TABLE OF CONTENTS

Chapter **Page**

CHAPTER 1

INTRODUCTION

1.1 Overview

Automated microwave measurement laboratory offered in the Department of Electrical and Computer Engineering aims to have the student able to assemble, program, and utilize sophisticated automated microwave measurement systems, with an appreciation for the capabilities and the limitations of the microwave measurements and of the automated system. To introduce students with the fundamentals of high-frequency measurements and the latest techniques for accuracy-enhanced microwave measurements, automated network analyzers and high-speed wafer probes are used in conjunction with state-of-the-art calibration techniques. Error correction for accuracy-enhanced measurement is performed using Agilent-VEE Pro, ADS and RMB on workstations. Data acquisition and network optimization are achieved through the controllers.

1.2 Motivation

In the past, students performed automated microwave measurements with HPBASIC programming which is a traditional textual language and forced programmers to define execution order at too low a level. Due to the time limitation in the laboratory, the students find it learning a new syntax based programming is frustrating and time

consuming. In contrast, VEE programs can be created by placing graphical objects on the work area and wiring them together with lines. This graphical view is the actual program. A student does not need to perform any cryptic and error-prone steps like preprocessing, compiling, or linking. Programs are easy to be edited and modified. The most appealing feature is the availability of hundreds of powerful functions to help students analyze the measurement data.

1.3 Organization

This thesis describes how the VEE programming can replace the previously written HPBASIC programming in three automated microwave measurement labs and emphasizes the importance of applying VEE to enhance students' learning process. Chapter 2 gives a thorough background on VEE programming and states its powerful features. Chapter 3 discusses the application of VEE in three automated measurement lab session. Each lab section in this chapter is first briefly explained based on the background lab, and then emphasized on the advantage of using Visual programming over the previously used HPBASIC programming in terms of simplicity, readability and modifiability in the program development process. The measurement results and plots obtained from VEE programs are presented and discussed at the end of each section of chapter 3.

CHAPTER 2

BACKGROUND

2.1 Introduction

Agilent VEE is a graphical programming environment optimized for use with electronic instruments. The simplest way to state its value is the phrase “Better measurements faster.” It maximizes the benefit of the two dimensions of graphics, for faster measurement analysis results. Software prototyping is noticeably faster with Agilent VEE than with lower-level languages. VEE handles day-to-day programming tasks in instrument control, measurement processing and test reporting. It automates instrument configuration, streamlines test sequencing, and simplifies application development. VEE was first introduced as HP VEE version 1.0 from Hewlett-Packard Company in 1991. Version 2.0, released in late 1992, extended its capabilities. Version 3.0 entered the market in February 1995 with major changes. Version 4.0 was released in February 1997. In 1999, Agilent was spun off from Hewlett-Packard Company, and VEE has been continued to support it as Agilent VEE version 5.0 in 1999. Then, Version 6.0 was released in March 2000. The current VEE 6.1 release in March 2002 provides support for Microsoft Windows XP, MATLAB® 6.1 integration, improved ActiveX capabilities, FlexLM® licensing and other improvements.

2.2 Building Blocks

Agilent VEE programs are created as data flow diagrams, or programs by selecting and placing graphical objects on the screen and connecting them with lines. The program is a collection of icons, each of which executes complied code to perform a function such as transforming data. They interact with graphical user interface, communicating with instrumentation or other networked computers, executing complied code or system calls, or evaluating condition which control program behavior. These icons communicate with each other primarily by passing data back and forth to each other along “wires” drawn between them. In some cases, they communicate by passing control flags to indicate that events have occurred. These programming elements are predefined from a series of hierarchical menus or defined by the developer.

VEE integrates the entire view of the application on one panel. The developer can promote interface elements to a panel view that serves as the general user’s application interface. The interface effectively restricts the view of application to only those elements needed for input and output. For development purposes, the interface elements are visible and functional within the program code itself. For expediency, many programs simply dispense with the panel view. Otherwise, it makes debugging easy by allowing the I/O to be viewed along with the program execution.

2.3 Programming Basics

2.3.1 Objects and Functions

Managing code side and structure mainly involves reducing the complexity of a panel to a reasonable level. VEE allows a hierarchical decomposition of a problem by

encapsulating groups of code elements into a User Object or User Function. In either case, the encapsulate code not only can be reduced to an icon on a panel, but also can be duplicated and called throughout the program, much like a subroutine. Duplicating a User Object makes a separate copy of the code, which then can be modified to produce a slightly different version. On the other hand, User Functions store the code in a single separate location, allowing multiple copies of its icon representation to call the same piece of code. In this way, one change to the User Function code affects the change for every procedure that calls it. Both features allow the developer to save code in Object Libraries for later use. VEE objects, the building blocks of a program, appear as rectangular boxes in the work area. These boxes performed the program actions. The boxes are connected by lines. These lines carry program data. Data travels out from the right side of an object (the output pins) and enters the next object on the left side (the input pins). Data flows from the left to right (as viewed by any given object). Since the execution of code is determined by dataflow through wires, VEE can multitask by running multiple execution threads and User Functions in parallel.

2.3.2 Menus

VEE has three types of menus from which to select: Pull-down menus that allow the user to access objects, perform actions, and set states. Object menu are specific to selected object. Cascading menus are sub-menus available from the main menus and object menus. Some menu options bring up dialog boxes. Menu options are accessed by clicking and dragging the mouse. Objects are manipulated easily in the same manner. They also can be collapsed into icons or double click to reveal their Open Views that

allow the user to see and edit an object's function. For instant, the number of input or output terminals may be set from an Object's Open View.

Over 100 other VEE objects are organized in five drop-down menus. Just click a main menu, follow the appropriate submenu (if any), click the desired object, and place it on the work area. Here is brief overview of the five drop-down object menus:

Flow These objects influence program flow. Some objects perform traditional execution control, such as conditional branching and looping. Others alter program and data timing with operations like delay and gate.

Device This is a grouping of miscellaneous objects. It includes useful submenus like Virtual Sources and ActiveX Controls. There are handy utility objects, like a counter, timer, and comparator. The Import Library and Sequencer objects are here. (If the object you want is not found in the first menu you check, look for it in the Device menu.)

I/O These objects interact with software and hardware outside the VEE application. This menu includes objects that communicate with instruments, files, printers, other processes, HP-UX pipes, and more. The first menu pick is the powerful Instrument Manager.

Data These objects declare, generate, and access data. This menu includes graphical input objects (push buttons, knobs, sliders, etc.), constants of every data type, global variables, and handy functions like Unbuild Array, Merge Record, Concatenator, and more.

Display These objects help you view data. This menu includes graphical indicators (meters, thermometers, etc.), alphanumeric displays, waveform (oscilloscope) displays, various X-Y plots, polar spectrum displays, and more.

2.4 VEE and Instruments

VEE programming environment strongly supports electronic instrumentation and measurement tasks. It is ideally suited to develop applications with external instruments accessed by GPIB or RS-232 interfaces. It is standards friendly due to improved ISA, PCI, CPCI, and PCMCIA supports. Programs and graphics can be monitored from web as remote diagnostics. For those early prototyping stages, VEE provides simulated signal sources and displays. Experiment can be done with program flow and data processing without any external hardware. For real-world data, VEE provides an instrument Manger and a Dynamic I/O Server to simplify the tasks of discovering, configuring, and managing external instruments. VEE supports several types of instrument drivers and lets the programmer choose the driver that best suits his or preferences and/or the characteristics of an instrument. For those times when the instruments are not available, just pressing one button in the instrument Manager to take a driver “off line” and continue developing the program. It carries out smart measurements by taking any measurement and control any PC cards or instrument from any vendor, utilizing over 500 National instruments LabWindows/CVI drivers, embedding I/O configuration for easier unlimited runtime distribution, programming the properties of instruments, automatically verifying instrument addresses and other parameters at runtime, automatic error troubleshooting for instrument timeouts.

2.5 Multi-Language Environment with ActiveX control

VEE provides a focused measurement analysis environment that allows the programmer to use its capabilities in conjunction with the best from many languages such

as C, Visual Basic, LabVIEW, VB for applications, MATLAB, VB Script, MATLAB script and Visual C++ through ActiveX by calling DLLs. In other words, VEE integrates into one environment what used to require two separate software packages one for measurements/data collecting, and one for spreadsheet/math analysis. It can easily leverage contribution from any market-leading instruments or PC cards, software applications, programming languages, or manufacturing equipment. For example, collaboration between Agilent and The MathWorks has resulted in leading-edge analysis and visualization capabilities of MATLAB embedded into an easy graphical programming environment set a new performance point that blends power AND ease of use for engineers. In addition of VEE's hundreds of functions, hundreds of powerful MATLAB functions and The MathWorks Signal Processing Toolbox functions inside VEE can now be accessed.

CHAPTER 3

APPLICATION OF VEE

3.1 Automated Detector Measurements

3.1.1 Introduction

Detectors use a nonlinear device to achieve frequency conversion of an input signal. Microwave diodes are most commonly used as the nonlinear element. All experimental measurements made in the microwave region depend upon the ability to detect the presence of RF power. The square law devices used for the detection of low-level RF power include the crystal diodes and thermally sensitive devices such as bolometers, barretters and thermistors. They derive their name from the fact that the output, a voltage or a current, is approximately proportional to the square of the input RF voltage or current. Thus, this output is proportion to the applied input power. In this detector response measurement, VEE programs will play an important role in checking the response of a crystal detector by using it to detect RF energy from a microwave source.

3.1.2 Experiment Description

The new lab manual is rewritten such a way that students are instructed to follow the two tutorials. The tutorials and a detailed lab instruction are provided in the appendix A. The first tutorial gives the students a complete guide to write a basic VEE program. The program accepts the frequency of 700 MHz and start and stop power of –40 and 10

dBm entered by the user. Then, it processes them to determine power steps and communicates with the measurement equipments, which are microwave source HP 8350B, and digital voltmeter (DVM) HP 3457A. Finally, it retrieves the measured output voltage from DVM, and instantaneously displays the plots of output voltage (mV) vs. input power (dBm) and log(mV) vs. input power (dBm). Then, both output voltage in mV and log(mV), and input power (dBm) are stored in a citifile. The measurement system was set up is diagrammed in the figure 3.1. The second tutorial gives a guide to student to make up a program which will read citifile created from the first program and display the stored data.

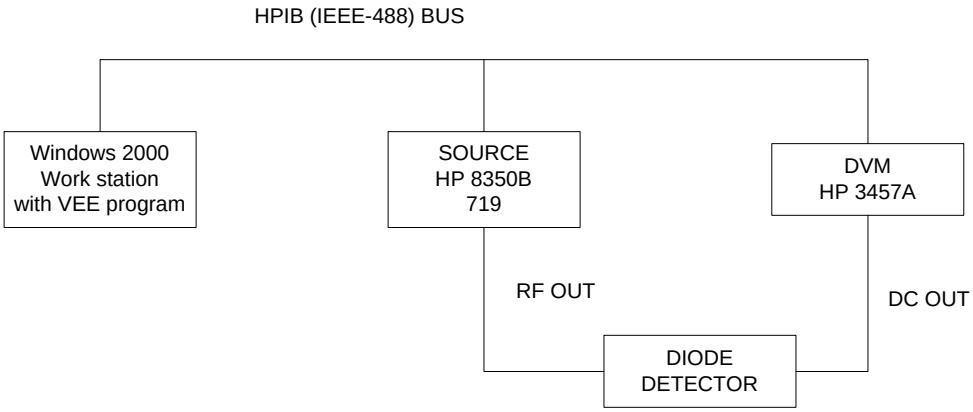


Figure 3.1: Automated Measurements of the Diode Detector

3.1.3 Improvements

Significant changes in the lab procedure have been made and are detailed as follows:

- The written programs are the flowcharts of the program design and thus by avoiding bulky text coding. Figure 3.2 and figure 3.3 are program flow charts and an actual program.
- Step by step tutorials on writing the first measurement VEE program are introduced and found to be helpful because this is the lab where students are

exposed to VEE environment. Programs required for next measurement labs are based on this lab tutorial. Only a few modifications have to be made in next labs.

