ENGR 210/EEAP 240 Lab1
Computer-Based Instrumentation

Background

This lab introduces the basic ideas of computer-controlled instruments and graphical computer programming. Both areas require an entire book to thoroughly study the topics. However, the following short introduction to graphical programming will allow you to do simple programming of instrumentation using one of the most widely-used computer programs, LabVIEW®. Incidentally, LabVIEW® stands for Laboratory Virtual Instrument Engineering Workbench.

Graphical Computer Programming

Traditional computer programming involves setting down a list of tasks for the computer to execute in the given sequential order. Each instruction is executed in the order of appearance in the list. Often, the availability of data determines the order given to these instructions. For example, instruction 3 in Figure 1-1 requires data calculated in instruction 2. Therefore, instruction 2 must execute before instruction 3 and it has a *data dependency* on instruction 2. Note that instruction 2 itself also has a data dependency on instruction 1. Because instruction 4 does not require a result from any other instruction in the sequence, it has no data dependencies on instructions 1, 2, or 3. Summarizing, instructions 2 and 3 are data dependent and must be executed in sequence; however, instruction 4 is data independent and it does not matter when it executes.

1. Add A to B
2. Add C to Sum of A and B
3. Divide Sum of A, B, and C by 3
4. Subtract A from B

Figure 1-1. A Sequence of Instructions

This discussion of data dependency leads to a new way of programming. If you specify the operations and the data dependencies, the computer can execute the instructions in any order that protects the data dependencies. Now you need a way of easily specifying data dependencies. So, if you can draw a block for each operation and connect the blocks to show the dependencies, you can program the computer by drawing a picture. For most people, pictures are much easier to understand than a list of instructions.

LabVIEW® programming consists of drawing pictures that specify data dependencies. The LabVIEW® programming environment includes a large set of blocks to specify operations and a Wiring tool to connect them together. As an example, Figure 1-2 below shows the operation of multiplying two numbers and displaying the result.

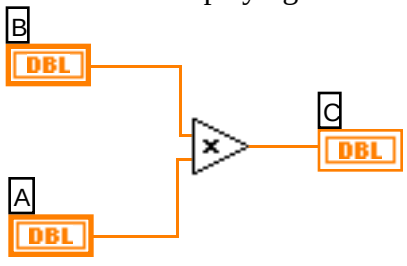


Figure 1-2. Program for Multiplying Two Numbers and Displaying the Result

A LabVIEW® program, called a virtual instrument (VI), is a two-window system. The code is in one window and the user interface (inputs and outputs) appear in a separate window. The program window is the diagram window, and the user inputs and outputs are in the front panel window. Figure 1-2 shows a sample program that would appear in the diagram window. The numbers are entered into the computer and displayed in the panel window shown in Figure 1-3. The two boxes on the left (labeled **A** and **B**) are controls, and the box on the right (labeled **C**) is the output or indicator. (The **X** and **=** are only displays showing the operation of the VI and not inputs or outputs.) The three boxes are associated with like labeled boxes in the diagram window shown in Figure 1-2.

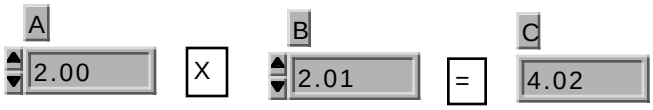


Figure 1-3. Front Panel for a Two-Number Multiplication Program

Figure 1-4 shows a more complicated program. This program reads a voltage using a Analog/Digital (abbreviated A/D) Converter card installed in the PC and plots the number corresponding to the voltage measurement on a scrolling chart. The gray box around the program is a While Loop. The program elements inside the While Loop will execute repeatedly as long as the While Loop control is true. That is, as long as the variable **stop** is false. **Stop** is the button on the front panel shown in Figure 1-5. When the user presses the button, **stop** becomes true, and the value which feeds into the While Loop control is inverted (changed to false). When the While Loop in this example stops, there are no other program elements to execute, and the program stops running.

