming environments are established, mature products that have not been upgraded in some time. These environments do not take advantage of modern development tools and capabilities of today's operating systems. LabWindows/CVI is a modern programming environment, built around a multiwindow development approach. LabWindows/CVI has a Project window that tracks all the source files, user interface files, instrument drivers, and libraries or object modules being used in your program, so that you do not have to track them. In addition, the debugging and editing tools in LabWindows/CVI rival those found in general-purpose programming environments. These tools include variable displays, watch windows, dynamic breakpoints, conditional breakpoints, a full-functioning source code editor, and a 32-bit ANSI C compiler/linker. Although these tools offer the power and flexibility of other professional C environments, LabWindows/CVI also offers fast and easy program development tools for the novice programmer. For more information on the debugging and editing tools in LabWindows/CVI, refer to Application Note 51, *Debugging and Editing Tools for Instrumentation Programmers*.
Converting Your Programs from BASIC to LabWindows/CVI

After you decide to move from BASIC to C using LabWindows/CVI, you must decide what you are going to do with your existing BASIC programs. If your BASIC programs run on the same platforms as LabWindows/CVI\(^1\) you can continue to use these programs as you develop new programs in LabWindows/CVI. However, you may decide to translate your BASIC programs to run in LabWindows/CVI. Reimplementing these programs using LabWindows/CVI should be much faster and easier than your original programming effort.

Most of the simple programming structures and operations such as declaring variables, for loops, while loops, subroutines, and functions will translate directly from BASIC to C. You simply must learn the equivalent syntax for using the programming structures in C. The application-specific operations are the areas where you will save the most time in developing with LabWindows/CVI. Areas such as instrument control and user interface design and manipulation can be accomplished much more efficiently and effectively using LabWindows/CVI. The following examples may help put your translation effort into perspective.

Instrument Control Operations

Instrument control in LabWindows/CVI is best accomplished using an instrument driver. As discussed earlier, when you use an instrument driver, you no longer need to learn the low-level command strings and data parsing and scaling information for your instrument. However, if you have an HP-BASIC program that uses \texttt{INPUT} and \texttt{OUTPUT} calls to control an instrument with ASCII command strings, you can use the exact same strings in \texttt{ibwrt} and \texttt{ibrd} calls in LabWindows/CVI.

In HP-BASIC, you may use the following sequence of commands to make a measurement from a voltmeter:

\begin{verbatim}
10 OUTPUT @Device;"*RST" !** Reset measurement state.
20 OUTPUT @Device;"*CLS" !** Reset status system.
30 OUTPUT @Device;":MEAS:VOLT:DC? 10,0.01" !** Make measurements.
40 !
50 ENTER @Device;Reading !** Retrieve a reading.
\end{verbatim}

You could do the same thing in LabWindows/CVI using the following set of function calls:

\begin{verbatim}
ibwrt (Device, "*RST", 3); /* Reset measurement state */
ibwrt (Device, "*CLS", 3); /* Reset status system */
ibwrt (Device, "*MEAS:VOLT:DC? 10,0.01", 21); /* Make measurements */
ibrd (Device, Reading, 100); /* Retrieve a reading */
\end{verbatim}

\(^1\) Contact National Instruments for current and anticipated LabWindows/CVI platform compatibility.
User Interface Operations

Creating interactive user interfaces for your program is a common task for modern test systems. With LabWindows/CVI, you create your graphic displays in a drag-and-drop graphical editor—there is no source code required to build a GUI. Therefore, you can save hours of development time by building your user interface interactively instead of creating it through function calls.

With HP-BASIC PLUS, you can create graphs, strip charts, and widgets (user interface controls) with a set of function calls. Compare the difference in using source code or an interactive editor for building your GUIs:

With HP-BASIC PLUS, you can build a graph control for displaying waveform data with the following function calls:

```plaintext
10   ! Create the XY graph and make it invisible until all
20   ! modifications are complete.
30   ASSIGN @Xy TO WIDGET "XY GRAPH"; SET("VISIBLE":0)
40   !
50   ! Modify the attributes of the graph.
60   !
70   CONTROL @Xy; SET("SHARED X":1)
80   CONTROL @Xy; SET("MOVABLE":0)
90   CONTROL @Xy; SET("TRACE COUNT":1)
100  CONTROL @Xy; SET("X":0,"Y":0)
110  CONTROL @Xy; SET("HEIGHT":Height,"WIDTH":Width)
120  CONTROL @Xy; SET("TITLE":"Fourier Series - Square Wave")
130  !
140  ! Set attributes unique to each axis.
150  !
160  CONTROL @Xy; SET("CURRENT AXIS":"X")
170  CONTROL @Xy; SET("AXIS LABEL":"Radians")
180  CONTROL @Xy; SET("AUTOSCALE":1)
190  CONTROL @Xy; SET("CURRENT AXIS":"Y")
200  CONTROL @Xy; SET("AXIS LABEL":"Amplitude")
210  CONTROL @Xy; SET("AUTOSCALE":1)
220  !
230  ! Set attributes unique to each trace.
240  !
260  CONTROL @Xy; SET("TRACE PEN":1)
270  CONTROL @Xy; SET("TRACE LABEL":"1st Order")
280  CONTROL @Xy; SET("X DATA":X_data(*))
290  CONTROL @Xy; SET("Y DATA":Y1_data(*))
300  !
310  ! Modifications are complete, so show the graph.
320  !
330  CONTROL @Xy; SET("VISIBLE":1)
340  PAUSE
```
With LabWindows/CVI, you can create the graph control by selecting it from the **Create** menu in the user interface editor and placing it on your screen, as shown in Figure 6.

![Graph Control Menu](image)

**Figure 6. Creating a Graph Control Using LabWindows/CVI**

You set the attributes for the graph control using the dialog boxes (shown in Figure 7) in the User Interface Editor. Once you have created and customized your graphics objects for displaying data and retrieving user input, you save these objects in a user interface resource (UIR) file.
To use the graphic objects from your program, you simply load them into memory using the `LoadPanel` function, and display the objects using the `DisplayPanel` function. You can then use a series of functions for passing data back and forth between your graphics objects and your program. For example, traces are plotted to a graph control using a single `PlotY` function. You can set the parameters in the `PlotY` function shown below to configure the color, plot style, point style, and trace style for each trace.

```c
handle = LoadPanel (0, "display.uir", PANEL);
DisplayPanel (handle);
    PlotY (handle, PANEL_GRAPH, Y1, 100, VAL_DOUBLE, VAL_THIN_LINE
          VAL_EMPTY_SQUARE, VAL_SOLID, 1, VAL_RED);
```

As you can see, developing GUIs in LabWindows/CVI is much more intuitive than using function calls to set up your graphics. You can quickly design all of your controls and indicators up front, specifying the labels, colors, position, size, range, precision, and specific operation attributes. You only have to write code that passes data between your program and these user interface objects. In addition, you can dynamically change any of the control attributes with function calls from your program, if required.
LabWindows/CVI—C for the Instrumentation Programmer

LabWindows/CVI offers an interactive C programming environment that most BASIC users can become comfortable with quickly. The development tools in LabWindows/CVI are designed to bring BASIC users up to speed quickly and remove the traditional complexities associated with building C programs. However, LabWindows/CVI does much more than provide a compiled C programming environment that looks and acts like interpreted BASIC. LabWindows/CVI adds development tools specifically designed to accelerate your instrumentation system development efforts. With the help of instrument drivers, the user interface tools, and the advanced code-generation, debugging, and editing tools in LabWindows/CVI, you can build instrumentation programs faster, more efficiently, and more effectively than you ever have before.