COMPONENT HIGH FREQUENCY CHARACTERIZATION IN ARBITRARY SOURCE AND LOAD IMPEDANCES

by

Michael W. Allender

Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master of Science in Engineering

in the

Electrical Engineering

Program

YOUNGSTOWN STATE UNIVERSITY

June, 1995

COMPONENT HIGH FREQUENCY CHARACTERIZATION

IN ARBITRARY SOURCE AND LOAD IMPEDANCES

Michael W. Allender

I hereby release this thesis to the public. I understand this thesis will be housed at the Circulation Desk of the University library and will be available for public access. I also authorize the University or other individuals to make copies of this thesis as needed for scholarly research.

Signature:

Michael W. Allender 5/22/95

Student Date

Approvals:

Salvatore R. Pansino 5/31/95

Thesis Advisor Date

5/22/95

Committee Member Date

Samuel J. Skarote 5/31/95

Committee Member Date

6/7/95

Dean of Graduate Studies Date

ABSTRACT

The purpose of this thesis is to develop a technique that takes ordinary S-parameters, measured in a standard 50 Ω characteristic impedance test environment, and mathematically produce a new set of S-parameters based on any arbitrary source and load impedance. A review of S-parameter theory is first presented followed by a brief discussion on today's network analyzer which is the most common test instrument used to measure S-parameters. The mathematical algorithms to convert 50 Ω S-parameters to arbitrary impedance S-parameters are then derived and incorporated into a computer program written by the author. This program, called CONVERTZ, automatically downloads 50 Ω S-parameters from a network analyzer, applies user-defined source and load impedances, and uploads the new S-parameters to the network analyzer's memory. The various advantages and features of CONVERTZ are then discussed followed by two sample applications for the program. The first deals with the design and measurement of a resistive attenuator and how CONVERTZ can be used as a tool for performance verification. The second is a thorough study regarding the relationship that exists between a device characterized by CONVERTZ and the total amount of energy that can be radiated from a system containing this same device.

ACKNOWLEDGMENTS

I would like to express my sincerest thanks to Jonathan Gallo and Mussie Pietros for their many hours of effort and technical support throughout the creation of this thesis. I would also like to thank Kin Moy, my supervisor at Delphi Packard, for permitting me access to the test equipment necessary for the experimentation contained in this thesis as well as for being on my thesis committee. A special thanks also goes to my thesis advisor, Dr. Salvatore Pansino, for all his advice and patience, as well as to Professor Samuel Skarote for being part of my committee. Finally, I would like to thank my entire family and especially my wife, Judy, for all the sacrifice, encouragement and support that motivated me to complete this work.

TABLE OF CONTENTS

PAGE

LIST OF SYMBOLS

SYMBOL	DEFINITION
a	Incident power wave
$\hat{a}$, $\hat{b}$, $\hat{V}$, $\hat{I}$	Column vector representation of a, b, V, and I
b	Reflected power wave
C, **D**	Diagonal matrices contained in power wave definitions
$\mathbf{C}_{new}$, $\mathbf{D}_{new}$	**C** and **D** matrix with arbitrary source and load impedance applied
$\overline{\mathbf{D}}$	Complex conjugate of **D** matrix
DUT	Device Under Test
I	Current flowing in or out of a network port
I	Identity matrix
RE	Radiated Emissions
Re Z	Real part of the complex impedance variable, Z
s_{11}	Input reflection coefficient S-parameter
s_{12}	Reverse transmission S-parameter
s_{21}	Forward transmission S-parameter
s_{22}	Reverse reflection coefficient S-parameter
S	Scattering matrix
$\mathbf{S}_{new}$	Scattering matrix with arbitrary source and load impedance applied
V	Voltage across a network port
Z	Impedance looking out of a network port
Z	Impedance matrix
$\mathbf{Z}_{50\Omega}$	Impedance matrix derived from 50 Ω S-parameters

SYMBOL	DEFINITION
Z^*	Complex conjugate of impedance variable, Z
Z_S	Source impedance seen by the DUT
Z_L	Load impedance seen by the DUT
Z_O	Characteristic impedance
ω	Angular frequency (radians/sec)

LIST OF FIGURES

FIGURE PAGE

FIGURE **PAGE**

FIGURE **PAGE**

CHAPTER I

INTRODUCTION

Today's proliferation of radio frequency (RF) devices are making component high frequency characterization more and more critical. With device operating frequencies becoming higher and higher, the most common characterization method of choice utilizes a parameter set called scattering or S-parameters. The reason for the widespread popularity of S-parameters is due to the relative ease in which they can be measured and worked with at RF and microwave frequencies as compared to other network parameters such as hybrid (H), admittance (Y), and impedance (Z). The importance of S-parameters as well as a brief history and physical meaning will be reviewed in Chapter II.

Various test instruments are readily available from a number of manufacturers which provide for very fast and accurate acquisition of S-parameters at a reasonable cost. The most common instrument on the market today is the automatic vector network analyzer which will be thoroughly reviewed in Chapter III. Today's network analyzer coupled with an S-parameter test set can measure a two-port device's four S-parameters over a user-specified frequency range in a matter of seconds. These analyzers are designed for a specific characteristic impedance, Z_o, which is typically 50 Ω. In other words, the S-parameters of a device measured with 50 Ω test equipment reflect how the device will perform if and only if it is placed in an environment that has a 50 Ω source and load impedance. If the actual application does not contain a 50 Ω source and load, however, the actual performance may vary greatly from the measured values.

Delphi Packard Electric Systems, based in Warren, Ohio, manufactures various wiring assemblies and components which require high frequency characterization. One such product is ignition cable. Ignition cable has the potential to radiate high frequency energy which could possibly interfere with the automobile's entertainment system if not properly designed and evaluated. Another Delphi Packard product which requires high frequency characterization is filtered header connectors which are connectors that contain various filter elements such as capacitors and ferrites. Each filtered header connector is required to meet certain high frequency performance specifications set either by Delphi Packard or by Delphi Packard customers.

Delphi Packard's Electromagnetic Compatibility (EMC) laboratory is equipped with network analyzers and other high frequency test equipment used to characterize the above mentioned devices as well as many others. These instruments provide an essential function starting at the design phase of a product all the way through final validation. Although all of the data generated by these instruments serves a useful purpose, it is still dependent on a 50 Ω source and load impedance. Since the automotive environment presents such a wide variety of impedance values other than 50 Ω, questions regarding "true" product performance has been raised from within Delphi Packard as well as by Delphi Packard customers. Therefore a tool was needed to enable accurate measurement (or prediction) of a device's performance in source and load impedances representative of the automotive environment.

This thesis addresses this need by developing a mathematical algorithm in Chapter IV which takes a component's existing 50 Ω S-parameters and generates a new set of S-parameters based on a user-defined source and load impedance. Chapter V then incorporates these algorithms into a computer program called "CONVERTZ" which utilizes the remote programmability of the network analyzer to automatically extract the 50 Ω S-parameters, apply the user-defined source and load impedances, and finally upload the new S-parameters back to the analyzer's memory. Chapter VI contains some useful applications for CONVERTZ including a correlation study that compares the s_{21} parameter generated by CONVERTZ and radiated emissions data taken in Delphi Packard's absorber lined chamber.

CHAPTER II

S-PARAMETER REVIEW

2.1 S-Parameter History

Before developing the methods for device characterization in arbitrary source and load impedances, it is important to have a good understanding of S-parameters and what they mean in a practical sense. Therefore, a brief review of S-parameter theory as well as basic measurement techniques will be presented in this chapter.

S-parameters were originally developed in the late 1930's for nuclear-physics applications[1] where a collision between a subatomic particle and a nucleus results in a nuclear reaction. The incoming particle can be thought of as traveling down an "input" channel while the resulting radiation from the collision is said to be scattered into multiple "output" channels. This scattering phenomena resulted in the name of scattering or S-parameters. In the 1950's, S-parameters were adapted for microwave measurements[2] due to the inadequacies of other parameter sets such as hybrid (H), admittance (Y), and impedance (Z) parameters at higher frequencies. By reviewing these traditional parameters, it will become apparent why an additional method of network characterization was needed.

2.2 S-Parameter Development

For a two-port linear device such as the one shown in Figure 2.1, the H, Y, and Z-parameter sets are given by

<u>Hybrid (H) Parameters</u>

$$V_1 = h_{11}I_1 + h_{12}V_2 \tag{1a}$$

$$I_2 = h_{21}I_1 + h_{22}V_2 \tag{1b}$$

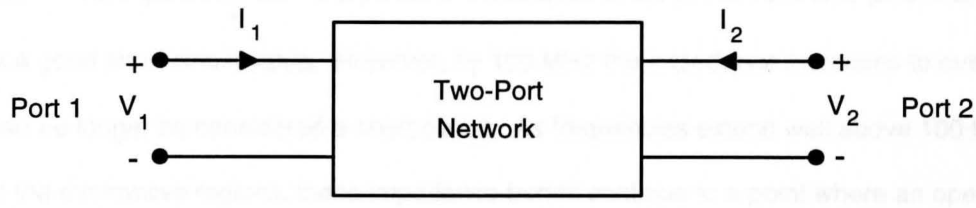

Figure 2.1 Two-Port Linear Device

Admittance (Y) Parameters

$$I_1 = y_{11}V_1 + y_{12}V_2 \tag{2a}$$

$$I_2 = y_{21}V_1 + y_{22}V_2 \tag{2b}$$

Impedance (Z) Parameters

$$V_1 = z_{11}I_1 + z_{12}I_2 \tag{3a}$$

$$V_2 = z_{21}I_1 + z_{22}I_2 \tag{3b}$$

As can be seen from these equations, all of the variables are either terminal voltages or currents with the only difference between parameter sets being the choice of independent and dependent variables. The parameters themselves are simply the constants which relate these variables. Looking at the H-parameters (1), in order to determine h_{11}, V_2 is set equal to zero by short circuiting Port 2 and taking the ratio of V_1 to I_1. To determine h_{12}, I_1 is set equal to zero by open circuiting Port 1 and taking the ratio of V_1 to V_2. As can be seen, the key to determining these parameter sets is the ability to open circuit and short circuit the device under test (DUT).

As frequency increases, however, it gets more and more difficult to achieve the required open and short circuit conditions primarily due to a device's lead inductance and capacitance. For example, a very small parasitic capacitance of 10 pF at 10 kHz has an impedance of $(2*\pi*f*C)^{-1}$ = 1.6 MΩ which is a good open circuit value. By 100 MHz, however, the impedance resulting from this same capacitance has reduced to 160 Ω which no longer resembles an open circuit. The same can be said regarding what would seem to be a negligible inductance. The impedance due to a parasitic inductance of 0.1 μH at 10 kHz is $(2*\pi*f*L) = 6\ m\Omega$ which is a good short circuit value. However, by 100 MHz the impedance increases to over 60 Ω which can no longer be considered a short circuit. As frequencies extend well above 100 MHz and into the microwave regions, these impedance trends continue to a point where an open circuit actually looks more like a short circuit and a short circuit more like an open circuit.

Methods do exist which can achieve the required open and short conditions such as stub tuning. Performing stub tuning requires adjustments at each frequency test point in order to reflect short or open circuit conditions to the DUT terminals. However, for swept frequency measurements with a reasonable resolution (large number of test frequencies) stub tuning is a rigorous and impractical process in today's environment. Also, for DUT's containing transistors, the open or short circuit conditions supplied to the device can result in an oscillatory condition. Therefore, a need for a new parameter set existed for device characterization at high frequencies based on something other than terminal voltages and currents.

With a wide variety of high frequency equipment readily available for generating and measuring high frequency traveling waves (such as synthesized sweepers, directional couplers/bridges, power splitters and tuned receivers), traveling waves became the obvious variable choice for a new parameter set. K. Kurokawa fully described this traveling wave parameter set in 1965[3] by choosing a linear transformation of the terminal voltages and currents which produced the desired mathematical units. Any linear transformation of terminal voltages and currents is valid as long as the transformation is not singular. The new parameter set describes the incident and reflected power waves, a_j and b_j, of a network where a_j and b_j are defined by

$$a_j = \frac{V_j + Z_j I_j}{2\sqrt{|\text{Re}\, Z_j|}} \tag{4}$$

$$b_j = \frac{V_j - Z_j^* I_j}{2\sqrt{|\text{Re}\, Z_j|}} \tag{5}$$

where V_j and I_j are the voltage and current flowing into the j^{th} port of the network respectively and Z_j is the impedance looking out from the j^{th} port. The asterisk (*) denotes the complex conjugate of the impedance value. Notice that the square of the magnitude of a_j and b_j have the units of power, hence the name, power waves.

In Figure 2.2a, the new variables are illustrated for a one-port network. The incident traveling wave (a_1) is delivered to the one-port network, or "load", and some of this energy (b_1) is reflected back to the source depending on the mismatch between the transmission line's characteristic impedance and the input impedance of the network.

Figure 2.2b shows the power waves for a two-port network. Looking from the Port 1 side, the incident traveling wave entering the network is again a_1. Some of the energy is reflected (b_1) based on the mismatch between the transmission line's characteristic impedance and input impedance of the network. The remainder of this energy, assuming no internal absorption, is transmitted through the network and is represented by b_2. Looking from the Port 2 side of the

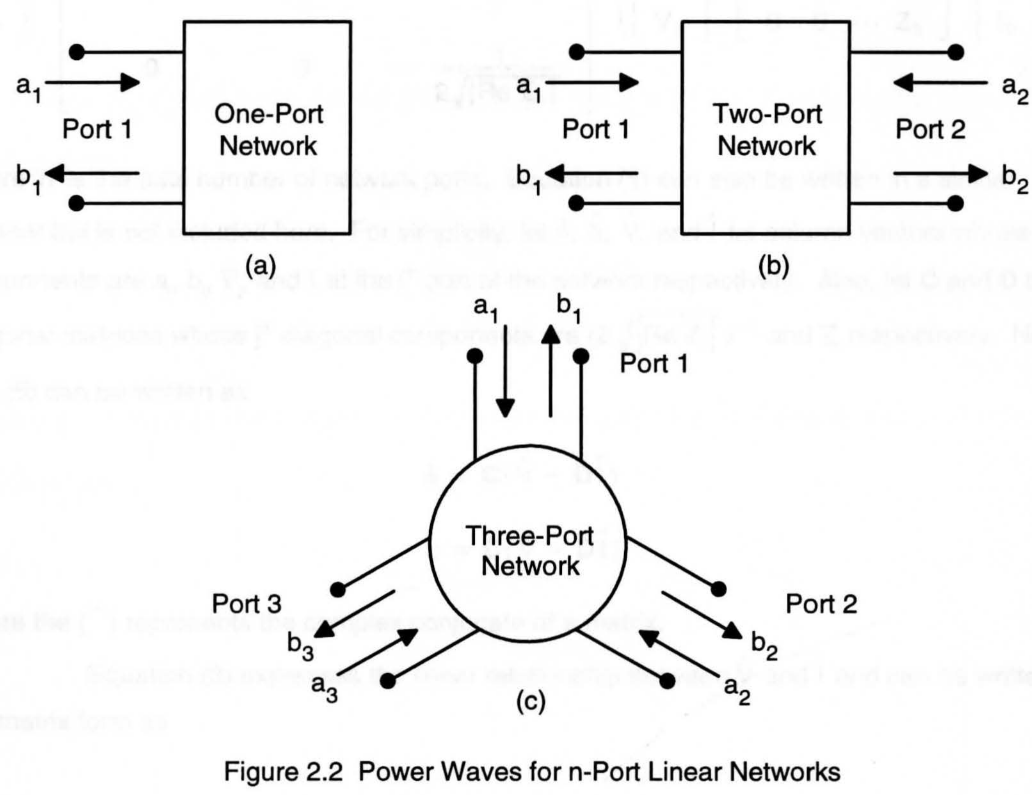

Figure 2.2 Power Waves for n-Port Linear Networks

network, a_2 is now the incident wave entering the device and b_2 is the reflected energy due to the impedance mismatch at the interface. The transmitted energy through the network is b_1, again assuming no internal absorption by the network.

Figure 2.2c illustrates a three-port device which is simply an extension of the previous two cases. For example, looking at Port 1, the incident energy (a_1) enters the device. Some of the energy is reflected (b_1) as before, and some of the energy is transmitted through the network to Port 2 (b_2) and some to Port 3 (b_3). Extending this thought process, it is clear that the definition of power waves can be applied to any arbitrary n-port network.

Based on the above discussion which shows that each port of a linear n-port network can have an associated incident and reflected power wave, equation (4) is recalled and written in matrix form:

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \frac{1}{2\sqrt{|\mathrm{Re}\, Z_1|}} & 0 & \cdots & 0 \\ 0 & \frac{1}{2\sqrt{|\mathrm{Re}\, Z_2|}} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \frac{1}{2\sqrt{|\mathrm{Re}\, Z_n|}} \end{bmatrix} * \left(\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix} + \begin{bmatrix} Z_1 & 0 & \cdots & 0 \\ 0 & Z_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & Z_n \end{bmatrix} * \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{bmatrix} \right) \quad (6)$$

where 'n' is the total number of network ports. Equation (5) can also be written in a similar manner but is not included here. For simplicity, let $\hat{a}$, $\hat{b}$, $\hat{V}$, and $\hat{I}$ be column vectors whose j^{th} components are a_j, b_j, V_j, and I_j at the j^{th} port of the network respectively. Also, let **C** and **D** be diagonal matrices whose j^{th} diagonal components are $(2\sqrt{|\mathrm{Re}\, Z_j|})^{-1}$ and Z_j respectively. Now (4) and (5) can be written as

$$\hat{a} = \mathbf{C}(\hat{V} + \mathbf{D}\hat{I}) \quad (7)$$

$$\hat{b} = \mathbf{C}(\hat{V} - \overline{\mathbf{D}}\hat{I}) \quad (8)$$

where the ($\overline{\ }$) represents the complex conjugate of a matrix.

Equation (3) expresses the linear relationship between $\hat{V}$ and $\hat{I}$ and can be written in the matrix form as

$$\hat{V} = \mathbf{Z}\hat{I} \tag{9}$$

where **Z** is the impedance matrix, and $\hat{V}$ and $\hat{I}$ are defined as in (7) and (8). Since the power waves, a and b, are simply the result of a linear transformation of V and **I**, it can be deduced that a linear transformation must also exist between a and b. This can be written in the form

$$\hat{b} = \mathbf{S}\hat{a} \tag{10}$$

where **S** is called the power wave scattering matrix. Expanding (10) yields

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} s_{11} & s_{12} & s_{13} & \cdots & s_{1n} \\ s_{21} & s_{22} & s_{23} & \cdots & s_{2n} \\ s_{31} & s_{32} & s_{33} & \cdots & s_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ s_{n1} & s_{n2} & s_{n3} & \cdots & s_{nn} \end{bmatrix} * \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix} \tag{11}$$

where 'n' again represents the total number of network ports.

