

ECE 451

Automated Microwave Measurements Laboratory

Experiment No. 2

Automated RF Power Detection Using LabVIEW

Introduction

Experiment design is undoubtedly the most difficult aspect of an engineering project. However, after an engineer successfully establishes an experimental procedure, he or she often has to run this experiment under many different conditions or many different times. Often, the economy of scale in doing so is such that it is very worthwhile for the engineer to automate the measurement. Automation not only speeds up the measurement process, but it also can significantly decrease the likelihood of operator error in a measurement.

Purpose

In this lab session, you will use LabVIEW to automate your measurements from Experiment 1. LabVIEW is a graphical language based on C that is incredibly useful for controlling external hardware. LabVIEW has all of the same programming constructs (conditionals, loops, variables, etc.) that you would use in any other language.

Automating this simple task should give you insights into how to go about automating more complicated apparatuses that you might come across in your career.

Useful-Links

1. LabVIEW tutorial videos on NI <http://www.ni.com/academic/students/learn-labview/>
2. LabVIEW Basics <http://www.ni.com/white-paper/7466/en/>
3. LabVIEW VISA Overview <http://www.ni.com/support/visa/vintro.pdf>

Procedure

Software used: National Instruments LabVIEW and Agilent Advanced Design System (ADS).

Equipment used: Same as Experiment 1 (list equipment explicitly in your report)

Begin by reconstructing the setup from Experiment 1 (shown in Figure 1). You may use either the self-contained crystal detector or the one that is part of the slotted line depending on availability.

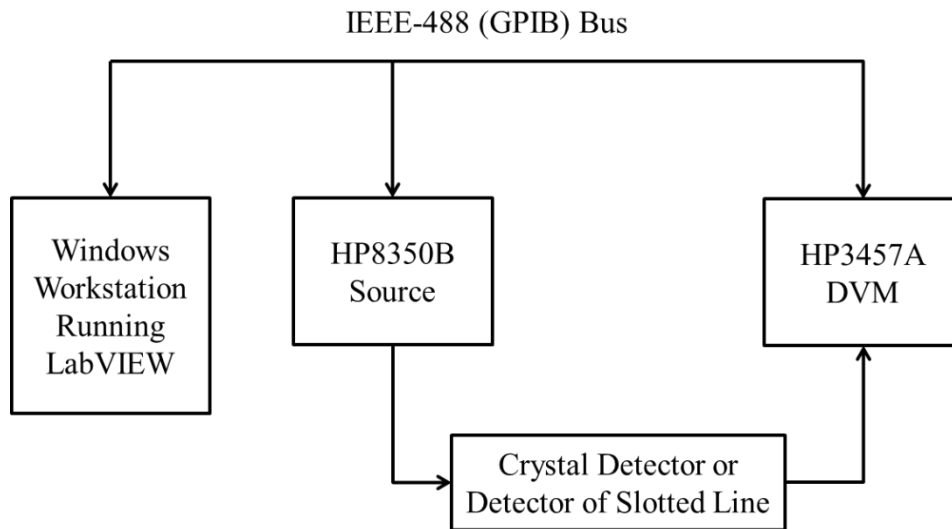


Figure 1: Block Diagram for this experiment.

1) Sign on to the Windows Workstation. In general, do not save anything to C:\. This is so that your files are available to you regardless of which computer you use and so that they are not lost when the Network Administrator does his or her periodic cleaning of the C:\.

2) Make a directory, Lab2, under LabVIEW in your personal drive. It should be:
 U:\LabVIEW\Lab2
 Later you may add other content to this personal directory.

3) Follow the LabVIEW tutorial #1 (see Appendix 2).

In this tutorial, you are writing a LabVIEW Virtual Instrument (VI) in which the user defines the minimum power, maximum power, number of power points, and source frequency. The program will communicate with the instruments and collect the detector output voltages and convert them to the logarithmic voltage format, and display both the linear and logarithmic values on the computer screen. The program will then save the data in LVM file format (which is easy to use within LabVIEW, but not compatible with other applications such as Agilent ADS). Finally, you will add functionality to the program so that the user can switch the save file format from LVM to CITIfile (which can later be imported into ADS). You will use a custom VI made by ECE 451 TAs to generate the CITIfile in the correct format. Figure 2 contains the suggested program flow chart for measuring and saving values. There is also a step-by-step guide for created the VI in the LabVIEW tutorial #1. **Include a printout of your completed front panel for your report.**

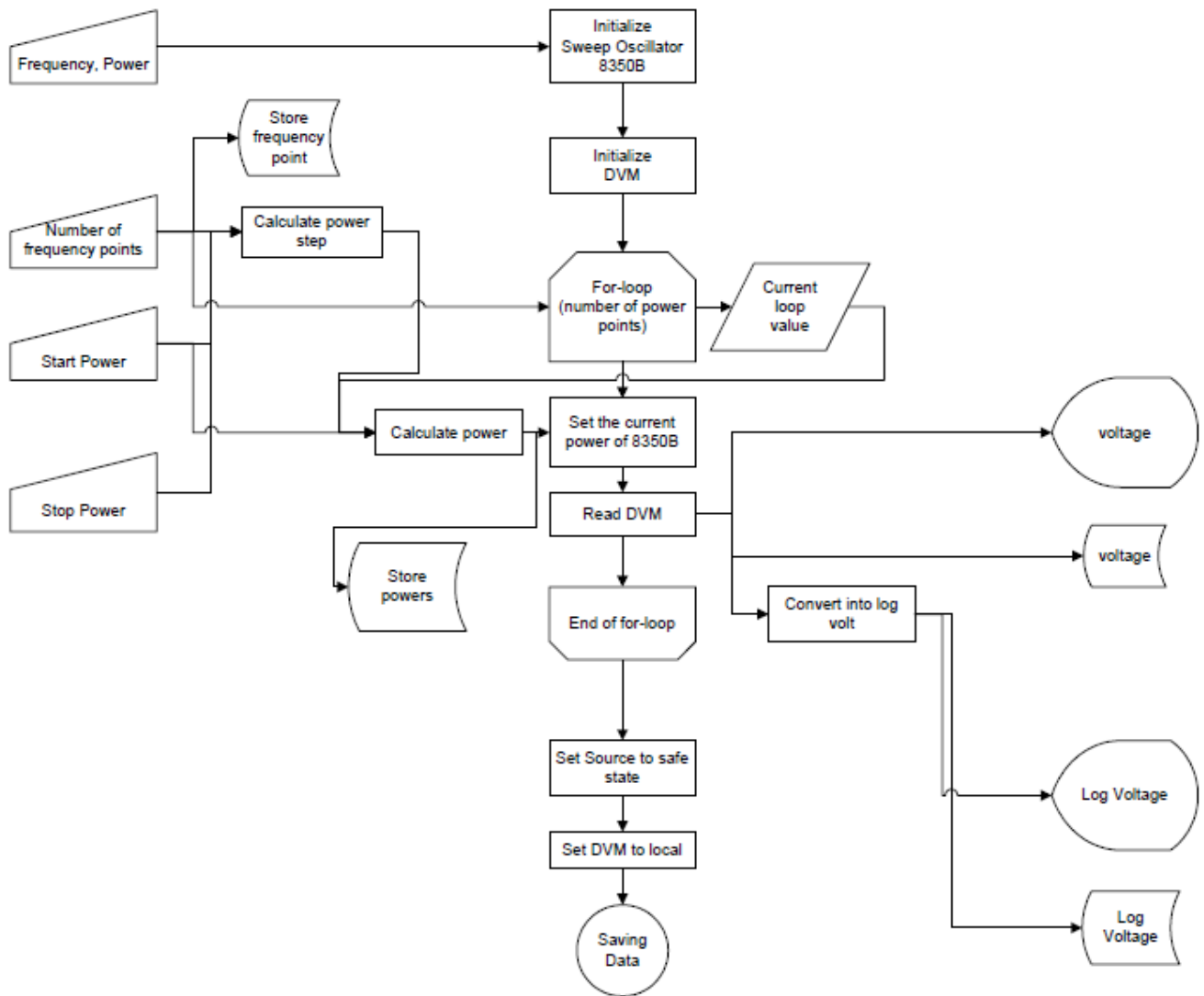


Figure 2: Suggested Program Flow Chart for Measuring and Saving Measured Values

4) Obtain the detector output voltage and logarithmic voltage plots from the first program. **Submit the CITIfile printout along with your lab notebook.**

5) Follow the LabVIEW tutorial #2 (see Appendix 3).

