ECE 202 – LAB 2
VIRTUAL INSTRUMENTATION

BEFORE YOU BEGIN

EXPECTED KNOWLEDGE

Familiarity with programming at the level covered in CS 161 and EAS 102.

PREREQUISITE LABS

ECE 202 Lab 1: Introduction to LabVIEW.

EQUIPMENT

TX1 Digital Multimeter.
Arbitrary Waveform Generator.

MATERIALS

One resistor (any value).
Screen Capture Program such as WinGrab

PRELAB

None.

INTRODUCTION

LabVIEW is a graphical programming environment that combines graphical development with the flexibility of a programming language. It offers an environment that is tightly integrated with measurement hardware for engineers and scientists to produce solutions for data acquisition, data analysis, and data presentation. One of the key strengths of a package like LabVIEW is that it enables engineers to control test equipment remotely through software.

OBJECTIVES

In the first lab we created a virtual instrument that had two threads running simultaneously. In this lab we will design a virtual instrument to control a Tektronix TX1 digital multimeter and implement this virtual instrument as a subVI (similar to a C/C++ function). We will then redesign our virtual instrument such that the subVI is executed only if a change is made to the user input controls. We are using the program we wrote in the first lab as a base for this experiment to show that we can create a program to communicate with a lab instrument that simultaneously accomplishes other tasks (in this case counting loops). These tasks are not related and counting loops is merely a way of simulating a thread which must execute on a regular basis and shouldn’t stall. At the end of this lab we should have a program which allows us to communicate back and forth with the DMM while the loop count smoothly increments.
BUILDING A SUBVI FOR CONTROLLING THE TX1

A subVI (or subVirtual Instrument) is similar to a C/C++ function where we essentially have a box with inputs and outputs available for our use. If we built it, we know exactly what it does. However, many times we will use subVIs that we didn’t build. With some of those we can see the Diagram Panel, but with others we simply know what they do and how to connect them.

National Instruments provides a website (http://www.ni.com/labview/sample_over.htm) with instrument drivers for many instruments common in electrical engineering environments. We will use the driver available through this site to build a small VI that will enable us to control a TX1 multimeter. Specifically, our VI will be able to read the DMM’s battery level and to read an AC voltage currently being measured by the meter. We will then convert our VI into a subVI so that it can import and export data like a function. Finally we will implement our subVI into the program we wrote in Lab 1.

1. Open LabVIEW and select create a new VI.

2. From the functions palette on the diagram panel, select I/O>Instrument drivers. Since we will need this menu for a few items, click on the upper left corner “tack” icon. This allows the menu to be conveniently displayed until you are finished with it.

With instrument communication, we need to start (or initialize) a session with the instrument. The major objective in the initialization is to tell the program what instrument we are communicating with. We will use a subVI to accomplish this task. As you will see, the subVI has inputs and outputs. One of the inputs tells the computer what port we are going to be communicating through. This data is passed out of the subVI from a connection called the “instrument handle.” From this point on, when we want to communicate with that instrument, we need only use the handle. When we are done communicating with the instrument, we close the session. This is done mostly as “house keeping.” If our system was very large and we communicated with many instruments, having open “non-active” sessions could slow down our performance.

3. Select Tektronix TX1 and TX3 DMMs->tktxx Initialize. Place the icon on the diagram panel.

4. From the Instrument Drivers menu, select Tektronix TX1 and TX3 DMMs->Application Functions->tktxx Single Point Measurement and place it on the diagram panel.

5. From the functions palette, select Structures->Case and place it around the two subVIs in your diagram panel as shown in Figure 1. The case structure is placed on the Diagram Panel in the same manner as the while loop (see Lab 1).

The case structure functions two ways in LabVIEW. The first way is as a true or false structure. If you connect a wire from a boolean control to the question mark on the border, the top label will indicate true or false (press the arrow to the left or right to see the other case). If you wire a digital control to the question mark the top label now indicates “0, default”. This is the same as
using the case structure in C or C++. You now can program in a large number of cases. We will see in the following sections how to implement this functionality in LabVIEW.

Note: You can use the align objects function to the right of the font indicator in the toolbar to align the subVIs. Simply select all objects to be aligned with the arrow tool, then click on the align button to see the available options.