From (11) it can be seen that for a one-port network (n=1), one equation exists with one S-parameter. The equation is simply

$$b_1 = s_{11}\, a_1 \tag{12}$$

For a two-port network (n=2), two equations exist with a total of four S-parameters. These equations are

$$b_1 = s_{11}\, a_1 + s_{12}\, a_2 \tag{13a}$$

$$b_2 = s_{21}\, a_1 + s_{22}\, a_2 \tag{13b}$$

A three-port network (n=3) consists of three equations with nine total S-parameters. These equations are

$$b_1 = s_{11}\, a_1 + s_{12}\, a_2 + s_{13}\, a_3 \tag{14a}$$

$$b_2 = s_{21}\, a_1 + s_{22}\, a_2 + s_{23}\, a_3 \tag{14b}$$

$$b_3 = s_{31}\, a_1 + s_{32}\, a_2 + s_{33}\, a_3 \tag{14c}$$

Following the above thought process, it is easy to see the pattern that relates the total number of network ports to the corresponding number of S-parameters. The number of S-parameters is simply equal to the square of the number of network ports (# S-parameters = # ports^2).

From this point on, this thesis will limit its discussion to two-port networks only. One reason for this is based on the fact that anything derived for two-port networks can be easily extended to n-ports as well. The second reason is that all examples throughout this thesis consist of two-ports.

2.3 Measuring S-Parameters

Now that there is a new parameter set based on incident and reflected power waves rather than terminal voltages and currents, the method for measuring S-parameters needs to be understood. Recalling H, Y, and Z-parameters, the measurements are accomplished by open-circuiting and short-circuiting the input or the output resulting in a simple ratio for the corresponding parameter. A similar approach is used for measuring S-parameters in that all but one of the input signals must be forced to zero in order for a simple ratio to result.

Looking at (13a), if a_2 is set to zero, s_{11} becomes the ratio of b_1 to a_1. Similarly from (13b), if a_2 is again zero, s_{21} becomes the ratio of b_2 to a_1. These two S-parameters (s_{11} and s_{21}) are known as the forward S-parameters since the stimulus is applied to the input, or in the forward direction of the network. The typical measurement setup for the forward S-parameters is shown in Figure 2.3. Here a signal source provides the incident power (a_1) to Port 1 of the DUT through a directional coupler with negligible loss. Based on the input impedance of the DUT and characteristic impedance of the measurement system (usually 50 Ω), some of the incident power is reflected back towards the source (b_1) and appears at the sampling port of the directional coupler. By knowing the incident power delivered to the DUT and measuring the amount reflected back using some type of RF receiver, the first S-parameter can be obtained using

$$s_{11} = \frac{b_1}{a_1}\Big|_{a_2=0} \tag{15}$$

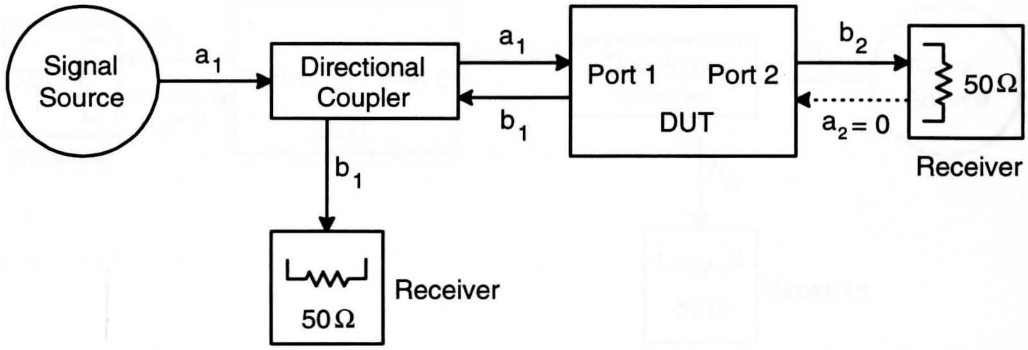

Figure 2.3 Measurement Setup for Forward S-Parameters S_{11} and S_{21}

which is known as the input reflection coefficient. The remainder of the incident power is either absorbed by the DUT and/or transmitted to Port 2 as b_2. Again by knowing the initial incident power and measuring the amount exiting Port 2, the second S-parameter is obtained from (13b) resulting in

$$s_{21} = \frac{b_2}{a_1}\Big|_{a_2=0} \tag{16}$$

which is called the forward transmission.

As can be seen by (15) and (16), the only time the forward S-parameter equations are valid is when $a_2 = 0$. This condition can only be met if Port 2 of the DUT is terminated with a load which matches the characteristic impedance of the test system. With a matched load impedance, all of the transmitted power (b_2) will be dissipated by the load resistor thereby eliminating any reflections and making $a_2 = 0$.

Figure 2.4 shows that the reverse S-parameters (s_{12} and s_{22}) are measured much the same way as the forward S-parameters. The only difference is that the DUT is essentially flipped around so that Port 2 becomes the incident port and a_1 is now forced to be zero. From (13a) it can be seen that if a_1 is set to zero

$$s_{12} = \frac{b_1}{a_2}\Big|_{a_1=0} \tag{17}$$

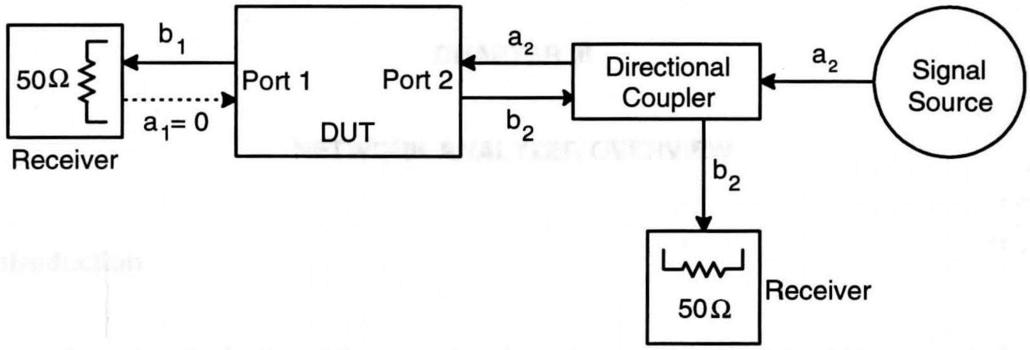

Figure 2.4 Measurement Setup for Reverse S-Parameters S_{22} and S_{12}

which is known as the reverse transmission parameter. Likewise from (13b)

$$s_{22} = \frac{b_2}{a_2}\Big|_{a_1=0} \qquad (18)$$

which is called the output reflection coefficient. Again, it is imperative that the output of the network (which is Port 1 in this case) be terminated in a matched resistive load to ensure the condition that $a_1 = 0$ is met.

CHAPTER III

NETWORK ANALYZER OVERVIEW

3.1 Introduction

Now that the basics of S-parameters have been reviewed, it would be a good idea to gain an understanding of how a modern test instrument makes these measurements as well as describe its operational features. An automatic network analyzer (ANA) coupled with an S-parameter test set is the primary instrumentation used to characterize devices and/or networks at RF and microwave frequencies. Basically, today's network analyzer test system consists of a signal source, signal separation devices, multiple receivers, and a display.

3.2 Measuring Forward S-Parameters

A block diagram showing how these components are packaged together for measuring the forward S-parameters, s_{11} and s_{21}, is illustrated in Figure 3.1. An RF signal of a particular frequency is generated by the Signal Source and travels to a Power Splitter contained in the S-Parameter Test Set. The power is then split and half is transmitted to Receiver 'R' while the other half travels through Directional Coupler #1 to Port 1 of the Test Set due to the "Forward" setting of the Switch Control. Both of the signals that appear at the two outputs of the Power Splitter are identical in both magnitude and phase and are considered the power waves a_1. Notice that the input of the DUT is connected to Port 1 of the Test Set while the output of the DUT is connected to Port 2. Depending on the impedance mismatch at the input of the DUT interface, some of a_1 will be reflected back through Port 1 of the Test Set as the power wave b_1. Directional Coupler #1 samples this signal which is then measured by Receiver 'A'. The remainder of this reflected power is dissipated by the internal source resistance. The portion of power wave a_1 that

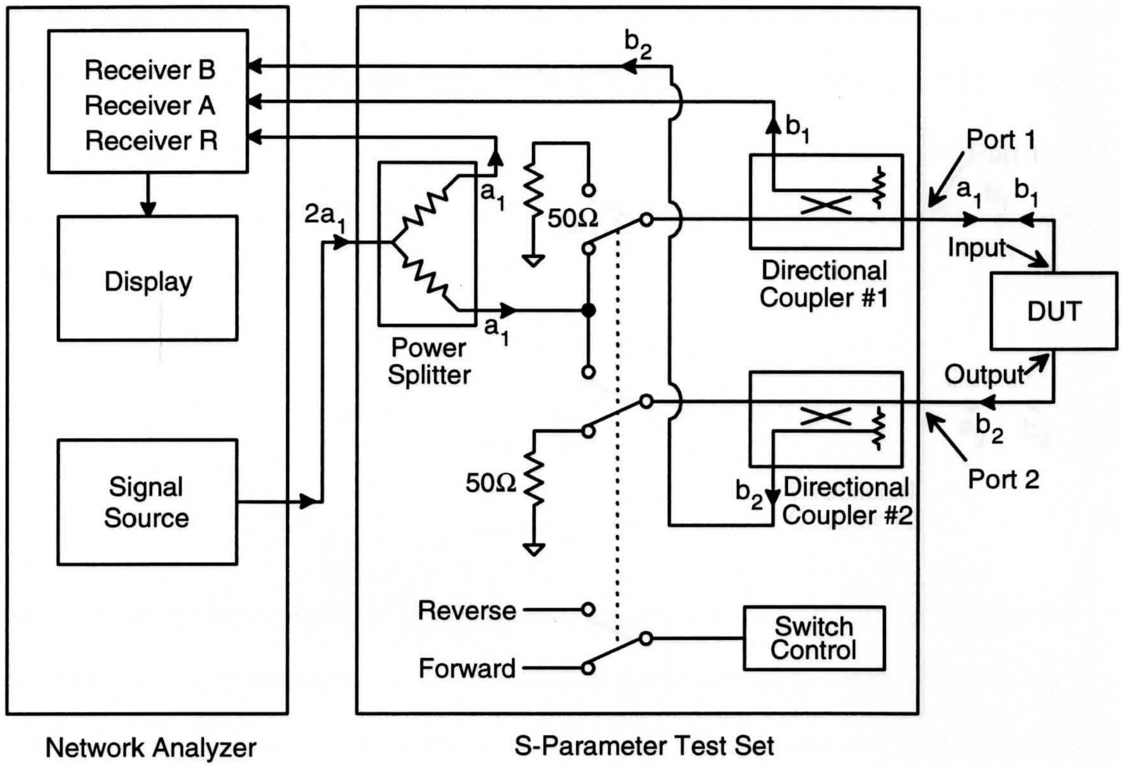

Figure 3.1 Simplied Block Diagram of Network Analyzer Test System in Foward Configuration

is not reflected at the DUT interface is either dissipated by the DUT itself (as heat) and/or passed through the DUT to Port 2 of the Test Set as power wave b_2. Directional Coupler #2 samples b_2 and is measured by Receiver 'B'. The remainder of b_2 is completely dissipated by the internal load resistance of 50 Ω.

Recalling the forward S-parameter equations

$$s_{11} = \frac{b_1}{a_1}\Big|_{a_2=0} \tag{19}$$

$$s_{21} = \frac{b_2}{a_1}\Big|_{a_2=0} \tag{20}$$

the requirement that a_2 must be zero is met because the switched 50 Ω internal load resistance completely dissipates the transmitted signal b_2, therefore no signal is reflected back as a_2. Now,

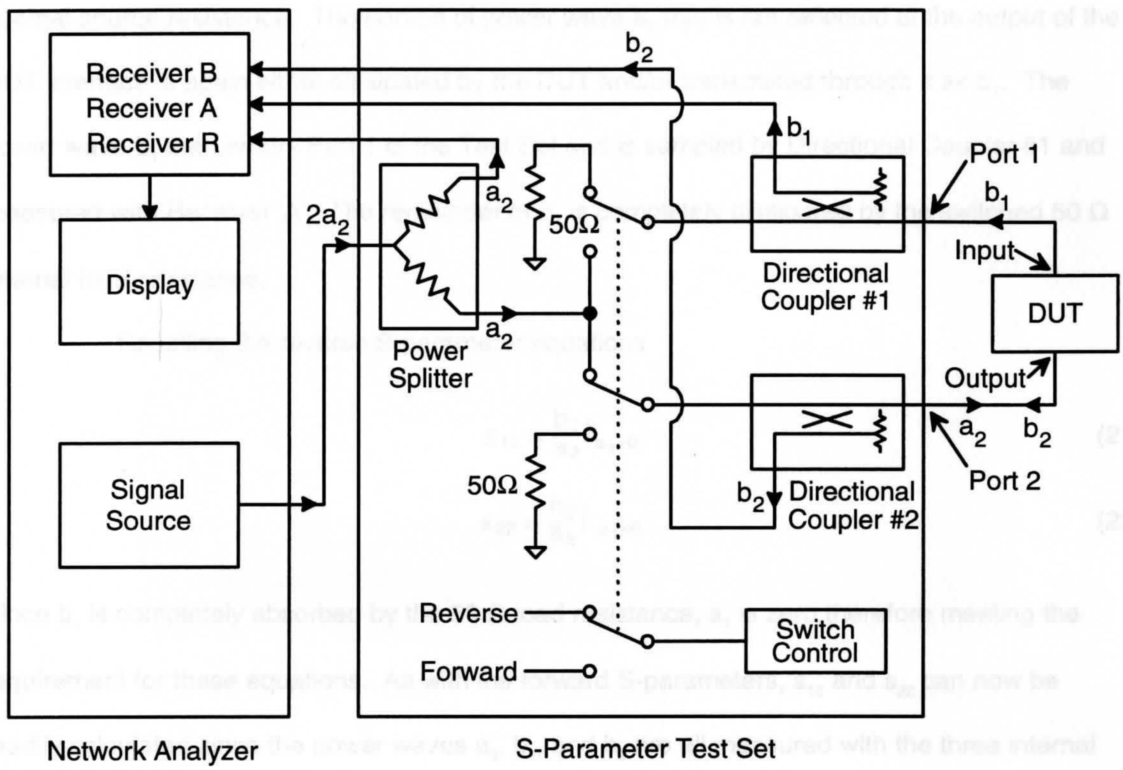

Figure 3.2 Simplied Block Diagram of Network Analyzer Test System in Reverse Configuration

s_{11} and s_{21} can be easily calculated since the power waves a_1, b_1, and b_2 are all measured with the three internal receivers of the network analyzer.

3.3 Measuring Reverse S-Parameters

To measure the reverse S-parameters, s_{12} and s_{22}, the Switch Control is set to the "Reverse" direction as shown in Figure 3.2. As in the forward case, the power from the signal source is split in two where one half again travels to Receiver 'R' but the other half, due to the new switch setting, travels through Directional Coupler #2 and exits Port 2 as the power wave a_2. Depending on the impedance mismatch presented at the output of the DUT interface, some of a_2 is reflected back through Port 2 of the Test Set as b_2. Once again, Directional Coupler #2 samples the signal and measures it with Receiver 'B'. The remainder of b_2 is dissipated by the internal source resistance. The portion of power wave a_2 that is not reflected at the output of the DUT interface is again either dissipated by the DUT and/or transmitted through it as b_1. The power wave b_1 then enters Port 1 of the Test Set and is sampled by Directional Coupler #1 and measured with Receiver 'A'. The remainder of b_1 is completely dissipated by the switched 50 Ω internal load resistance.

Recalling the reverse S-parameter equations

$$s_{12} = \frac{b_1}{a_2}\bigg|_{a_1=0} \quad (21)$$

$$s_{22} = \frac{b_2}{a_2}\bigg|_{a_1=0} \quad (22)$$

Since b_1 is completely absorbed by the 50 Ω load resistance, a_1 is zero therefore meeting the requirement for these equations. As with the forward S-parameters, s_{12} and s_{22} can now be readily calculated since the power waves a_2, b_1, and b_2 are all measured with the three internal receivers of the network analyzer.

3.4 Network Analyzer Features

The process just described was for making S-parameter measurements at a single frequency. To fully characterize a DUT, however, usually requires that S-parameter measurements be made over an entire range of frequencies. This is easily accomplished with today's network analyzers because they incorporate a synthesized sweeping source which produces a swept RF signal (over a specified frequency range) with very high frequency stability and very little phase noise.[4] Of course each network analyzer has a finite frequency capability based on the operational frequency characteristics of the internal components including the signal source, receivers, and the various signal separation devices. A good rule of thumb is that the higher the operational frequency range of the analyzer, the higher the cost of the instrument. Delphi Packard utilizes a HP 8753C network analyzer test system, shown in Figure 3.3, that

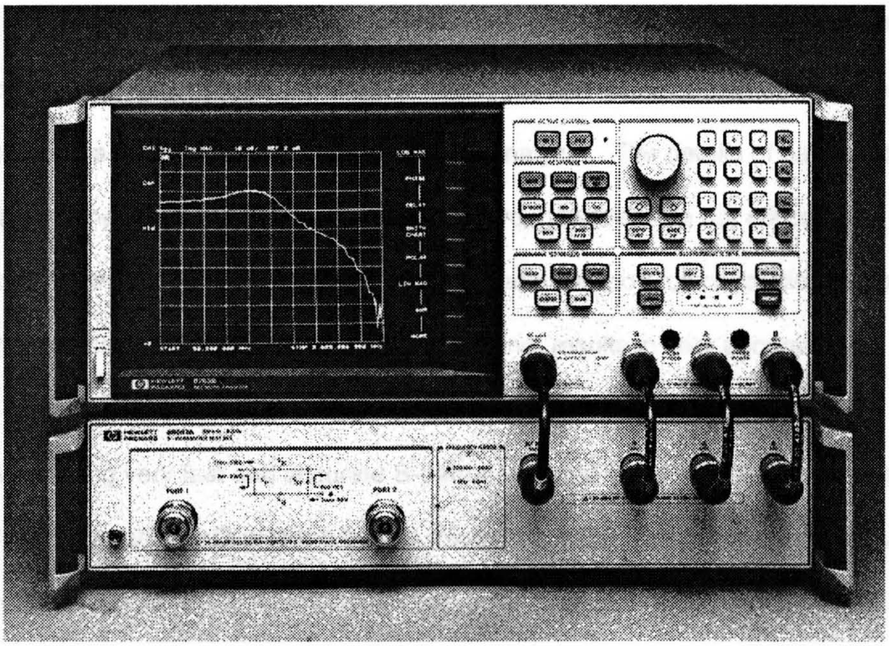

Figure 3.3 Delphi Packard's HP 8753C Network Analyzer Test System

operates from 300 kHz to 3 GHz. This figure illustrates the network analyzer (top) and S-parameter test set (bottom) as well as the interconnecting cables between the two.