In this tutorial, you are writing a second program that will read the LVM file you have created with the first program and display the contents, which are the normal detector output voltage and logarithmic voltage. **Include a printout of the front panel of this VI in your report.**

6) Follow the short ADS tutorial (see Appendix 4)

Read the CITIfile data and obtain the plot of detector output voltage, and logarithmic voltage.

7) Obtain the plots from ADS dataset window. **Print them and include them in your report.**

Appendix List:

- (1) HP-IB Program Codes (Note: GPIB and HP-IB are the same thing)
- (2) HP3457A Reading and Changing the HP-IB Address
- (3) LabVIEW tutorial 1 (measuring, saving data in LVM and CITIfile)
- (4) LabVIEW tutorial 2 (reading saved data)
- (5) ADS tutorial

ECE 451 Experiment 2: Automated Measurements TA Questions

Theory:

1. What is LabVIEW? Describe the benefits of LabVIEW and why it is used so much in industry today?

Conclusion:

1. Briefly (1-2 paragraphs) detail the major operations and commands used in your program.
2. Compare detector response (and calculate the slope) to that obtained in Lab 1. What is the square law range for both?
3. Comment on any “bumps” that may appear within this square law region. Why do you think these bumps exist? (Hint: This question has nothing to do with theory or the operation of the crystal detector. Rather we are observing non-ideal behavior in our test equipment. While you were acquiring your data, you might have heard a click from the microwave source. This is a relay, which is used for switching.)

Appendix 1: HP3457A Reading and Changing the HP-IB Address

Reading the HP-IB Address

Before you can operate the HP 3457 from remote, you need to know its HP-IB address. The address was displayed during the power-on sequence. If you cannot recall the address, press:



A typical display is:



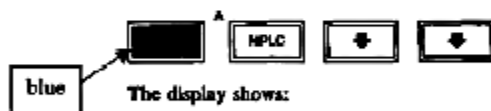
The displayed response is the device address. When sending a remote command, you append this address to the HP-IB interface's select code (normally 7). For example, if the select code is 7 and the device address is 22, the combination is 722.

Changing the HP-IB Address

NOTE

All examples in this manual assume an HP-IB address of 22. We recommend you retain address 22 to simplify programming.

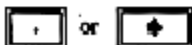
Every device on the HP-IB bus must have a unique address. If you need to change the HP 3457's address, press:



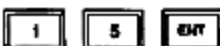
The display shows:



Press:



You can now enter the new address. For example, press:



You have now changed the address from 22 to 15. If you want to change the address back to 22, repeat the above procedure but use 22 instead of 15 in the last step.

Figure 3: HP3457A Reading and Changing the GPIB Address

Appendix 3: LabVIEW tutorial 1 (measuring, saving data in LVM and CITIfile)

Objective

The goal of this tutorial is to be able to write a simple virtual instrument (VI – similar to a program in other programming languages) that accepts the inputs (frequency, power level etc.) from the user, processes them, communicates with the measurement equipment, retrieves the measured raw data from the equipment, analyzes them and presents them to the user in a meaningful form. Using this program, a student should also be able to save the data into a file for later usage.

The concept of LabVIEW programming resembles that of a program flow chart. A box represents each instruction or I/O operation. Boxes are in turn connected with data flow paths (wires). One exceptionally useful aspect of LabVIEW is the “Show Context Help” under the “Help” category. This will create a window that will provide a description of any element that you run your mouse over. Anything in the tutorials that you do not understand can usually be explained with this. **Before starting the program, all instruments should be turned on and connected through the GPIB (IEEE-488) bus.**

Starting LabVIEW

Select *Programs>National Instruments LabVIEW 2013 (32-bit)* from the Start Menu to load the program. Then select *Blank VI*. You will see the blank Front Panel window and Block Diagram of your new program, as shown in Figure 4. You can switch between Front Panel and Block Diagram windows by pressing Ctrl-E.

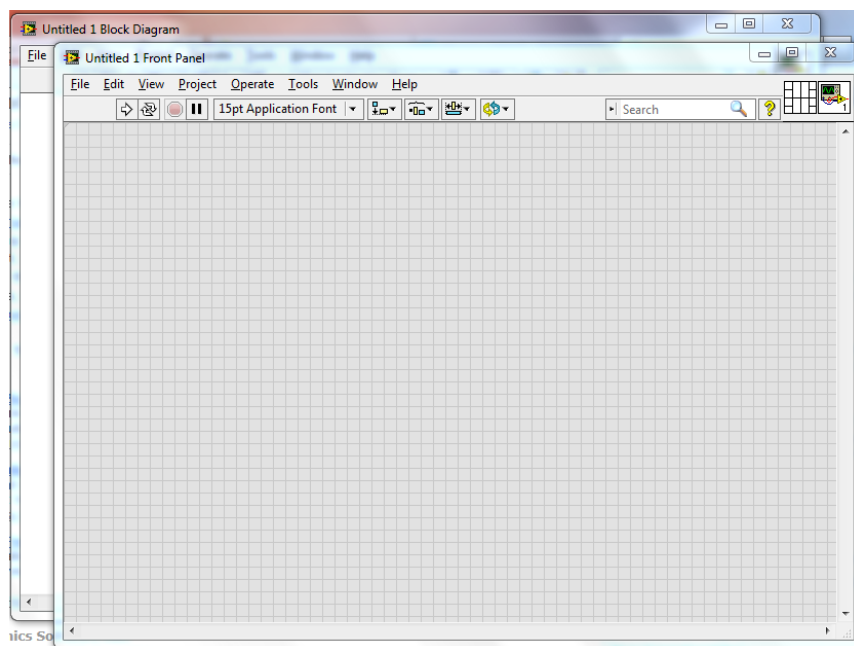


Figure 4: LabVIEW Front Panel and Block Diagram Windows

Communicating with the instruments

VISA resource box must be created on the Front Panel to communicate with each instrument. To do that, right-click on any blank space of the Front Panel and select Modern>I/O>VISA Resource. A combo box titled “VISA Resource Name” will appear. Place it anywhere on the front panel. Change its title to represent a device you want to communicate with, e.g. Source, or DVM (for Digital Voltmeter). From the drop-down listbox, you will be able to select the desired GPIB address that corresponds to the actual device, as shown in Figure 5. LabVIEW automatically detects all the devices connected to GPIB bus, and offers them in the drop-down listbox. Choose the correct address based on which device you want to control. You can usually find the devices GPIB address by pressing a few buttons on the front panel of that device.

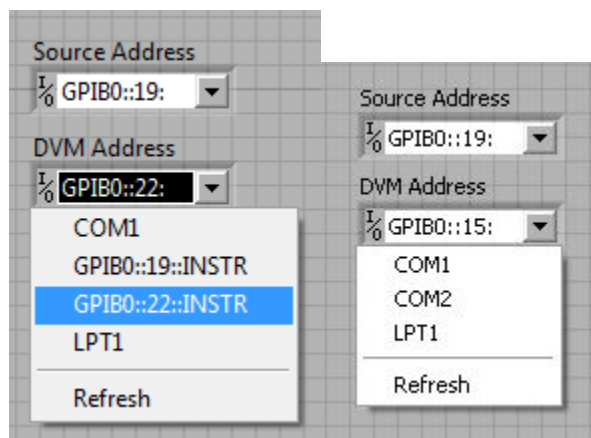


Figure 5: Selecting the GPIB address using VISA Resource box.