- Data collected can be simultaneously plotted during the measurement. This is a big advantage of introducing of VEE for this purpose. Moreover, students have the privilege of using the built in objects for plotting. Program outputs a single text file which stores input power array, output voltage array and output logarithmic millivoltage array. Writing measurement data into a citifile is easy to implement with VEE as it shown in figure 3.4. A sample citifile is shown in figure 3.5. The output text files (citifile) are editable by a simple word editor and usable from other application software such as Microsoft Excel and Agilent ADS. Reading a citifile and displaying the contents can be found in the tutorial section of the updated lab instruction in appendix A. Previously, the data was first collected and then saved in arrays. After that, each array was stored as a file which is only readable from a program developed from HPBASIC. Lastly, a prewritten GRD_91 HPBASIC program retrieves the file, displayed and printed results as postscript files.
- User-friendly input and output boxes in the panel, which is displayed in the figure 3.6, has replaced traditional text prompting commands.
- Students write their own second program, which is shown in figure 3.7 will read the citifile created from the first program in order to analyze the data. An output voltage at a certain power level can be easily retrievable. This was a challenging task with HPBASIC. The slope of the detector response curve can be calculated

by the use of markers as seen in figure 3.9 (b). The second program teaches students the file retrieval process.

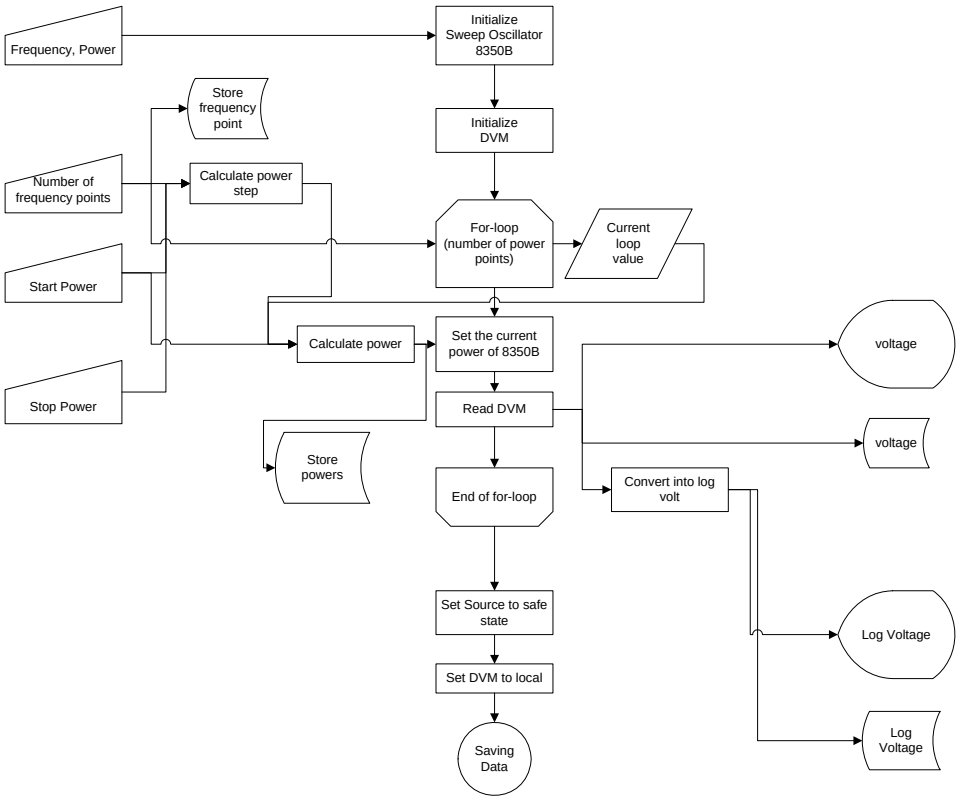


Figure 3.2: Suggested Program Flow Chart

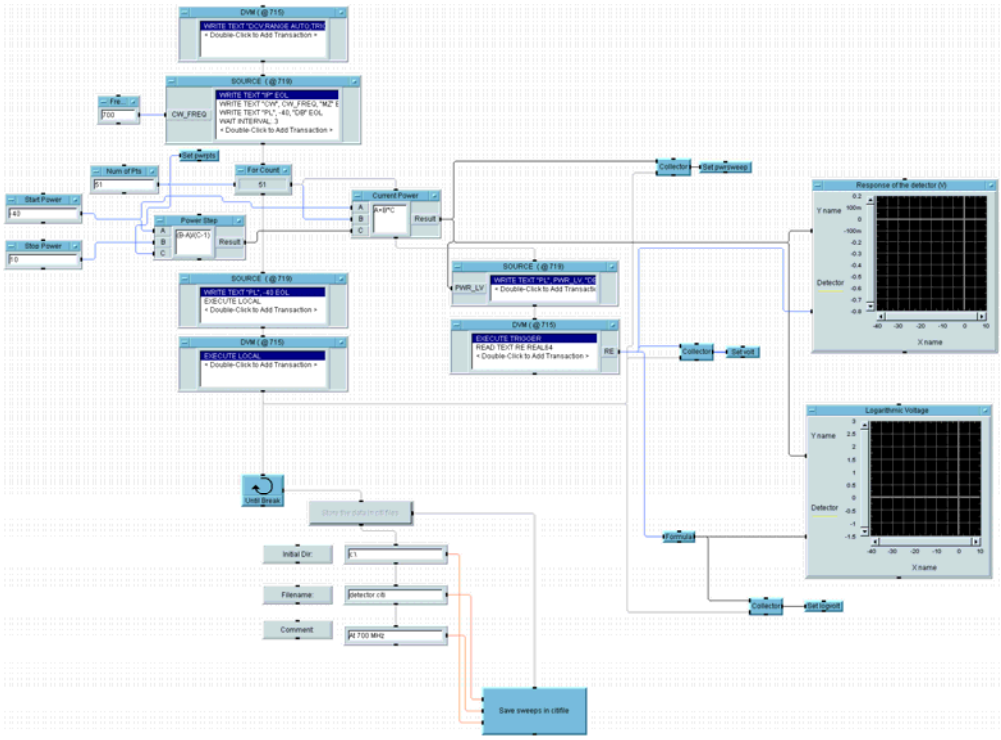


Figure 3.3: A VEE program based on the program flow chart

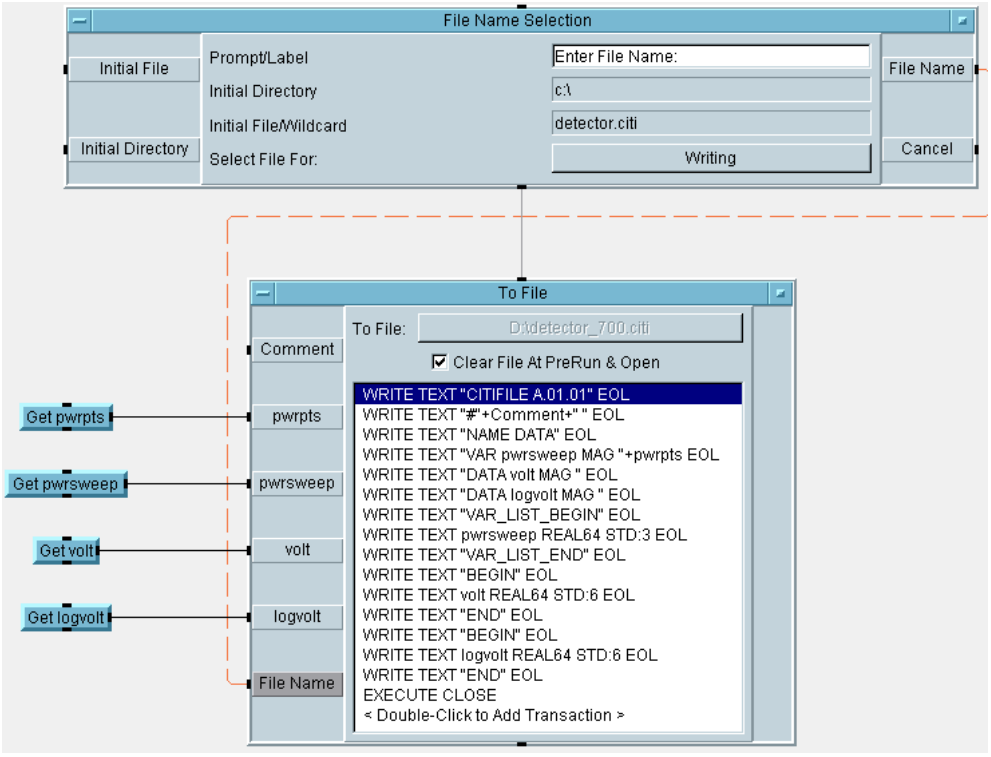


Figure 3.4: Writing data into a citifile in VEE

citifile_format - Notepad

File Edit Format Help

```
CITIFILE A.01.01
#At 650 MHz
NAME DATA
VAR pwrsweep MAG 51
DATA volt MAG
DATA logvolt MAG
VAR_LIST_BEGIN
-40
-39
.
.
.
9
10
VAR_LIST_END
BEGIN
-7.438E-005
-8.598E-005
.
.
-0.641046
-0.7237
END
BEGIN
-1.12854
-1.0656
.
.
2.80689
2.85956
END
```