Today's network analyzers are microprocessor based which allow for very fast yet highly accurate measurements. For instance, measuring the s_{11} parameter from 300 kHz to 3 GHz with 401 data points takes under 3 seconds. The accuracy is incorporated by allowing the user to perform a system calibration which essentially removes repeatable systematic errors from the raw data that the analyzer collects. The errors that can be removed include directivity, source match, load match, isolation, reflection tracking, and transmission tracking, each in both the forward and reverse direction. There are several calibration routines the network analyzer incorporates which can remove one or more of these errors depending on the accuracy and S-parameter the user desires. To measure all four S-parameters with the highest accuracy possible requires a two-port full calibration be performed which effectively removes all twelve error terms. To execute a two-port full calibration requires known standards to be consecutively placed at the outputs of the test port extension cables at the network analyzer's prompting. These calibration standards consist of a short termination, 50 Ω termination, shielded open termination, and an interconnecting thru. These standards are very precise and maintain their respective values over the entire frequency range of the network analyzer. The details of each error term and the corresponding effect of each calibration standard, however, are not critical for development of this thesis. What is critical is the fact that a two-port full calibration performed at the ends of the test cables, known as the calibration plane, results in very accurate and repeatable S-parameter data.

Generating very fast and accurate S-parameter data is just the beginning of the network analyzer's capability. Due to the built-in microprocessor, many convenient features have been incorporated such as multiple rectangular and polar display formats. For example, the user can choose to display an s_{11} measurement in various formats such as linear or logarithmic magnitude, phase, real, imaginary, impedance, as well as Standing Wave Ratio (SWR). Other network analyzer features that are specific to the HP 8753C, however not exclusively, include multiple trace display where up to two live traces and two memory traces can be displayed simultaneously. Trace math is also incorporated which allows the user to add, subtract, multiply, and divide various live traces as well as memory traces. Two noise reduction techniques are also available including sweep-to-sweep averaging and smoothing which effectively lowers the noise floor of the instrument. Direct plotter output is another nice feature where a touch of a button dumps the entire contents of the CRT directly to a plotter for an instant hardcopy handy for reports, displays, or presentations.

One final feature to mention is the network analyzer's HP-IB (Hewlett-Packard Interface Bus) which is also known as the GPIB (General Purpose Interface Bus). This is the remote programming digital interface based on the IEEE 488.1 and IEC-625 worldwide standards for interfacing test instruments. This allows the analyzer to be completely controlled via an external computer which can send instructions to and receive data from the analyzer. The remote operator has the same control of the instrument over the bus as a local operator using the front panel buttons and knobs. This is extremely useful for automated measurements as well as for saving calibration and trace data on disk for future use and analysis.

3.5 Network Analyzer Test Sequence and Output

To perform S-parameter measurements on a network analyzer test system, a certain sequence of events is usually followed. At Delphi Packard, the following steps are commonly performed to characterize a DUT.

1. Preset the network analyzer to return it to a known default state
2. Enter the desired start and stop frequency
3. Enter the number of data points to be taken
4. Select the S-parameter to be measured and the display format
5. Perform an instrument calibration
6. Connect the DUT to Ports 1 and 2 of the analyzer using the extension cables
7. Scale the measurement using the AUTOSCALE feature
8. Plot the data and/or store it to disk

Figure 3.4 is an actual graph obtained from the HP 8753C network analyzer by following the above steps. Since many of these graphs are contained throughout this thesis, it will be useful to explain the graph layout along with the various abbreviations and notations that are included on each. First of all, the x-axis contains the frequency information while the y-axis consists of the actual S-parameter values. The start and stop frequency information is displayed at the bottom of the graph which for this case is 200 MHz and 500 MHz respectively. The top left of the graph contains specific measurement information about the data. This graph shows that the s_{21} parameter (S21) was measured on Channel 1 (CH 1) and is displayed in a log magnitude (log MAG) format. The top middle of the graph describes the particular scaling information of the displayed trace. For this example each division of the y-axis corresponds to 10 dB (10 dB/) with

Figure 3.4 Sample Output Graph from HP 8753C Network Analyzer

the reference line (REF) being equal to 0 dB. The reference line is indicated by the small triangle along the y-axis. To illustrate how to read the S-parameter data from this graph, the key is to realize that each division below the reference line is 10 dB less than the value of the reference line. Since the reference line is 0 dB, the next line down is -10 dB, the next is -20 dB, and so on. Therefore, at 200 MHz the s_{21} value is 0 dB and at 500 MHz the s_{21} value is -10 dB. Finally, the last bit of information contained on the graph is the "Cor" located along the y-axis. This stands for "Correction" and indicates that a valid calibration has been performed and is in effect for the displayed data.

CHAPTER IV

S-PARAMETERS WITH ARBITRARY SOURCE AND LOAD IMPEDANCE

4.1 Need for Arbitrary Impedance S-Parameters

As briefly mentioned in Chapter I, each network analyzer test system has a specific characteristic impedance. For instance, all of the rigid coaxial cable, connectors, and even the test port extension cables are designed to have the same characteristic impedance. All other components such as the signal source, receivers, directional couplers, and power splitter have input and/or output impedances that match this characteristic impedance. The reason for this, of course, is to maximize signal transmission to the test port by minimizing extraneous reflections due to impedance mismatches that can contribute to measurement error.

The most common characteristic impedance for network analyzer test systems, as well as other RF test equipment, is 50 Ω. This is due primarily to the U.S. military's influence where the majority of Mil Specs require that the test configuration consist of a 50 Ω source and a 50 Ω load impedance. Not all network analyzer systems are exclusively 50 Ω, however. Due to the large cable television industry, 75 Ω systems can be purchased as well as a handful of other characteristic impedance systems.

Typically, S-parameter data found in RF and microwave data books specify that measurements were taken in a 50 Ω test system. This statement is critical since S-parameter measurements are highly dependent on the values of the source and load impedance applied to the input and output of the DUT. For example, the circuit shown in Figure 4.1 consists of a shunt resistor to ground (R_A) as the DUT which is connected between a load resistor R_L and a signal source with an internal resistance of R_S. To calculate the forward transmission parameter, s_{21}, the equation for this parameter must first be derived. Recalling (16)

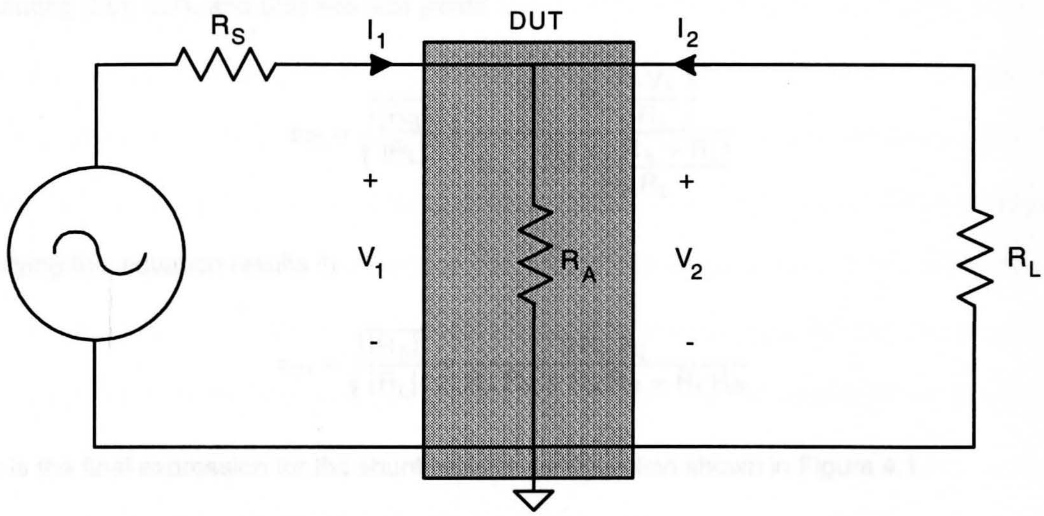

Figure 4.1 Shunt Circuit Configuration for Measuring S_{21}

$$s_{21} = \frac{b_2}{a_1}\Big|_{a_2=0} \tag{23}$$

and substituting in the definitions of a_1 and b_2 from (4) and (5) yield

$$s_{21} = \frac{(V_2 - Z_2^* I_2)}{2\sqrt{|\mathrm{Re}\, Z_2|}} * \frac{2\sqrt{|\mathrm{Re}\, Z_1|}}{(V_1 + Z_1 I_1)} \tag{24}$$

where by definition $Z_1 = Z_S = R_S + jX_S$ and $Z_2 = Z_L = R_L + jX_L$. If the source and load impedance consist of a real part only (i.e. X_S and $X_L = 0$), then

$$s_{21} = \sqrt{\frac{|R_S|}{|R_L|}} * \frac{V_2 - R_L I_2}{V_1 + R_S I_1} \tag{25}$$

From Figure 4.1, the following circuit relationships can be readily obtained

$$V_1 = V_2 \tag{26}$$

$$I_2 = -\frac{V_2}{R_L} = -\frac{V_1}{R_L} \tag{27}$$

$$I_1 = \frac{V_1 (R_A + R_L)}{R_A R_L} \tag{28}$$

Substituting (26), (27), and (28) into (25) yields

$$s_{21} = \sqrt{\frac{|R_S|}{|R_L|}} * \frac{V_1 - R_L\left(\frac{-V_1}{R_L}\right)}{V_1 + \frac{R_S V_1(R_A + R_L)}{R_A R_L}} \tag{29}$$

Simplifying this equation results in

$$s_{21} = \sqrt{\frac{|R_S|}{|R_L|}} * \frac{2 R_A R_L}{R_A R_L + R_A R_S + R_L R_S} \tag{30}$$

which is the final expression for the shunt resistor configuration shown in Figure 4.1.

Allowing R_A to be 25 Ω and the source and load resistance to be 50 Ω, equation (30) yields an s_{21} value of 0.5. Converting this result to the more typical log magnitude format yields $20*\log_{10}(0.5) = -6.02$ dB. Therefore, if the 25 Ω shunt resistor configuration were to be measured in a 50 Ω network analyzer test system, the result would be -6.02 dB. In a practical sense, this means that if the DUT were inserted into a circuit with a 50 Ω source and load impedance, any signal incident to the input of the DUT would be reduced by 6.02 dB when measured at the output.

If this same DUT were inserted into a circuit similar to the one described above with the exception that the load impedance is changed from 50 Ω to 5000 Ω, equation (30) says that the s_{21} value is 0.0664 or $20*\log_{10}(0.0664) = -23.55$ dB. Therefore, if an unsuspecting engineer measures the DUT with a 50 Ω network analyzer and expects to induce a 6 dB loss in an actual application where R_L = 5 kΩ, he would be in for a big surprise to find a 23.6 dB loss instead. To take this one step further, lets assume that both the source and load impedance are 5 kΩ in the actual environment the 25 Ω DUT is to be placed. Again by utilizing (30), $s_{21} = 0.0099$ or $20*\log_{10}(0.0099) = -40.09$ dB which is even further from the network analyzer's -6.02 dB measurement.

With the huge differences in measured versus actual values of the S-parameters described above, one might question the worth of a network analyzer test system as well as the data obtained from it. The value of this measurement system lies in the same fact that one might question its merit - the measured data is dependent on the 50 Ω source and load impedance of the system. By standardizing on specific source and load values for measurement systems, the engineers' job of comparing similar products by different manufacturers is made easier. For instance, when comparing the performance of RF amplifiers by viewing data sheets from different suppliers, the engineer knows the comparison is valid since the measurements are all made with the same impedance test system. On the other hand, if there was no standardization, supplier 'A' might measure their amplifier performance in a 2 kΩ test system while supplier 'B' might measure theirs in a 25 Ω test system. This would make comparing the products from these two suppliers very difficult if not outright impossible.

4.2 Arbitrary Impedance S-Parameter Techniques

Now that the importance as well as the limitations of 50 Ω network analyzer test systems have been discussed, it is time to address the question that is the heart of this thesis - how do networks/devices perform when taken from their 50 Ω test environment and placed in the actual environment for which they are designed. A few methods which address this question were researched and then deemed unfeasible for one reason or another. One such method is to use an S-parameter test set in the characteristic impedance of interest.[5] This method is practical if and only if a couple of characteristic impedances are needed for device characterization. The drawbacks for such an approach include the high cost of purchasing multiple S-parameter test sets as well as the fact that test sets cannot accommodate mixed impedances (i.e. Z_S must always equal Z_L). Another drawback is that S-parameter test sets can only be purchased in very limited characteristic impedance values. This is unacceptable for Delphi Packard use since the impedance values presented by automotive systems vary from a couple Ohms to Megaohms.

Another approach that was investigated involved performing the network analyzer's built-in error correction routines, as described in Chapter III, but replacing the 50 Ω calibration standard with a calibration standard representative of the actual impedance value.[5] Without going into detail as to why this method works and the complex error models associated with the built-in calibration algorithms, this technique was also deemed unfeasible. The reason being that a precision standard is required for each and every value of source and load impedance desired. This is not practical because precision standards are only manufactured in values that correspond to the characteristic impedance of common S-parameter test sets which are very limited as previously mentioned. Limited success can be achieved in constructing home-made standards with thin-film resistors for instance, although over a couple hundred megahertz the values of these "loads" start degrading dramatically. Since Delphi Packard has need to characterize devices/networks to at least 1 GHz, this method was also ruled out.

A third method investigated was a software package marketed by Hewlett-Packard called MDS or Microwave Design System. MDS is basically an integrated CAE (Computer-Aided Engineering) package designed for microwave and RF engineers.[6] The software allows the user to input schematic drawings which can then be used by the built-in linear or nonlinear simulator for high frequency circuit analysis and optimization. Many powerful and useful features have been incorporated in this software but one feature really stands out with regards to this thesis topic. This feature is its ability to remotely control various network analyzer systems over the HP-IB, including Delphi Packard's HP 8753C, and download 50 Ω S-parameter data to a file. The data can then be incorporated into a model which allows the user to alter the value of the source and load impedance, in R + jX format, and new S-parameters based on these new impedance values can be calculated.

Although this software apparently provides an immediate solution for the task at hand, the major drawback to this approach is cost. For the software license alone, the price is approximately $22,000. Add to that the cost of a mid-range workstation to run the program and the price quickly approaches $35,000. Since Delphi Packard has no need for the RF and microwave design features of this software, which is its primary function, it is hard to justify the cost based on the one small feature that would be useful. Although the outright purchasing of this software was not feasible in this case, the approach used to create arbitrary source and load impedance S-parameters based on existing 50 Ω S-parameter data was an intriguing one. A phone call to Hewlett-Packard did not result in a copy of the algorithms used in their software, however it did result in a reference to an IEEE transaction authored by K. Kurokawa from which their algorithms are derived. The derivations which comprise the remainder of this chapter are based on Kurokawa's paper,[3] although the results take on a slightly different form with much more detail and explanation included here for completeness.

4.3 Derivation of Arbitrary Impedance S-Parameter Algorithms

As described in Chapter II, S-parameters are the result of a linear transformation of voltage and current defined in Figure 2.1 for a two-port network. Since the H, Y, and Z-parameters are also a linear transformation of this same voltage and current, there must be a way to convert between S-parameters and H, Y, and Z-parameters. Lets concentrate on S-parameter to Z-parameter conversion and vice-versa. From (9) and (10)

$$\hat{V} = \mathbf{Z}\hat{I} \tag{31}$$

$$\hat{b} = \mathbf{S}\hat{a} \tag{32}$$

where $\hat{a}$, $\hat{b}$, $\hat{V}$, and $\hat{I}$ are the column vectors previously defined, and **Z** and **S** are the impedance and scattering matrix respectively. Recall, also, the definition of the power waves, a and b, in matrix form from (7) and (8)

$$\hat{a} = \mathbf{C}(\hat{V} + \mathbf{D}\hat{I}) \tag{33}$$

$$\hat{b} = \mathbf{C}(\hat{V} - \overline{\mathbf{D}}\hat{I}) \tag{34}$$

where **C** and **D** are diagonal matrices whose j^{th} diagonal components are $(2\sqrt{|\mathrm{Re}\,Z_j|}\,)^{-1}$ and Z_j respectively. Substituting (33) and (34) into (32) yields

$$\mathbf{C}(\hat{V} - \overline{\mathbf{D}}\hat{I}) = \mathbf{S}\mathbf{C}(\hat{V} + \mathbf{D}\hat{I}) \tag{35}$$

Substituting (31) into (35) and factoring out the $\hat{\mathrm{I}}$ term results in

$$\mathbf{S}\,\mathbf{C}(\mathbf{Z}+\mathbf{D}) = \mathbf{C}(\mathbf{Z}-\overline{\mathbf{D}}) \tag{36}$$

Rearranging (36) to solve for the S-matrix as a function of the Z-matrix yields

$$\mathbf{S} = \mathbf{C}(\mathbf{Z}-\overline{\mathbf{D}})(\mathbf{Z}+\mathbf{D})^{-1}\,\mathbf{C}^{-1} \tag{37}$$

Now, rearranging (36) to solve for the Z-matrix as a function of the S-matrix yields

$$\mathbf{Z} = \mathbf{C}^{-1}(\mathbf{I}-\mathbf{S})^{-1}(\mathbf{S}\,\mathbf{D}+\overline{\mathbf{D}})\,\mathbf{C} \tag{38}$$

where **I** is a 2x2 identity matrix for a two-port network.

Now lets derive the equations that take an original set of 50 Ω S-parameter data and produce an entirely new set of S-parameter data based on a user-defined source and load impedance. The whole concept is really quite simple once it is realized that Z-parameters are not dependent on source and load impedance values as S-parameters are. Getting straight to the task at hand, a two step process is employed where the first step is to take the original 50 Ω S-parameter data and convert it to Z-parameters by utilizing (38). Even if the original S-parameters are measured in a 1 kΩ test system, the resulting Z-parameters will be the same as the ones obtained in the 50 Ω measurement system. Once the Z-parameters are calculated, step two is to insert these values into

$$\mathbf{S}_{\mathrm{new}} = \mathbf{C}_{\mathrm{new}}(\mathbf{Z}_{50\Omega}-\overline{\mathbf{D}_{\mathrm{new}}})(\mathbf{Z}_{50\Omega}+\mathbf{D}_{\mathrm{new}})^{-1}\,\mathbf{C}_{\mathrm{new}}^{-1} \tag{39}$$

which is a slight modification of (37) where $\mathbf{S}_{new}$, $\mathbf{C}_{new}$ and $\mathbf{D}_{new}$ are the **S, C** and **D** matrices respectively when Z_S and Z_L are replaced by the new source and load impedance Z_{Snew} and Z_{Lnew}. The $\mathbf{Z}_{50\Omega}$ matrix contains the Z-parameters calculated in step one. Although this two-step process appears to be quite simple, to actually derive individual equations for each S-parameter becomes a tedious process as the reader will soon discover.