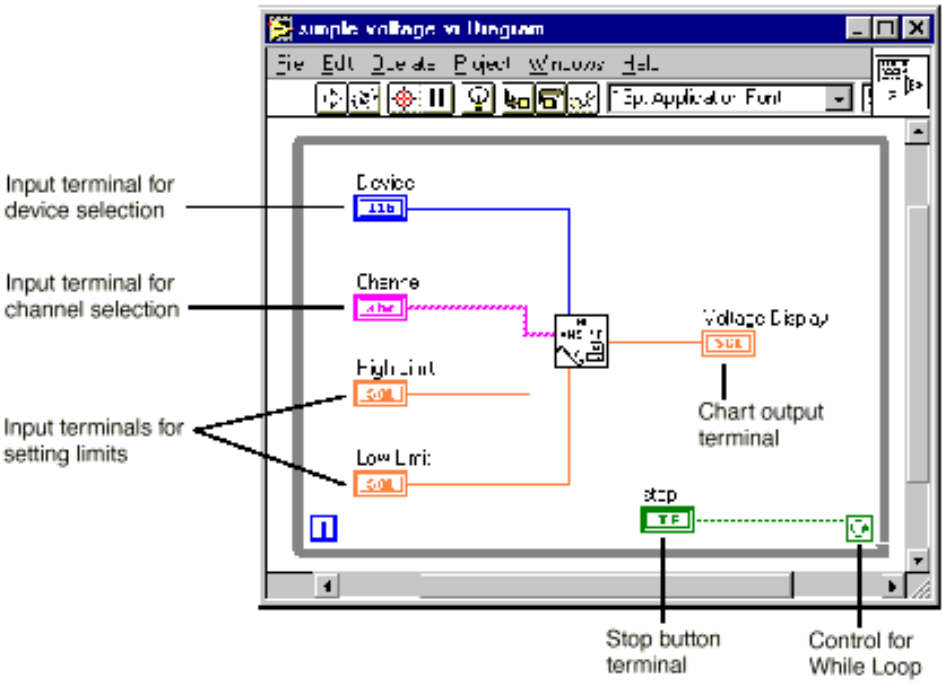


Figure 1-4. LabVIEW® Program to Read a Voltage from a Single Channel and Display

In the center of the While Loop, you see the "work" being done in the loop. The block labeled **AI ONE PT** performs the operation of getting a voltage from the channel specified by **Channel** and the device specified by **Device**. NOTE: An A/D converter card will have several channels, typically 8-16. **AI ONE PT** reads one voltage reading from the specified channel on the specified device. The output of **AI ONE PT** is a list of voltages

in a structure called an array. The terminal labeled **Voltage Display** is the connection point for the chart in the front panel shown in Figure 1-5. The chart is updated every time a new number is input to it — the new number is graphed along with all the previous numbers in a scrolling manner. The programmer can specify a limit on the number of points that can be graphed at one time.

Now that you have examined two simple programs, you will look at how to write the program. That is, what are the operations performed (mouse clicks and keyboard operations) to write a program?

The **AI ONE PT** block in Figure 1-4 performs the DAQ (data acquisition) procedure. It is not the only piece of programming needed to read a voltage level. National Instruments has completed a major portion of the programming by writing software that drives the DAQ system. That means that all you need to do is include the **AI ONE PT** block in the program. In later labs you will take a look at what other work the computer does in the process of getting the measurement.

