Worldwide Technical Support and Product Information
ni.com

National Instruments Corporate Headquarters
11500 North Mopac Expressway  Austin, Texas 78759-3504  USA  Tel: 512 683 0100

Worldwide Offices
Australia 1800 300 800, Austria 43 0 662 45 79 90 0, Belgium 32 0 2 757 00 20, Brazil 55 11 3262 3599, Canada 800 433 3488, China 86 21 6555 7838, Czech Republic 420 224 235 774, Denmark 45 45 76 26 00, Finland 385 0 9 725 725 11, France 33 01 48 14 24 24, Germany 49 0 89 741 31 30, India 91 80 41190000, Israel 972 0 3 6393737, Italy 39 02 413091, Japan 81 3 5472 2970, Korea 82 02 413091, Malaysia 1800 887710, Mexico 01 800 010 0793, Netherlands 31 0 348 433 466, New Zealand 0800 553 322, Norway 47 0 66 90 76 60, Poland 48 22 3390150, Portugal 351 210 311 210, Russia 7 095 783 68 51, Singapore 1800 226 5886, Slovenia 386 3 425 4200, South Africa 27 0 11 805 8197, Spain 34 91 640 0085, Sweden 46 0 8 587 895 00, Switzerland 41 56 200 51 51, Taiwan 886 02 2377 2222, Thailand 662 278 6777, United Kingdom 44 0 1635 523545

For further support information, refer to the Additional Information and Resources appendix. To comment on National Instruments documentation, refer to the National Instruments Web site at ni.com/info and enter the info code feedback.
Contents

Student Guide
A. NI Certification ..................................................................................................... ix
B. Course Description ............................................................................................... x
C. What You Need to Get Started ............................................................................. x
D. Installing the Course Software .............................................................................. x
E. Course Goals ......................................................................................................... xi
F. Course Conventions .............................................................................................. xi

Lesson 1
Introduction to LabWindows/CVI

Introduction................................................................................................................. 1-1
Timesaving Features of LabWindows/CVI ............................................................... 1-2
LabWindows/CVI Libraries ....................................................................................... 1-4
Customizing the Library Menu ................................................................................... 1-6
LabWindows/CVI Windows—Workspace Window .................................................. 1-7
LabWindows/CVI Windows—Source Window ......................................................... 1-9
LabWindows/CVI Windows—User Interface Editor ................................................. 1-10
Exercise 1-1 Examining the LabWindows/CVI Environment ................................ 1-12
Exercise 1-2 Examining the LabWindows/CVI Source Window ......................... 1-15
LabWindows/CVI Menus ........................................................................................... 1-20
LabWindows/CVI Toolbars ...................................................................................... 1-21
LabWindows/CVI Options ......................................................................................... 1-22
Exercise 1-3 Examining the LabWindows/CVI Build Options Dialog Box .......... 1-24
Opening Function Panels ............................................................................................ 1-29
Function Panels........................................................................................................... 1-30
Function Panels—Help ................................................................................................ 1-31
Function Panels—Variables ....................................................................................... 1-32
Interactive Execution Window ................................................................................... 1-33
Exercise 1-4 Creating a LabWindows/CVI Project ................................................. 1-35
Exercise 1-5 Developing a LabWindows/CVI Program ......................................... 1-44
LabWindows/CVI Data Structures ............................................................................. 1-46
Scope of Variables ...................................................................................................... 1-47
Debugging................................................................................................................... 1-49
Compiling Files .......................................................................................................... 1-51
Excluding Lines .......................................................................................................... 1-52
Breakpoints ................................................................................................................ 1-53
Conditional Breakpoints ............................................................................................ 1-54
Single-Stepping .......................................................................................................... 1-55
Variables Window ....................................................................................................... 1-56
Variable Tooltips ......................................................................................................... 1-57
Watch Window ............................................................................................................. 1-58
Graphical Variable Watches ....................................................................................... 1-60
Exercise 1-6 Using the Variables Window .............................................................. 1-62
Contents

Exercise 1-7  Source Code Stepping and Debugging .............................. 1-66
Summary Lesson 1 ................................................................................ 1-70
Tips and Shortcuts ............................................................................. 1-71