First, to derive the Z-parameters required for step one, (38) is written in its expanded matrix form as

$$Z_{50\Omega} = \begin{bmatrix} \left(2\sqrt{|\mathrm{Re}\,Z_S|}\right)^{-1} & 0 \\ 0 & \left(2\sqrt{|\mathrm{Re}\,Z_L|}\right)^{-1} \end{bmatrix}^{-1} * \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix} \right\}^{-1}$$

$$* \left\{ \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix} * \begin{bmatrix} Z_S & 0 \\ 0 & Z_L \end{bmatrix} + \begin{bmatrix} Z_S^* & 0 \\ 0 & Z_L^* \end{bmatrix} \right\} * \begin{bmatrix} \left(2\sqrt{|\mathrm{Re}\,Z_S|}\right)^{-1} & 0 \\ 0 & \left(2\sqrt{|\mathrm{Re}\,Z_L|}\right)^{-1} \end{bmatrix} \quad (40)$$

where the * denotes the complex conjugate of the variable. Since $Z_S = R_S + jX_S$, then Re $(Z_S) = R_S$. Likewise, since $Z_L = R_L + jX_L$, then Re $(Z_L) = R_L$. Taking the transpose of the first matrix in (40) and combining the matrices in terms two and three result in

$$Z_{50\Omega} = \begin{bmatrix} 2\sqrt{|R_S|} & 0 \\ 0 & 2\sqrt{|R_L|} \end{bmatrix} * \begin{bmatrix} (1-s_{11}) & -s_{12} \\ -s_{21} & (1-s_{22}) \end{bmatrix}^{-1} * \begin{bmatrix} (s_{11}Z_S + Z_S^*) & s_{12}Z_L \\ s_{21}Z_S & (s_{22}Z_L + Z_L^*) \end{bmatrix}$$

$$* \begin{bmatrix} \left(2\sqrt{|R_S|}\right)^{-1} & 0 \\ 0 & \left(2\sqrt{|R_L|}\right)^{-1} \end{bmatrix} \quad (41)$$

Taking the transpose of the second matrix in (41) and multiplying the result with the first matrix yields

$$Z_{50\Omega} = k_1^{-1} * \begin{bmatrix} 2(1-s_{22})\sqrt{|R_S|} & 2s_{12}\sqrt{|R_S|} \\ 2s_{21}\sqrt{|R_L|} & 2(1-s_{11})\sqrt{|R_L|} \end{bmatrix} * \begin{bmatrix} (s_{11}Z_S + Z_S^*) & s_{12}Z_L \\ s_{21}Z_S & (s_{22}Z_L + Z_L^*) \end{bmatrix}$$

$$* \begin{bmatrix} \left(2\sqrt{|R_S|}\right)^{-1} & 0 \\ 0 & \left(2\sqrt{|R_L|}\right)^{-1} \end{bmatrix} \quad (42)$$

where k_1 is a constant defined by

$$k_1 = (1 - s_{11})(1 - s_{22}) - s_{21}s_{12} \quad (43)$$

Now, multiplying the first two matrices of (42) and simplifying yields

$$\mathbf{Z}_{50\Omega} = k_1^{-1} * \begin{bmatrix} 2\sqrt{|R_S|}\left\{ \begin{matrix} (1 - s_{22})(s_{11} Z_S + Z_S^*) \\ + Z_S s_{12} s_{21} \end{matrix} \right\} & 2\sqrt{|R_S|}\, s_{12}(Z_L + Z_L^*) \\ 2\sqrt{|R_L|}\, s_{21}(Z_S + Z_S^*) & 2\sqrt{|R_L|}\left\{ \begin{matrix} (1 - s_{11})(s_{22} Z_L + Z_L^*) \\ + Z_L s_{21} s_{12} \end{matrix} \right\} \end{bmatrix} * \begin{bmatrix} \left(2\sqrt{|R_S|}\right)^{-1} & 0 \\ 0 & \left(2\sqrt{|R_L|}\right)^{-1} \end{bmatrix} \tag{44}$$

Finally, the last matrix multiplication and simplification of (44) results in

$$\mathbf{Z}_{50\Omega} = k_1^{-1} * \begin{bmatrix} (1 - s_{22})(s_{11} Z_S + Z_S^*) + Z_S s_{12} s_{21} & s_{12}\sqrt{\dfrac{|R_S|}{|R_L|}}\,(Z_L + Z_L^*) \\ s_{21}\sqrt{\dfrac{|R_L|}{|R_S|}}\,(Z_S + Z_S^*) & (1 - s_{11})(s_{22} Z_L + Z_L^*) + Z_L s_{21} s_{12} \end{bmatrix} \tag{45}$$

For the most common case where the original S-parameters in (45) are taken in a 50 Ω test system, $Z_S = R_S + jX_S = 50 + j0$ and $Z_L = R_L + jX_L = 50 + j0$. Substituting these values into (45) result in the much more simplistic form of

$$\mathbf{Z}_{50\Omega} = k_1^{-1} * \begin{bmatrix} 50[(1 - s_{22})(1 + s_{11}) + s_{12} s_{21}] & 100 s_{12} \\ 100 s_{21} & 50[(1 - s_{11})(1 + s_{22}) + s_{21} s_{12}] \end{bmatrix} \tag{46}$$

If the original S-parameter data was taken in a 1 kΩ test system rather than a 50 Ω test system, Z_S would then be $R_S + jX_S = 1000 + j0$ and $Z_L = R_L + jX_L = 1000 + j0$. Substituting these values into (45) results in the Z-matrix, $\mathbf{Z}_{1000\Omega}$, that when solved for individual Z-parameters results in the same values found from solving $\mathbf{Z}_{50\Omega}$. Expanding out the individual Z-parameter terms from (46) results in the following four equations

$$z_{11_{50\Omega}} = 50\frac{(1 - s_{22})(1 + s_{11}) + s_{12} s_{21}}{(1 - s_{11})(1 - s_{22}) - s_{21} s_{12}} \tag{47}$$

$$z_{12_{50\Omega}} = \frac{100 s_{12}}{(1 - s_{11})(1 - s_{22}) - s_{21} s_{12}} \tag{48}$$

$$z_{21_{50\Omega}} = \frac{100 s_{21}}{(1 - s_{11})(1 - s_{22}) - s_{21} s_{12}} \tag{49}$$

$$z_{22_{50\Omega}} = 50\frac{(1 - s_{11})(1 + s_{22}) + s_{21}s_{12}}{(1 - s_{11})(1 - s_{22}) - s_{21}s_{12}} \tag{50}$$

Keep in mind that these equations are valid if and only if the S-parameters are derived or measured in a 50 Ω test system.

Now that a set of equations for calculating 50 Ω Z-parameters have been derived for step one, a set of equations are now needed for step two that use these Z-parameters to produce a new set of S-parameters ($\mathbf{S}_{new}$) based on an arbitrary source and load impedance. Writing (39) in full matrix form results in

$$\mathbf{S}_{new} = \begin{bmatrix} \left(2\sqrt{|\mathrm{Re}\,Z_{S_{new}}|}\right)^{-1} & 0 \\ 0 & \left(2\sqrt{|\mathrm{Re}\,Z_{L_{new}}|}\right)^{-1} \end{bmatrix} * \left\{ \begin{bmatrix} z_{11_{50\Omega}} & z_{12_{50\Omega}} \\ z_{21_{50\Omega}} & z_{22_{50\Omega}} \end{bmatrix} - \begin{bmatrix} Z^*_{S_{new}} & 0 \\ 0 & Z^*_{L_{new}} \end{bmatrix} \right\}$$

$$* \left\{ \begin{bmatrix} z_{11_{50\Omega}} & z_{12_{50\Omega}} \\ z_{21_{50\Omega}} & z_{22_{50\Omega}} \end{bmatrix} + \begin{bmatrix} Z_{S_{new}} & 0 \\ 0 & Z_{L_{new}} \end{bmatrix} \right\}^{-1} * \begin{bmatrix} \left(2\sqrt{|\mathrm{Re}\,Z_{S_{new}}|}\right)^{-1} & 0 \\ 0 & \left(2\sqrt{|\mathrm{Re}\,Z_{L_{new}}|}\right)^{-1} \end{bmatrix}^{-1} \tag{51}$$

where again the * represents the complex conjugate of the variable. Also, Re (Z_{Snew}) = R_{Snew} and Re (Z_{Lnew}) = R_{Lnew} as before. To simplify (51), the transpose of the last matrix is taken and the matrices in the second and third terms are combined to give

$$\mathbf{S}_{new} = \begin{bmatrix} \left(2\sqrt{|R_{S_{new}}|}\right)^{-1} & 0 \\ 0 & \left(2\sqrt{|R_{L_{new}}|}\right)^{-1} \end{bmatrix} * \begin{bmatrix} \left(z_{11_{50\Omega}} - Z^*_{S_{new}}\right) & z_{12_{50\Omega}} \\ z_{21_{50\Omega}} & \left(z_{22_{50\Omega}} - Z^*_{L_{new}}\right) \end{bmatrix}$$

$$* \begin{bmatrix} \left(z_{11_{50\Omega}} + Z_{S_{new}}\right) & z_{12_{50\Omega}} \\ z_{21_{50\Omega}} & \left(z_{22_{50\Omega}} + Z_{L_{new}}\right) \end{bmatrix}^{-1} * \begin{bmatrix} 2\sqrt{|R_{S_{new}}|} & 0 \\ 0 & 2\sqrt{|R_{L_{new}}|} \end{bmatrix} \tag{52}$$

Multiplying the first two matrices of (52) and transposing the third results in

$$\mathbf{S}_{new} = k_2^{-1} * \begin{bmatrix} \dfrac{\left(z_{11_{50\Omega}} - Z^*_{S_{new}}\right)}{2\sqrt{|R_{S_{new}}|}} & \dfrac{z_{12_{50\Omega}}}{2\sqrt{|R_{S_{new}}|}} \\ \dfrac{z_{21_{50\Omega}}}{2\sqrt{|R_{L_{new}}|}} & \dfrac{\left(z_{22_{50\Omega}} - Z^*_{L_{new}}\right)}{2\sqrt{|R_{L_{new}}|}} \end{bmatrix} * \begin{bmatrix} \left(z_{22_{50\Omega}} + Z_{L_{new}}\right) & -z_{12_{50\Omega}} \\ -z_{21_{50\Omega}} & \left(z_{11_{50\Omega}} + Z_{S_{new}}\right) \end{bmatrix} * \begin{bmatrix} 2\sqrt{|R_{S_{new}}|} & 0 \\ 0 & 2\sqrt{|R_{L_{new}}|} \end{bmatrix} \tag{53}$$

where k_2 is a constant defined by

$$k_2 = \left(z_{11_{50\Omega}} + Z_{S_{new}}\right)\left(z_{22_{50\Omega}} + Z_{L_{new}}\right) - z_{12_{50\Omega}} z_{21_{50\Omega}} \tag{54}$$

Multiplying the first two matrices of (53) and simplifying yields

$$\mathbf{S}_{new} = k_2^{-1} * \begin{bmatrix} \dfrac{\left(z_{11_{50\Omega}} - Z^*_{S_{new}}\right)\left(z_{22_{50\Omega}} + Z_{L_{new}}\right) - k_3}{2\sqrt{|R_{S_{new}}|}} & \dfrac{z_{12_{50\Omega}}\left(Z_{S_{new}} + Z^*_{S_{new}}\right)}{2\sqrt{|R_{S_{new}}|}} \\ \dfrac{z_{21_{50\Omega}}\left(Z_{L_{new}} + Z^*_{L_{new}}\right)}{2\sqrt{|R_{L_{new}}|}} & \dfrac{\left(z_{22_{50\Omega}} - Z^*_{L_{new}}\right)\left(z_{11_{50\Omega}} + Z_{S_{new}}\right) - k_3}{2\sqrt{|R_{L_{new}}|}} \end{bmatrix} * \begin{bmatrix} 2\sqrt{|R_{S_{new}}|} & 0 \\ 0 & 2\sqrt{|R_{L_{new}}|} \end{bmatrix} \tag{55}$$

where k_3 is defined as $z_{21_{50\Omega}} * z_{12_{50\Omega}}$. Finally, multiplying the last two matrices of (55) and simplifying the results yields

$$\mathbf{S}_{new} = k_2^{-1} \begin{bmatrix} \left(z_{11_{50\Omega}} - Z^*_{S_{new}}\right)\left(z_{22_{50\Omega}} + Z_{L_{new}}\right) - k_3 & 2R_{S_{new}}\sqrt{\dfrac{|R_{L_{new}}|}{|R_{S_{new}}|}}\, z_{12_{50\Omega}} \\ 2R_{L_{new}}\sqrt{\dfrac{|R_{S_{new}}|}{|R_{L_{new}}|}}\, z_{21_{50\Omega}} & \left(z_{22_{50\Omega}} - Z^*_{L_{new}}\right)\left(z_{11_{50\Omega}} + Z_{S_{new}}\right) - k_3 \end{bmatrix} \tag{56}$$

Expanding out the new individual S-parameter terms results in the following four equations

$$s_{11_{new}} = \frac{\left(z_{11_{50\Omega}} - Z^{*}_{S_{new}}\right)\left(z_{22_{50\Omega}} + Z_{L_{new}}\right) - z_{21_{50\Omega}}\, z_{12_{50\Omega}}}{\left(z_{11_{50\Omega}} + Z_{S_{new}}\right)\left(z_{22_{50\Omega}} + Z_{L_{new}}\right) - z_{21_{50\Omega}}\, z_{12_{50\Omega}}} \tag{57}$$

$$s_{12_{new}} = \sqrt{\frac{\left|R_{L_{new}}\right|}{\left|R_{S_{new}}\right|}}\; \frac{2\,R_{S_{new}}\, z_{12_{50\Omega}}}{\left(z_{11_{50\Omega}} + Z_{S_{new}}\right)\left(z_{22_{50\Omega}} + Z_{L_{new}}\right) - z_{21_{50\Omega}}\, z_{12_{50\Omega}}} \tag{58}$$

$$s_{21_{new}} = \sqrt{\frac{\left|R_{S_{new}}\right|}{\left|R_{L_{new}}\right|}}\; \frac{2\,R_{L_{new}}\, z_{21_{50\Omega}}}{\left(z_{11_{50\Omega}} + Z_{S_{new}}\right)\left(z_{22_{50\Omega}} + Z_{L_{new}}\right) - z_{21_{50\Omega}}\, z_{12_{50\Omega}}} \tag{59}$$

$$s_{22_{new}} = \frac{\left(z_{22_{50\Omega}} - Z^{*}_{L_{new}}\right)\left(z_{11_{50\Omega}} + Z_{S_{new}}\right) - z_{12_{50\Omega}}\, z_{21_{50\Omega}}}{\left(z_{11_{50\Omega}} + Z_{S_{new}}\right)\left(z_{22_{50\Omega}} + Z_{L_{new}}\right) - z_{21_{50\Omega}}\, z_{12_{50\Omega}}} \tag{60}$$

where the values for the 50 Ω Z-parameters come from (47) through (50). Due to the complexity associated with plugging in equations (47) through (50) into each of the new S-parameters in (57) through (60), it was decided to best leave this rigorous number crunching task for a computer.

In summary, this chapter has derived a technique which takes S-parameters that have been gathered from a 50 Ω test system and essentially strips away the source and load impedance dependency by converting them to Z-parameters using (47) - (50). These Z-parameters can then be employed in (57) - (60) to apply a new source and load impedance of any value and calculate the resulting S-parameters. The only drawback of this technique is the fact that all four 50 Ω S-parameters have to be known in order to calculate any one of the arbitrary impedance S-parameters.

CHAPTER V

ARBITRARY IMPEDANCE S-PARAMETER SOFTWARE

5.1 Using the CONVERTZ Program

The algorithms derived in Chapter IV are implemented in a software program written by the author and named CONVERTZ - pronounced CONVERTS. The name CONVERTZ is a shortened form of "Convert Impedance" where impedance is ordinarily represented with the letter "Z". This program is written in TransEra's High Tech Basic (HTBasic) which is a PC-based programming language tailored for instrumentation control. This fact as well as the ability to define complex data types are the main reasons for choosing this language to implement the code. Appendix A contains the code for the CONVERTZ program in its entirety.

Figure 5.1 illustrates the test equipment and configuration needed to run CONVERTZ. The heart of this setup is the HP 8753C Network Analyzer coupled with the HP 85046A

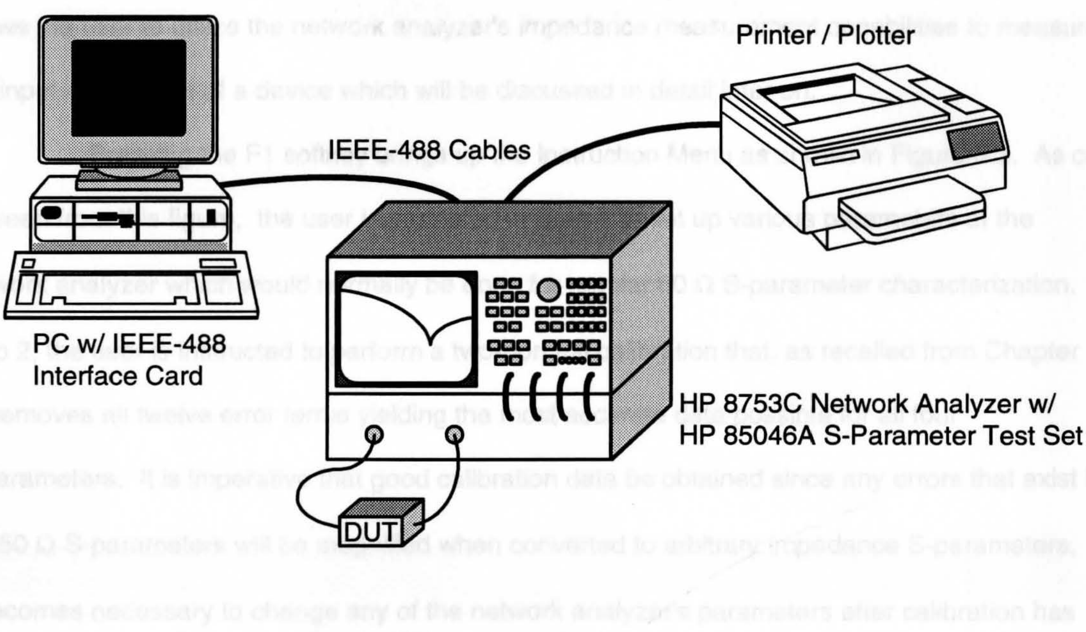

Figure 5.1 Required Equipment Configuration for CONVERTZ

S-Parameter Test Set. This network analyzer test system is connected to a PC and a plotter by IEEE-488 cables. The PC must contain an IEEE-488 interface card in order to communicate over the bus. CONVERTZ can be used with any brand IEEE-488 card as long as the proper driver is loaded with HTBasic. CONVERTZ will not work, however, with any network analyzer test system other than the HP 8753C due to the instrument's specific instruction set.