Figure 3.5: A citifile containing measurement results

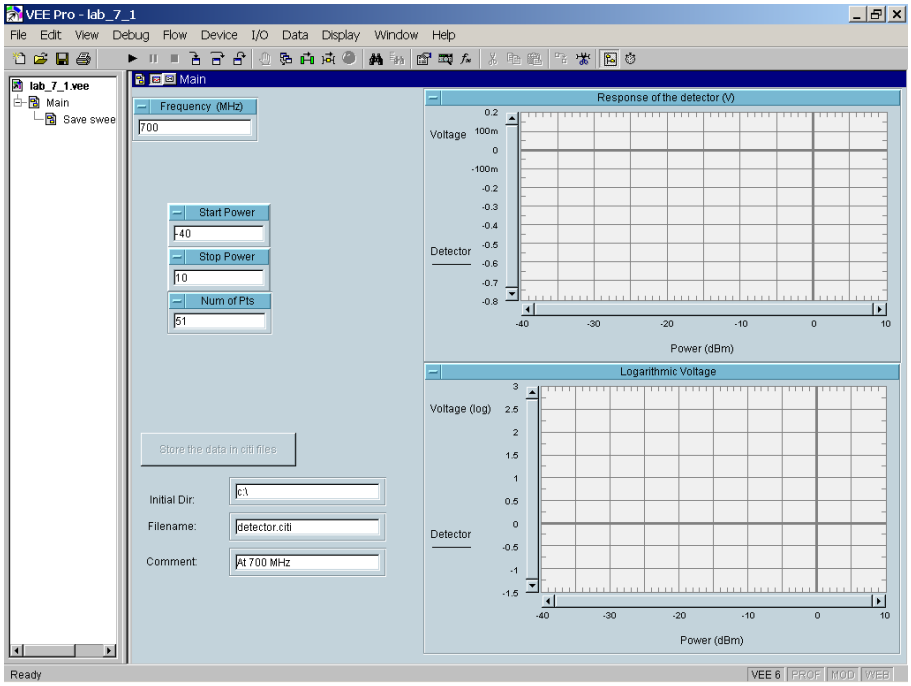


Figure 3.6: The panel of the measurement VEE program

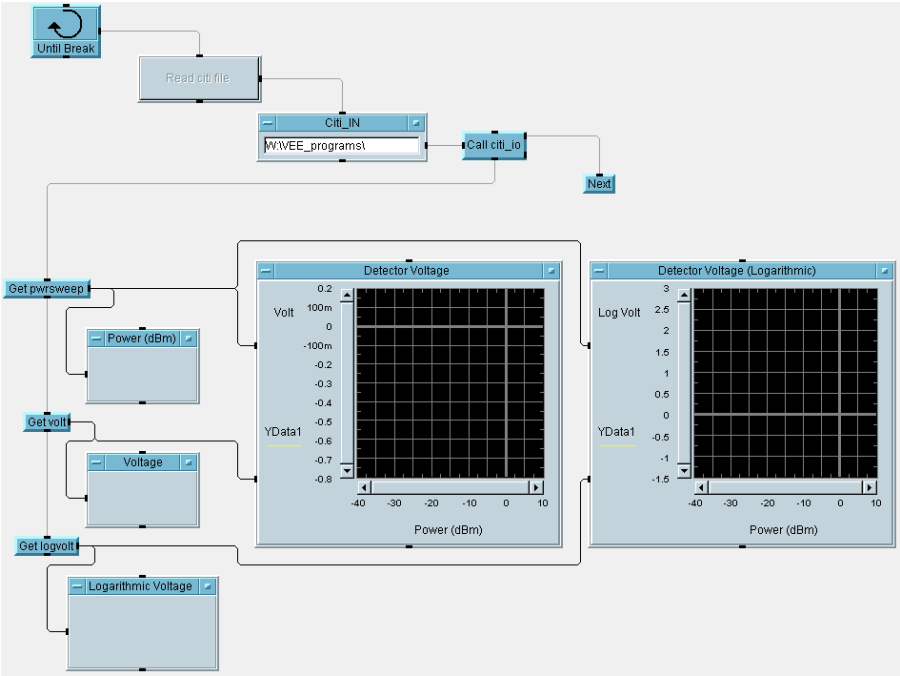


Figure 3.7: A VEE program to retrieve the citifile

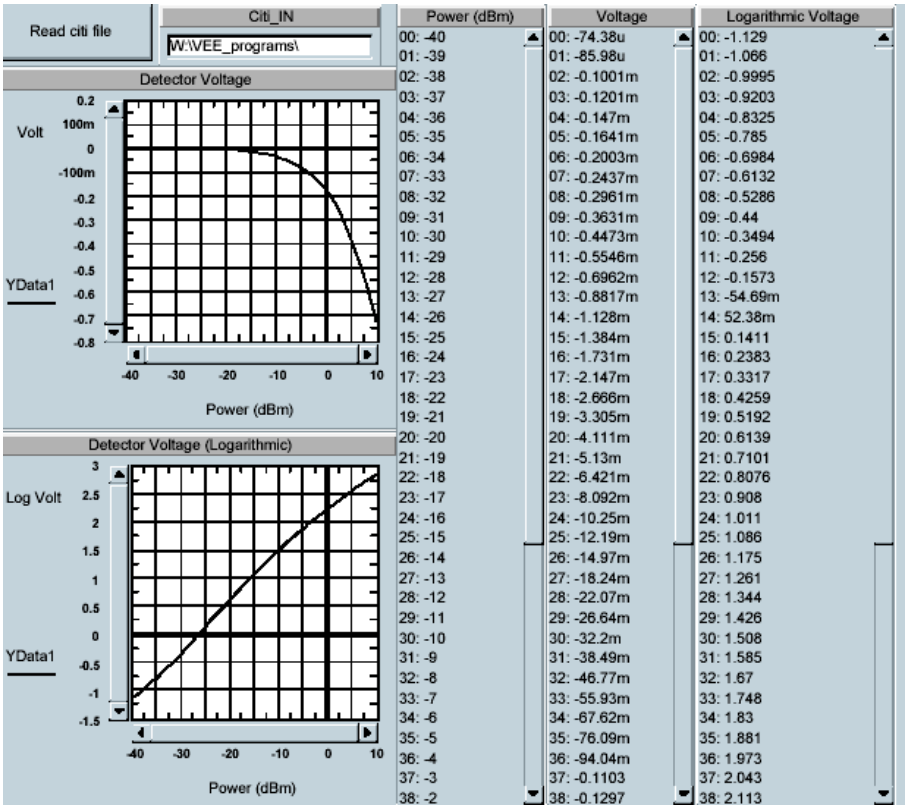
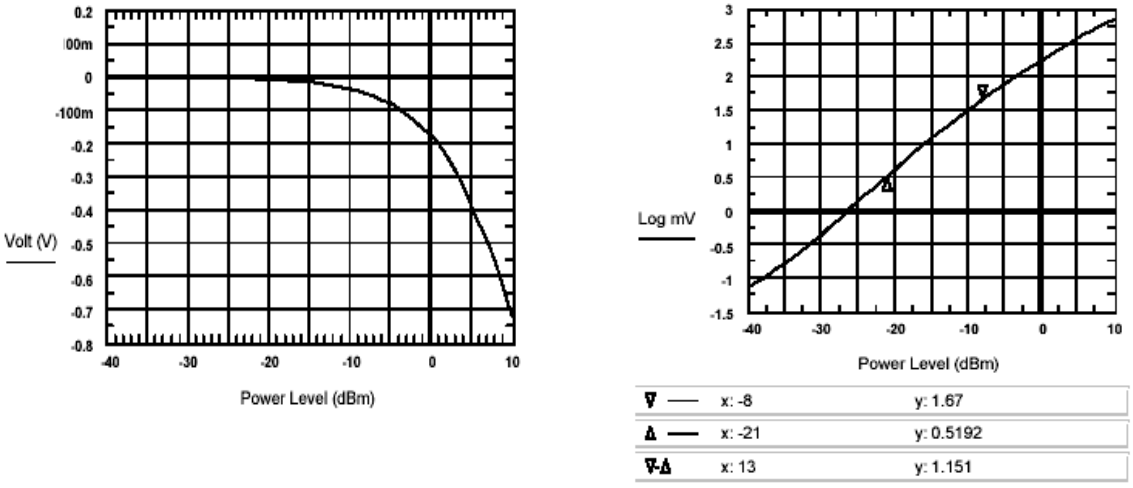


Figure 3.8: A front panel of data retrievable program

3.1.4 Measurement Plot Analysis

The main objective of the automated detector measurements lab is to see the linear response of the voltage to the input power. This was fulfilled by the use of two VEE programs. Referring to figure 3.9 (b), the slope of detector response can be calculated with the help of markers available in VEE program. From the survey conducted during the normal lab session, all students finished writing programs in 3-hour lab time.



(a) Normal output voltage (b) Logarithmic output voltage

Figure 3.9: Detector response. (a) Normal output voltage. (b) Logarithmic output voltage.

3.2 Automated Scalar Reflectometer Measurements

3.2.1 Introduction

Reflectometers are used to perform simple scalar error corrections. A reflectometer is a circuit that uses a directional coupler to isolate and sample the incident and reflected powers from a mismatched load. It is the major part of a scalar and vector network analyzer, as it can be used to measure the reflection coefficient of a one-port network and, in more general configuration, the S parameter of a two-port network. In operation it provides a sample of the incident wave, and a sample of reflected wave. Using the linear detectors at port 3 and port 4, these waves can be converted into DC voltages that are then monitored with voltmeters. Due to the non-ideal behavior of realistic directional couplers, they have a finite directivity which means errors are introduced in incident and reflected voltages. The other errors are due to the source load interaction and coupler variations. The goal of this lab to measure the incident and reflected voltages of standard devices short, open, a student unknown and a shorted, 15 foot length of RG/8U coaxial cable. Errors from the measurement data of the student unknown and a shorted, 15-foot coaxial cable are then removed with the measurement data of short and open standards. The effect of the source-load interaction can be removed by taking the ratio of the reflected power and incident power. In this experiment, the first measurement program takes care of this job. Then, the error contributed by coupler variations in the student unknown measurement is removed by using the power ratio of standards and the power ratio of devices that need to be corrected. The second program is designed for this

purpose. The next part of the lab is to measure the incident voltage of a shorted cable. This voltage will be used to find the VSWR of the microwave source.

3.2.2 Experiment description

The detail lab instruction is provided in the appendix B. In brief, student will write three VEE programs. The measurement setup is diagrammed in figure 3.10. The first program is to make scalar reflectometer measurements to retrieve |S11| in dB over the 300 MHz to 1300 MHz frequency range every 5 MHz (201 points) and keep the power level constant at 10 dBm. The program flow chart is diagrammed as in figure 3.11. The actual VEE program can be seen in figure 3.12. The front panel of the program is displayed in figure 3.13. The program commands HP8350B source to sweep the frequency and two HP3457A digital voltmeters to read the incident and reflected voltages which are linearly dependent on sample powers taken from HP 778D 20dB-coupler. Knowing that the detector response function has a slope of 0.1, the voltages detected can be easily converted into powers, and can be displayed from the VEE program.

