

# ENGR 210/EEAP 240

## Lab3: Computer-Based Instrumentation

### Background

This lab involves two very broad areas of study in instrumentation: analog-to-digital conversion 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 use a computer to take simple measurements.

### 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 3-1 requires data calculated in instruction 2. Therefore, instruction 2 must execute before instruction 3. Therefore, instruction 3 has a *data dependency* on instruction 2. Note that instruction 2 has a data dependency on instruction 1. Because instruction 4 does not require the result from any other instruction in the sequence, it has no data dependencies on instructions 1, 2, or 3. Therefore, instructions 3 and 4 are data independent. Because instruction 4 is data independent with all the other instructions, 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 C

Figure 3-1. 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 3-2 below shows the operation of multiplying two numbers and displaying the result.

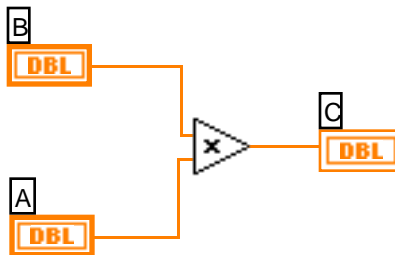


Figure 3-2. Program for Multiplying Two Numbers and Displaying the Result

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 3-2 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 3-3. 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 3-2.



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

Figure 3-4 shows a more complicated program. This program reads a voltage and adds it to a 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 3-5. 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.

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**. **AI ONE PT** reads one voltage reading from each of the specified channels 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 3-5. The update of the chart is such that every time a new number is input to it, the new number is graphed along with all the previous numbers. The programmer can specify a limit on the number of points that can be graphed at one time.

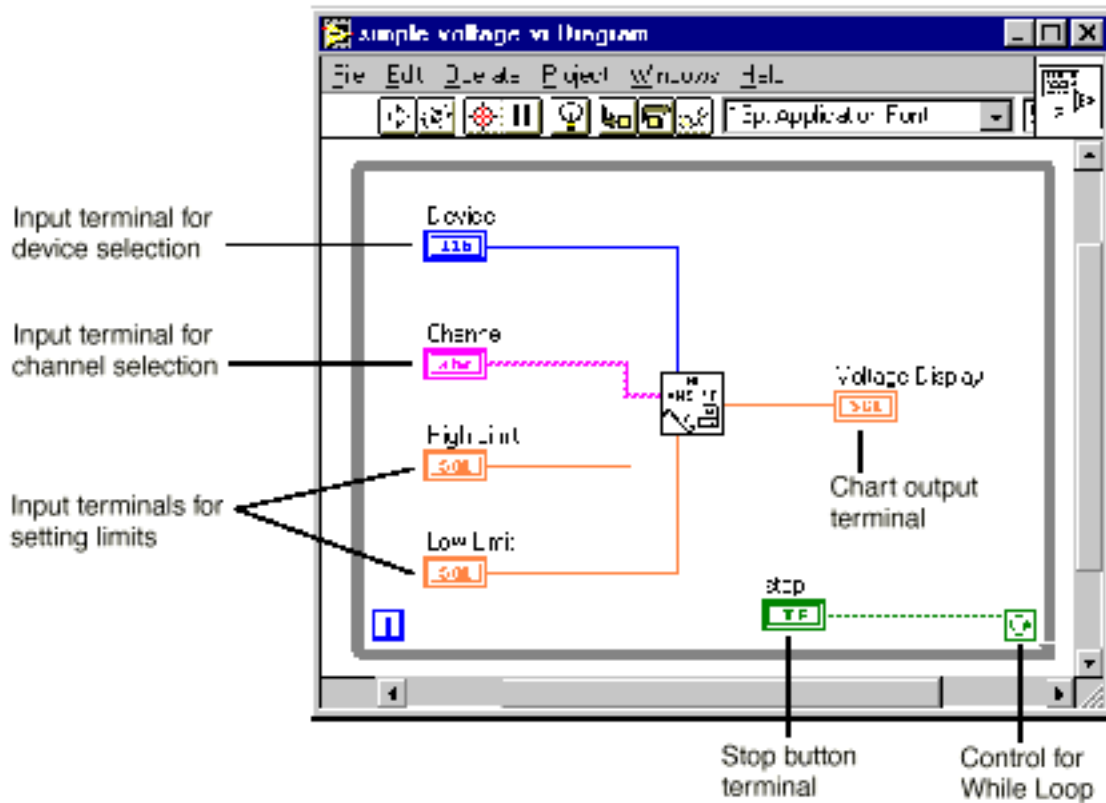


Figure 3-4. LabVIEW Program to Read a Voltage from a Single Channel and Display

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?