Simultaneously with creating the VISA Resource box on the Front Panel, an equivalently named box was created on the Block Diagram. This box will be used to provide the identifier of the device to all the other device control boxes on the Block Diagram (used to Open and Close communication, Init the device, Set parameters, Assert trigger and Read values), as will be seen shortly.

The quick way to copy something is to click on it, then holding “Ctrl,” drag it away. This produces a second copy. Add two VISA Resource boxes to the Front Panel, label them “DVM Address” and “Source Address,” (like in Figure 5) and on the Block Diagram connect each to its own Open box (found in Instrument I/O>VISA>VISA Advanced menu). The output of VISA Resource box should be connected to the “VISA Resource Name” input of the Open box (uppermost input on the left-hand side of the box).

Next, add two Write boxes (found in Instrument I/O>VISA submenu), and connect the corresponding resource name inputs with outputs. You can also change the labels of Open and Write boxes to remind you of their functions, as shown in Figure 6.

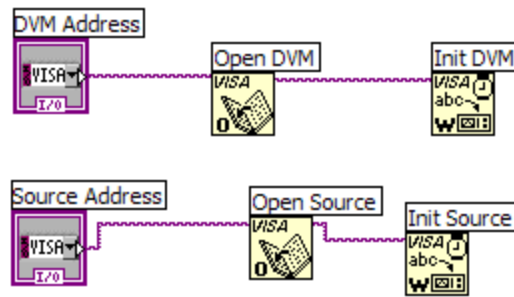


Figure 6: Open and Write boxes that will be used to initialize the devices.

In order to actually initialize the devices, we need to send them appropriate strings. For initializing the DVM, create on the Block Diagram a String Constant (found in Programming>String palette), fill it with appropriate text for initializing the DVM (as shown in Figure 7), create an End Of Line constant (in the same palette), append it to the String Constant using the Concatenate Strings box (again in the String palette), and connect the appended string as an input to “write buffer” terminal of our Init DVM box (as shown in Figure 7).

Initializing the source is a slightly different procedure, since we want to be able to define the sweeping frequency at runtime. To do that, we first create a String Control on the Front Panel (found in Modern>String & Path menu) and label it Frequency. Next, by double-clicking on it, we find its corresponding box on the Block Diagram; its output will be used as an input to the Build Text VI that we will also insert (from Express>Output menu). By double-clicking the Build Text icon, we can define its functionality, as shown in Figure 7. We now connect the output of Frequency box to the “freq” input of Build Text VI, and its output, in turn, to the “write buffer” input of Init Source box.

We now want to add a time delay of, say, 300 ms, in order for our source to have time to stabilize its output. We do that by inserting the Time Delay VI (from Express>Exec Control menu), as shown in Figure 7, and creating the time constant (a quick way to do that is to right-click its “Delay Time” terminal, and to select Create>Constant from the shortcut menu). To conserve screen real estate, we can right-click on two Express VIs we just added and select “View as Icon” option.

Since we don’t want our Time Delay to be executed before we send the initialization data to the source, we need to be able to control the flow of our program. One handy way to do that is by using the “error out” and “error in” terminals, as shown in Figure 7. Error data flow is indicated by a thick pink wire in the figures. **In the version you are using, these thick wires are yellow, black, and white instead.**

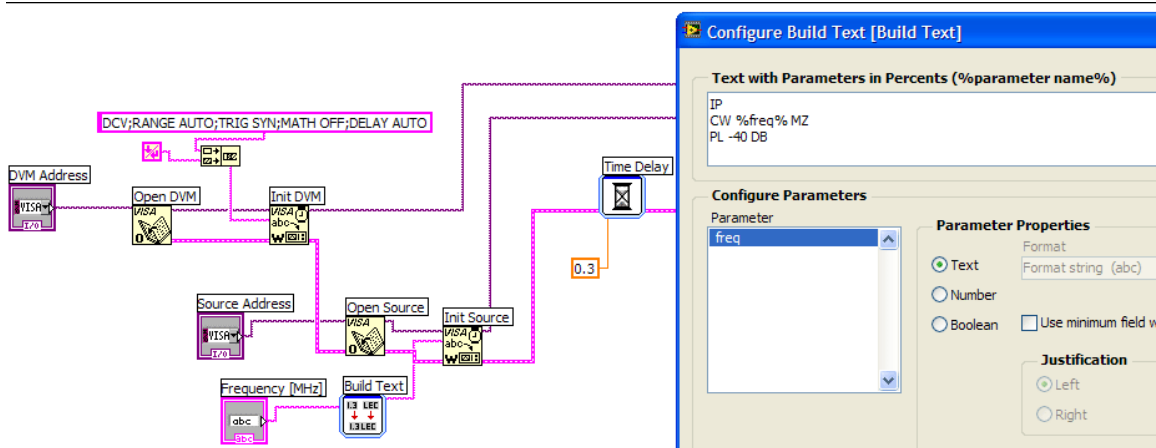


Figure 7: Completed block diagram for initializing the devices.

Now we're ready to move on to measuring the data. On the Front Panel, add three Numeric inputs (Numeric>Numeric Control), and label them as shown in Figure 8. These will serve as inputs to our "for" loop that will be doing the measurements. To have the data represented as integers (as opposed to default of double-precision real numbers), right-click each of the boxes on Block Diagram, select Representation>I32 (actually, in this case, any integer type would do). Apply several math operations, as shown in Figure 8, to obtain the step size for our sweep.

Add a large "for" box (Programming>Structures>For Loop), and connect Start Power, Step, Number of Points, and the two instrument addresses to the left-hand side of the "for" box – those will serve as its inputs. A nice programming practice is to set the cursor to "busy" during measurements; this is done by inserting the Set Busy box (Programming>Dialog & User Interface>Cursor).

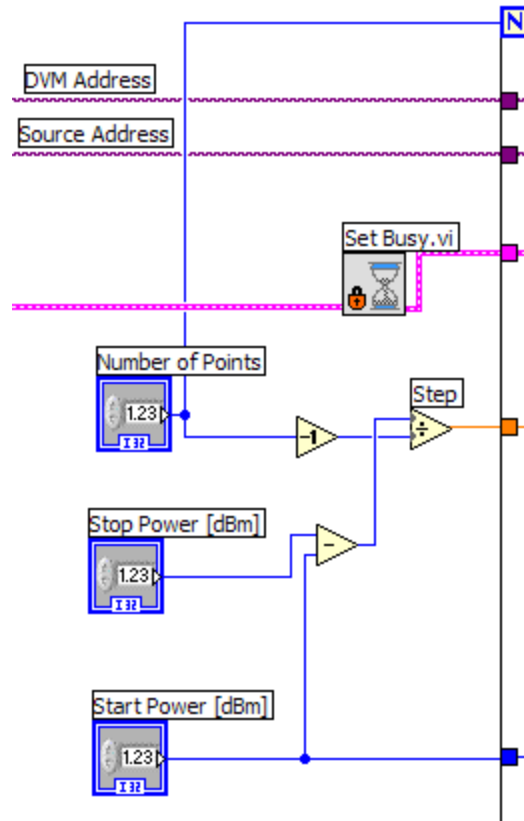


Figure 8: Inputs to the "for" loop.

The “for” loop has two significant objects automatically created: “N” holds the total number of repeats, and “i” holds the current iteration of the loop (ranging from 0 to N-1). We use “i” to calculate the current value of power to be sent to the source. In each iteration of the “for” loop, we first build the string to be sent to the source, by using Build Text VI (labeled “Build Power”) whose behavior is depicted in Figure 9.

Next, we write the string to the source, wait 300 ms for the output to stabilize, trigger the DVM by using Assert Trigger (found in Instrument I/O>VISA), read up to 16 digits of voltage by using VISA Read (again found in Instrument I/O>VISA), and convert the string that was read to a number by using Fract/Exp String To Number (found in Programming>String>String/Number Conversion). This process is shown in Figure 9.