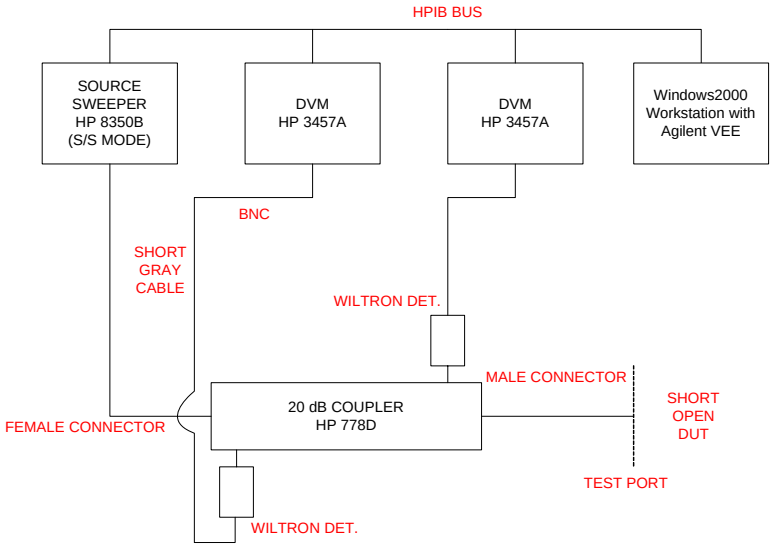


Figure 3.10: Experiment setup for the measurement program

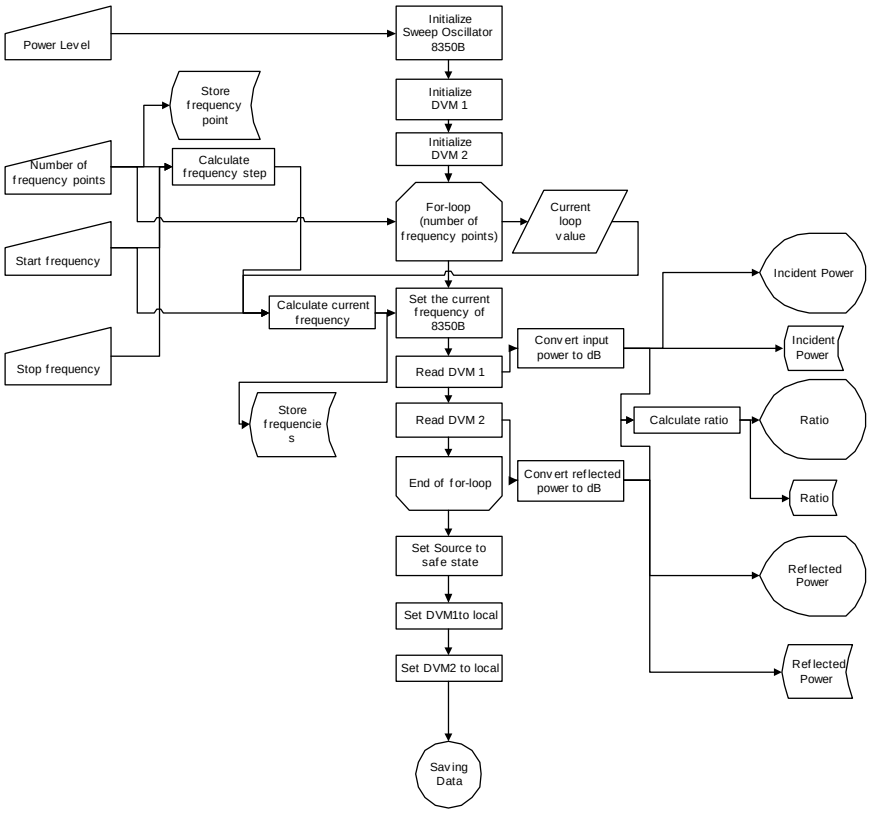


Figure 3.11: Program flow chart of the measurement program

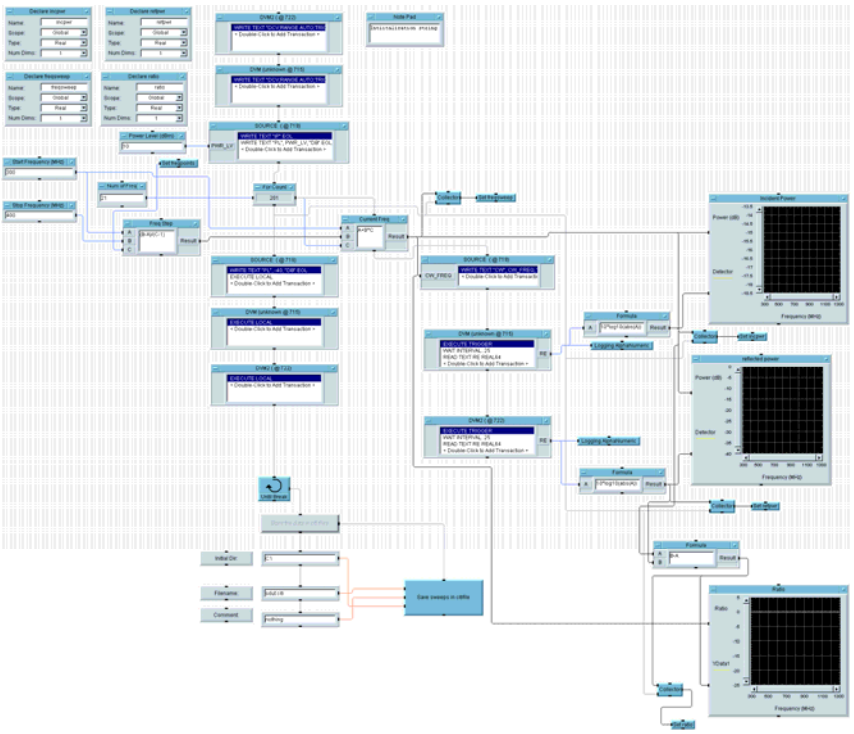


Figure 3.12: The measurement VEE program

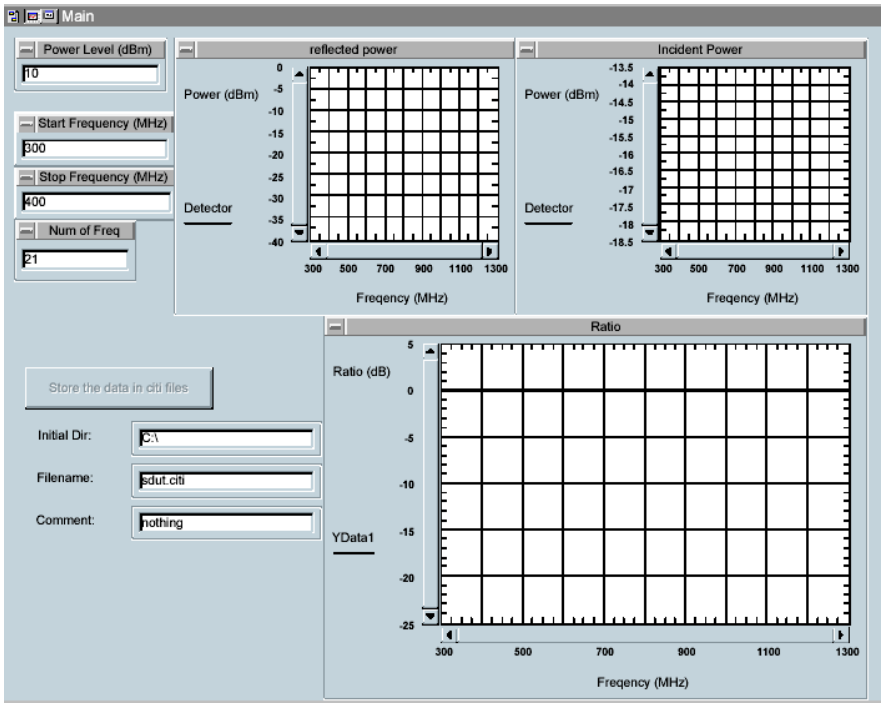


Figure 3.13: The front panel of measurement program

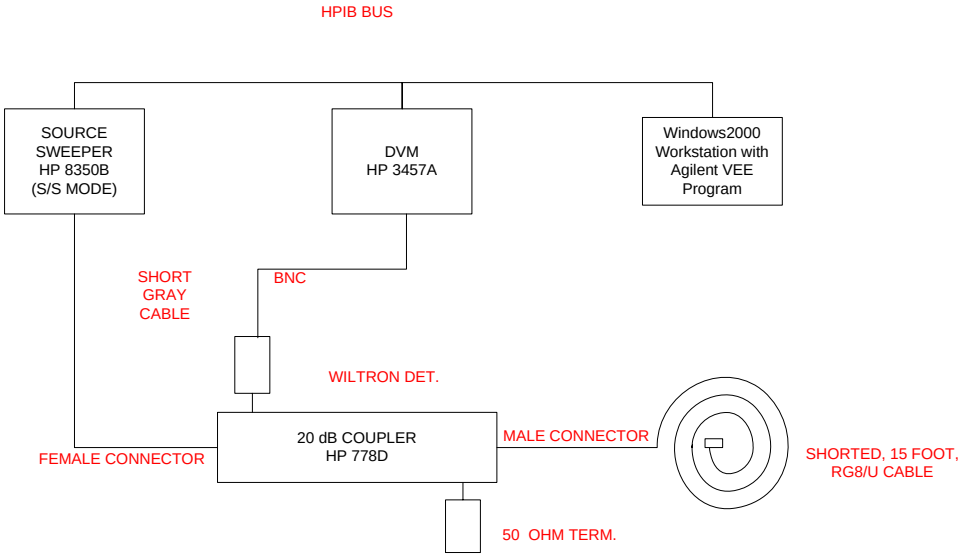


Figure 3.14 Experiment setup to measure the incident voltage of 15-foot shorted cable

The measurement program outputs four different files for short, open, student unknown and 15-foot cable. The second program will read three citifiles to correct errors. The structure of the program can be observed in figure 3.15. The front panel can be designed to appear as in figure 3.16. The last part of the lab is to modify the measurement program to measure the incident voltage of the 15-foot cable to calculate the VSWR of the microwave source. The modified measurement program is shown in figure 3.17. The front program panel can be designed as in figure 3.14.

3.2.3 Improvements

- Students are able to constantly observe the plots of the incident and reflected powers, and the ratio of both powers on the panel display while performing the measurement as in figure 3.10. In the past, these plots can only be viewed with the help of GRD_91 developed with HPBASIC only after the measurement program stops running. Besides, students are allowed to see the effects of coupler variations and source-load interaction from the incident power plots which were not observable previously.
- The number of output files reduces from 8 to 4 as a result of using citifile format. However, each array of data was used to be stored as an HPBASIC file. In the new lab manual, each citifile represents each device measurement data that contains 4 arrays of data namely —frequency, incident power, reflected power and the ratio of incident and reflected power.