Below is a tutorial. Read through the lab procedure and come to lab prepared to execute the lab exercise. The **AI ONE PT** block in Figure 3-4 performs the DAQ 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. Next, you will take a look at what other work the computer does in the process of getting the measurement.

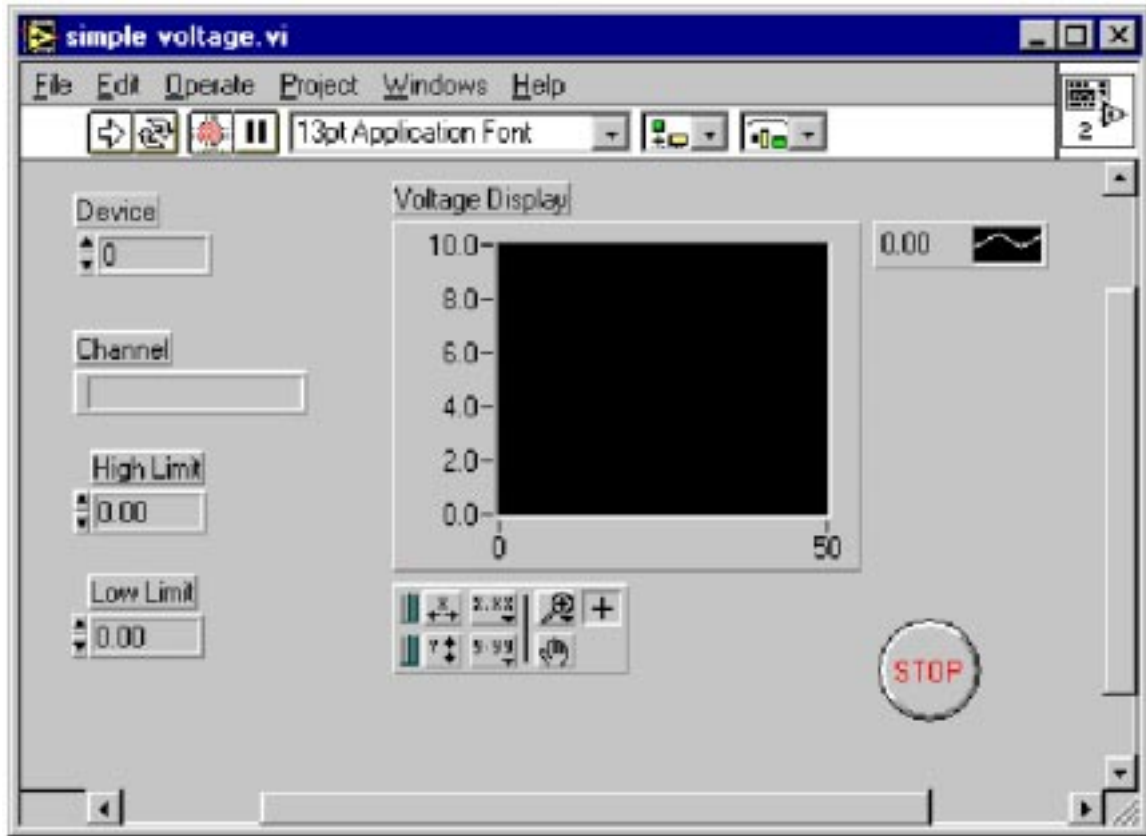


Figure 3-5. Front Panel of a Program that Reads and Displays a Voltage Waveform

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 are set to digitize voltages which lie in the range of 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 and how to acquire data.

## References

Gary W. Johnson (1996), *LabVIEW Graphical Programming*, McGraw-Hill, Inc.  
 Lisa K. Wells and Jeffrey Travis (1997), *LabVIEW for Everyone: Graphical Programming Made Even Easier*, Prentice-Hall PTR, Upper Saddle River, NJ.  
*LabVIEW Tutorial*, National Instruments, Inc., Austin, TX.

## Lab Procedure

### Introduction

In this lab, you will write a LabVIEW virtual instrument (VI) program to read voltages and display them. The idea is to build an instrument that works like a graphing digital

multimeter. Your multimeter must be able to set the range of values for the readings (resolution) and display the reading on a digital readout and a chart.

Your lab report will consist of a floppy disk with your LabVIEW VI and a text file called `README.TXT`. The text file will explain what your VI does and describe the various files on the disk (there should be only two files).

## Equipment Needed

For this lab exercise, you will need:

A workstation with LabVIEW software. You will have to do this lab in the lab because only the lab computers have the data acquisition cards in them.