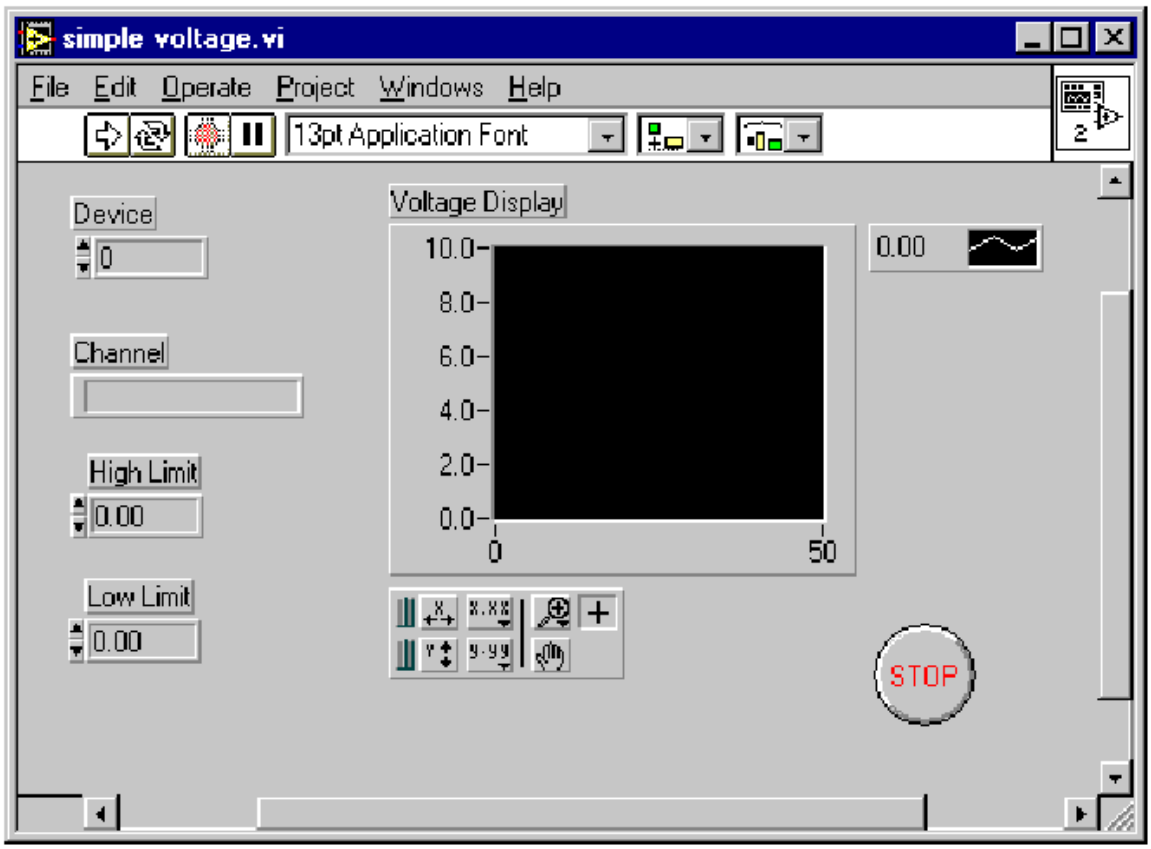


Figure 1-5. Front Panel for Program that Reads & Displays a Voltage Waveform

Outside the computer, the voltage to measure is at some unknown level. A real number represents that level. Remember that a real number can be any number between positive and negative infinity and can have an infinite number of digits. The PCI-1200 NI-DAQ data acquisition boards we have in the EECS Circuits Lab are set to digitize voltages which lie in the range 0 to +10 volts. The input analog value can be any value in between these limits. Once the value is inside the computer, you represent that value with a digital number with some limited number of digits. In a coming weeks, you will study this area of

resolution (# of digits) further. For now, you will concentrate on the software side of DAQ.

This lab is basically a tutorial. In later labs you will write your own programs and use more complex LabVIEW® programs. Read through the lab procedure and perform the lab exercise indicated below.

References

1. John K. Eaton and Laura Eaton, LabTutor: A Friendly Guide to Computer Interfacing and LabVIEW® Programming, Oxford University Press, 1995, ISBN 0-19-510044-1, $54.95.
2. Gary W. Johnson, LabVIEW® Graphical Programming : Practical Applications in Instrumentation and Control , McGraw-Hill, 1997, ISBN 0-07-032915-X, $50.00.
3. Lisa K. Wells and Jeffrey Travis, LabVIEW® for Everyone : Graphical Programming Made Even Easier, Prentice Hall, 1996, ISBN 0-11-268194-3, $55.00.
4. *LabVIEW® Tutorial*, National Instruments, Inc., Austin, TX.

NOTE: References 1-3 have been ordered and will be placed on reserve in the Kelvin Smith Library for ENGR210. Reference 4 is a CD-ROM which we are trying to make available in the EECS Circuits Lab.