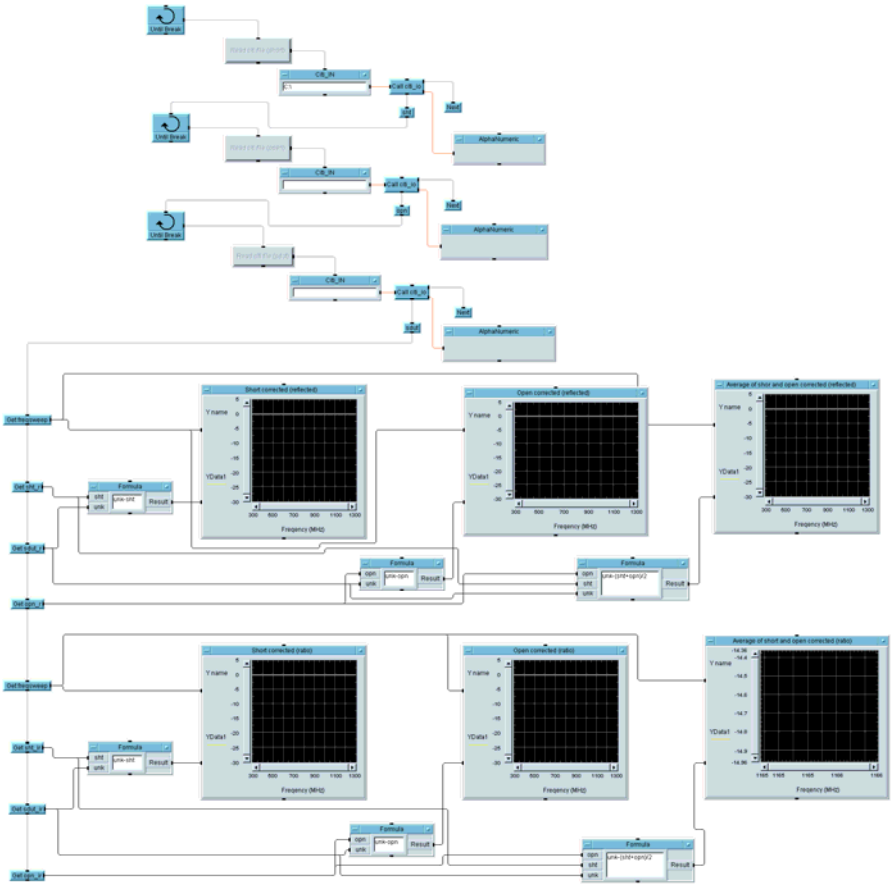


Figure 3.15: The error correction program

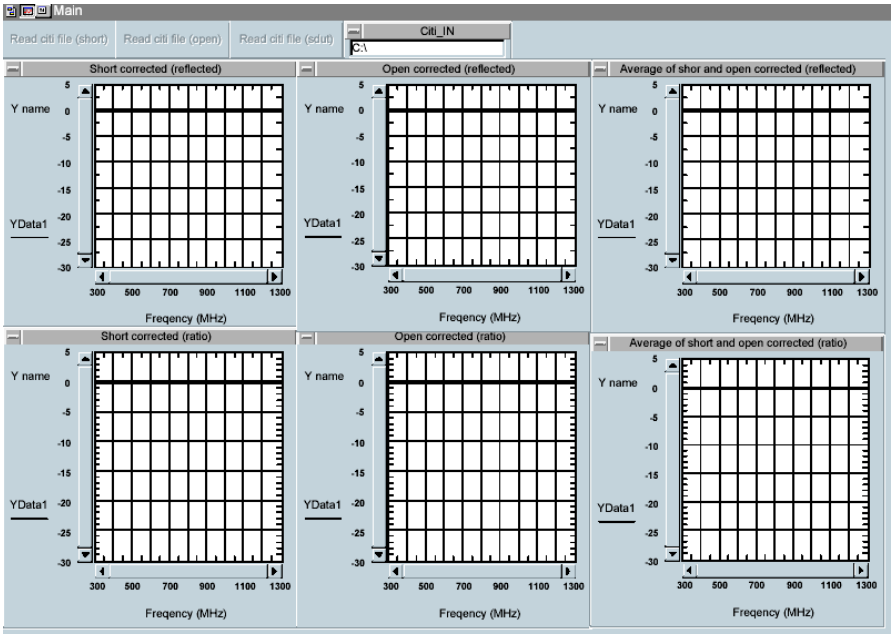


Figure 3.16: The front panel of scalar error correction program

- The second VEE program is designed to run once for a device under test. All of the 6 types of error correction are performed in a single run. Once the program reads all necessary citifile, all error corrected plots are displayed as in figure 3. It saves the work of creating the output files that will be read from another HPBASIC plot program to create postscript files. On the other hand, each run of the old HPBASIC was originally designed to carry out a specific type of error correction and output files were created to allow students to view the plots. The new VEE program design also reduces the workload when repetitive run is necessary.

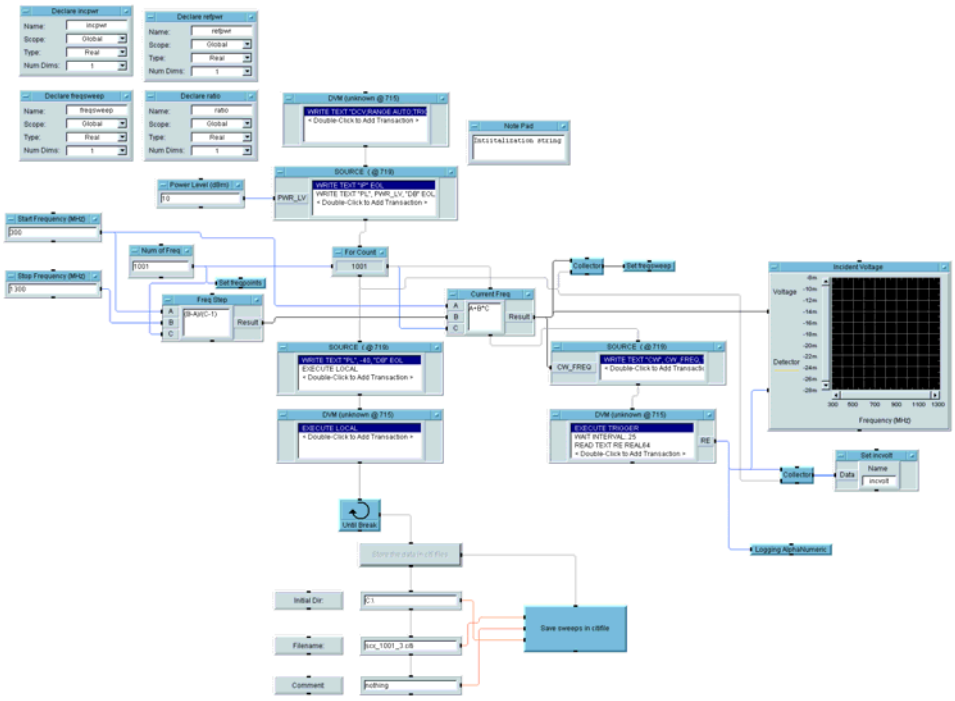


Figure 3.17: The modified version of the measurement program

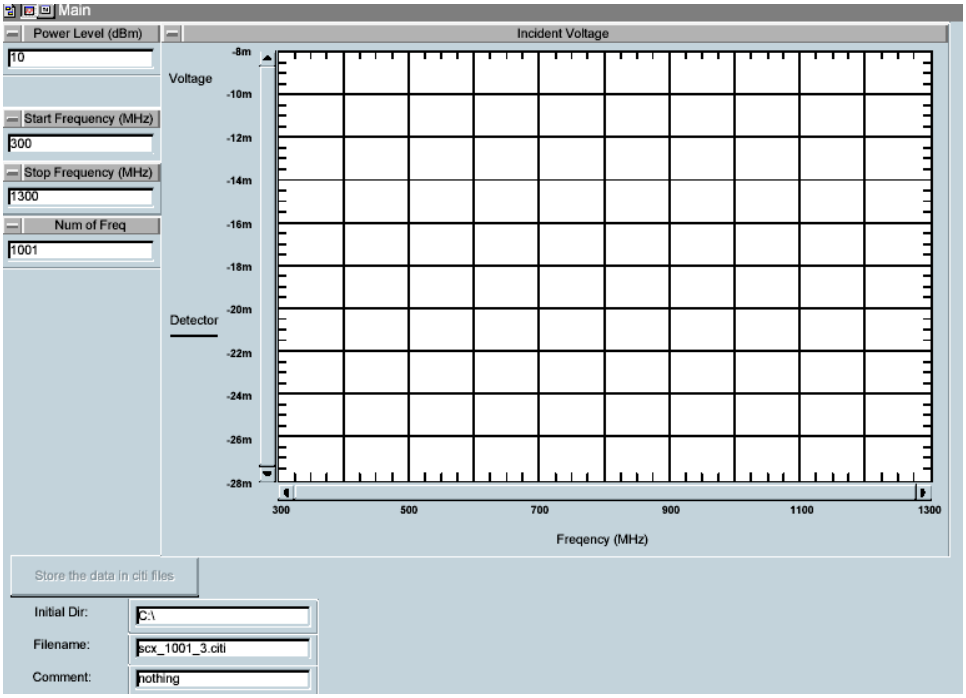


Figure 3.18: The front panel of a VEE program which measures the incident voltage of a shorted cable

3.2.4 Measurement Plot Analysis

All the plots obtained from VEE programs are presented. From figure 3.17 to figure 3.20, each depicts the measured incident power, reflected power and ratio of incident and reflected powers for each device. Power ratio plots clearly show source-load interaction correction. From figure 3.21 to figure 3.22, each shows the different error corrections required from the lab manual. Figure 3.23 allows us to see the incident voltage of the shorted 15-foot coaxial cable vary with frequency while measurement is being performed. An incident voltage can be read from the plot with the help of built in maker function since the minimum and maximum voltages are included in the calculation VSWR of the source.

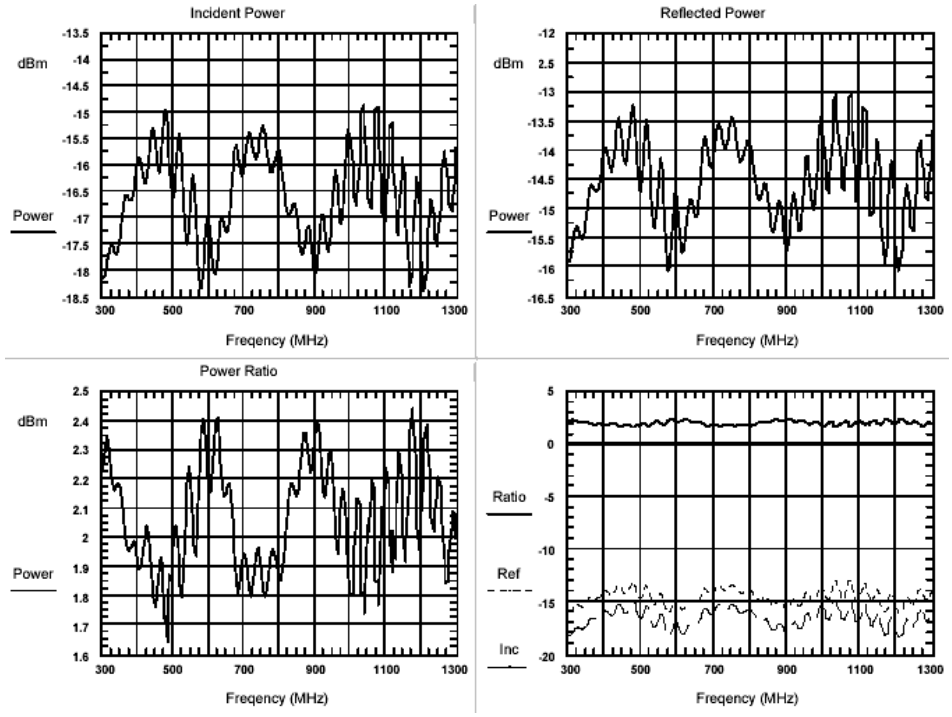


Figure 3.19: Short measurement screenshot.

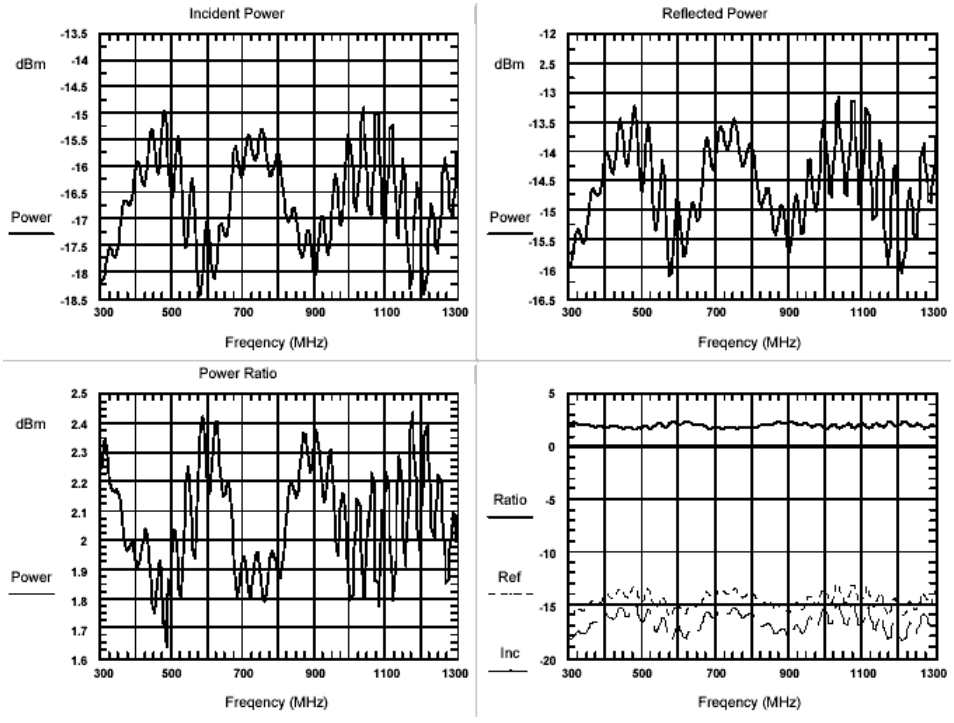


Figure 3.20: Open measurement screenshot.