A blank formatted (virus-free) 3.5-inch floppy disk (high or low density). **THE FLOPPY IS YOUR RESPONSIBILITY**

Some voltage sources (batteries are a good source). **THESE WILL BE PROVIDED IN THE LAB.**

## Lab Work, Part 1

### 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. After completing this tutorial, you should be able to do the requirements for the lab. 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. 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:

Insert your *blank* floppy disk into the disk drive. Check the disk directory to make sure your disk is formatted and is free of any viruses. A simple way to achieve this is to format the disk, but you will lose any information stored on the disk.

Start the LabVIEW application from the **LabVIEW** group in the **Start** menu of the task bar.

When prompted to open an existing or new VI, select **New VI**.

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.

From the **Tools** palette, select the Positioning tool, . (Table 3-1 describes several commands and tools, including the Positioning tool.)

From the **File** menu, select **Save As** and save the file to your floppy disk (drive a:) under a suitable name. The file extension must be `*.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.

Review the commands and tools in Table 3-1.

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	

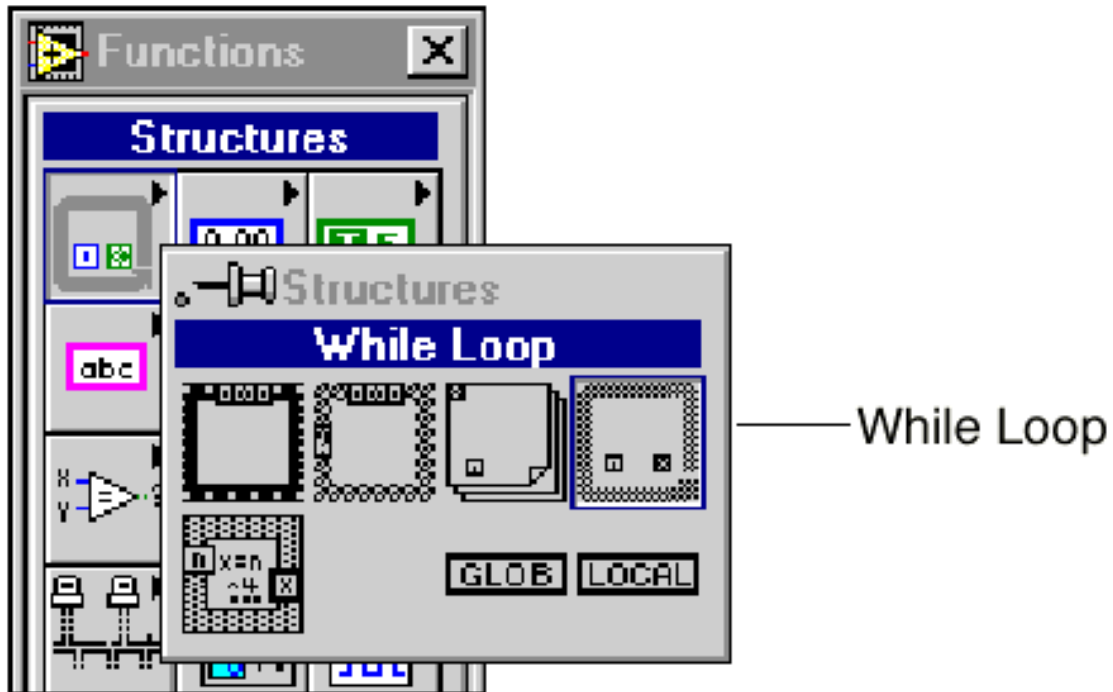
Table 3-1. Table 1: Some Useful Commands and Tools

## 2. Virtual instrument programming:

Click on the diagram window (the window with the white background) to bring it to the front.

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).

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.



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). Choose the icon that looks like a pair of dice to add the **Random Number Generator** function to your diagram.

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.

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

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.

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.

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

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.

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.

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 your floppy disk. (You do not need to rename the file.)

5. To control the While Loop:

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. You may select any button, as long as it is a button and not an LED, light, or switch. Type "STOP" as a label for the button and click the Enter button on the Toolbar.

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.

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

Click on the Diagram window.

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 floppy 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.

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."

Use the Positioning tool to arrange the controls to make your front panel look presentable. 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 3-6 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.