Lesson 2
Graphical User Interface

Introduction ...................................................................................... 2-1
Graphical User Interface (GUI) ....................................................... 2-2
User Interface Editor ....................................................................... 2-3
Creating a GUI ................................................................................ 2-4
Creating a Panel ............................................................................. 2-5
Panel Properties ............................................................................ 2-6
Creating Controls .......................................................................... 2-7
Control Properties ......................................................................... 2-8
Control Labels and Shortcut Keys ................................................ 2-9
Custom Controls ........................................................................... 2-10
Tab Order ......................................................................................... 2-11
Graph Control Features .................................................................. 2-12
User Interface Header File ............................................................. 2-14
Exercise 2-1  Running a Program Containing a GUI ......................... 2-16
Exercise 2-2  Experimenting with User Interface Features .................. 2-18
GUI Controls .................................................................................. 2-20
Exercise 2-3  Building a GUI .......................................................... 2-26
User Interface Library ..................................................................... 2-30
Controlling a GUI ........................................................................... 2-32
Set Control Value .......................................................................... 2-34
Get Control Value .......................................................................... 2-35
Set Control Attribute ...................................................................... 2-36
Get Control Attribute ...................................................................... 2-39
Panel Attributes ............................................................................. 2-40
Event-Driven Programming ............................................................ 2-41
Viewing Run-Time Events ............................................................... 2-42
LabWindows/CVI GUI Events ......................................................... 2-43
Control Modes ................................................................................ 2-44
Processing Events .......................................................................... 2-45
Relationship between RunUserInterface and QuitUserInterface .... 2-46
CodeBuilder .................................................................................... 2-47
Exercise 2-4  Writing an Event-Driven Program ................................. 2-50
Exercise 2-5  Developing a Well-Behaved Application ....................... 2-55
Pop-Up Panels ................................................................................. 2-56
Timer Controls ................................................................................ 2-57
Strip Charts ...................................................................................... 2-58
Exercise 2-6  Using a Timer Control ................................................ 2-62
Exercise 2-7  Continually Incrementing/Decrementing a Counter
(Additional Challenge Exercise) ...................................................... 2-66
Lesson 3
Instrument Drivers

Introduction.................................................................................................................3-1
Calling Functions from Other Modules.................................................................3-2
What is an Instrument Driver?................................................................................3-3
Instrument Driver Files..........................................................................................3-4
Loading an Instrument Driver...............................................................................3-5
Instrument Driver Function Tree ..........................................................................3-6
Creating an Instrument Driver ..............................................................................3-7
Creating a .fp File ..................................................................................................3-8
Creating Instruments..............................................................................................3-9
Creating Classes.................................................................................................3-10
Creating Function Trees.......................................................................................3-11
Creating Function Panels.....................................................................................3-12
Documenting the Instrument Driver .....................................................................3-13
Generating the Header File and the Source Code Skeleton.................................3-14
Header File and the Source Code Skeleton ...........................................................3-15
Editing Instrument Driver Files ...........................................................................3-16
Exercise 3-1 Creating and Using an Instrument Driver .........................................3-19

Lesson 4
Formatting and Scanning

Introduction............................................................................................................4-1
Data Representation and Conversion....................................................................4-2
LabWindows/CVI vs. ANSI C................................................................................4-3
Formatting and Scanning Functions.......................................................................4-4
Format String Specifiers.......................................................................................4-7
Array Formatting Example....................................................................................4-9
Array Scanning Example.......................................................................................4-10
Removing Headers...............................................................................................4-11
Modifying Byte Order..........................................................................................4-12
Discarding Unwanted Data..................................................................................4-13
Formatting Examples............................................................................................4-14
Exercise 4-1 Using a Scanning Function...............................................................4-16
Summary Lesson 4.................................................................................................4-17

Lesson 5
Distributing LabWindows/CVI Applications

Introduction............................................................................................................5-1
Release Executables...............................................................................................5-2
Creating Release Executables..............................................................................5-3
Necessary Files......................................................................................................5-4
Distributing Executables.......................................................................................5-5
Exercise 5-1 Creating a Distribution Kit...............................................................5-7
Lesson 6
Data Acquisition