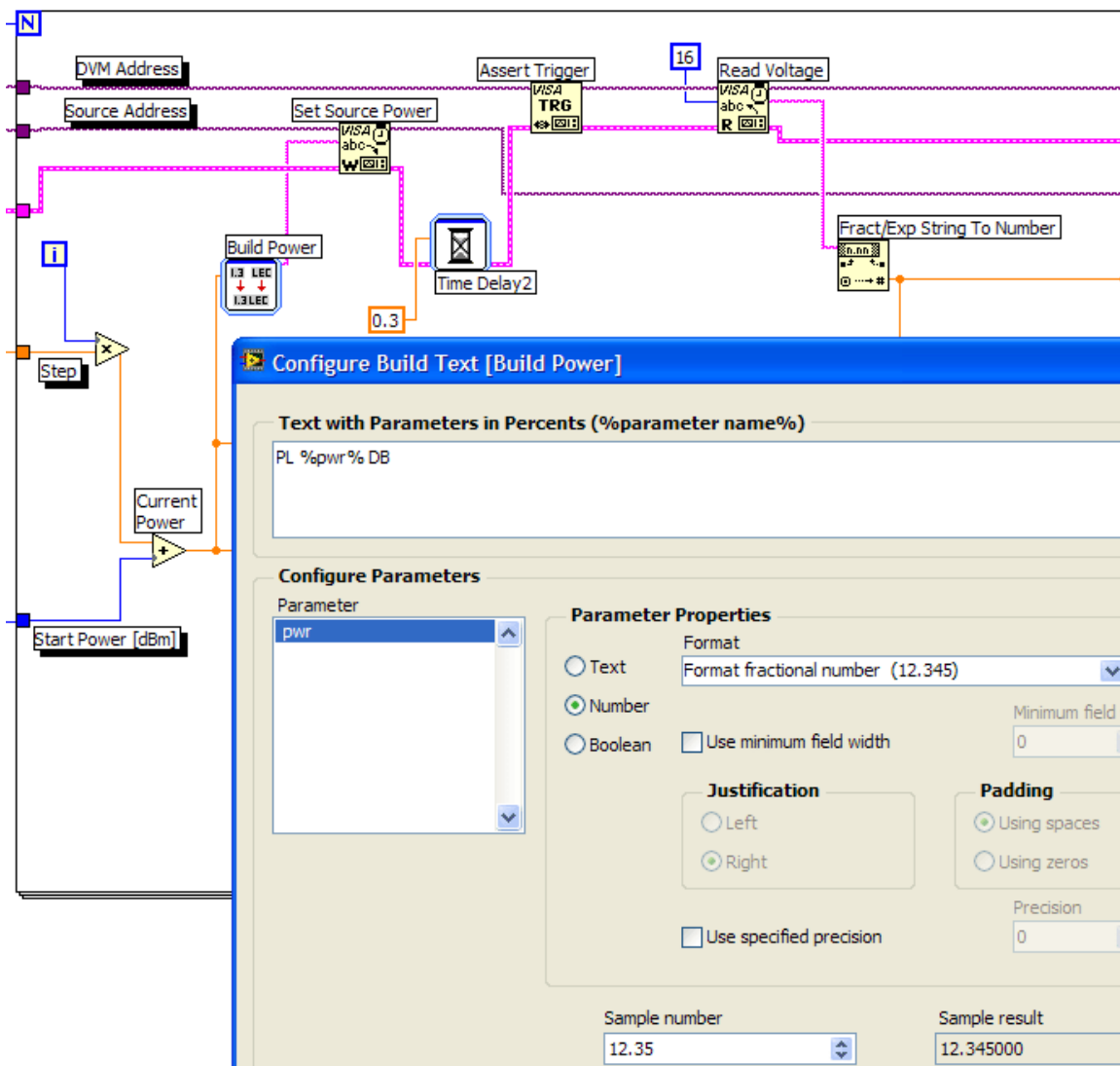


Figure 9: Measuring data in the "for" loop.

If we want to follow our measurements in real time, one possible solution is to insert plots of read data into the “for” loop. To do that, first insert two Express XY Graph objects (found in Express>Graph Indicators menu) to the Front Panel, as shown in Figure 15. Corresponding Build XY Graph VIs will automatically be added to the Block Diagram. Make sure (by double-clicking on them) that “Clear data on each call” is turned off, since we want graphs of complete measurements, not just single points.

Both graphs should have the current value of power connected to their X inputs. The linear graph will have just the read value of voltage as its Y input, while we would need to calculate the log value of voltage (in units of dBm) as shown in Figure 10. Log is located in Mathematics >> Elementary >> Exponential.

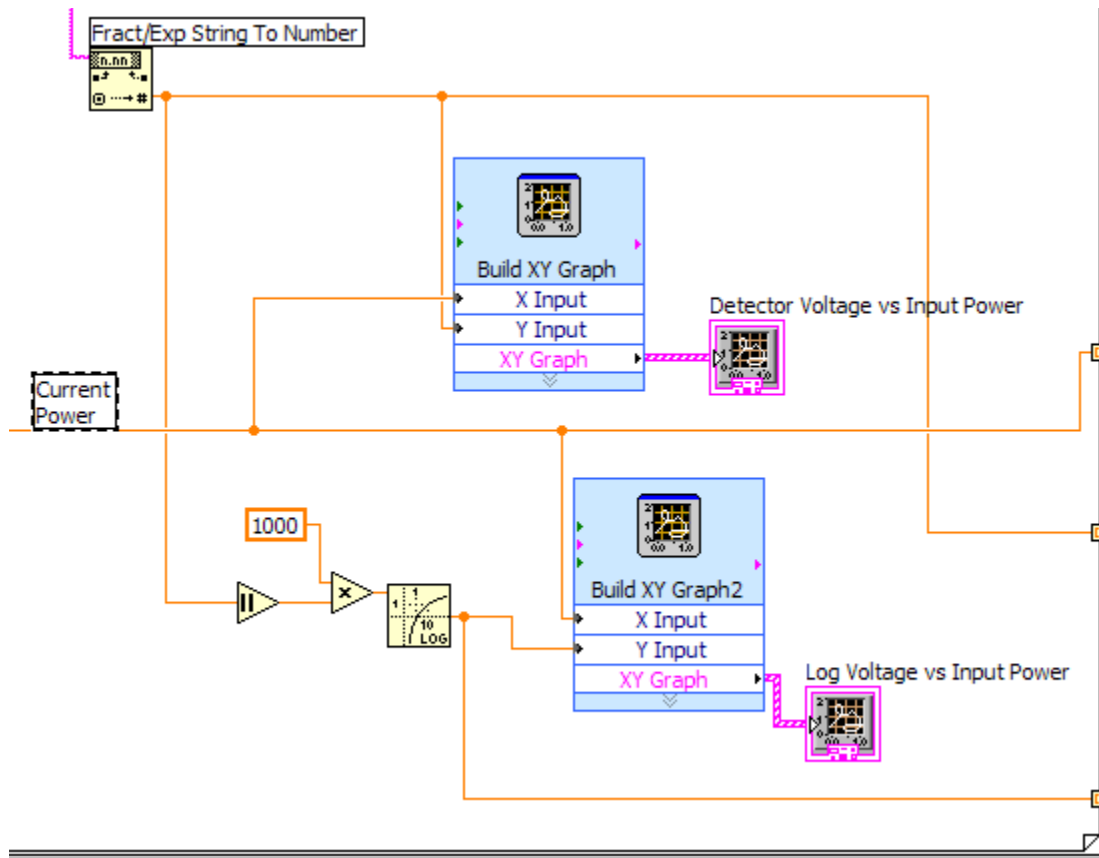


Figure 10: Plotting the measurements in real time.

After reading the data and finishing with “for” loop, we would first want to unset the Busy cursor, power down our source (e.g. to -75dBm) and close communication with the instruments, as depicted in Figure 11.