Lab Procedure

For this lab exercise, you will need a PC with LabVIEW® software. You can either do this in the lab or from any computer with access to the CWRUnet software library. However, there WILL be a checkout of your program by the TAs in the lab.

A LabVIEW® Tutorial:
Displaying Scaled Random Numbers on a Chart

The following tutorial will help you learn the basics of LabVIEW® programming. Read each step completely before executing the step. By the end of the tutorial, you will have written a VI that displays scaled random numbers on a chart. Be sure that everyone in the group gets a chance at the computer during the tutorial.

Before starting, it is very important to know that LabVIEW® 4.1 does not have an undo feature. Later versions of LabVIEW® will have this feature. However, to the best of our knowledge you can do this lab in LabVIEW® 4 or 5. In either case, you should save early and often. You can easily revert to the previously saved version, if necessary. Also, if the system should stop responding due to some problem caused by an unknown source, you can get back to a known state very easily.

1. Setup:

a. Start the LabVIEW® application from the **LabVIEW®** group in the **Start** menu of the task bar.
b. When prompted to open an existing or new VI, select **New VI**.
c. When the new VI windows appear, select **Show Tools Palette** from the **Windows** pull down menu in the LabVIEW® window to display the **Tools** palette. Note that this palette will often come up automatically.
d. From the **Tools** palette, select the Positioning tool,  . (Table 1-1 describes several commands and tools, including the Positioning tool.)
e. From the **File** menu, select **Save As** and save the file to your floppy disk (drive a:) or hard under a suitable name. If you are working in the circuits lab we recommend that you save your file to a floppy which you can take with you since we cannot guarantee that it will remain on the hard drive. When you save the file it must be saved with the file extension .vi. It is a good idea to save the file every few minutes during the development process. Save the file after making a change you want to keep.
f. Review the commands and tools in Table 1-1.

Command/Tool	**Purpose**	**Used When**	**Picture**
Delete key	Deletes selected objects	There are unwanted objects in the program	<Delete>
Ctrl-S	Saves files	You want to save your Changes	<Ctrl-S>
Positioning tool	Moves and selects objects	You need to move or delete program elements or insert new ones	
Wiring tool	Connects objects together	Program elements must be connected to allow data to flow between them	
Ctrl-B	Removes all broken wires	There are several unwanted wires in the program; use with caution	<Ctrl-B>
Operating tool	Changes values	You need to change a value in a front panel object	
Text tool	Edits text	You need to change a label or a comment	

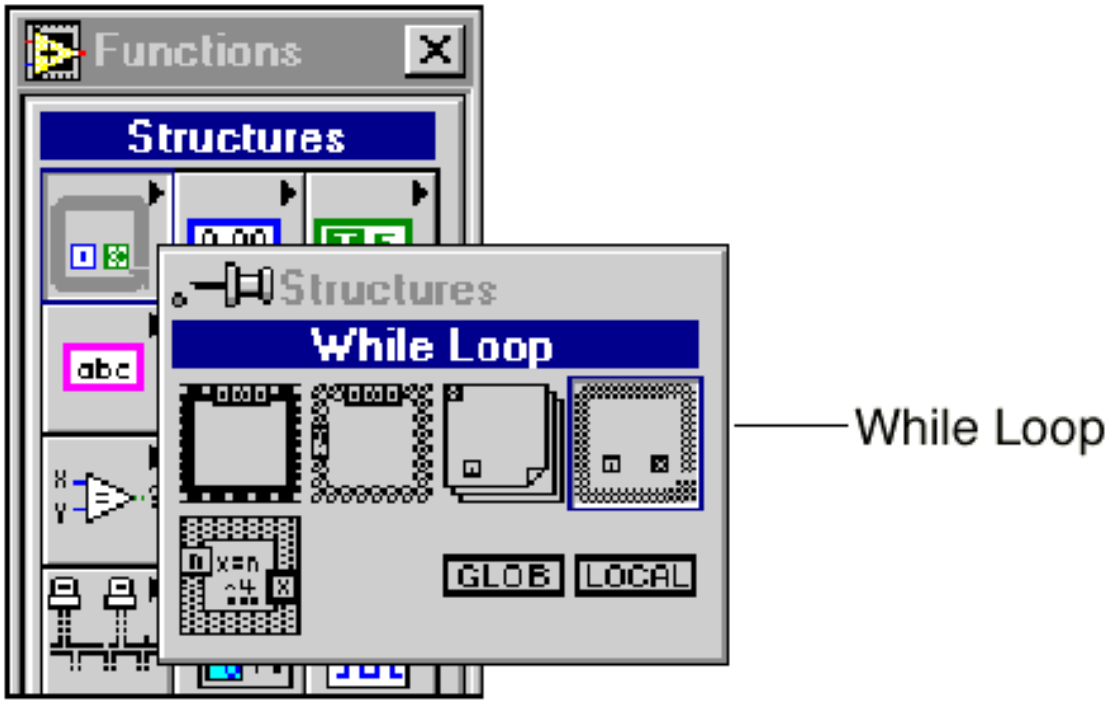
Table 1-1. Table 1: Some Useful Commands and Tools