Introduction................................................................................................................. 6-1
Overview and Configuration ...................................................................................... 6-2
Measurement Software Framework............................................................................ 6-4
MAX ........................................................................................................................... 6-5
Channels and Tasks .................................................................................................... 6-7
Exercise 6-1  MAX................................................................................................. 6-9
Analog Input ............................................................................................................... 6-13
Single Point Analog Input........................................................................................... 6-17
Exercise 6-2  Single Point Acquisition .................................................................... 6-19
Multiple Point Acquisition.......................................................................................... 6-23
Exercise 6-3  Multiple Point Acquisition ............................................................... 6-25
Continuous Acquisition .............................................................................................. 6-30
Exercise 6-4  Continuous Acquisition .................................................................... 6-32
Single Point Analog Output........................................................................................ 6-38
Exercise 6-5  Voltage Output ................................................................................. 6-40
Multiple Point Voltage Output ................................................................................... 6-44
Continuous Multiple Voltage Output ......................................................................... 6-45
Exercise 6-6  Continuous Voltage Output ............................................................... 6-47
Digital Input and Output ............................................................................................. 6-54
Making Digital I/O Signal Connections ..................................................................... 6-55
Digital Input ................................................................................................................ 6- 56
Digital Output ............................................................................................................. 6-57
Exercise 6-7  Digital I/O......................................................................................... 6-59
Counters ...................................................................................................................... 6-63
Exercise 6-8  Simple Event Counting..................................................................... 6-65
DAQ Assistant Overview ........................................................................................... 6-70
Creating a Task with the DAQ Assistant.................................................................... 6-71
Configuring a Task with the DAQ Assistant ............................................................. 6-72
Generating Example Code .......................................................................................... 6-74
Exercise 6-9  DAQ Assistant.................................................................................. 6-76

Lesson 7
Instrument Control

Introduction.................................................................................................................. 7-1
Instrument Control Overview .................................................................................... 7-2
GPIB Communications ............................................................................................... 7-4
Hardware Specifications ............................................................................................. 7-5
GPIB Software Architecture ...................................................................................... 7-7
Configuring GPIB Board and Instruments ................................................................ 7-8
Exercise 7-1  GPIB Configuration with MAX ....................................................... 7-10
Virtual Instrument Software Architecture ............................................................... 7-12
VISA Programming Terminology ............................................................................. 7-14
VISA Resource Name ............................................................................................... 7-15
Contents

VISA Example............................................................................................................7-16
Exercise 7-2 Programming with VISA.................................................................7-18
What is the Instrument I/O Assistant? .................................................................7-24
Communicating with an Instrument .................................................................7-25
Exercise 7-3 Using the Instrument I/O Assistant ............................................7-28
Instrument Drivers ............................................................................................7-31
Instrument Driver Model ..................................................................................7-32
Component Functions .......................................................................................7-33
Instrument Driver Network (IDNET) ...............................................................7-35
Exercise 7-4 Instrument Drivers ..................................................................7-37
Serial Communication ......................................................................................7-41
Serial Hardware Connection .............................................................................7-42
Serial Communication Parameters .................................................................7-44
Using the Instrument I/O Assistant for Serial Communication ......................7-46
Exercise 7-5 Serial Write & Read ..................................................................7-48

Appendix A
Regular Expression Characters

Appendix B
File I/O

Appendix C
Formatting and Scanning Modifiers

Appendix D
ASCII Character Code Equivalents

Appendix E
Additional Information and Resources

Course Evaluation
Introduction
This lesson discusses how to use instrument drivers in LabWindows/CVI. You also learn how to create instrument drivers in this lesson.
Calling Functions from Other Modules

You can call functions from other modules by adding to your project a .c file that contains the functions you want to call. To make the function calls, pass the necessary parameters to the functions that are defined in the other module. Also, ensure that the file calling the function in the other module contains the necessary #include statements. Notice that in the example in the figure above, names_displayName and names_storeName are defined in the names.c module. The phonebook.c module, which contains main, also contains the #include “names.h” statement because the functions must be prototyped before you can use them within phonebook.c.

If you access variables in different modules, you must pay particular attention when defining the variables. For more information, refer to the Scope of Variables section in Lesson 1, Introduction to LabWindows/CVI.
What is an Instrument Driver?