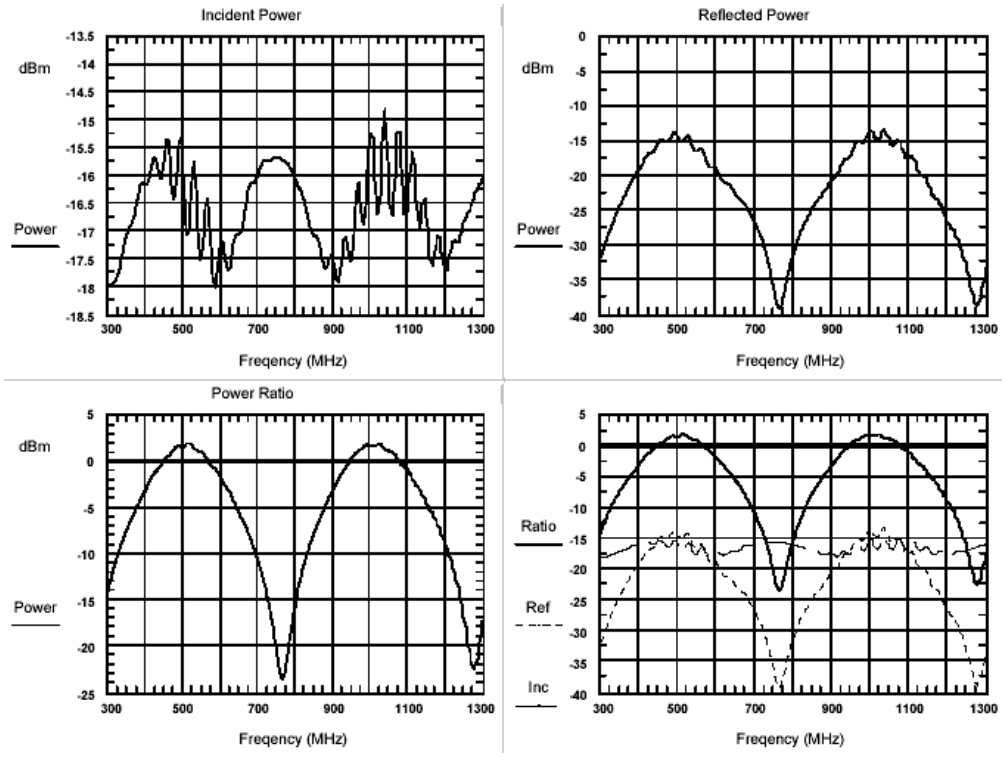


Figure 3.21: Student Unknown measurement screenshot.

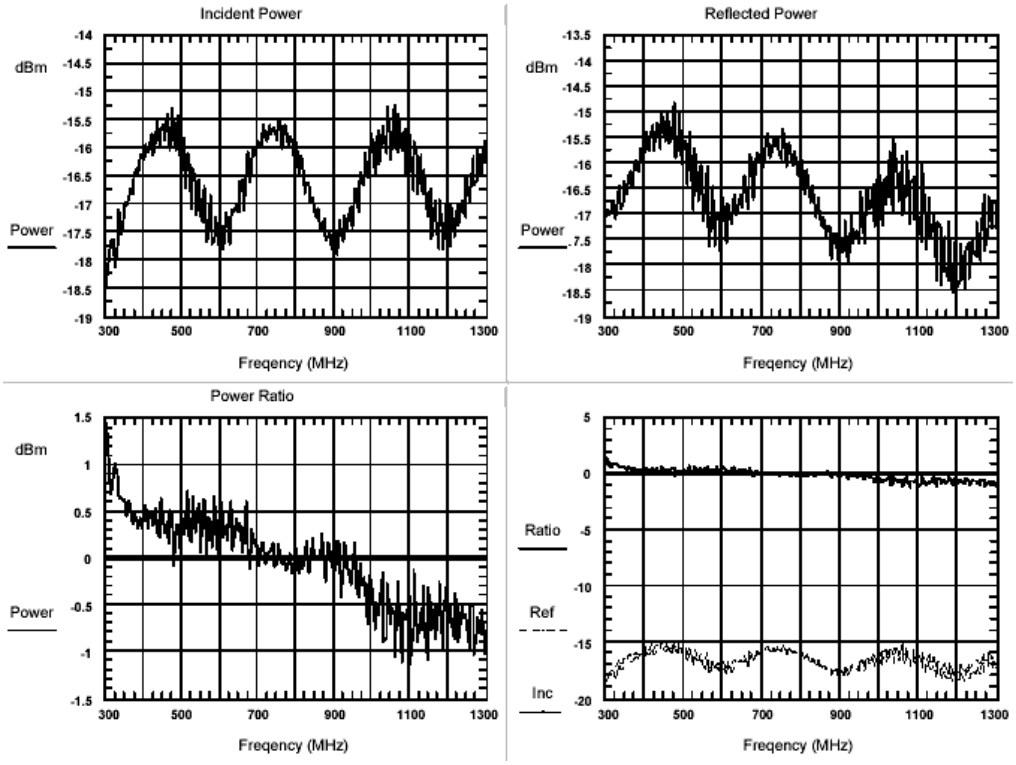


Figure 3.22: Shorted 15-foot cable measurement screenshot.

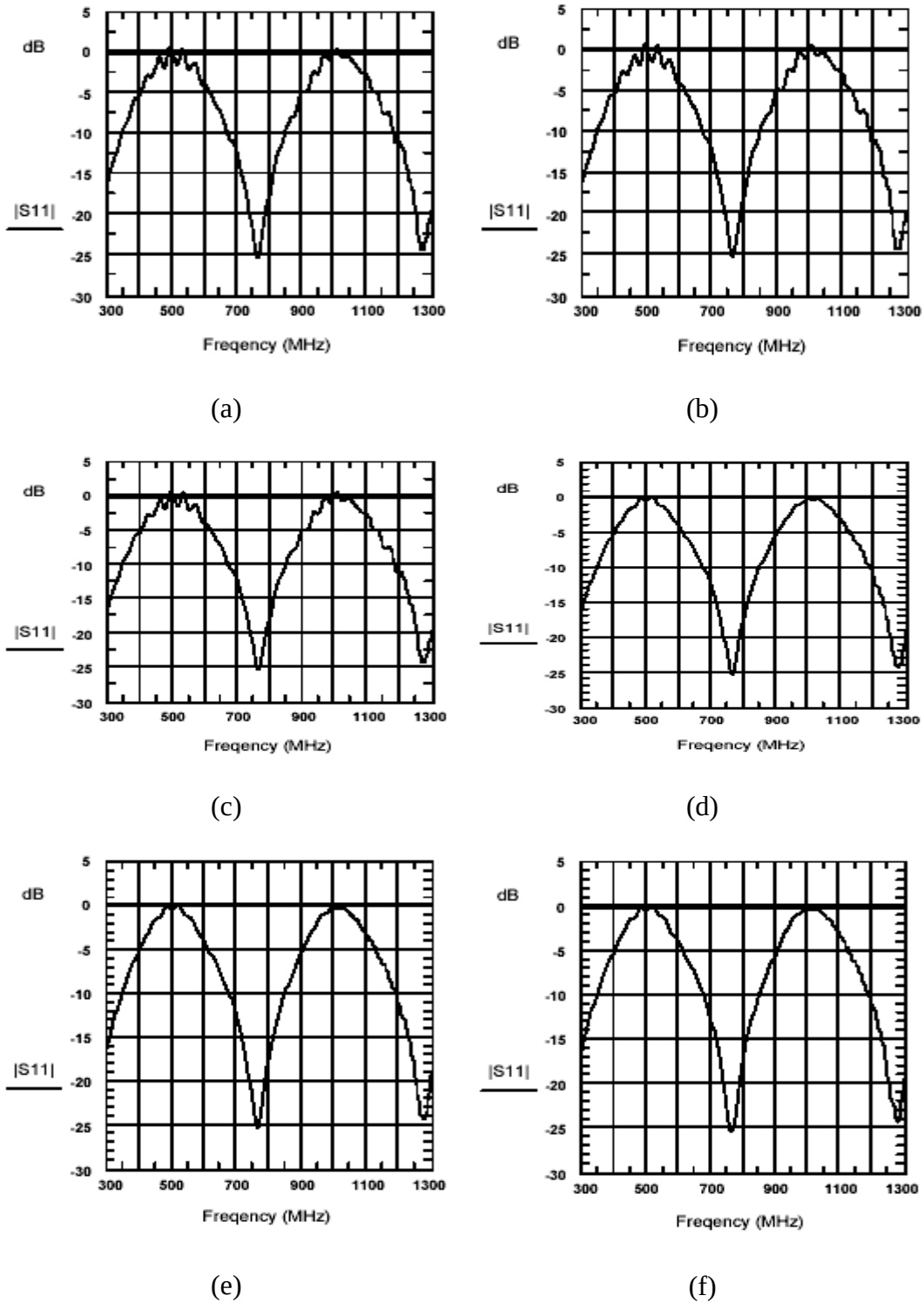


Figure 3.23: Corrected return loss plots of student unknown: (a) Short corrected. (b) Open corrected. (c) Average corrected. (d) Ratio short corrected. (e) Ratio open corrected. (f) Ratio average corrected.