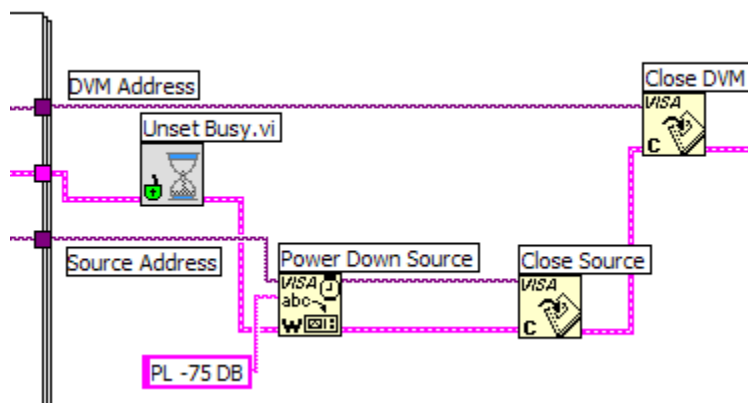


Figure 11: "Cleaning up" after measuring.

Finally, we want to save our measured data to a file, in order to be able to analyze it later. LabVIEW's "for" loop automatically collects all the data coming out of the loop, and creates arrays out of it; the process is called "auto-indexing" (if needed, this behavior could be changed by right-clicking the point where the data leaves the loop - in our case we would want to uncheck auto-indexing for the DVM Address, Source Address and Error wires).

To make use of LabVIEW's full potential in working with files, one easy way is to convert the three arrays of data (power, voltage and logvolt) to Waveform data types, by using the Build Waveform box (found in Programming>Waveform palette), as illustrated in Figure 12.

Next, we set the array names to represent the type of data measured, by using Set Waveform Attribute (found in Programming>Waveform) to set the "NI_ChannelName" attributes of the waveforms, also depicted in Figure 12.

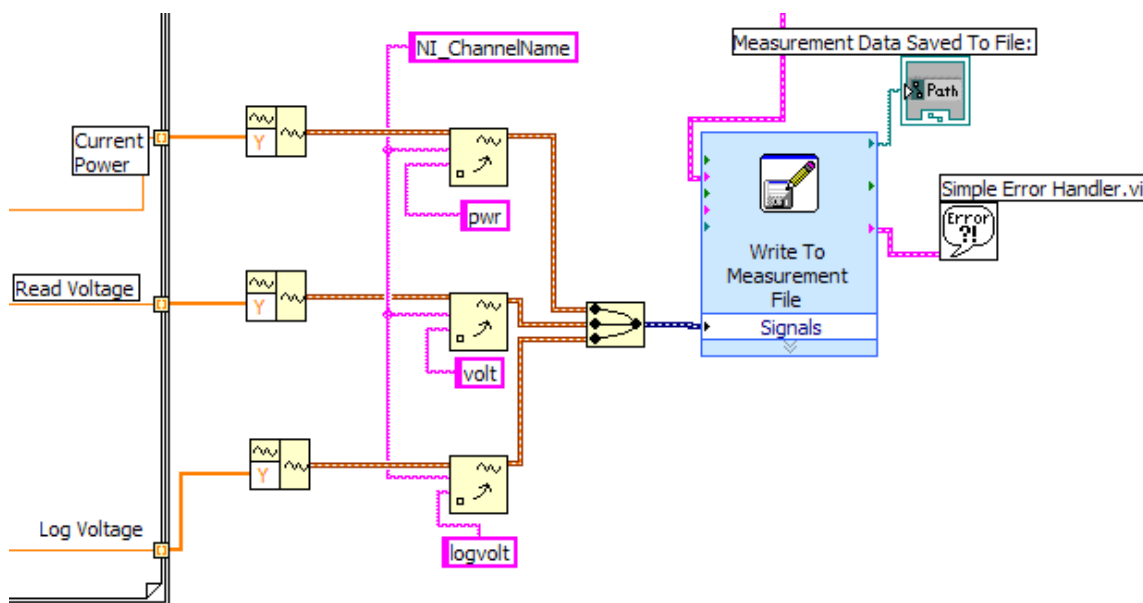


Figure 12: Formatting and saving the data.

We then aggregate the three waveforms, by using the Merge Signals box (found in Express>Signal Manipulation), and feed them to the Write To Measurement File VI (found in Express>Output), the settings of which are visible in Figure 13.

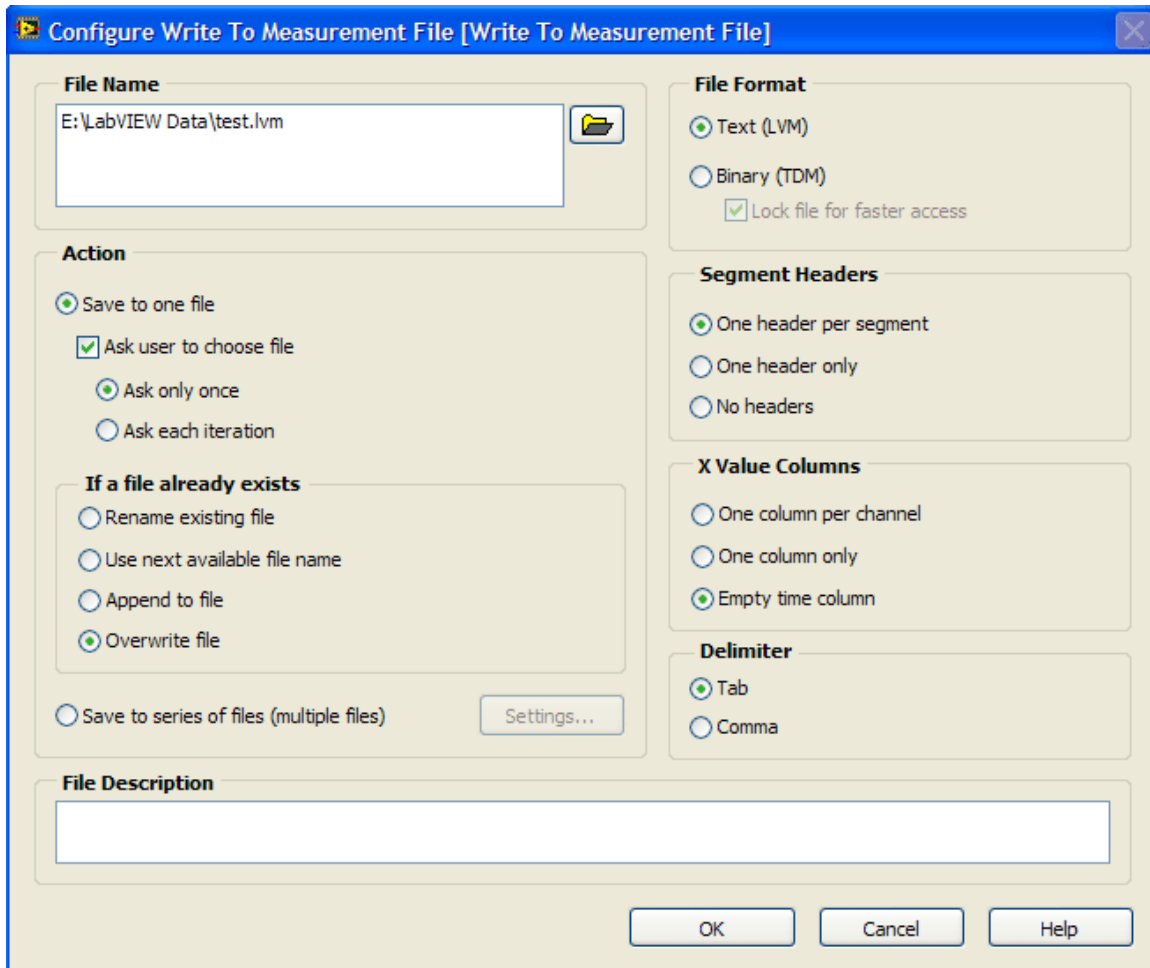


Figure 13: Write to Measurement File configuration.