As can be seen by the length of the code in Appendix A, the CONVERTZ program is quite rigorous due mainly to the design of the user interface and the error handling capabilities that are incorporated. Great lengths had to be taken to make this program user-friendly and crash-proof because it will most likely be used by a variety of people at Delphi Packard. Therefore, rather than delving into the intricacies of the code, it will be more productive to examine the functionality and features of CONVERTZ as would be required for a new user.

Figure 5.2 illustrates the Main Menu which is displayed when CONVERTZ is first loaded. The keys labeled at the bottom of the screen are known as softkeys which correspond to the F1 through F8 function keys on a standard 101/102 enhanced keyboard. The Main Menu allows the user two options other than exiting the program. The F1 key actually begins the 50 Ω S-parameter conversion process which has been developed throughout this thesis. The F5 key allows the user to utilize the network analyzer's impedance measurement capabilities to measure the input impedance of a device which will be discussed in detail later on.

Pressing the F1 softkey brings up the Instruction Menu as shown in Figure 5.3. As can be seen from this figure, the user is prompted in Step 1 to set up various parameters of the network analyzer which would normally be done for regular 50 Ω S-parameter characterization. In Step 2, the user is instructed to perform a two-port full calibration that, as recalled from Chapter III, removes all twelve error terms yielding the most accurate data possible for all four S-parameters. It is imperative that good calibration data be obtained since any errors that exist in the 50 Ω S-parameters will be magnified when converted to arbitrary impedance S-parameters. If it becomes necessary to change any of the network analyzer's parameters after calibration has

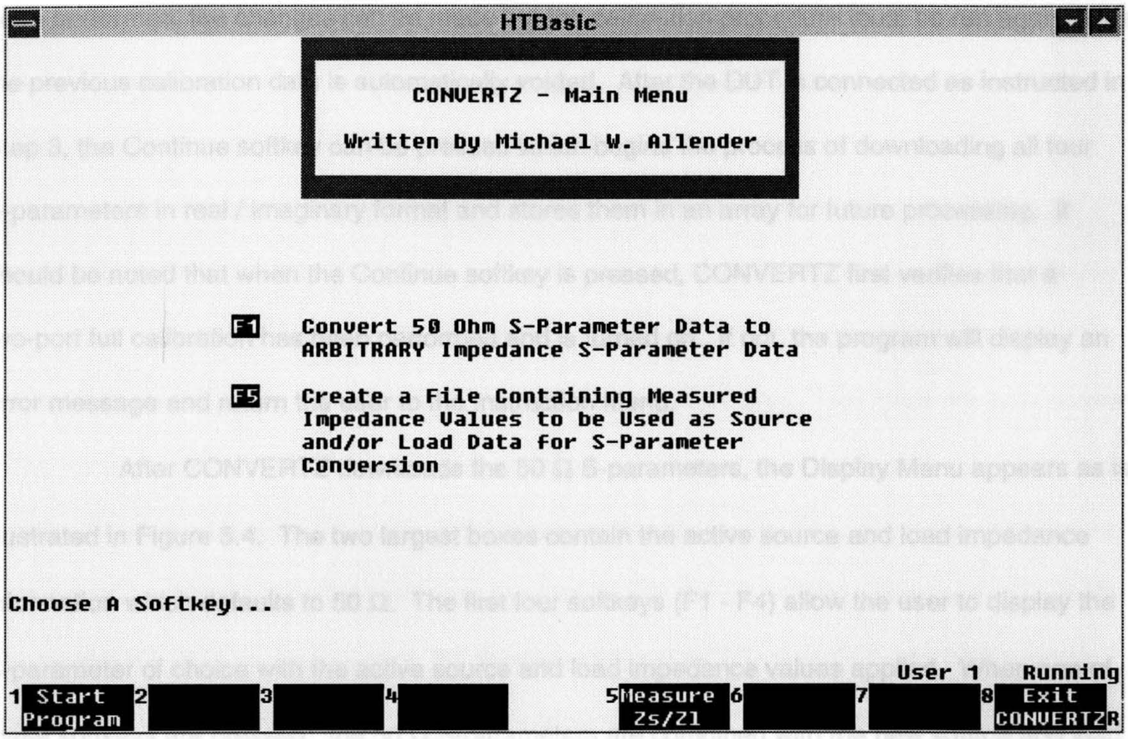

Figure 5.2 Main Menu of CONVERTZ

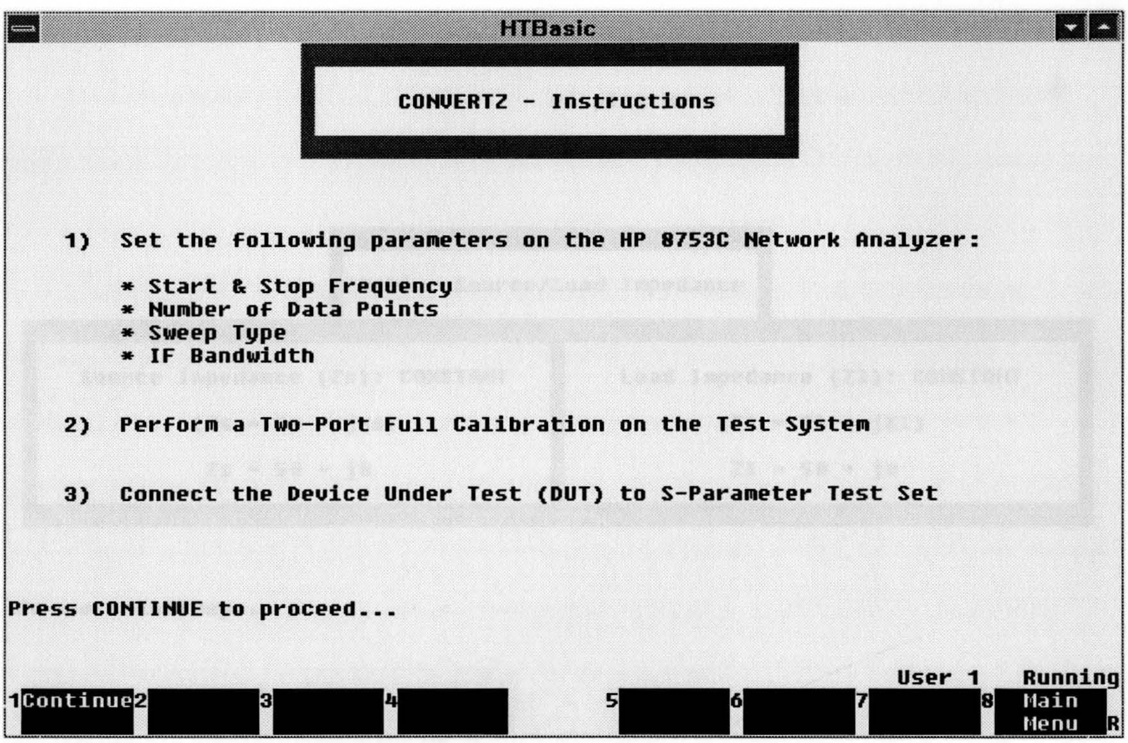

Figure 5.3 Instruction Menu of CONVERTZ

been performed, the changes can be made but the calibration procedure must be run again since the previous calibration data is automatically voided. After the DUT is connected as instructed in Step 3, the Continue softkey can be pressed which begins the process of downloading all four S-parameters in real / imaginary format and stores them in an array for future processing. It should be noted that when the Continue softkey is pressed, CONVERTZ first verifies that a two-port full calibration has been performed and is turned on. If not, the program will display an error message and return the user to the Instruction Menu.

After CONVERTZ downloads the 50 Ω S-parameters, the Display Menu appears as is illustrated in Figure 5.4. The two largest boxes contain the active source and load impedance information which defaults to 50 Ω. The first four softkeys (F1 - F4) allow the user to display the S-parameter of choice with the active source and load impedance values applied. When one of these softkeys are pressed, the 50 Ω S-parameters are combined with the new source and load impedance information using the algorithms derived in Chapter IV. The selected S-parameter

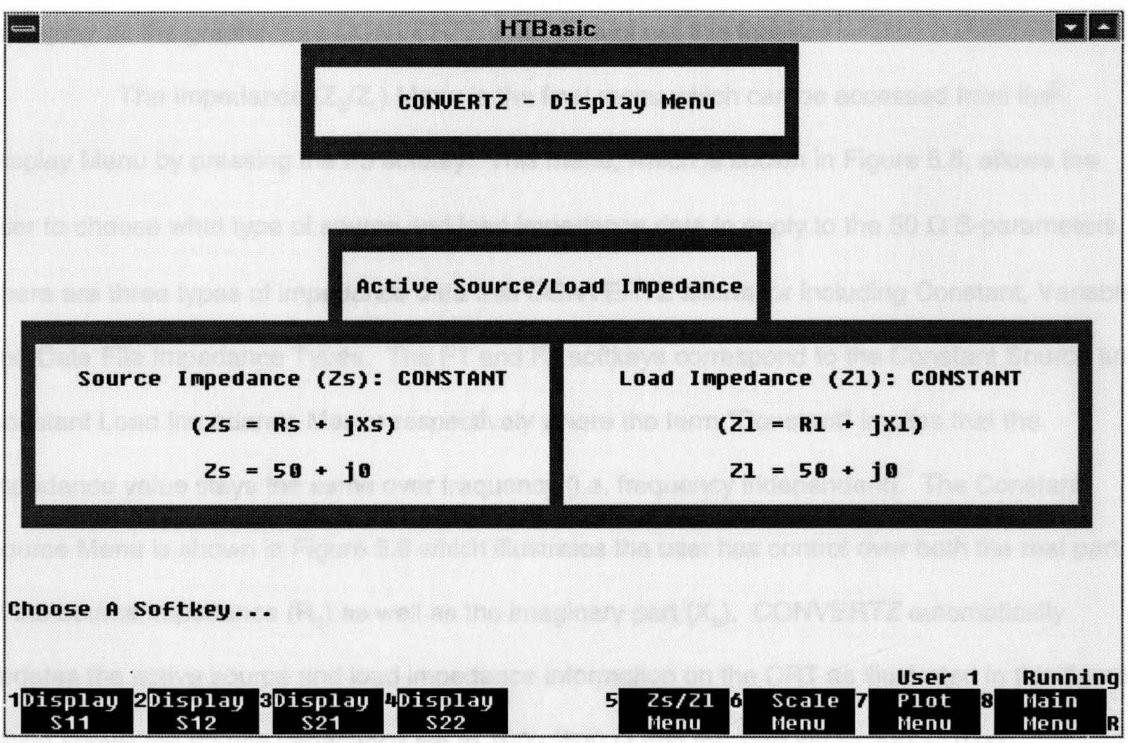

Figure 5.4 Display Menu of CONVERTZ

data is then uploaded back to the network analyzer for viewing and formatting. By uploading the data to the network analyzer in this manner, a special display and plotting routine did not have to be incorporated into CONVERTZ resulting in significant time savings.

When CONVERTZ uploads the new S-parameter data to the network analyzer, it automatically issues an "AUTOSCALE" command which scales the data to best fit in the display area. If the user desires to modify the scaling to enable easier comparison to previous data, for example, the Scale Menu can be entered by pressing the F6 softkey. This gives the user control of the various scaling functions built into the network analyzer such as setting the reference line position or value as well as the number of S-parameter units per division. Once the user has the data scaled as desired, the Plot Menu can be entered with the F7 softkey. This menu allows the user to plot the data, as displayed on the network analyzer's CRT, directly to any plotter which understands the Hewlett-Packard Graphics Language (HP-GL). From this menu the user can also choose to route the plotter information to an ASCII data file instead of a plotter which can then be imported into any PC software package with a HP-GL import filter. This feature was used to display all the graphs from CONVERTZ used throughout this thesis.

The Impedance (Z_S/Z_L) Menu is the final menu which can be accessed from the Display Menu by pressing the F5 softkey. This menu, which is shown in Figure 5.5, allows the user to choose what type of source and load impedance data to apply to the 50 Ω S-parameters. There are three types of impedance data that CONVERTZ allows for including Constant, Variable, and Data File Impedance Types. The F1 and F2 softkeys correspond to the Constant Source and Constant Load Impedance Menus respectively where the term "Constant" implies that the impedance value stays the same over frequency (i.e. frequency independent). The Constant Source Menu is shown in Figure 5.6 which illustrates the user has control over both the real part of the source impedance (R_S) as well as the imaginary part (X_S). CONVERTZ automatically updates the active source and load impedance information on the CRT as illustrated in this figure which shows the source impedance set to 100 - j590 Ω and the load set to 300 + j10 Ω.

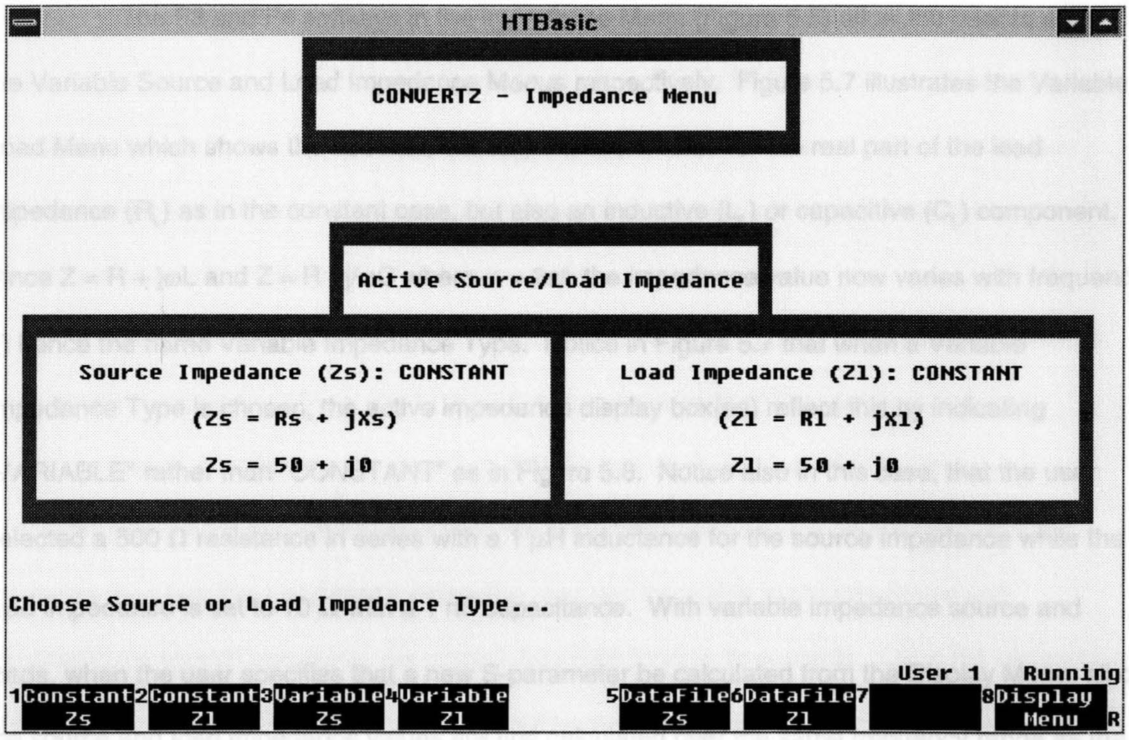

Figure 5.5 Impedance Menu of CONVERTZ

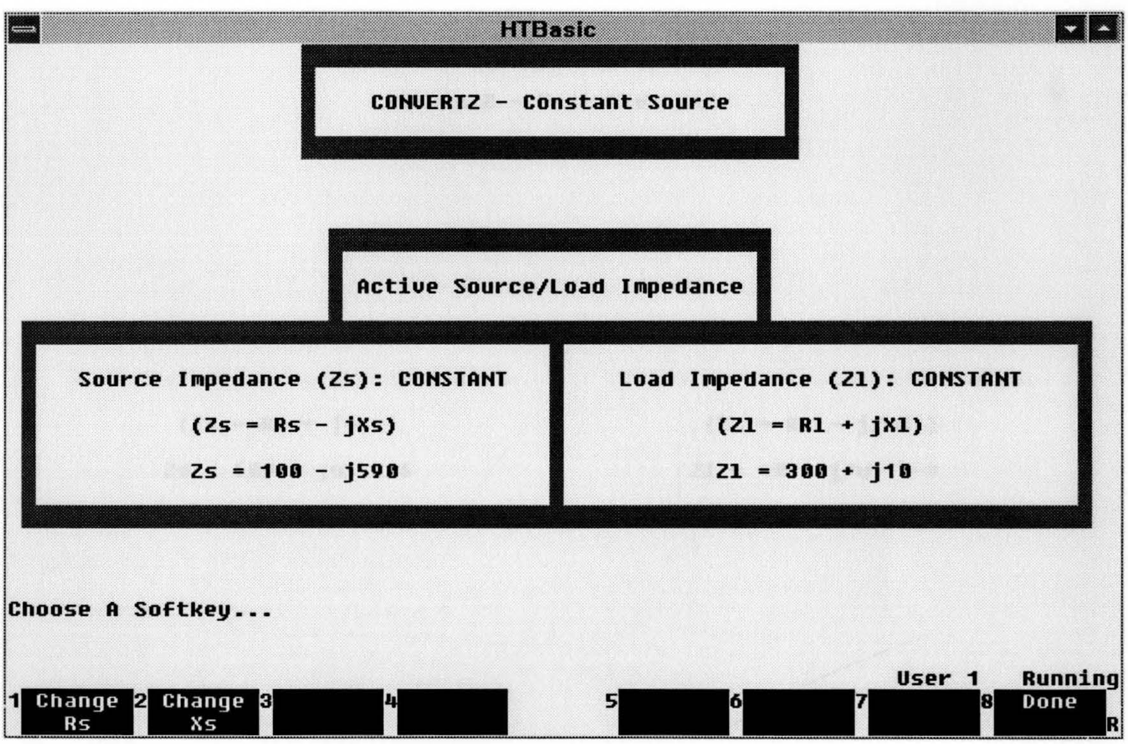

Figure 5.6 Constant Source Menu of CONVERTZ

The F3 and F4 softkeys in the Impedance Menu (Figure 5.5) allow the user to enter the Variable Source and Load Impedance Menus respectively. Figure 5.7 illustrates the Variable Load Menu which shows the user can not only choose a value for the real part of the load impedance (R_L) as in the constant case, but also an inductive (L_L) or capacitive (C_L) component. Since $Z = R + j\omega L$ and $Z = R - j/\omega C$ where $\omega = 2\pi f$, the impedance value now varies with frequency (f) hence the name Variable Impedance Type. Notice in Figure 5.7 that when a Variable Impedance Type is chosen, the active impedance display box(es) reflect this by indicating "VARIABLE" rather than "CONSTANT" as in Figure 5.6. Notice also in this case, that the user selected a 500 Ω resistance in series with a 1 μH inductance for the source impedance while the load impedance is set to 10 Ω with a 1 nF capacitance. With variable impedance source and loads, when the user specifies that a new S-parameter be calculated from the Display Menu, all of the source and load impedance values are first calculated over the same frequency range as the 50 Ω S-parameters. These impedance values along with the 50 Ω S-parameters are then utilized