In the next sections you will be wiring the subVIs together. You can accomplish this by selecting the wiring tool and moving it over the subVI of choice. As with the wait function we saw in the first lab, when the wiring tool passes over the subVI, the selected terminal will flash and the name will momentarily appear.


7. Wire the Initialize->Error Out terminal to the Single Point Measurement->Error In terminal.

The error output from the initialize function can be wired through the instrument driver functions. This allows us to propagate the error through for an entire session. At the end of the instrument driver functions we will pass the error data to an indicator on the Front Panel.

8. From the instrument driver subpalette, select Tektronix TX1 and TX3 DMMs->tktxx Close.vi and place it on the diagram panel.

10. Wire the Single Point Measurement-> Error Out terminal to the Close->Error In terminal as shown in Figure 2.

![Figure 2.](image)

11. Right click on the Initialize->Resource Name terminal and select create constant. With the lettering tool enter the value “ASRL1::INSTR”.

The string we are entering is a VISA instrument handle, and it indicates where the instrument can be found on the system. In this case we are telling LabVIEW that we are working with an instrument that is connected to Com Port 0 (Serial Port).

12. Right click on the Initialize->ID Query terminal and select Create Constant. The value that appears defaults to False.

13. Wire this False icon to the Initialize->Reset Device terminal as well. With the Hand Tool, change the False icon to True. The results are shown in Figure 3.
14. With the Wiring Tool, right-click on Single Point Measurement->Reading and select Create Indicator.

15. With the Wiring Tool, right-click on Close->Error Out and select Create Indicator.

By doing steps 13 and 14 you have created indicators on the front panel that are receiving the output from these terminals when the program is run. In this case we want to see the AC voltage reading and all possible errors.

We have placed a case structure around these instrument control VIs. As mentioned earlier, this doesn’t have to be a true or false case. The following tasks will change the true/false statement into a case statement.

16. On the Front Panel, select the Controls Palette->List and Ring and Enum->Menu Ring and place it on the Front Panel.

![Figure 3.](image)

The menu ring functions as a “pop-up” digital control and allows us to enter words as values instead of just being able to enter numbers. Imagine having to remember that 0 means to read the AC value and 1 means to read the battery level. This may be adequate for a very small program that only you are going to use. The menu ring allows us to enter text for each number. For example, instead of 0, we enter Read AC Value into the first entry location. We then enter
Read Battery Level into the second entry location instead of 1. While programming we may get lost as to the meaning of each value. Consequently, LabVIEW provides a digital value display with the menu ring control. The current value is for the text shown. This will become apparent below. When we wire this control into the case structure, the text that corresponds to value 0 will select case 0 and the text that corresponds to value 1 will select case 1. Many more cases can be added.

17. Label the control “Select Action”.

18. Click the Lettering Tool inside of the box and enter “Read AC Value.”

19. Right-click on the control and select Add Item After.

20. Click the Lettering Tool inside of the box and enter “Read Battery Level.”

If we now select the Hand Tool and click on the Select Action control, we will see that the control currently has two options. Many more options could be added in this manner.

21. Right-click on the control and select Visible Items->Digital Display as shown in Figure 4.

We now see a number to the right of the display that indicates the digital value of the selection. Notice that our first entry has the value 0 and the second entry has the value 1.

Figure 4.
22. On the Diagram Panel, move the Select Action icon outside of the case statement and wire the icon to the case statement question mark.

When this is done, you will notice that indicator at the top of the case statement no longer says false. Now the indicator displays “0,Default”. This case corresponds to our Read AC Voltage option of the Select Action control on the Front Panel.

23. Click on the arrow to the right of the “0,Default” indicator to show the frame for case 1.

Note: by default there are two cases provided. If additional cases are desired they must be added by right clicking on the label at the top of the structure and selecting Add Case.

24. From the Instrument Driver palette, select Tektronix TX1 and TX3 DMMs->Initialize and place it on the diagram panel inside of case statement 1.

25. From the Instrument Driver palette, select Tektronix TX1 and TX3 DMMs->Utility Functions->tktxx Read Battery Level and place it on the diagram panel inside of case statement 1.