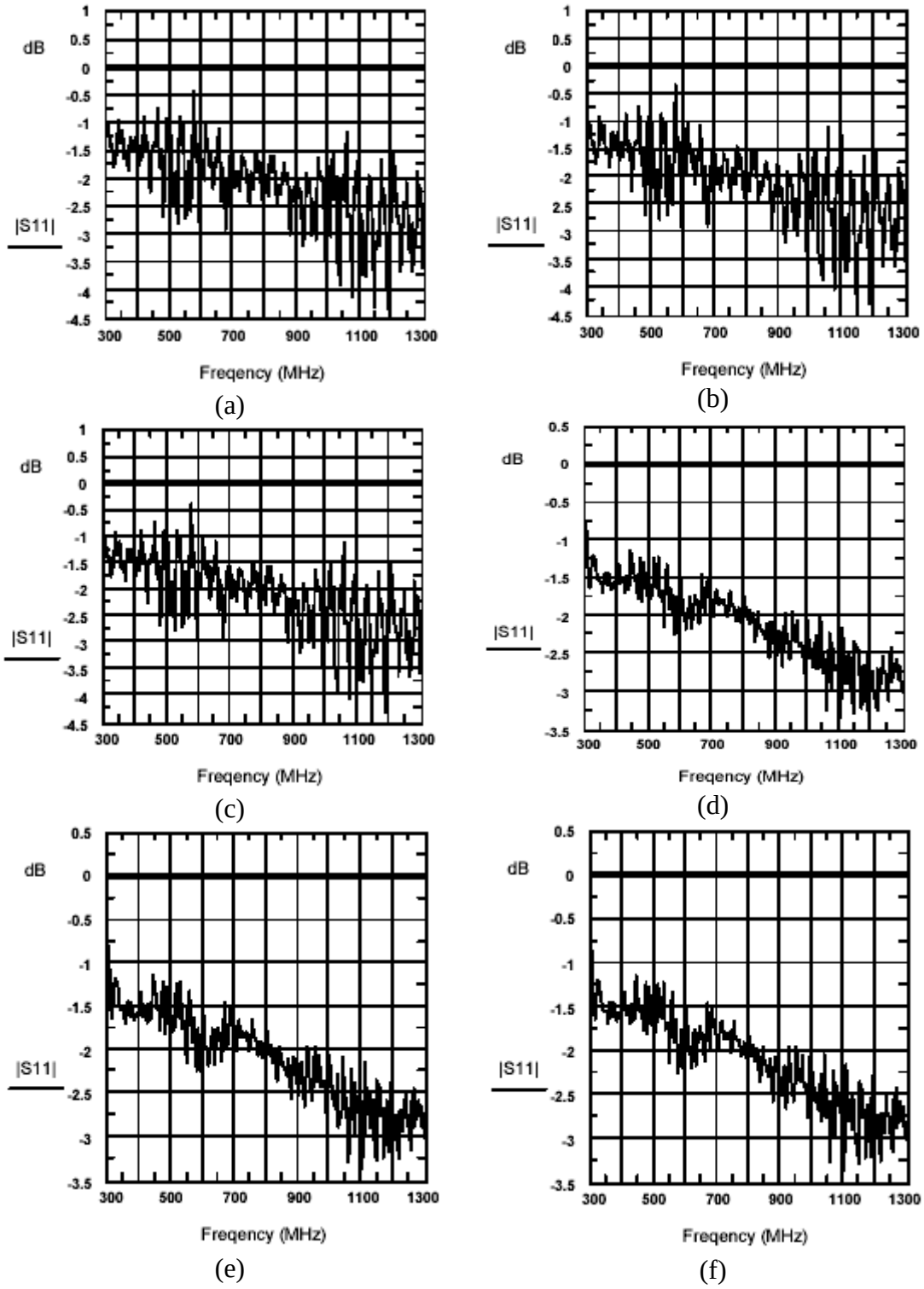


Figure 3.24: Corrected return loss plots of 15-foot cable: (a) Short corrected. (b) Open corrected. (c) Average corrected. (d) Ratio short corrected. (e) Ratio open corrected. (f) Ratio average corrected.

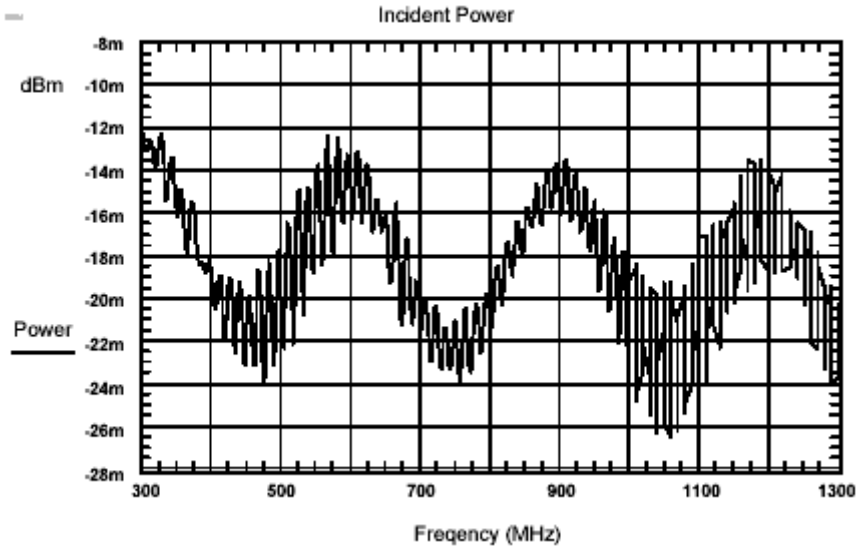


Figure 3.25: Incident voltage of a shorted 15-foot coaxial cable.

3.3 Automated Network Analyzer Error Corrections

3.3.1 Introduction

Error corrections in a network analyzer is called calibration which is a process consists of measuring a number of known calibration standards. A model of the characteristics of the microwave hardware in the measurement is called an error model or error adapter based on the resulting data. Then, the network analyzer corrects measurements by removing the systematic measurements errors. This lab introduces the one port error correction. Four types of one port error correction have been discussed in the lab. It is required to measure standard devices such as a short, an open and a match termination and devices under test such as a student unknown, 2.5 VSWR load and 5.0 VSWR load. The goal is that the first VEE program to measure uncorrected S11 of all devices and store them in citifiles, and the second program to make error corrections and store the corrected result in citifiles.

3.3.2 Experiment Description

An updated lab instruction is provided in the appendix C. In brief, a VEE program is written to communicate with a HP8510C network analyzer to measure a device's S11, which is then stored as a citifile. The measurement setup is diagrammed in figure 3.26. A 15-foot coaxial cable is inserted between the test port and the device to accentuate systematic errors. A program flowchart, an actual program, and front panel are shown in figure 3.27, 3.28 and 3.29 correspondingly. A second error correction program is designed to carry out 4 different types of error corrections for a 1-port device under test in a single run. This program produces a citifile which stores four different error term

correction types of corrected S11 of the device under test. Figure 3.30 shows its front panel. Four |S11| dB vs. frequency plots representing four different error term models were included in the second VEE program for a quick check for the program correctness. The program structure is depicted in figure 3.31. A student lab report requires |S11| in dB, Phase of S11 and S11 on Smith Chart for each error term model of a device under test. This can be accomplished by creating a third VEE program or using ADS. A short tutorial for retrieving citifiles and displaying the dataset in terms of plots in ADS is provided with the lab instruction. By this far, a student should be able to extract and plot the data from citifile with a VEE program.

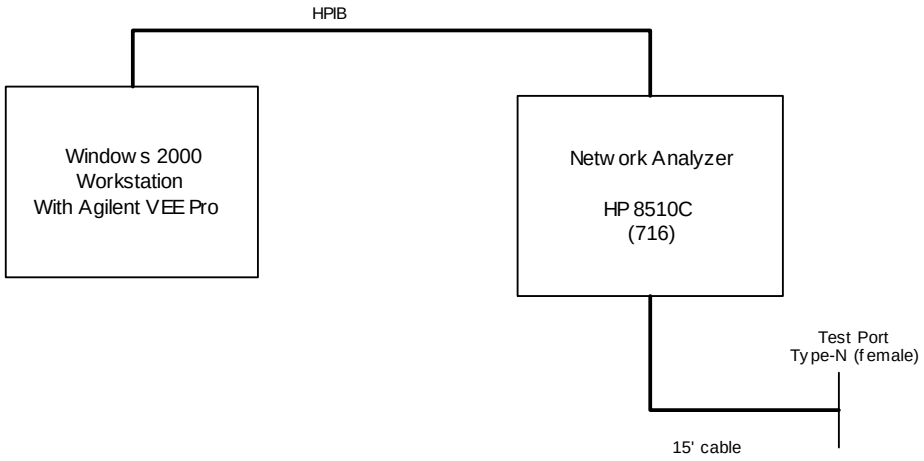


Figure 3.26: Network analyzer and workstation

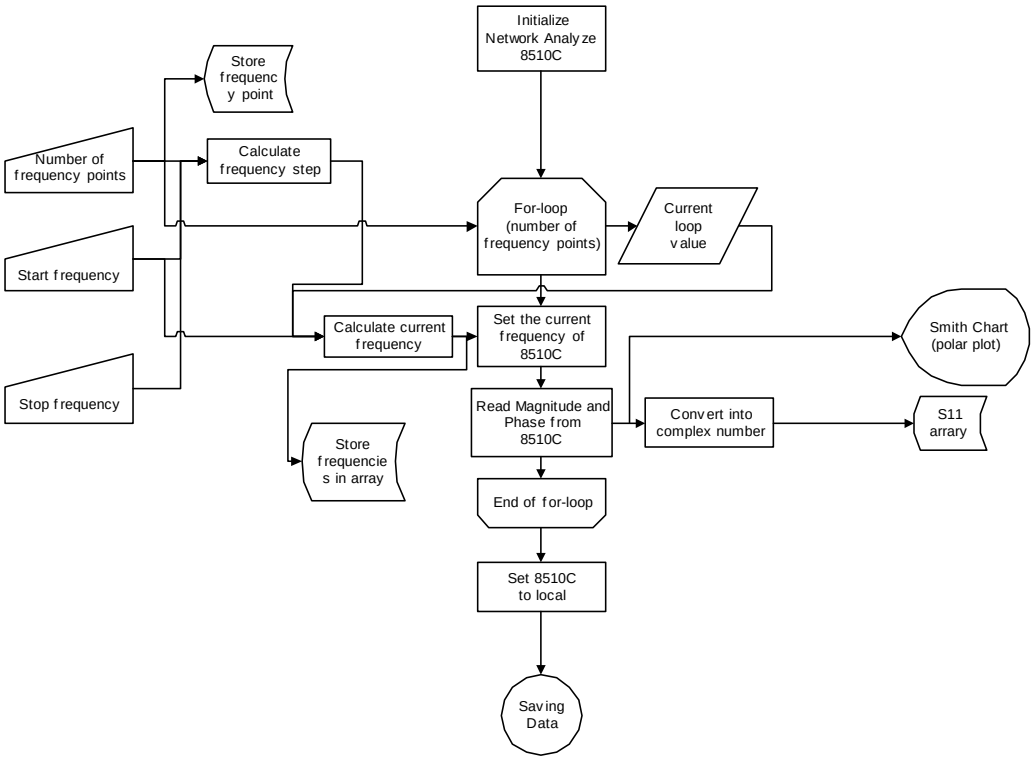


Figure 3.27: Program flow chart of the measurement program.

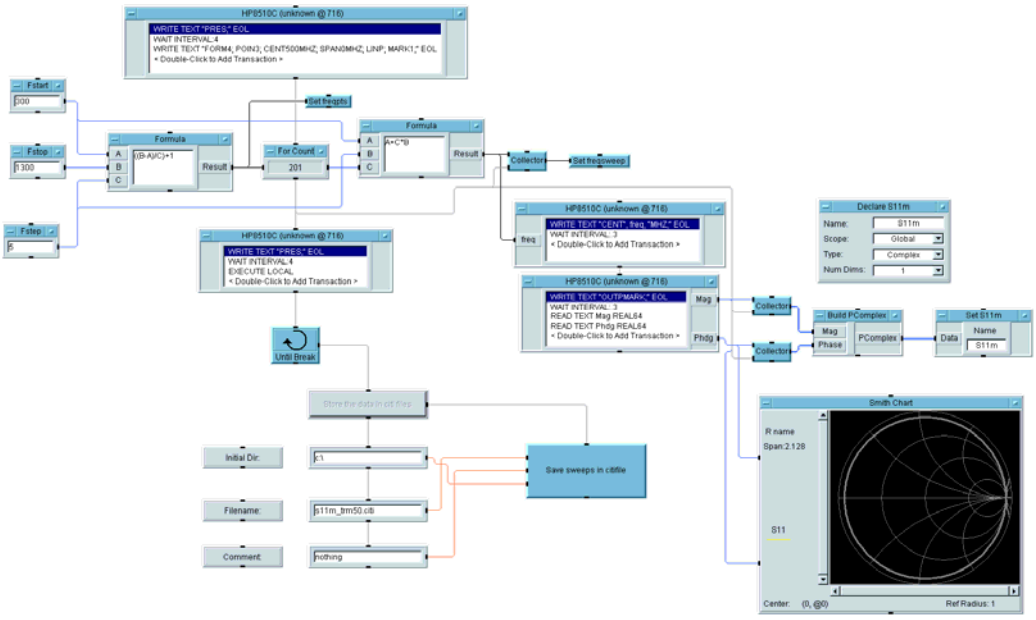


Figure 3.28: The measurement VEE program.