- Collection of functions
- Organized as a separate module
  - .c
  - .lib
  - .obj
  - .dll
- Function panel (.fp) front end

What is an Instrument Driver?
As discussed, you can have multiple .c files in one project. You also can separate your functions into multiple files to organize the functions logically. An instrument driver is a collection of functions, organized in a .c, .lib, .obj, or .dll file, with a function panel front end. The organization is almost identical to the multiple modules model discussed previously. An instrument driver defines function panels, which you use throughout this course, to map to functions in your own .c files.

In this lesson, you will learn to create an instrument driver as a library of functions. Under this framework, you will learn how instrument drivers in LabWindows/CVI help simplify instrument control.
**Instrument Driver Files**

- **Source Code**—.c
- **Include File**—.h
- **Function Panels**—.fp
- **Attribute File**—.sub
- Files must reside in the same directory

---

**Instrument Driver Files**

The following file types are associated with LabWindows/CVI instrument drivers.

- The instrument driver program, which can be a .c, .obj, .dll, or .lib file
- The instrument driver include (.h) file, which contains function prototypes and constant declarations
- The instrument function panel (.fp) file, which contains information that defines the function tree, the function panels, and the help text
- An ASCII text (.doc) file, which contains documentation for the instrument driver
- A .sub file, which defines all attributes available for the instrument when using IVI instrument drivers
Loading an Instrument Driver

To use an instrument driver, load the instrument driver in the Library Tree. Right-click the Instruments folder and select Load Instrument. In the Load Instrument dialog box, select the instrument driver you want to load, and click Load. You also can load instrument drivers by dropping .fp files onto the Library Tree directly.

To load an instrument driver through the Instrument menu, select Instrument»Load. When you load the instrument driver, its name appears under the Instrument menu.

You also can load an instrument driver by adding its function panel to your project. When you use this method, LabWindows/CVI reloads the instrument driver every time you launch LabWindows/CVI. For this reason, National Instruments recommends that you load instrument drivers by adding the function panels to your project.
Instrument Driver Function Tree

Use the instrument driver exactly as you use the standard LabWindows/CVI libraries. Navigate through the **Instruments** folder of the Library Tree until you find the function you want to use. Select the function, and its function panel appears, representing the instrument driver function with a control for each parameter. By manipulating the controls, you can construct a specific function call that you can either interactively execute or paste into a program.
Creating an Instrument Driver

- Create function tree (.fp file)
- Create function panels for functions
- Create documentation directly from Function Tree Editor
- Create header file and source code skeleton directly from Function Tree Editor
- Complete all function definitions in source code skeleton

Creating an Instrument Driver

To create an instrument driver, you must first create the function tree in the function panel file (.fp). The function tree contains all the function names categorized in different classes. When you create the function tree, you can create a function panel for each function. Fill in the online help as you create your functions. When you write the function panel help for all the functions in your function tree, you can automatically create a .doc file that contains all the help documented on each function. LabWindows/CVI can automatically create a source code skeleton and a header file that contains the function prototypes of all the functions in the function panel file. When you create the source code skeleton and the header file, add all the necessary code to create working functions.
Creating an .fp File

To create a new instrument driver, select **File»New»Function Tree (*.fp)**. The Function Tree Editor appears. In this window, you can define all the functions in the instrument driver.
Creating Instruments

To name the instrument, select `Create»Instrument`. Enter the name, prefix, and default qualifier of the instrument driver in the Create Instrument Node dialog box. The name of the instrument is the name that appears in the `Instrument` menu and `Instruments` folder after the instrument is loaded. The prefix of the instrument is prepended to each function to create the function name. The prefix is useful in your code to help determine which instrument driver each function call originated from. The default qualifier provides information to LabWindows/CVI about access options and conventions. LabWindows/CVI uses this qualifier for all functions in the instrument driver unless a function specifies a different qualifier. To enable the `Default Qualifier` field, select `Options»FP File Format` and select `CVI 5.5 and later` as the `Default Format for New Files`. 
Creating Classes

Your instrument driver is made up of different functions, which you can categorize into classes. To create a class, select **Create»Class**. Classes are useful in organizing your functions into logical groups.