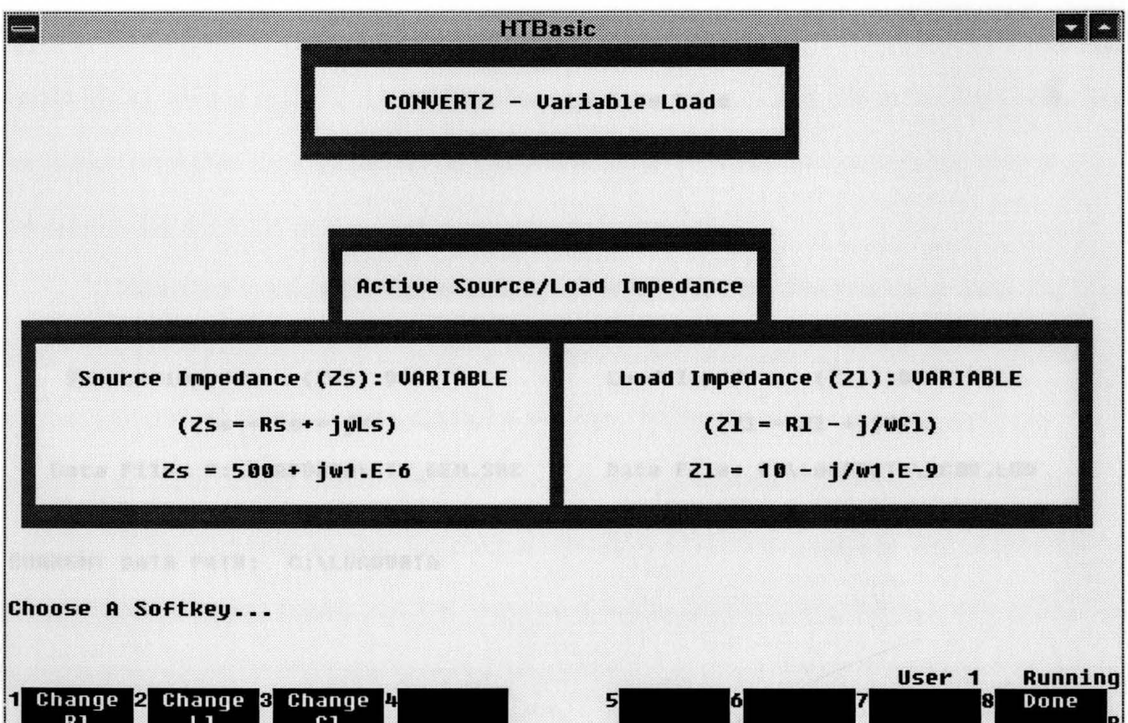

Figure 5.7 Variable Load Menu of CONVERTZ

in the conversion equations of Chapter IV to form the new S-parameter values displayed on the network analyzer. This Variable Impedance Type gives CONVERTZ much more flexibility over the Constant Impedance Type, especially for Delphi Packard since many of the loads in the automotive environment include a section of wiring harness which is largely inductive and therefore varies with frequency.

The final impedance type, and perhaps the most useful, is the Data File Impedance Type which consists of a file stored on the PC with measured impedance values in real / imaginary format over frequency. The Data File Source and Load Impedance Menus can be accessed with the F5 and F6 softkeys from the Impedance Menu in Figure 5.5. Figure 5.8 illustrates the Data File Load Menu where the user has the ability to change drives and subdirectories as well as catalog the current data path to view its contents. Once the desired path is set, as viewed below the active source impedance box, the user can press the F1 softkey to input the filename which contains the desired impedance values. Notice in this figure that the

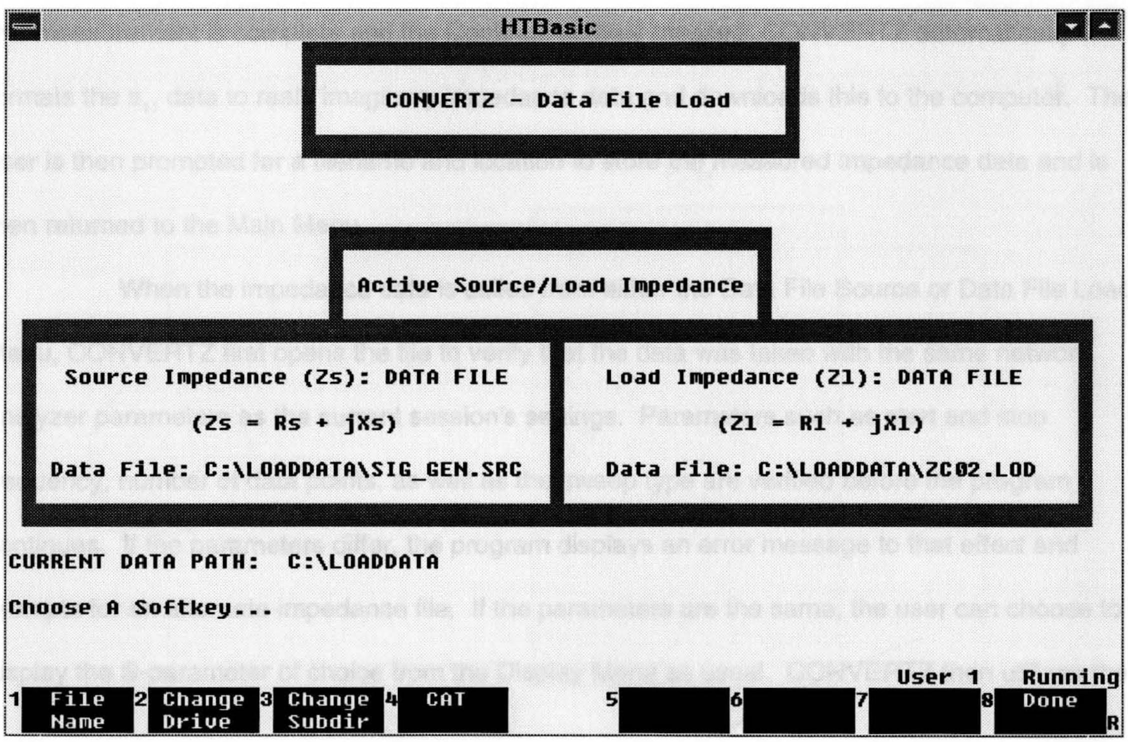

Figure 5.8 Data File Load Menu of CONVERTZ

active impedance display boxes now contain "DATA FILE" rather than "VARIABLE" or "CONSTANT". Also note in this example that the source impedance data file is located in C:\LOADDATA and is called SIG_GEN.SRC while the load impedance data file is in the same location but called ZC02.LOD. Once the desired data files are inputted, the user can press the Done softkey (F8) which returns the user to the Impedance Menu.

The impedance data files are created by using the impedance measurement capabilities of the network analyzer as briefly mentioned earlier. The network analyzer calculates impedance by taking the corrected s_{11} data and applying the formula

$$Z = Z_O * \frac{1 + s_{11}}{1 - s_{11}} \tag{61}$$

where Z_O is the system characteristic impedance which is 50 Ω for the HP 8753C. Recall from the Main Menu in Figure 5.2 that the F5 softkey allows the user to measure the input impedance of the connected device. When this softkey is pressed, the user is prompted to make a calibrated s_{11} measurement of the desired source or load and store the data in the memory of Channel 1. When this measurement is complete and the Continue softkey pressed, CONVERTZ automatically formats the s_{11} data to real / imaginary impedance data and downloads this to the computer. The user is then prompted for a filename and location to store the measured impedance data and is then returned to the Main Menu.

When the impedance data is called from either the Data File Source or Data File Load Menu, CONVERTZ first opens the file to verify that the data was taken with the same network analyzer parameters as the current session's settings. Parameters such as start and stop frequency, number of data points, as well as the sweep type are verified before the program continues. If the parameters differ, the program displays an error message to that effect and prompts for an alternate impedance file. If the parameters are the same, the user can choose to display the S-parameter of choice from the Display Menu as usual. CONVERTZ then utilizes the impedance values from the data files to apply to the previously measured 50 Ω S-parameters in order to calculate the new S-parameters.

It should be noted that the examples used in Figures 5.6 - 5.8 for the various types of impedance data show both the source and load impedance type being the same. For example, Figure 5.7 is an illustration of the Variable Load Menu which shows both the source and load impedance types as "VARIABLE". Due to the flexibility of CONVERTZ, mixing of impedance types is completely valid so it is possible, for instance, to have a "CONSTANT" source impedance while using a "DATA FILE" for the load impedance values.

5.2 Example Data from CONVERTZ

Now that there is a flexible tool developed which converts existing 50 Ω S-parameter data to arbitrary source and load S-parameter data, lets connect a real device to the network analyzer and look at some output generated by CONVERTZ. The simplistic DUT design shown in Figure 5.9 was constructed using two 23 Ω SMD (surface mount device) resistors. The network analyzer parameters were then set and a two-port full calibration was performed as CONVERTZ

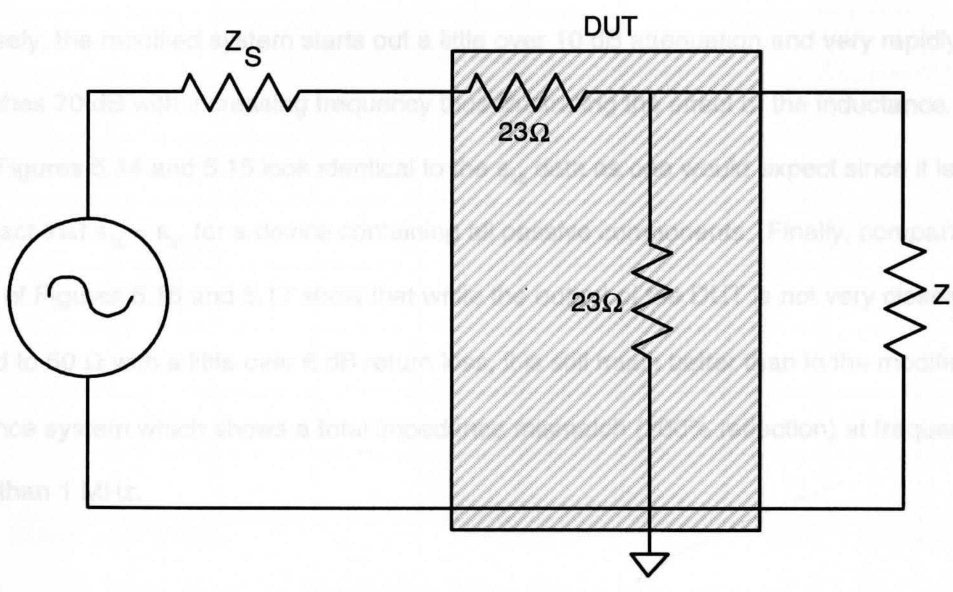

Figure 5.9 Resistive DUT for CONVERTZ Demonstration

instructs. After the program finished downloading the 50 Ω S-parameters and entered the Display Menu, all four of the S-parameters were consecutively displayed using the default 50 Ω source and load impedance values. These graphs of s_{11}, s_{12}, s_{21}, and s_{22} are shown in Figures 5.10, 5.12, 5.14, and 5.16 respectively. Next, the source impedance was changed to a constant 500 + j0 Ω while the load impedance was changed to a variable 10 + jω10e-6 Ω and the four S-parameters were again calculated. The graphs for the modified s_{11}, s_{12}, s_{21}, and s_{22} are shown in Figures 5.11, 5.13, 5.15, and 5.17 respectively. Notice in the top left corner of each graph that CONVERTZ uses the graphic capabilities of the 8753C to print specific information about the data. The first line specifies the S-parameter which is presently displayed while the next two lines contain the active source and load impedance information that the DUT is characterized with.

Comparing the S-parameters measured in the 50 Ω system to the S-parameters in the modified impedance system of above, one can clearly see the dramatic difference that source and load impedance make. Looking at the s_{11} data in Figures 5.10 and 5.11 show that the input impedance of the DUT is much more closely matched to the 50 Ω system than the modified impedance system which is critical for maximizing power transfer. Figures 5.12 and 5.13 compare the s_{12} data of the DUT showing less than 10 dB attenuation over frequency in the 50 Ω system. Conversely, the modified system starts out a little over 10 dB attenuation and very rapidly approaches 70 dB with increasing frequency thus illustrating the effect of the inductance. The s_{21} data in Figures 5.14 and 5.15 look identical to the s_{12} data as one would expect since it is a well known fact that $s_{12} = s_{21}$ for a device containing all passive components. Finally, comparing the s_{22} data of Figures 5.16 and 5.17 show that while the output of the DUT is not very closely matched to 50 Ω with a little over 6 dB return loss, it is still much better than in the modified impedance system which shows a total impedance mismatch (100% reflection) at frequencies greater than 1 MHz.

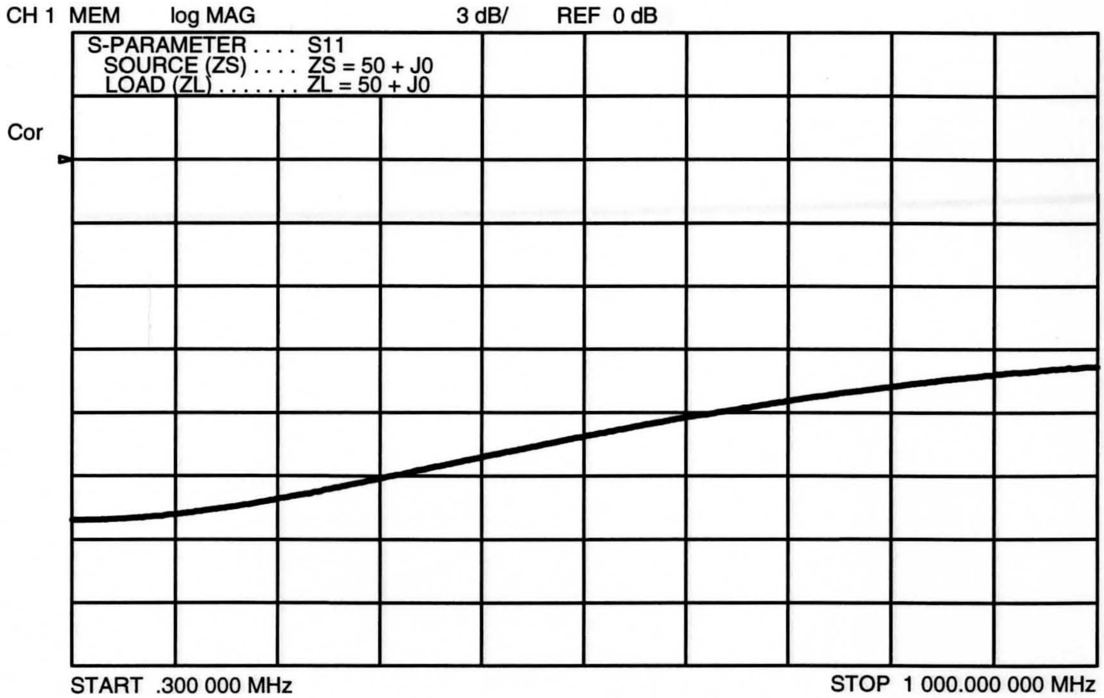

Figure 5.10 S_{11} Graph of Resistive DUT in Figure 5.9 with $Z_S = Z_L = 50\ \Omega$

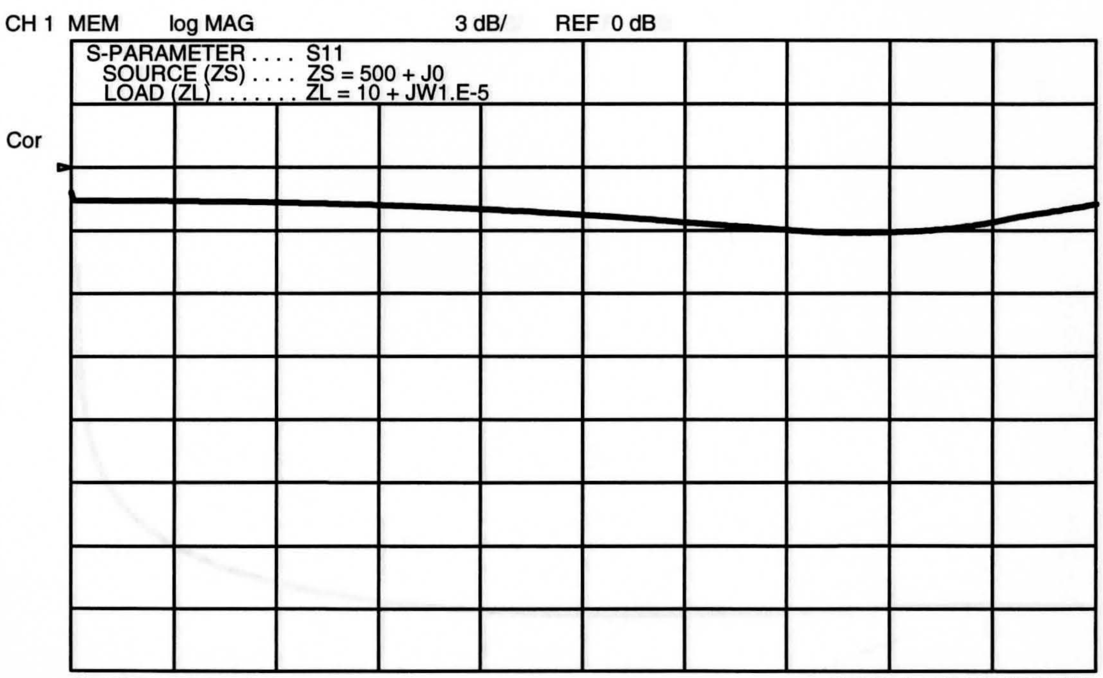

Figure 5.11 S_{11} Graph of Resistive DUT in Figure 5.9 with $Z_S = 500 + j0\ \Omega$ and $Z_L = 10 + j\omega 10e\text{-}6\ \Omega$