26. From the Instrument Driver palette, select Tektronix TX1 and TX3 DMMs->Close and place it on the diagram panel inside of case statement 1.

27. Wire the Instrument Handle and Error terminals as indicated before (same as case 0). The results are shown in Figure 5.

Figure 5.
28. Right-click on the Initialize->Resource Name terminal and select Create Constant. Name the same as before.

29. Right-click on the Initialize->ID Query terminal and select Create Constant.

30. Change the False icon to True and wire the icon into the Initialize->Reset Device icon.

31. Right-click on the Read Batter Level->Battery Level terminal and select create indicator.

Since we already have an error indicator, we would like to connect the error out from this case to the same error indicator we placed in case 0.

32. Click on the left arrow of the case statement indicator to get to case 0.

33. Move the Error Out indicator outside of the case statement.

34. The wire connecting them is now broken. With the Wiring Tool rewire the Close->Error Out terminal to the Error Out indicator.

35. Click on the right arrow of the case statement indicator to get to case 1.

36. Wire the Close->Error Out terminal to the small white box on the border of the case statement where the wire from the Error Out indicator is connected as shown in Figure 6.

When this is done, the small white box becomes a small purple box. The box is called a tunnel. In LabVIEW, whenever a tunnel is created in a case structure as data output, it must be connected for every case. A white box indicates that one or more frames have not connected to this tunnel. A black box indicated that all frames are connected.
At this point, you should play with the Front Panel to get things into a more understandable format. Move items around, color them differently, use the Control Palette->Decorations menu, etc. An example is shown in Figure 7.
Now we need to turn our VI into a subVI. This is done by telling LabVIEW how the terminals are setup. In the upper right corner of the Front Panel is an icon with a front panel of some instrument and a small yellow triangle. This is the standard LabVIEW icon. When you right-click on this icon several options appear.

Note: The icon appears on the Front Panel as well as the Diagram Panel. The options for both are different.

38. Right-click on the Front Panel icon and select Show Connector.

A box composed of several other small boxes replaces the icon image. The small boxes are the terminals. Keeping with the LabVIEW concept of dataflow, the boxes on the left side of the icon are usually for inputs, and the boxes on the right side of the icon are usually for outputs. You should have one input box and three output boxes at this point. If you do not, right-click on the terminal box and select Patterns. Select the appropriate pattern here.

Notice that when you place the mouse over the terminal box, the wiring tool appears. You are now going to wire the terminal to the correct control or indicator.

39. Left-click on the left terminal (there should be only one). The box will turn black. Then left click on the correct input (our program has only one…the Select Action input). The box now turns a different color (not always the same, but not black either). See Figure 8.
Figure 8.

The different color indicates that the terminal is now connected to a control or indicator. Black means that the terminal has been selected, but has not yet been connected.

40. Connect the output terminals in a similar manner (order does not matter). There should be an output for the AC Voltage Reading, the Battery Level, and the Error Out indicators.

41. Move the icon onto the body of the Front Panel and change to the any tool except the arrow tool. Move that tool back onto the terminal icon and click on any terminal. You will see a dashed line appear around the control or indicator to which it is connected. This will undoubtedly help to debug or understand other people’s programs later.

42. Finally, right-click on the Front Panel icon and select VI properties. Change the Category to Preferred Execution System then Instrument I/O.

LabVIEW uses six different execution systems to help control how things run. If you executed this VI as a subroutine in another program, it might very well “bog down” that program until it had finished executing. Using a different execution system facilitates multithreading and allows both sections of code to share processor resources.

Note: for those who are familiar with the concept, using a different execution system forces the system into a time sharing execution. Thus, the subVI will run for a few CPU cycles, then the user interface will run for a few cycles, etc. If we ran the system in the same preferred execution system, the program would execute in series. If the subVI needed to run for a long
period of time, as is the case here, other program functions would be stopped until the subVI had completed.

43. Save this VI as “Your Last Name TX1 control subVI.vi”.

44. At this point you should test your VI to make sure that it functions as you intended it to. Run through the options and see if you get the correct result for each function. Build a simple AC voltage source using the Arbitrary Function Generator and a resistor. Connect the DMM to that circuit so that you have an AC voltage to read. Turn the DMM on. Then run the program with the Read AC Voltage selected on the Front Panel. If everything has been done correctly, the AC Voltage indicator should now be displaying the same result as the DMM display. Now try the battery level.