**Note**  Function classes serve only to help present a hierarchical overview of your instrument driver to your users. LabWindows/CVI instrument driver classes are not the same as C++ classes or the object-oriented notions of classes. Placing functions in different classes does not affect their accessibility or functionality in any way.
Creating Function Trees

Select **Create»Function Panel Window** to create functions. Fill in the Create Function Panel Window Node dialog box. The **Name** field defines the name of the function panel window, which appears while you search through the instrument driver function tree. The **Function Name** field defines the actual name of the function that does a given operation. LabWindows/CVI precedes the function name with the instrument prefix when you paste the function in the source code. The **Function Name** field must follow ANSI C rules of naming a function. The following example shows the use of the prefix with the function name:

- Prefix: names
- Function Name: displayName
- Function Name in source code: names_displayName

The **Qualifier** field defines the default qualifier that provides information to LabWindows/CVI about access options and conventions.
Creating Function Panels

To create a function panel, double-click the name of a function defined in the Function Tree or highlight the function and select **Edit»Edit Function Panel Window** or press the **Enter** key. An empty function panel appears on which you can place input controls, slide controls, binary controls, ring controls, and numeric controls. The input controls are the input parameters of the function. You also can place output controls, which represent the output parameters of the function, on the function panel. In addition, you can specify a return value for the function.

To place input controls, output controls, and a return value on the function panel, select the object from the **Create** menu, right-click the panel and select a control from the context menu, or click one of the following icons on the toolbar of the function panel window.

- ![Icon](image1.png) Creates an input control
- ![Icon](image2.png) Creates a numeric control
- ![Icon](image3.png) Creates a slide control
- ![Icon](image4.png) Creates an output control
- ![Icon](image5.png) Creates a binary control
- ![Icon](image6.png) Creates a return value
- ![Icon](image7.png) Creates a ring control
- ![Icon](image8.png) Creates a global variable control
- ![Icon](image9.png) Creates a message control

Selecting a control opens a dialog box where you can fill in the parameters for each particular control.
Creating Function Panels (cont.)

When you create an input, output, or a return value, a dialog box appears requesting the following information:

- **Control Label**—The label of the control to place on the function panel.
- **Parameter Position**—The position of the parameter in the function call.
- **Data Type**—The data type of the parameter. To add user-defined data types to the list, select Options→Data Types.
- **Default Value**—The default value of the parameter.
- **Control Width**—The width of the control in pixels.
- **Display Format** (if output)—The format of numeric data that appears in the output control in the function panel.

Binary, slide, ring, or numeric controls require control-specific information that you define in the same way that you define control-specific information in the User Interface Editor.

You also can place global variable indicators on the function panel. These variables are updated as a result of an operation performed within the function. Select Create→Global Variable or click the Create Global icon.

You also can place text on the function panel. Select Create→Message or click the Create Message icon.
Documenting the Instrument Driver

You can document a function directly from the Function Panel Editor. To create a help window for the function, right-click an open area of the function panel and select Function Help from the context menu. In the Help Editor, you can enter help pertaining to the function.

You also can document the function parameters. Right-click the function parameter and select Control Help from the context menu. Use the Help Editor to enter the necessary help information for the parameter. LabWindows/CVI automatically adds the prototype of the function to the function help and adds the data type to the parameter help. You can view this additional information when the function panel is in operate mode.

If you enable Options»Help Style»New (Recommended) in the Function Tree Editor, you can use HTML tags in both the function panel help text and control help text you enter.

Another file type associated with an instrument driver is an ASCII text (`.doc`) file, which contains documentation for the instrument driver. LabWindows/CVI can create this file automatically, based on the help text you enter on the function panels. This document provides information such as the function tree layout, the data types associated with each function parameter, and the information you enter in the help windows of each function panel. In addition, LabWindows/CVI can automatically add general instrument driver information and can document the functions in Visual Basic or in C format. To generate a `.doc` file automatically, select Options»Generate Documentation in the Function Tree Editor. A dialog box appears, in which you can specify the type and format of the information to place in the document.
Generating the Header File and the Source Code Skeleton

LabWindows/CVI can automatically generate a header (.h) file containing the function prototypes of the functions defined in the function tree as well as a source code (.c) skeleton to simplify the coding of your functions. To generate these files automatically, select Tools»Generate New Source for Function Tree.

Note If you are not generating an IVI or VXIplug&play instrument driver, make sure that you do not enable the Options»IVI/VXIplug&play Style option. This option affects the skeleton code that LabWindows/CVI generates.
Header File and Source Code Skeleton

The figure above shows an example of the resulting source code skeleton and the header file. Observe that the functions contain a CVIFUNC define at the beginning of the function. This is the _stdcall calling convention defined in cvidef.h. For the source code to compile, you must add the #include <cvidef.h> line in the instrument driver header file.