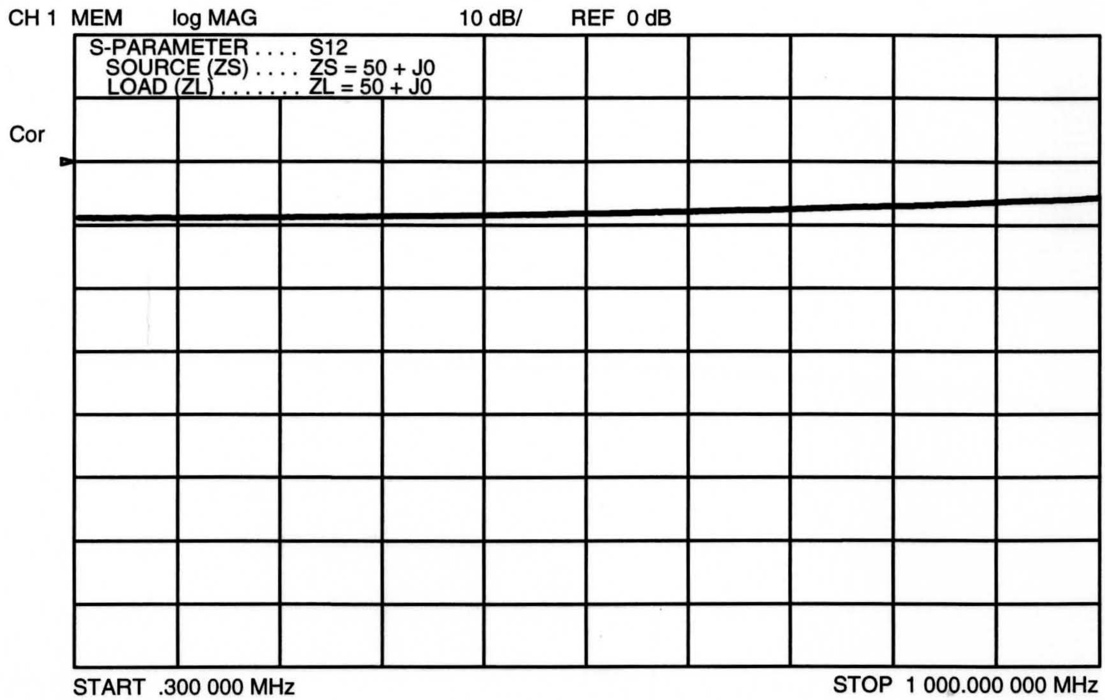

Figure 5.12 S_{12} Graph of Resistive DUT in Figure 5.9 with $Z_S = Z_L = 50\ \Omega$

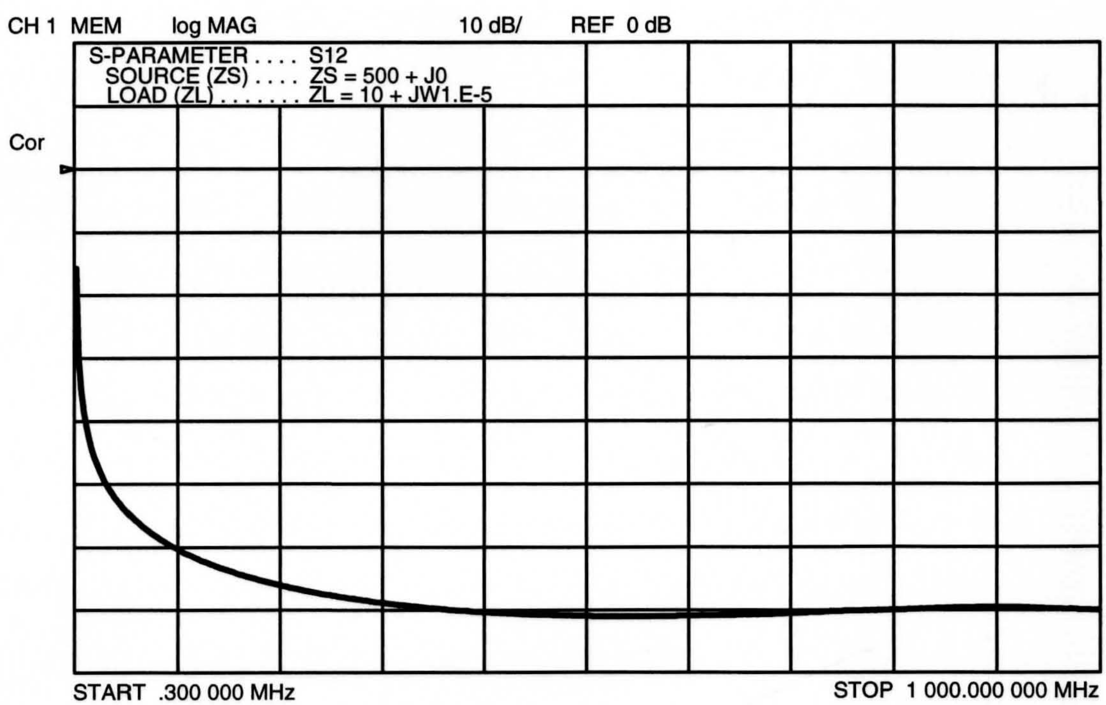

Figure 5.13 S_{12} Graph of Resistive DUT in Figure 5.9 with $Z_S = 500 + j0\ \Omega$ and $Z_L = 10 + j\omega 10e\text{-}6\ \Omega$

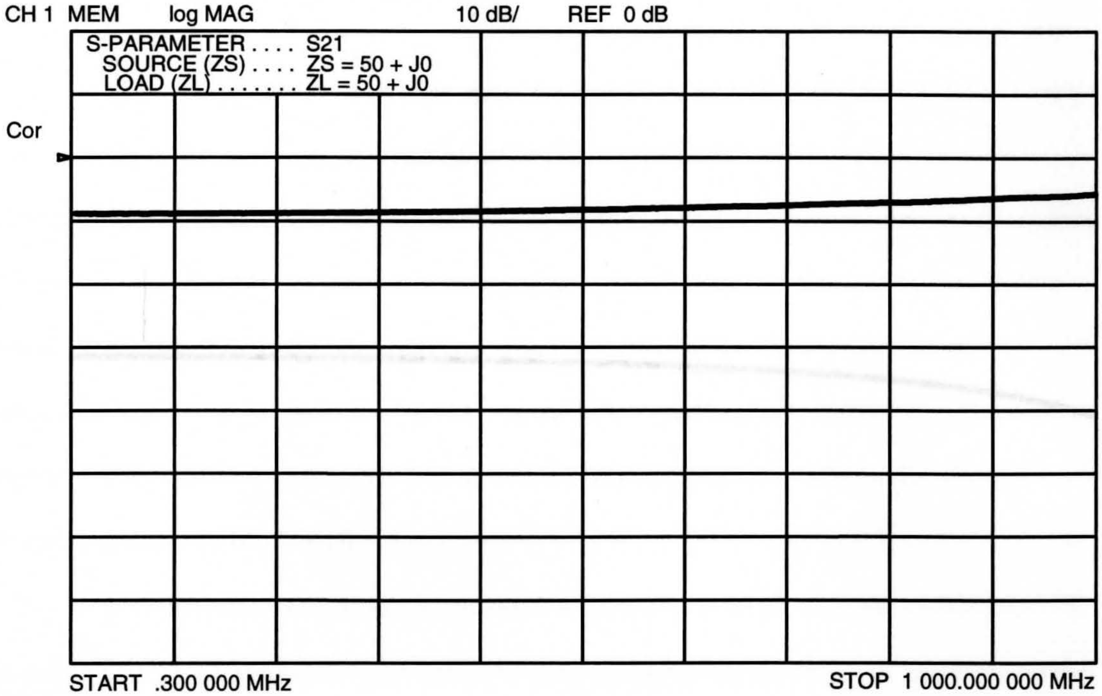

Figure 5.14 S_{21} Graph of Resistive DUT in Figure 5.9 with $Z_S = Z_L = 50\ \Omega$

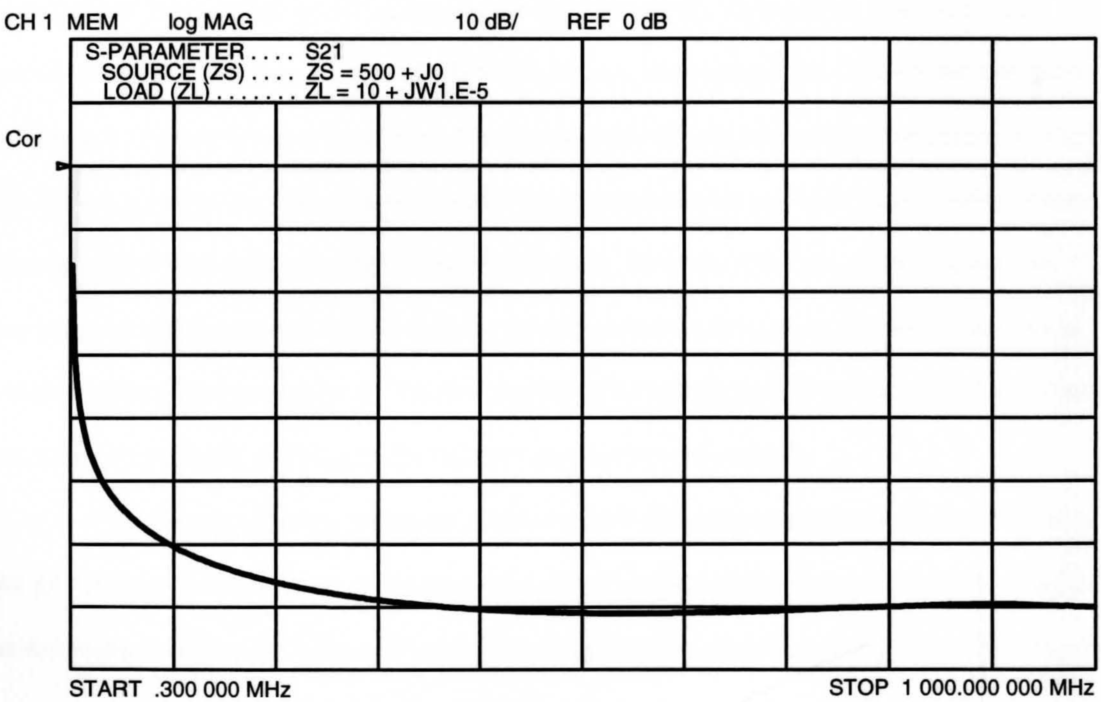

Figure 5.15 S_{21} Graph of Resistive DUT in Figure 5.9 with $Z_S = 500 + j0\ \Omega$ and $Z_L = 10 + j\omega 10e\text{-}6\ \Omega$

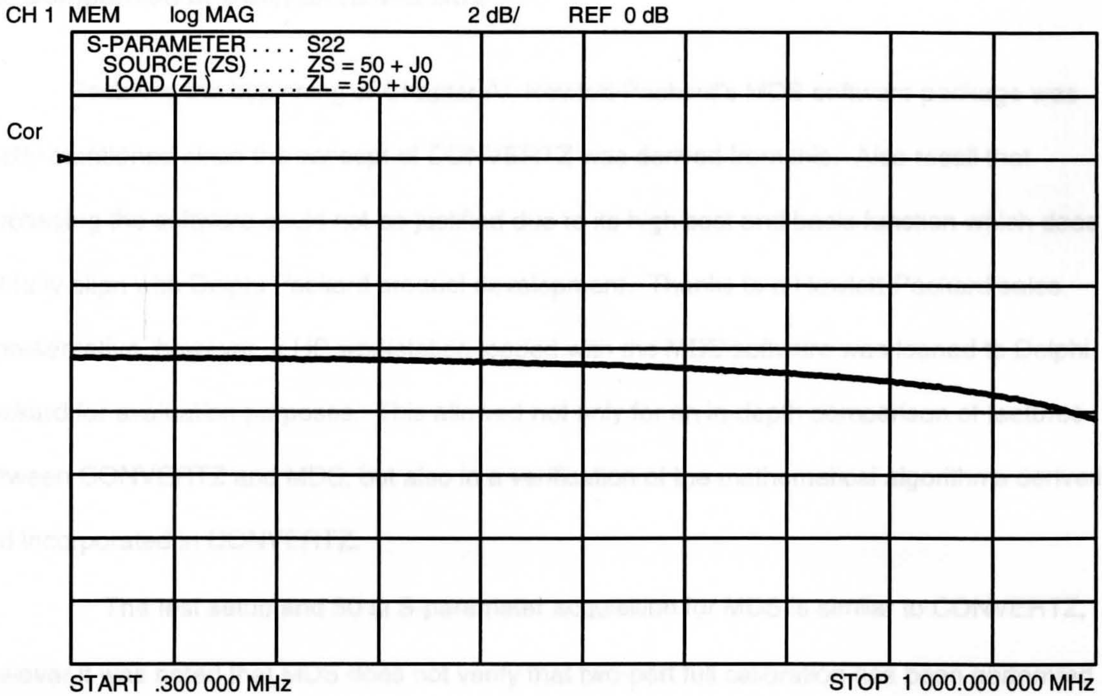

Figure 5.16 S_{22} Graph of Resistive DUT in Figure 5.9 with $Z_S = Z_L = 50\ \Omega$

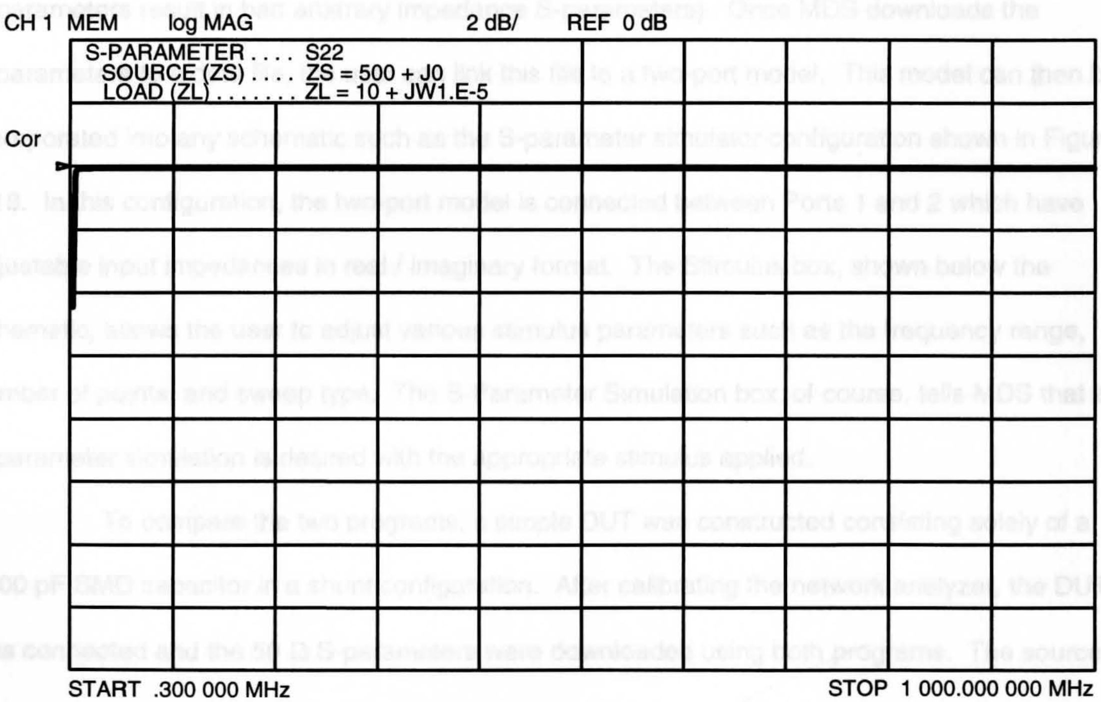

Figure 5.17 S_{22} Graph of Resistive DUT in Figure 5.9 with $Z_S = 500 + j0\ \Omega$ and $Z_L = 10 + j\omega 10e\text{-}6\ \Omega$

5.3 Comparison of CONVERTZ and MDS

Recalling the beginning of Chapter IV, Hewlett-Packard's MDS software package was briefly mentioned since the concept of CONVERTZ was derived from this. Also recall that purchasing the software could not be justified due to its high cost and basic function which does not fully align with Delphi Packard product development. Thanks to a Hewlett-Packard sales representative, however, a HP workstation loaded with the MDS software was loaned to Delphi Packard for evaluation purposes. This allowed not only for an in-depth comparison of features between CONVERTZ and MDS, but also in a verification of the mathematical algorithms derived and incorporated in CONVERTZ.

The test setup and 50 Ω S-parameter acquisition for MDS is similar to CONVERTZ, however it was noted that MDS does not verify that two-port full calibration has been performed before the 50 Ω data is downloaded. As mentioned earlier, the accuracy of the modified S-parameter data is highly dependent on the accuracy of the 50 Ω S-parameters (i.e. bad 50 Ω S-parameters result in bad arbitrary impedance S-parameters). Once MDS downloads the S-parameters to a data file, the user can link this file to a two-port model. This model can then be incorporated into any schematic such as the S-parameter simulator configuration shown in Figure 5.18. In this configuration, the two-port model is connected between Ports 1 and 2 which have adjustable input impedances in real / imaginary format. The Stimulus box, shown below the schematic, allows the user to adjust various stimulus parameters such as the frequency range, number of points, and sweep type. The S-Parameter Simulation box, of course, tells MDS that a S-parameter simulation is desired with the appropriate stimulus applied.

To compare the two programs, a simple DUT was constructed consisting solely of a 1000 pF SMD capacitor in a shunt configuration. After calibrating the network analyzer, the DUT was connected and the 50 Ω S-parameters were downloaded using both programs. The source impedance for the MDS schematic was changed from its 50 Ω default to 10 + j200 Ω and the load impedance was changed to 500 - j1500 Ω as shown in Figure 5.18. An S-parameter simulation

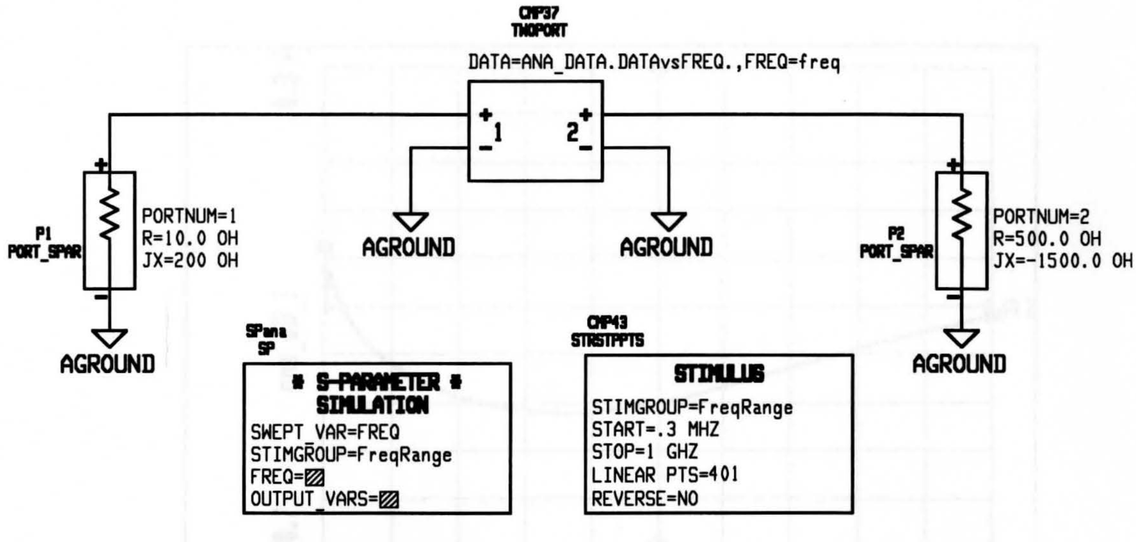

Figure 5.18 Schematic Interface of the MDS S-Parameter Simulator

was then run from 300 kHz to 1 GHz. The four S-parameters were also calculated using CONVERTZ with the same source and load impedance applied as above.