Note: The battery level can be checked by holding down the softkey #4 while turning on the instrument

Note: Unlike the last lab, we built this VI without the outside while loop. So, in order to check all of the functions, you will have to change the option, run the program, change the option, run the program, etc.

Answer Questions 1 – 2.

USING THE SUBVI

We now want to use the subVI that we have just written in the program that we wrote for the previous lab (entitled Lastname.vi). With the function already implemented and debugged, it should be much less difficult to implement it in our program. Furthermore, if we needed to use the function in another section of code, we have the subVI already written and we just need to connect it to function properly within the main program. Programmers who have been using LabVIEW for a long time usually have a large library of subVIs that accomplish various tasks.

45. Open the VI Lastname.vi. Your program should look like the program in Figure 9.
46. Click on the Dial control and delete it (simply press the delete key). Do the same with the tank control.

47. On the Diagram Panel, press Ctrl-B to remove all bad wires.

48. Select Functions Palette->Select a VI. You will be prompted to choose a VI. Select the TX1 control VI from the first half of this lab. Place the subVI in the second while loop as shown in Figure 10.
Note: you will once again want to expand your work area for the next part (particularly near the second while loop).

In the next section we will be accomplishing two objectives. The first is to create indicators and controls on the Front Panel of the main program that allow us to pass data to and from the subVI. The second objective is to code the system such that the subVI is only called when the menu ring selection on the Front Panel is changed. Obviously, we don’t want to be calling the subVI all of the time (that would put a tremendous strain on the resources of the system). However, we do want to update the value when the menu ring has changed values. In other words, we want to call the subVI only when we change from Read AC Value to Read Battery Level or vice versa.

49. On the Diagram Panel select the Function Palette->Structures->Case Structure, and place it around the TX1 Control SubVI.

50. Right-click on the left side of the while loop and select Add a Shift Register. See Figure 11.
51. Using the Wiring Tool, right-click on the TX1 Control SubVI->Select Action terminal and select Create Control. Remove the connection wire and move the control icon outside of the case statement (see Figure 12).

52. On the Functions Palette, select Comparison->Equal and place it outside of the case statement as well.

53. We can initialize the shift register by selecting Function Palette->Numeric->Numeric Constant, placing it outside of the while loop, and wiring it to the shift register (see Figure 12). The default value is 0 and is acceptable.

54. Wire the Select Action control to one left hand terminal of the Equal Function. Wire the other left hand terminal to the left shift register (see Figure 12).

55. Wire the output of the Equal Function to the question mark of the case statement.
56. Wire the Select Action icon to the TX1 control SubVI->Select Action terminal.

57. Right-click on the TX1 Control SubVI->Battery Level terminal and select Create Indicator.

58. Right-click on the TX1 Control SubVI->Reading terminal and select Create Indicator.

LabVIEW matches the new indicator to the indicator on the Front Panel of the subVI. If you look at the Front Panel of your program (Lastname.vi) you will see your controls from the TX1 control subVI have been duplicated.

59. Connect the Shift Register to the side of the case statement (see Figure 13).
60. In the False case wire the Select Action input through the case to the shift register on the other side. Again you will see the tunnel appear as a white box indicating that additional connections must be made.

61. Change to the True case and wire the shift register input through to the tunnel on the right side of the case statement (see Figure 14 and Figure 15).
Figure 15.

The system then works as follows. At the start of the program, the shift register is initialized to zero. If the select action control has the same value as the shift register, the true case is run. Running the true case passes the value (“0” in this case) back out to the shift register on the right side of the while loop. Now if the value of the select action control changes, the value of the equality function will be false. In this case we run the subVI, and pass the new value of the select action control back out to the shift register on the right side of the while loop.

62. Run the program from the Front Panel and make sure that both options are functioning. You will need to generate an AC voltage to take a reading. Verify your results with the readout from the TX1.

Note: The battery level can be checked by holding down the softkey #4 while turning on the instrument.

63. Save your program as LastnameLab202-2.vi.

Answer Questions 3 – 6.