This way, the data is saved in LabVIEW's proprietary text format¹ with the extension LVM, viewable in a text editor, but not directly importable into other programs, such as Agilent ADS.

¹ Note: The LabVIEW Measurement (.lvm) format is a text-based file format for one-dimensional data that you want to use with the Read LabVIEW Measurement File and Write LabVIEW Measurement File Express VIs.

The .lvm file is designed so it is easy to parse and easy to read when imported into a spreadsheet program, such as Microsoft Excel, or a text editor, such as Notepad. It supports multiple data sets, grouping of data sets, and the addition of data sets to existing files.

The file format is not designed for high-performance or for very large data sets, as is the case with all text-based formats. Use the binary file format, such as HDF5, for very large data sets.

Specification for the LabVIEW Measurement File (.lvm) is available at:

<http://zone.ni.com/devzone/conceptd.nsf/webmain/041828bc369ee1a686256d33005303fd>

In order to communicate with ADS, we would need to write a separate subroutine for saving the data into e.g. CITIfile² format, which falls out of the scope of this course because of the relative complexity of that subroutine (compared to the simple “Write to Measurement File” Express VI). However, for the purpose of demonstrating how to export data measured in LabVIEW to ADS, a complete subroutine for saving the data in CITIfile format can be found at V:\ece451 named p_lab2_save2citifile. Before adding that subVI, place a conditional loop to enable the user to choose which file format to use Express>Exec Control>Case Structure as shown in Figure 14.

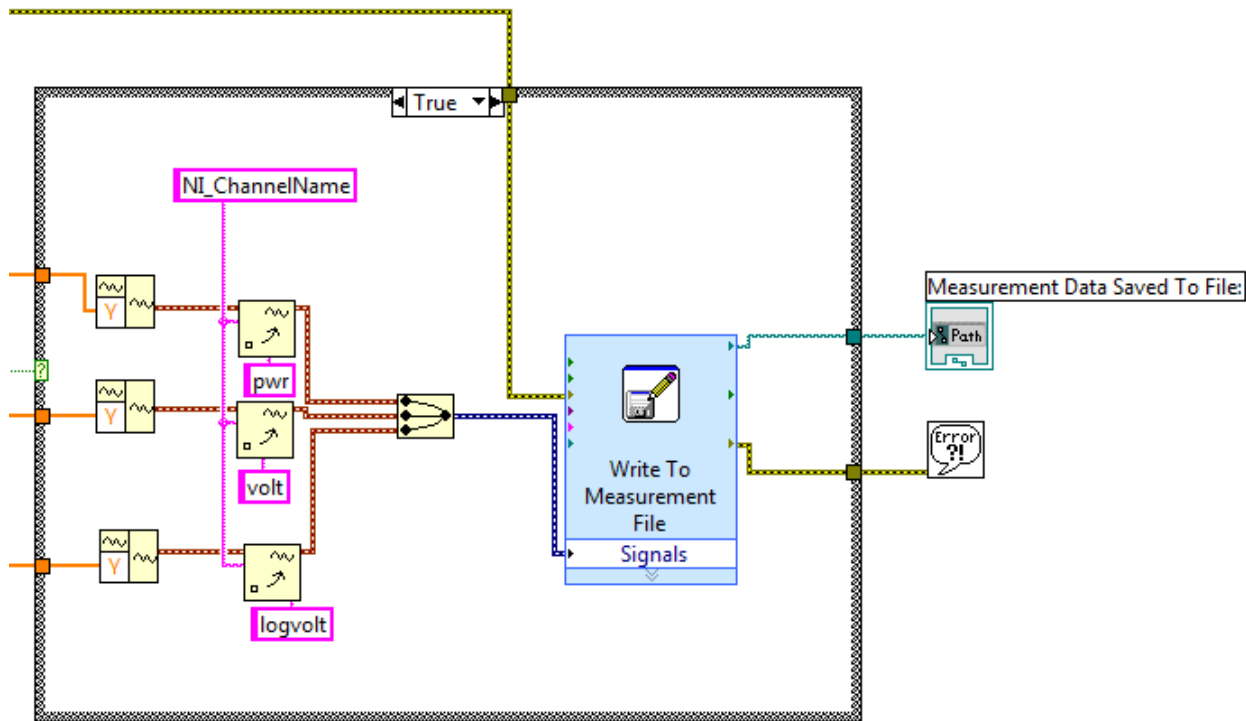


Figure 14: Inserting a conditional structure around the LVM generation code.

Now, switch to the false case by pressing one of the arrows to the right of the box that says “True.” Place the aforementioned subVI by right-clicking and selecting “Select a VI,” which then opens a dialogue box from which the p_lab2_save2citifile can be found. Create a Boolean control to the conditional structure by right clicking the left wire end of the green question mark box on the left of the structure and selecting Create>Control. Wire the subVI as shown in Figure 15.

² Note: Hewlett-Packard Co. (now Agilent Technologies) developed the CITIfile (Common Instrumentation Transfer and Interchange) format for computer/instrumentation data exchange and subsequently adopted it to the MDS and ADS EDA tools. CITIfile is suited for load-pull data since it can support an arbitrary number of dependent and independent variables. One requirement is that the independent variables must be methodically swept—that is, the same inner values of the sweep must be identical.

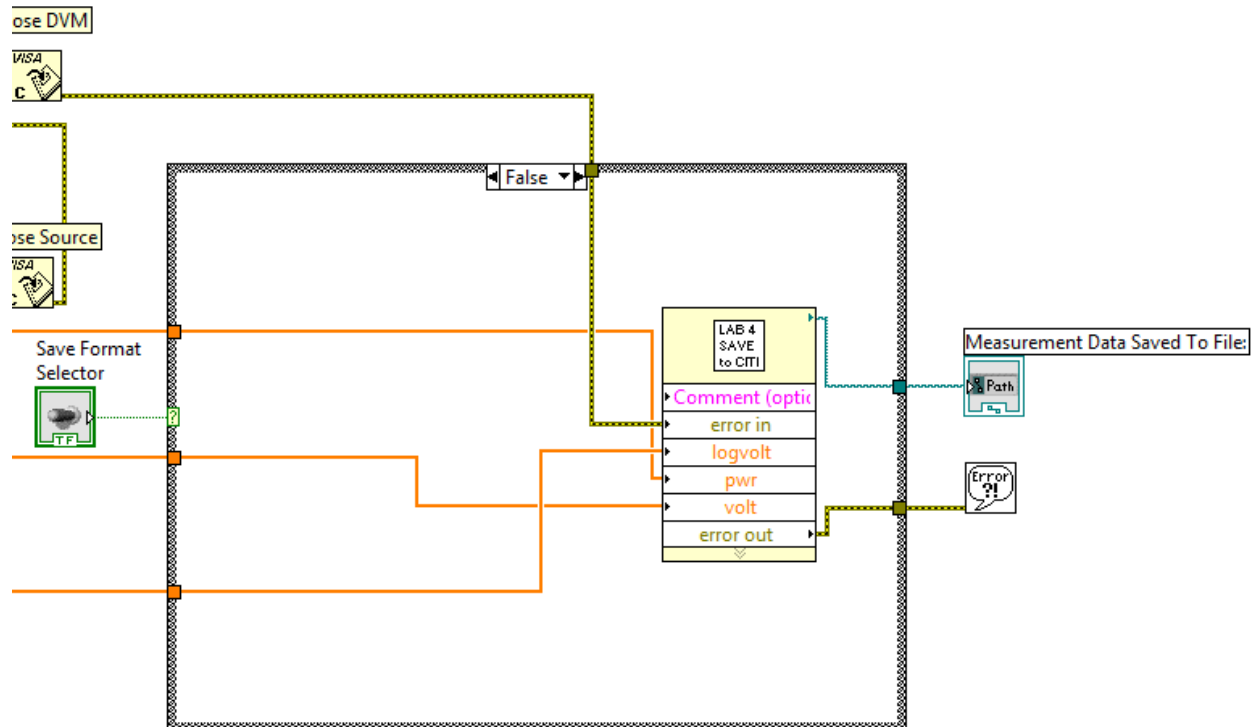


Figure 15: SubVI for saving the data in CITIfile Format.