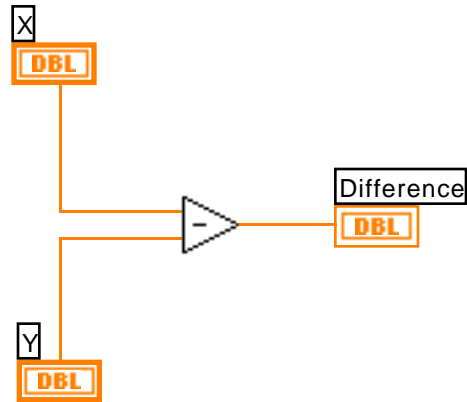


Figure 3-6. Subtract Operation in LabVIEW

9. 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 floppy disk. Run the VI with five different sets of values of Upper Limit and Lower Limit. (Use the Operating tool to change the values.) Record a few of the values you get. Did you get any values outside the limits? Write the numbers and your answer on your data sheet.

## Lab Work, Part 2

You will program a graphing digital voltmeter. This part of the lab requires you to do some experimenting and use what you have learned from Part 1. Below are some steps to guide you through this part of the lab.

### Building a Voltmeter with LabVIEW

Modify your program from Part 1 as shown below. First, save the program under a different and still meaningful name, such as **Graphing Digital Voltmeter.vi**.

The VI used to take voltage measurements is **AI Sample Channel** (located in **Data Acquisition » Analog Input » AI Sample Channel**). Remember that you find this menu by pointing the cursor where you want the icon to appear and clicking the right mouse button.

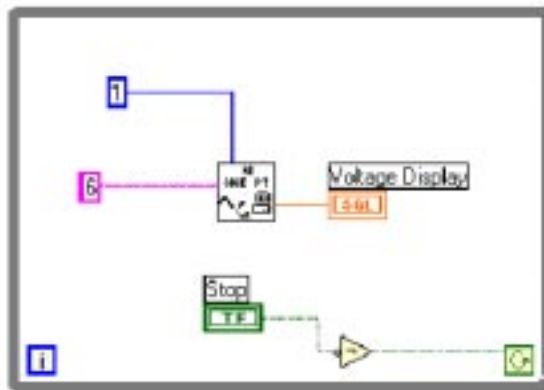
Open the Help window if it is not already open.

Use a digital control of type integer to set the device number for **AI Sample Channel**.

This is as opposed to the double precision control you used in part 1. Use a string control to set the channel number for **AI Sample Channel**. String controls work the same as digital controls, except they use a different data type. HINT: You can change the data type from DBL to things like integer or string by selecting the control in the diagram window and right clicking. In the resulting popup menu select **Representation** and then select the data type you want.

Remember to save your VI after every few changes, so you can return to a known state if necessary.

Don't forget to complete the rest of the lab procedure and take readings of each battery you were given.



(a) Diagram View



(b) Front Panel View

Figure 3-7. Sample VI that Reads Channel 6 from Device 1

### Lab Measurements

Set the device number and channel number on your panel to the values appropriate for the data acquisition board you are using. This will be posted on one of the boards in the lab.

A set of small alligator connectors is preconnected to the terminal block of the data acquisition board for you. A detailed chart of these screw connections is being posted on the course Web page for your reference.

Have your TA check your system and program.

Start your VI.

Measure and record the voltages of six different batteries. Be sure to record the labels of the batteries you actually measure — these are on the sides of the batteries.

## Extra Credit Items

There are a number of LabView Tutorials and References on the Web. Find these and write a short review of each tutorial.

Make your VI read and display the voltage of more than one channel.

Make your VI read several samples at one time but using a subVI designed to acquire a waveform. (This one is tough and requires that you figure out what a SubVI is.)

## Lab Report

This lab report will be considered an informal report. Prepare the report in the form of a plain text file called README.TXT which contains the following:

An introduction to your project similar to that of any other informal report. State what you have done and the objectives of the work.

An itemized description of all files on your floppy disk. That means the filenames, where they are located, and what function they perform.

A description of what your program does. If you have done any extra credit work, be sure to describe it in detail.

At least three uses for your voltmeter program.

Conclusions you have drawn about using LabVIEW to perform measurements. You should have some sort of idea about how easy or difficult LabVIEW is to use. Just give your personal thoughts.

Answer the following questions:

Run the random number tutorial VI several times and record the values of the random number after each run.

How do you know actually runs?

How many times does the loop execute each time you run the VI?

Do you get any random number values outside the limits?

7. (DATA SHEET) Enter the battery voltage readings you took during lab in the following format:

Voltage Source (label on battery)	Reading

You will be expected to hand in:

a floppy disk containing your VIs and a plain text file named README.TXT. **BE SURE TO PUT YOUR NAME ON YOUR DISK.**

A printout of README.TXT.

You should have started with a clean disk, so the only files on the disk are those requested in this lab manual without any potential viruses.