You now can begin coding all the functions of your instrument driver by filling the blanks.
Editing an Instrument Driver

Right-click the Instruments folder, select Edit Instrument, and select the instrument driver you want to edit.

Editing Instrument Driver Files

When working with instrument drivers, right-click the Instruments folder in the Library Tree and select Edit Instrument to either invoke the Function Panel Editor or to modify the relationship between the function panel (.fp) file and its associated program file.

- **Show Info**—Displays the names of the current function panel file and the attached program file.
- **Attach and Edit Source**—Searches the directory that contains the function panel for a filename that has the same prefix as the function panel file and a .c extension and opens the .c file for editing.
- **Detach Program**—Detaches the program file from the function panel.
- **Reattach Program**—Attaches a program file to a function panel. The Reattach program option searches the directory that contains the function panel for a program file that has the same prefix as the function panel file and a .lib, .obj, .dll, or .c extension.
- **Edit Function Tree**—Invokes the Function Tree Editor.
Exercise 3-1

Time to complete: 30 min.

- To create and use an instrument driver
- To use the Average function as an instrument driver

OBJECTIVE
Lesson 3  Instrument Drivers

Exercise 3-1  Creating and Using an Instrument Driver

Objective:  Part A: To create and use an instrument driver.
Part B: To use the Average function as an instrument driver.

Part A

Create a simple instrument driver that finds the average of two numbers. Because this math function does not relate to instrument control, you can focus on the instrument driver creation process. In exercises in later lessons, you have the opportunity to examine a true instrument driver that controls an instrument in more detail.

LabWindows/CVI uses function panels to simplify calling complex functions. A function panel easily converts into a static or dynamic link library. Create your instrument driver by creating a function panel.

1. Launch LabWindows/CVI and create a new project.
2. Create a new function tree by selecting File→New→Function Tree (*.fp). A function tree is a collection of function panels. A function panel contains exactly one function.
3. Add an instrument to the function tree. Select Create→Instrument. Name the instrument My Math Functions and enter Math as the prefix. The prefix is placed at the beginning of all function names. Leave the Default Qualifier control blank.

![Create Instrument Node]

Note  To enable the Default Qualifier field, select Options→FP File Format and select CVI 5.5 and later as the Default Format for New Files.

4. To organize these function panels, you can use classes to group similar functions. For example, you can group all math analysis function panels under a class called Analysis or all arithmetic function panels under a class called Arithmetic. For the instrument driver in this exercise, create a class called “Helpful Math Functions” by selecting Create→Class.
5. Now that you created a class for your helpful math functions, you can create a function panel. Select Create»Function Panel Window. Create a function called Average. Leave the Qualifier field blank. To make your function simple to use, name the function panel the same name as the function, as shown in the following figure.

6. Place this function in the Helpful Math Functions class.
When you finish, your instrument driver structure should look similar to the one in the following figure.
7. Double-click Average function panel to open it.

8. You must create the inputs and outputs for your Average function. Create a function that takes the average of two integers and returns the result, which must be a double. You also must return the status of the function in case an error occurs during the calculation. You need two inputs (integers), one output (double), and a return status (integer). Create and place these items on your function panel. Use intuitive names for your items so that you can recall what they are in your code.
When you finish, your function panel should resemble the one in the following figure. Your parameter names might be different.

Notice how the function is built in the bottom of the window.

9. Before you proceed, document the parameters on this function panel. To create help for your user, right-click a parameter and select Control Help from the context menu. Enter your comments in the Help Editor. You also can add help for functions, function classes, and instruments by right-clicking those items. Your user can read the help by right-clicking the controls and panel.
10. Save the file as `MyMath.fp` in the directory `C:\Exercises\CVI Basics I\Lesson 3\MyMath.fp`.

11. Create the source code skeleton and header file for this function tree. LabWindows/CVI creates the `.c` and `.h` files automatically for you based on the parameters you created on the function panel. Because this driver is a software driver and does not use IVI, VISA, or VXIplug&play architecture, make sure that you disable the Options»IVI/VXIplug&play Style option. Then, select Tools»Generate New Source for Function Tree.