We finally insert a File Path Indicator to the Front Panel (from Modern>String & Path) to be able to observe the actual location of the saved file, as shown in Figure 16, and end the dataflow with Simple Error Handler (from Programming>Dialog & User Interface). Figures 14-17 show the saving part of our program.

A printout .pdf of the Front Panel, Block Diagram, and accompanying information can be found at V:\ece451 titled “Lab2App3.”

Print out a final copy of your Front Panel and of the CITIfile that you generated for you and your lab partner to include in your lab reports. The CITIfile can be opened in any text editor, like Notepad or Notepad++.

Appendix 4: LabVIEW Tutorial 2 (reading saved data)

Objective

The goal of this tutorial is to write a program that will read the LVM file that you have created with the first program and display the contents, which are the normal detector output voltage and logarithmic voltage.

Before starting LabVIEW, all instruments should be turned on and connected through the GPIB (IEEE-488) bus.

Reading the data

One of the advantages of LabVIEW's LVM format is the ease with which we can retrieve saved data, as shown in Figure 16. We simply insert an Express VI named "Read from Measurement File" and error handling and optionally display the path and filename from which the user opted to read the data.

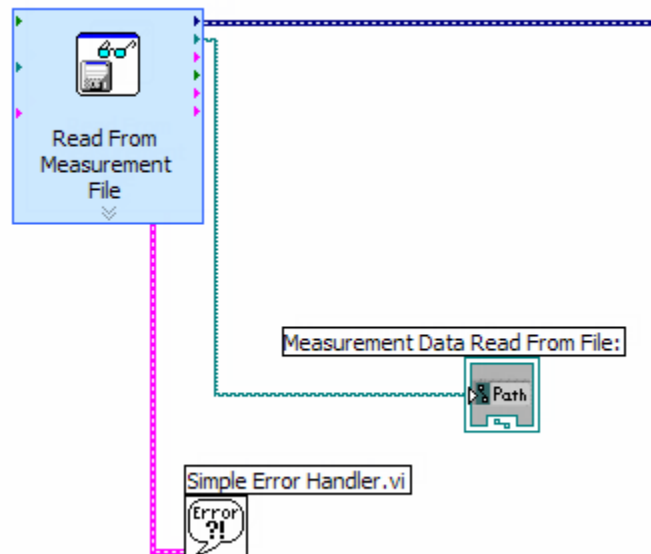


Figure 16: Reading data from LVM file.

Plotting the Read Data

Plotting the data is done in very much the same fashion as in Tutorial 1 – we simply break down the set of signals into individual arrays, using Split Signals (found in **Express>Signal Manipulation**), and then plot two graphs using standard Build XY Graph Express VI, as depicted in Figure 17.

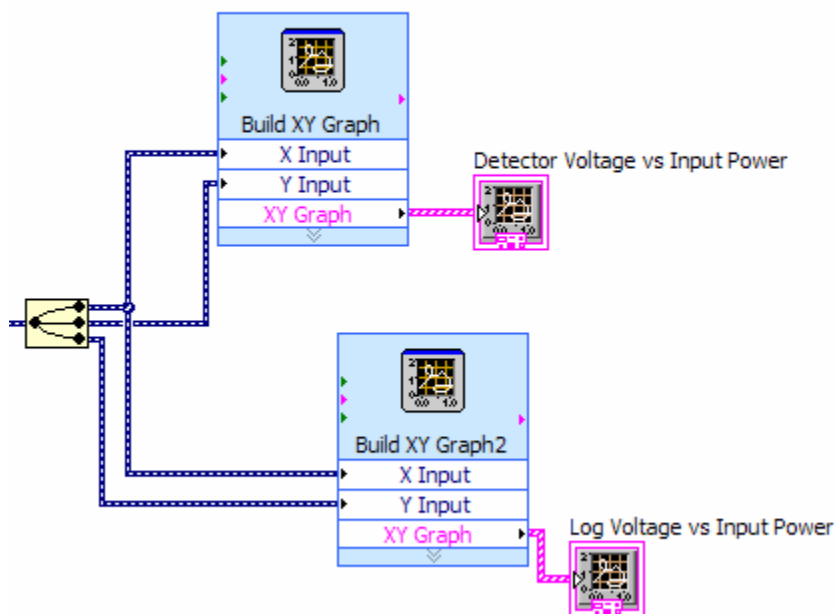


Figure 17: Plotting the voltage and logvolt measurements.

Creating a Table of Read Values

Creating a table of array values is also done using an Express VI, as shown in Figure 18.

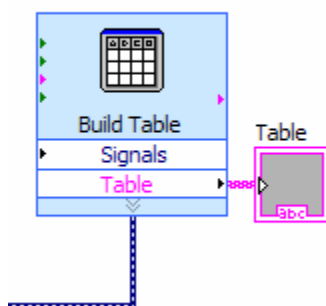


Figure 18: Displaying arrays of measured values in a table.

Unfortunately, extracting array names and labeling table columns is not as straightforward. We need to directly manipulate the Property Node of the table which contains column headers.

One possible way of achieving this, as suggested on the NI website, would be to add the `ex_GetAllExpressAttribs.vi` box (found in “V:\ece451”) and feed it with the set of signals we have read. Then, we would unbundle the signal names and group them into an array, with which we would feed the `Strings[]` Property Node of the table (created by right-clicking on Table object and then selecting **Create>Property Node>Column Header Strings**). The entire process is depicted in Figure 19.

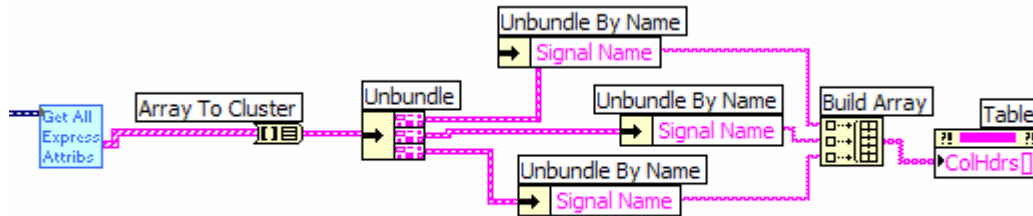


Figure 19: Adding column headers to the table.

A printout .pdf of the Front Panel, Block Diagram, and accompanying information can be found at V:\ece451 titled “Lab2App4.”

Print out a final copy of your Front Panel for you and your lab partner to include in your lab reports and close LabVIEW.

Appendix 5: ADS Tutorial (Reading CITIfile)

Objective

The goal of this tutorial is to show you how to load a CITIfile measurement into the Agilent Advanced Design System (ADS). An engineer frequently will need to load measured data into his or her simulation software of choice. This tutorial shows you how to do so for ADS.

Procedure

To start Agilent Advanced Design System, go into the start menu and type “advanced design system.” “Advanced Design System 2013.06 (32-bit Simulations)” should pop up. Select it to load the program. The intro screen in Figure 20 should pop up. Click “Create a new workspace.”



Figure 20: ADS Introduction Screen.

1. Click “Next” and then give your workspace the name “ECE451_Lab2” and then browse to select your network directory where you would like to save the workspace. After doing so, click “Finish.” Once you do so, the workspace window will appear automatically.
2. Click “Window” and select “New Data Display.” A new data display will pop up as shown in Figure 21.