2. Virtual instrument programming:

a. Click on the diagram window (the window with the white background) to bring it to the front.
b. Insert a While Loop into the diagram window. First, open the **Functions** palette by clicking the right mouse button with the cursor in the diagram window. Then move the pointer down to the **Structures** palette button (upper left button). When

the cursor reaches the button, a palette of program elements will appear. Click on the While Loop (icon on the far right in the top row).

c. The While Loop first appears in the diagram window as a box-shaped cursor. Insert the loop by placing the cursor in the upper left corner of the diagram window and clicking and dragging the icon to the lower right corner. Make the While Loop almost as large as the window, but don't fill the entire window.



d. Insert the **Random Number Generator** function into the While Loop. Click the right mouse button as before. This time, select the **Numeric** button by moving the cursor to the **Numeric** palette button (the center button in the top row) in the function window. Choose the icon that looks like a pair of dice to add the **Random Number Generator** function to your diagram.

e. Press <Ctrl-H> to open the Help window. Move the cursor to the pair of dice and click once. Read the information in the Help window. This help feature can be useful when determining what connections need to be made to a VI.

f. Click on the Panel window (the window with the grey background).

g. Insert a Waveform Chart. Right-click in the Panel window to bring up the **Controls** palette. Click on the **Graph** button (the right button in the second row) in the **Controls** palette. Choose **Waveform Chart** from the palette, move the cursor back to the Panel window, and click to insert the chart wherever you want.

h. Name the chart "Scaled Data" by typing the name and clicking on the Enter icon of the tool bar. You should see the text appear in a box near the upper left corner of the chart. If not, try pointing the cursor at the chart and clicking the right mouse button. In the menu that appears, select **Show » Label** from the submenu and type the title.

i. Point at the chart and click and hold the right mouse button. Select **Show » Digital Display** from the submenu.

j. Point the cursor at the chart and click and hold the right mouse button. Select **Find Terminal** in the pop-up menu and release the button. This should bring up the Diagram window, and the terminal for the chart will be highlighted with dashed lines.

k. Connect the Random Number Generator to the chart terminal. Select the Wiring tool from the **Tools** palette. (It looks like a spool of thread.) Use the Wiring tool to connect the output of the dice to the terminal for the Scaled Data chart by pointing the tool at the dice and clicking once. Move the tool to the indicator terminal (a small rectangle with DBL inside) and click once more. An orange line should appear. NOTE: Wiring can often give you problems. You can remove a wire by selecting it and using the CLEAR key. Also, you will occasionally find that your wires do not make a connection. This can be checked by using <ctrl>-B to check for broken (open) wiring connections. If you do have a broken connection you will not be able to run your VI.

l. Click on the Panel window. Now you can test your VI. Click on the Run button, the single arrow in the upper left corner. Run the VI several times and record the values of the random number after each run. Does the VI run? How do you know? How many times does the loop execute each time you run the program? Answer these questions on your data sheet.