13. You now have created two new files: `MyMath.c` and `MyMath.h`. All required functions and parameters are inserted for you to complete the code to run the function panels. Do not modify `MyMath.h` but add code to `MyMath.c` to complete the `Average` function. Your code should resemble the following code.

```c
#include "MyMath.h"
int Math_Average (int a, int b, double *average)
{
    *average = (a + b) / 2.0;
    return 0;
}
```
14. After completing your code skeleton, save all the files and open a new project by selecting **File»New»Project (*.prj)**. Now that you have finished both the function tree and source code for your instrument driver, you must associate the two. Select **Instrument»Edit**. In the dialog box that appears, select your **My Math Functions** instrument driver in the list and select **Reattach Program**. LabWindows/CVI searches for a source or object module to associate with this function panel. When you get the message that **MyMath.c** is installed as the program file for your instrument driver, your driver is ready to use. Click **OK**. Click **Done** to close the Edit Instrument dialog box.

**Part B**

Before you test your work in a program, run the function panel interactively.

1. Load the instrument driver you created (**MyMath.fp**). It might already be loaded. You can look in the **Instruments** folder of the Library Tree or in the **Instrument** menu to check if the instrument driver is already loaded.

2. From the Library Tree, select **My Math Functions»Helpful Math Functions»Average** to open the function panel you created called **Average**.
3. Enter values for your inputs and run the function panel. You should get the correct average and a status of 0. If you did not get correct results, right-click My Math Functions in the Library Tree, select Edit Instrument, and go from there. You might need to return to step 13 of Part A.

After you have determined that the function panel is working correctly, use the Average function in a simple program.

5. Save the source file as `Averaging.c` in the directory
   `C:\Exercises\CVI Basics I\Lesson 3\Averaging.c`.

6. Add the `Averaging.c` to the project.

7. Using the Library Tree, open the `Average` function panel.

8. Fill out the function panel with variables and declare each one using `<Ctrl-D>`.

**Note**  Be sure that `Averaging.c` is set as the target file.
Insert the function by selecting **Code»Insert Function Call** after you declare the variables. When you finish, return to your source code. You have created a simple program.

9. Insert code to set values for the variables in your program.

10. After the Math_Average function call, insert a `printf` similar to the one shown in the following code.

```
printf ("The average of %d and %d is %f\n", num1, num2, result);
```

11. Insert another `printf` function similar to the one shown in the following code.

```
printf("Press the ENTER key to close the Standard I/O window\n");
getchar();
```
Your program should look similar to the code shown in the following figure.

```
#include "KvMath.h"
#include "utility.h"
#include <ctime.h>
#include <ansi_c.h>

static double result;
static int numx = 42;
static int numy = 13;

int main (int argc, char *argv[])
{
    if (InitCVI RTE (0, argv, 0) == 0)
        return 1;
    Math_Average (numx, numy, &result);
    printf ("The average of %d and %d is %lf \n", numx, numy, result);
    printf ("Press the ENTER key to close the Standard I/O window\n");
    getchar();
    return 0;
}
```

12. Run the project. Save the project as Averaging.prj in the directory C:\Exercises\CVI Basics I\Lesson 3\Averaging.prj. Click Yes when prompted to include the header files that are needed. You should get correct results using your instrument driver.

13. Press <Enter> key to close the Standard I/O window and terminate program execution.

14. Modify the values of the parameters passed into Math_Average in the main function to test the instrument driver with other values.
Challenge

If you have extra time, try to complete the following task.

Create a function in the MyMath instrument driver that contains a calculator function. Write the code to run this function panel. A possible solution function panel is shown in the following figure.

End of Exercise 3-1
Summary Lesson 3

- LabWindows/CVI instrument drivers consist of the following files: the instrument driver program (.c, .lib, .obj, or .dll), a header (.h) file, the instrument function panel (.fp) file, and an ASCII text (.doc) file.
- A function tree defines all the functions of the instrument driver. You can categorize the functions under different classes through the Function Tree Editor.
- A function panel is a graphical representation of a function. You can add inputs, outputs, return values, and so on.
- LabWindows/CVI automatically generates a header file, a source skeleton code, and a document file containing all the help text entered on the function panels.