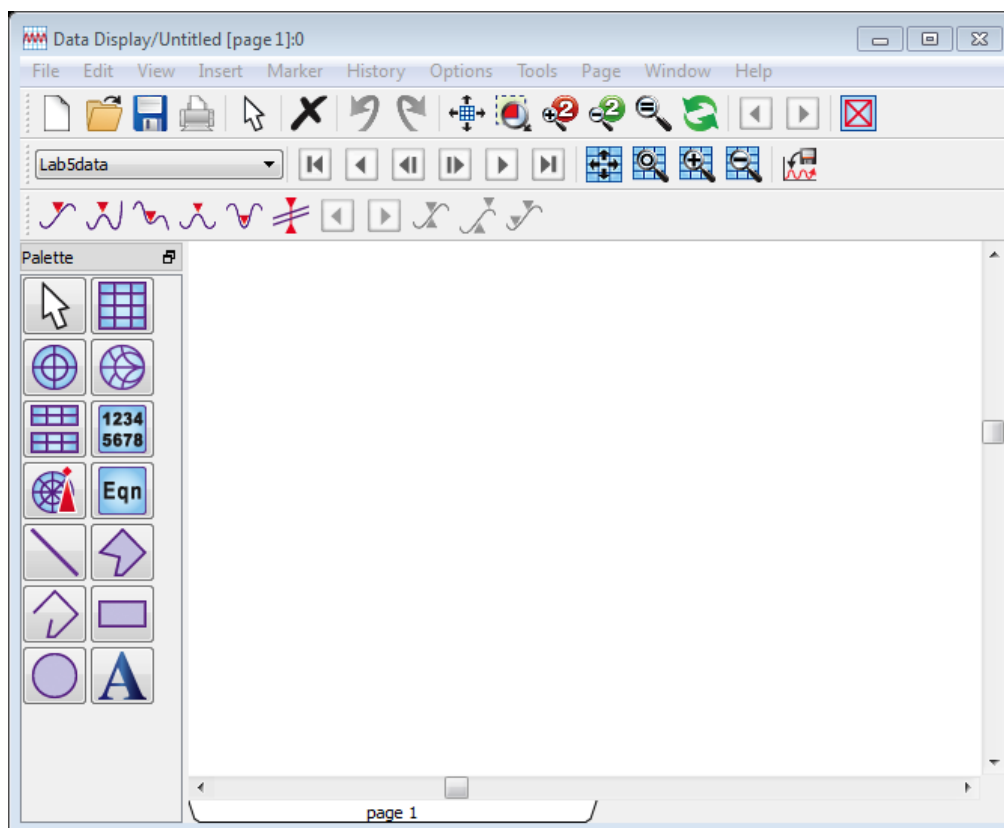


Figure 21: Data Display Window

3. On this new display, click “Tools” and select “Data File Tool...” The Data File Tool will pop up as well as the simulation status screen (the box with the options: File, Simulation/Synthesis, Text, Window). Ignore the simulation status screen for now and click on the data file window, shown in Figure 22.

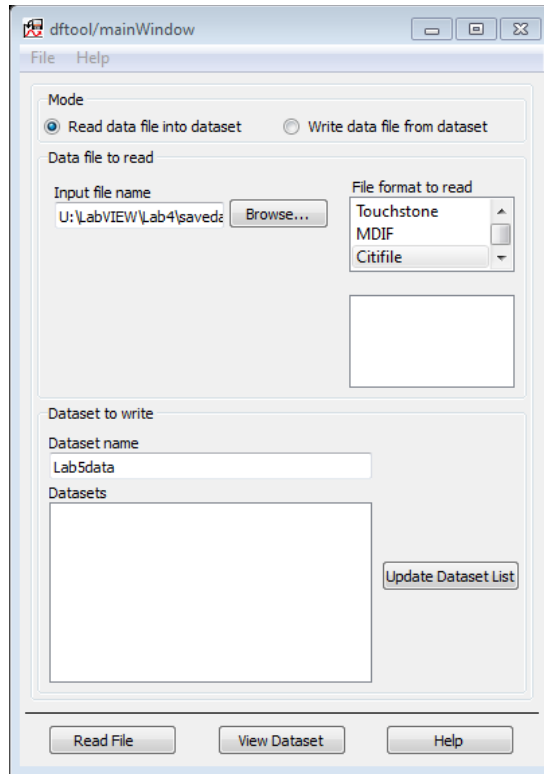




Figure 22: Data File Tool Window.

4. Click on “Browse...” and select the CITIfile you generated in Appendix 2 and click on Citifile. Then write “Lab2data” in the “Dataset name” field, as shown in Figure 22 and press “Read File.” The simulation tab should say “File read was successful.” Close the data file tool window.

5. Back on the Data Display Window, click on the Rectangular Window icon  and place it on the white space. After you do so, a dialog box will pop up. Select “logvolt” and then press “Add Vs.” and select “pwrswEEP.” Press “OK” twice and you will see plot of logvolt vs. pwrswEEP. Do the same to generate another plot with volt vs.

pwrswEEP. Next try pressing  and then clicking on the logvolt vs. pwrswEEP graph to place a marker. Place a marker at -10 dBm, as shown in Figure 23.

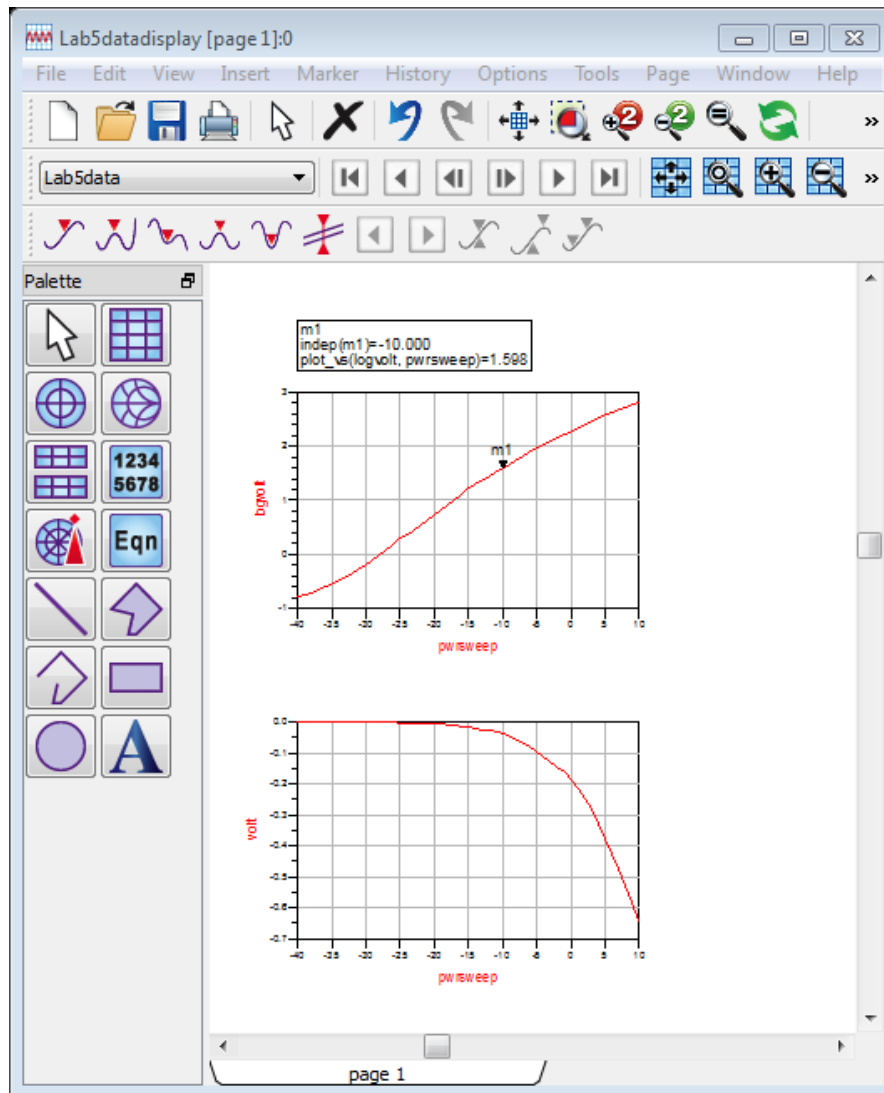


Figure 23: ADS Final Data Display Window.

6. Print out a copy of the Data Display for you and your lab partner for your lab reports and exit ADS.