3. You may want to use the Positioning tool to rearrange some of the icons to make the program more clear. In general, it is best to place input terminals on the left and output terminals on the right. Also, the wires in between should not cross unless absolutely necessary.

4. Save the VI to disk. (You do not need to rename the file.)

5. To control the While Loop:

a. Click on the Panel window. You will now insert a stop button to control the While Loop execution. You can find the button palette by choosing the Boolean controls from the **Controls** palette. Choose the "STOP" button, other controls may not behave the way we want. Click the Enter button on the Toolbar.

b. When you have the button in place, point at it with the Positioning tool. Click and hold on the right mouse button and select Find Terminal to bring up the Diagram window.

c. Insert a NOT element from the **Boolean** palette.

d. Click on the Diagram window.

e. Use the Wiring tool to connect the STOP terminal (small rectangle with TF written inside) to the input (left side) of the NOT element. Then connect the output (right side) of the NOT element to the conditional terminal that controls the While Loop. The conditional terminal  should be in the lower right corner of the While Loop.

6. Save the VI to your disk.

7. In the next few steps, you will add parameters to your program to provide a scaled random number between user-defined values of X and Y.

a. Go to the Panel window and insert two digital controls (under the **Controls » Numeric** palette). Name one of the controls “upper limit” and the other “lower limit.”

b. Use the Positioning tool to arrange the controls to make your front panel look presentable.

c. Switch to the Diagram window. Make sure the terminals for the new controls are inside the While Loop.

8. Before continuing, you need some background. Math operations are programmed using triangular icons. All operations used here are binary, meaning they have two inputs. For example, the subtract icon in Figure 1-6 has two inputs on the left and one output on the right. To subtract Y from X (as shown below), you connect **X** to the upper left corner and **Y** to the lower left corner. The difference (answer) is then available at the right corner.

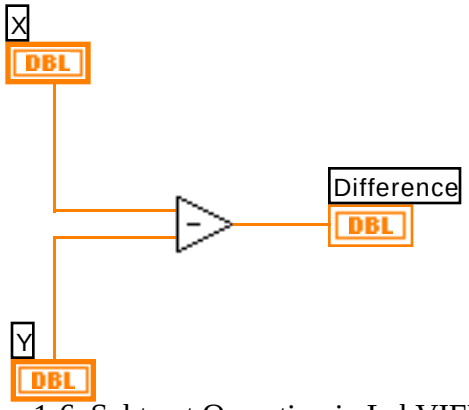


Figure 1-6. Subtract Operation in LabVIEW®

9. You will have to remove the orange wire between the random number generator and the chart terminal before implementing the following function. Insert a **Subtract**, a **Multiply**, and an **Add** function from the **Numeric** palette. Use the Wiring tool to connect these three arithmetic functions together with the **Random Number Generator** function and the chart terminal to provide the following function.

$$\text{Output} = [\text{Random} \times (\text{Upper} - \text{Lower})] + \text{Lower}$$

The operators you need to implement this equation are shown below:

10. Save the VI to your disk. Run the VI with ten different sets of values of Upper Limit and Lower Limit. (Use the Operating tool (it looks like a pointing finger) to change the values of the upper and lower limits. You can point to the arrows next to the digital control box or click on the number in the box to add values from the keyboard.) Record the upper and lower limits and the output values you get from using your VI in Data Table 1. You should read the output value from the digital display rather than trying to estimate from the chart.
11. Save your VI to a floppy disk. You will be e-mailed instructions about the checkout of your VI.

DATA AND REPORT SHEET FOR LAB 1

Student Name (Print): _____ Student ID: _____

Student Signature: _____ Date: _____

Student Name (Print): _____ Student ID: _____

Student Signature: _____ Date: _____

Student Name (Print): _____ Student ID: _____

Student Signature: _____ Date: _____

Lab Group: _____

Data Table 1. Results of Random Number Generator

Run #	Low Limit	High Limit	Output
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			

Program checked by _____

Questions

1. Does the VI run? How do you know?
2. How many times does the loop execute each time you run the program?
3. Did you get any output values outside the limits? Explain your answer.