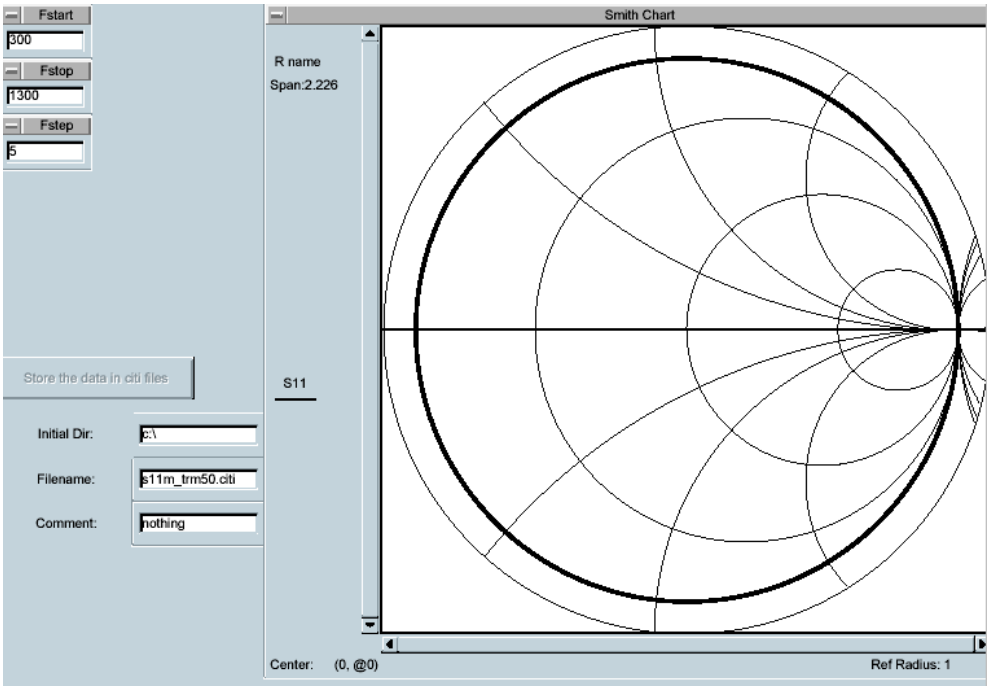


Figure 3.29: A front panel of measurement program

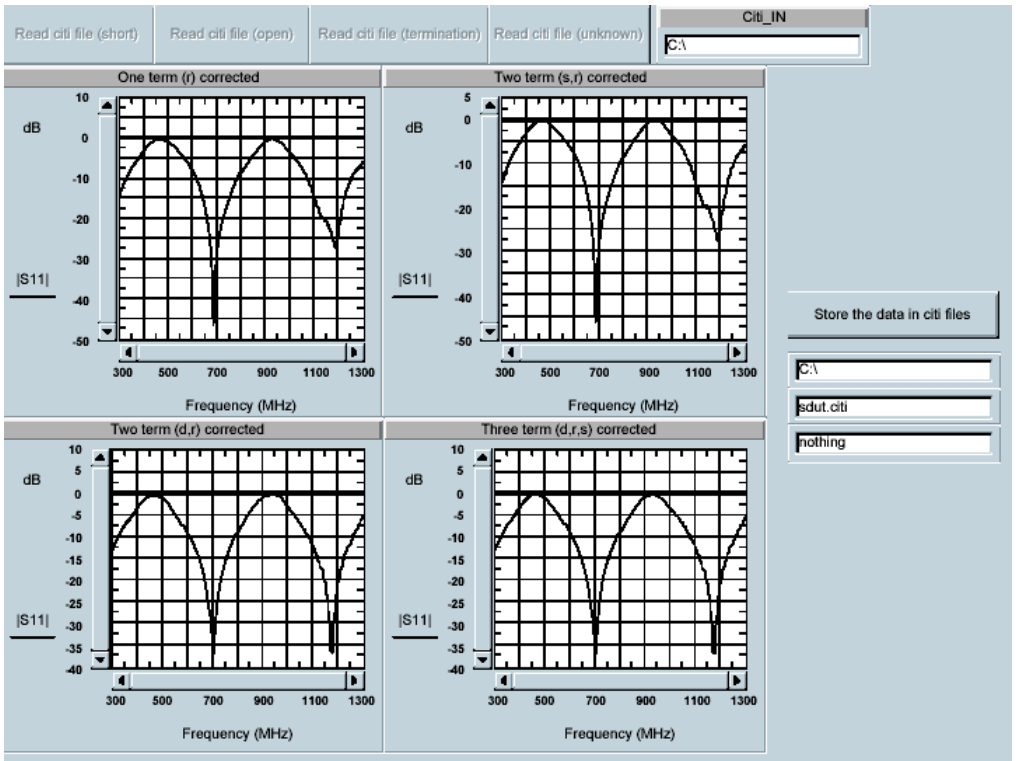


Figure 3.30: A front panel of error correction program

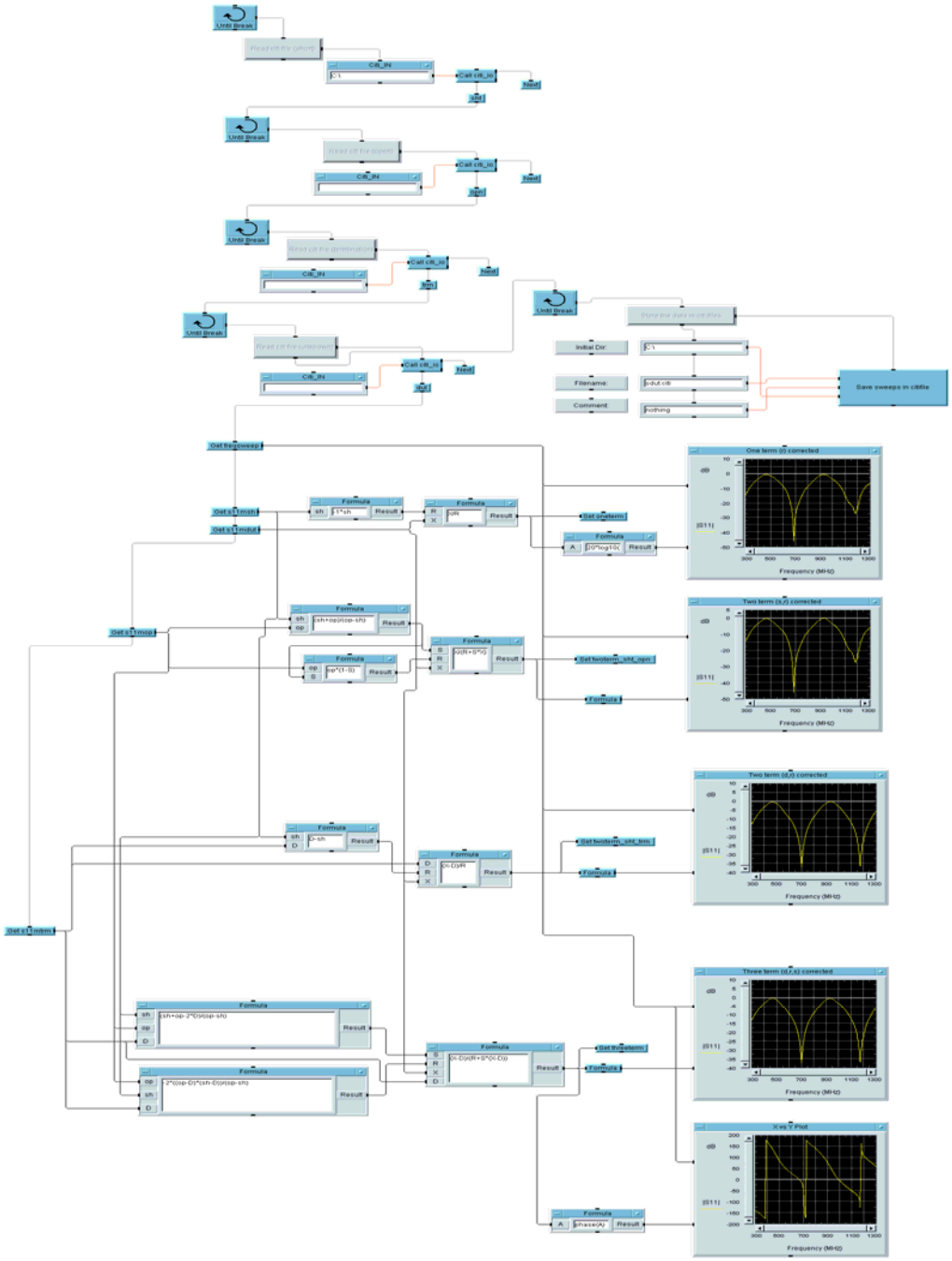


Figure 3.31: Vector error correction VEE program.

3.3.3 Improvements

- Students are able to monitor the Smith chart as it seen on the Network Analyzer while running the measurement program. This feature was not easy to implement with the program written with HPBASIC.
- When compared with the HPBASIC code for communication with HP8510C network analyzer, the implementation of measurement program becomes simple as it shown in figure 3.27 and 3.28.
- VEE error correction program will output a single citifile instead of 4 different files as required with HPBASIC. The program correctness can easily be checked with sample plots. For instance, |S11| in dB of any error term model correction as it displayed in figure 3.30 and 3.31. This will save tedious workload due to syntax errors frequently occurred in an HPBASIC program.
- Students are exposed with ADS design software when reporting the error corrected plots. It is obvious from the tutorial and from the figure 3.32, ADS proves that data acquisition and presentation them is a simple task.

3.3.4 Measurement Plot Analysis

A total of 12 plots represents four different error term models for a device under test as it shown in figure 3.32. Similarly, a 2.5 VSWR and 5.0 VSWR required the same error correction procedure. However, this will not be included in this section. Student can simply analyze the magnitude in dB, phase and complex plot of S11 for each error term correction from ADS data display environment.

CHAPTER 4

CONCLUSION

In the thesis, background on the VEE programming and its structure have been thoroughly discussed. Then, the improvements in lab procedures and the advantages of using VEE over HPBASIC have been clearly demonstrated with the aid of figures and plots. It is observed that the introduction of Agilent VEE programming enhances the automated microwave measurements laboratory. Giving lab instruction becomes simpler and learning environment becomes lively. The lab procedure becomes more concise and understandable. Students enrolled in this course in spring 2002 semester have successfully conducted three automated measurements experiments described in this thesis within the given time frame and they became familiar with VEE development in their second experiment. Students are able to easily capture the objective and requirement of the lab while performing the experiments because a textual programming language could barrier to fulfil the main objective the experiment. Possibility of improving the experiment with the aid of VEE programming is endless. It is suggested that the existing experiments will be constantly updating in the future to keep in touch with state of the art measurement technique.

REFERENCES

[1] R. Helsel and Hewlett-Packard, *Visual Programming for HP-VEE*. Englewood Cliffs, NJ: Prentice-Hall, 1998.

[2] E. C. Baroth, "A Review of HP VEE 4.0," EE-Evaluation Engineering, Nelson Publishing Inc, 1998, http://www.evaluationengineering.com/.

[3] Agilent VEE, Agilent Technologies, "Learn about VEE," April 17, 2002, http://www.get.agilent.com/gpinstruments/products/vee/support/.

[4] "Automated Microwave Measurements Laboratory," laboratory manual for ECE 351, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Spring 2002.

[5] P. W. Klock, *Automated Microwave Measurements: The Theory of Reflectometers*, University of Illinois at Urbana-Champaign, 2000.

[6] D. M. Pozar, *Microwave Engineering*, Reading, MA: Wiley & Sons, 1998.