Figures 5.19 and 5.20 contain the s_{11} data from the 1000 pF DUT comparing the MDS simulation data with the CONVERTZ data respectively. Figures 5.21 and 5.22 compare the s_{12} data while Figures 5.23 and 5.24 compare the s_{21} data. Finally, Figures 5.25 and 5.26 compare the s_{22} data obtained from both programs. It should be pointed out that the MDS graphs and the CONVERTZ graphs have identical scaling for ease of comparison. As can be seen from these figures, the MDS and CONVERTZ data look identical resulting in a very high confidence level in derived algorithms as well as their implementation in this thesis. Although the two programs generate the same results, CONVERTZ does have some notable advantages over MDS in terms of user-friendliness and flexibility. Since MDS is not specifically designed for S-parameter conversion as CONVERTZ is, the interface is more difficult to learn and use resulting in large amounts of training time for a new user. CONVERTZ, on the other hand takes no longer than 20 minutes to learn assuming familiarity with a network analyzer.

Another advantage of CONVERTZ over MDS is the display format for the S-parameter graphs. As can be seen from Figures 5.19 - 5.26, CONVERTZ displays only the pertinent

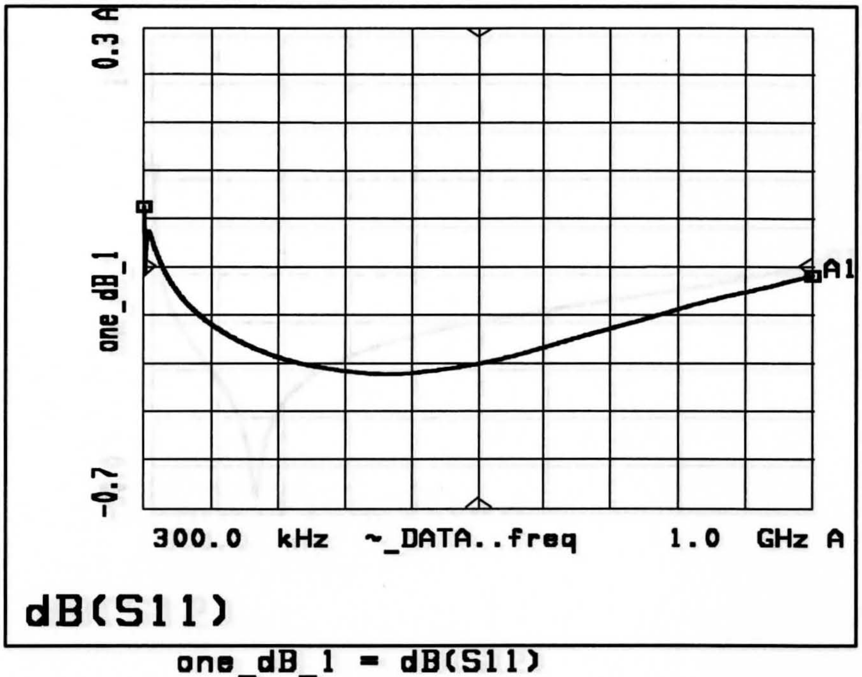

Figure 5.19 S_{11} Graph from MDS of 1000 pF Shunt Capacitor with Z_S = 10 + j200 Ω and Z_L = 500 - j1500 Ω

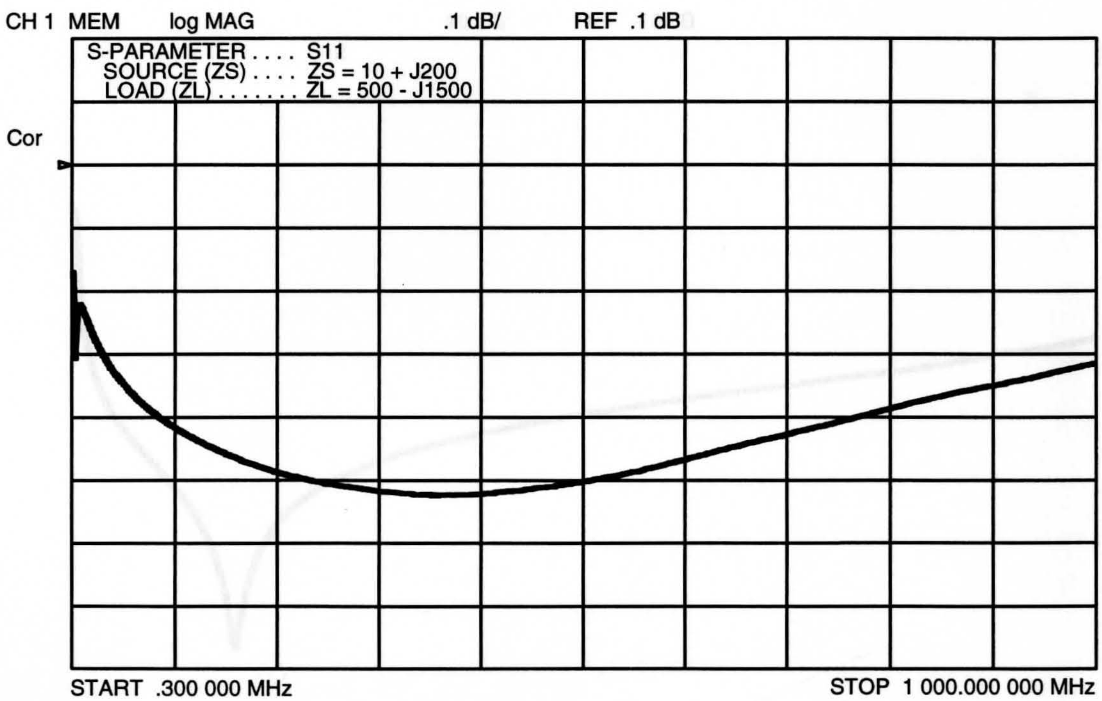

Figure 5.20 S_{11} Graph from CONVERTZ of 1000 pF Shunt Capacitor with Z_S = 10 + j200 Ω and Z_L = 500 - j1500 Ω

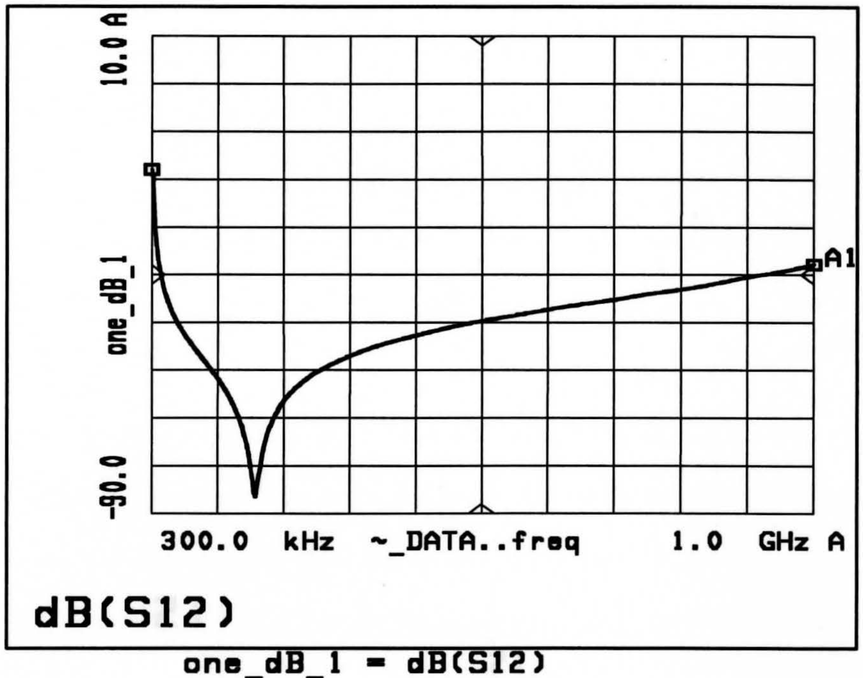

Figure 5.21 S_{12} Graph from MDS of 1000 pF Shunt Capacitor with Z_S = 10 + j200 Ω and Z_L = 500 - j1500 Ω

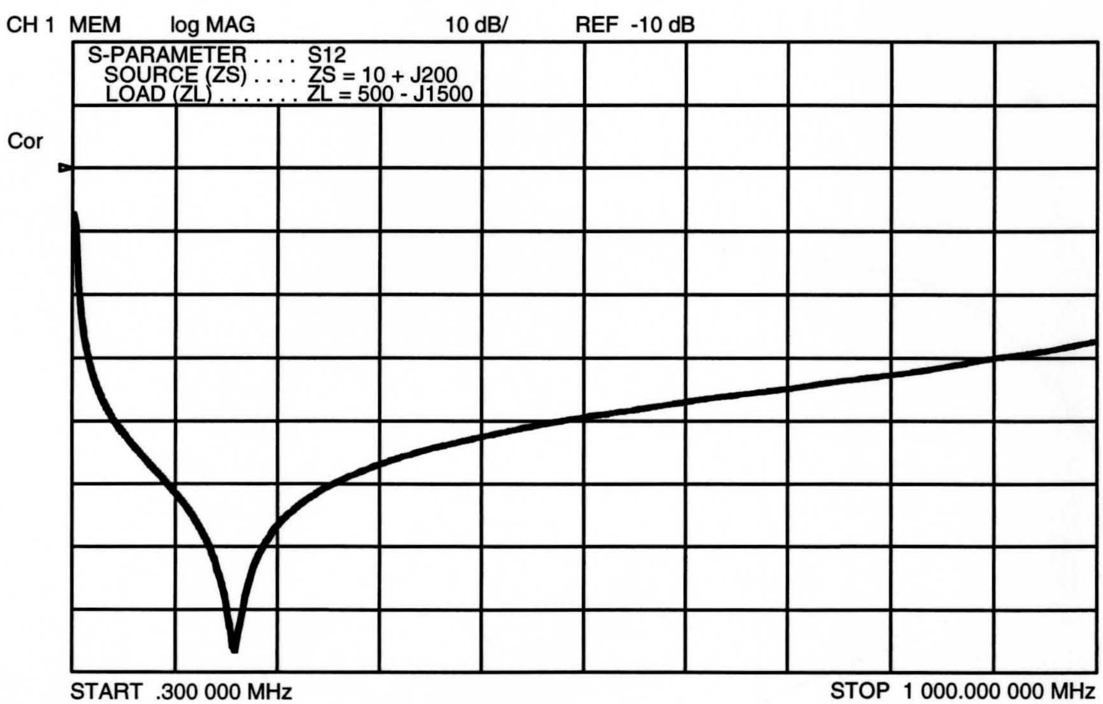

Figure 5.22 S_{12} Graph from CONVERTZ of 1000 pF Shunt Capacitor with Z_S = 10 + j200 Ω and Z_L = 500 - j1500 Ω

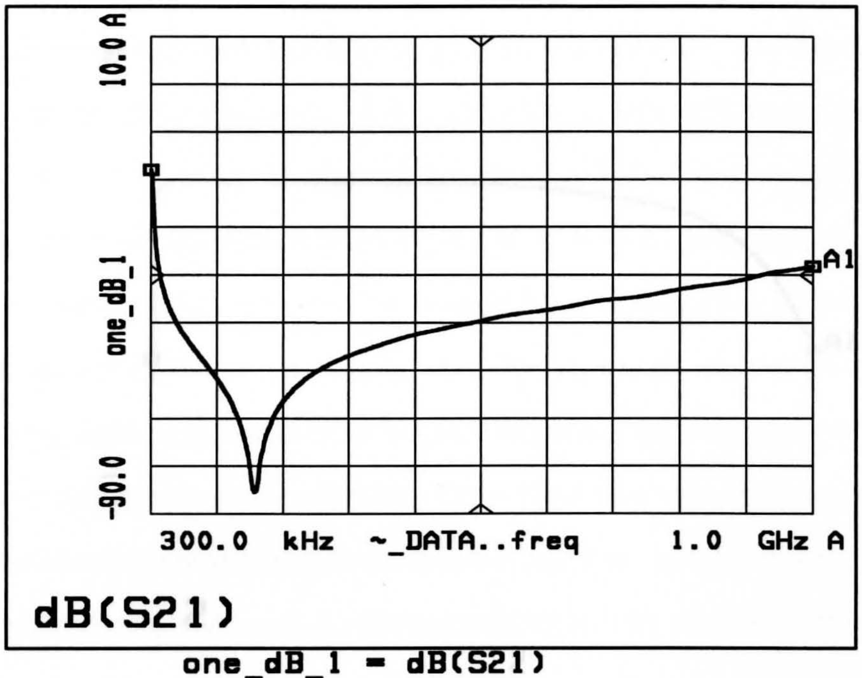

Figure 5.23 S_{21} Graph from MDS of 1000 pF Shunt Capacitor with Z_S = 10 + j200 Ω and Z_L = 500 - j1500 Ω

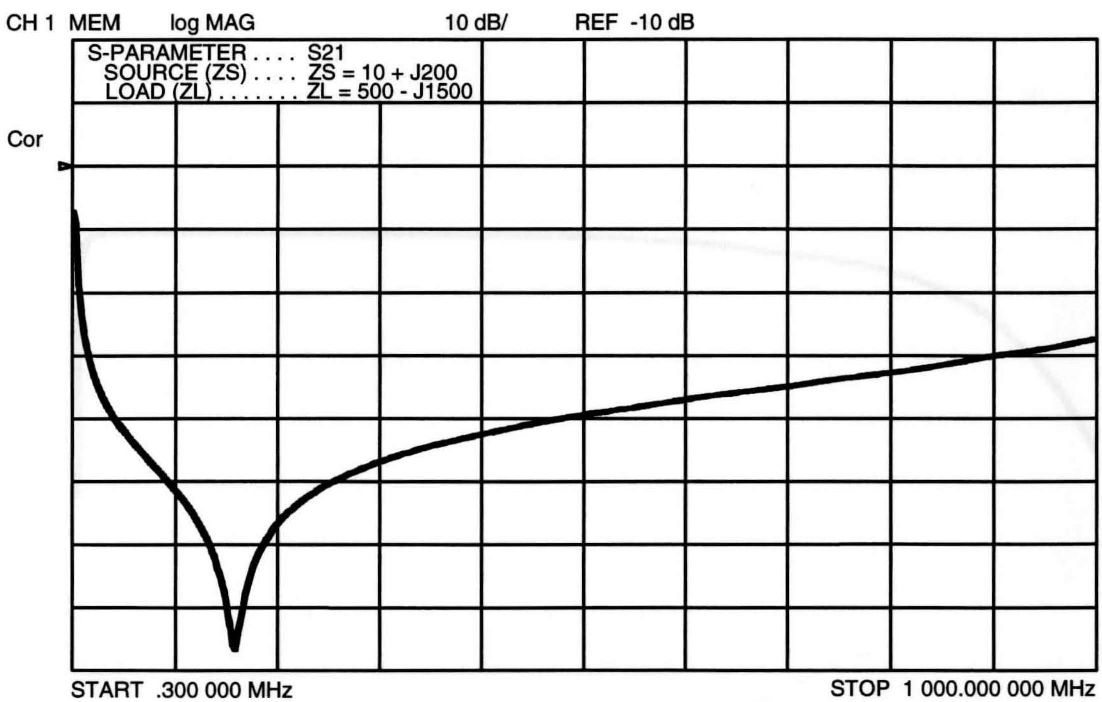

Figure 5.24 S_{21} Graph from CONVERTZ of 1000 pF Shunt Capacitor with Z_S = 10 + j200 Ω and Z_L = 500 - j1500 Ω

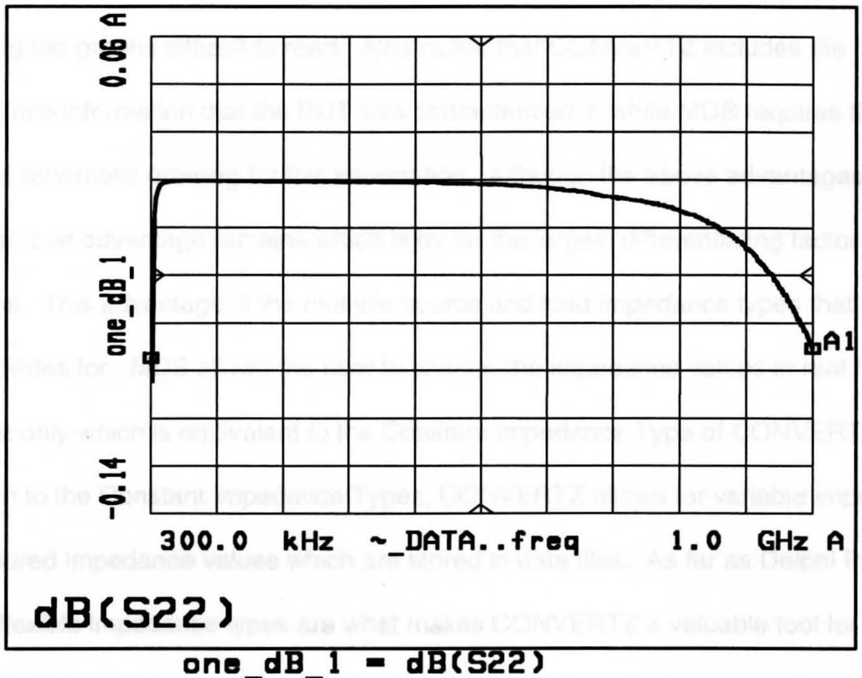

Figure 5.25 S_{22} Graph from MDS of 1000 pF Shunt Capacitor with $Z_S = 10 + j200\ \Omega$ and $Z_L = 500 - j1500\ \Omega$

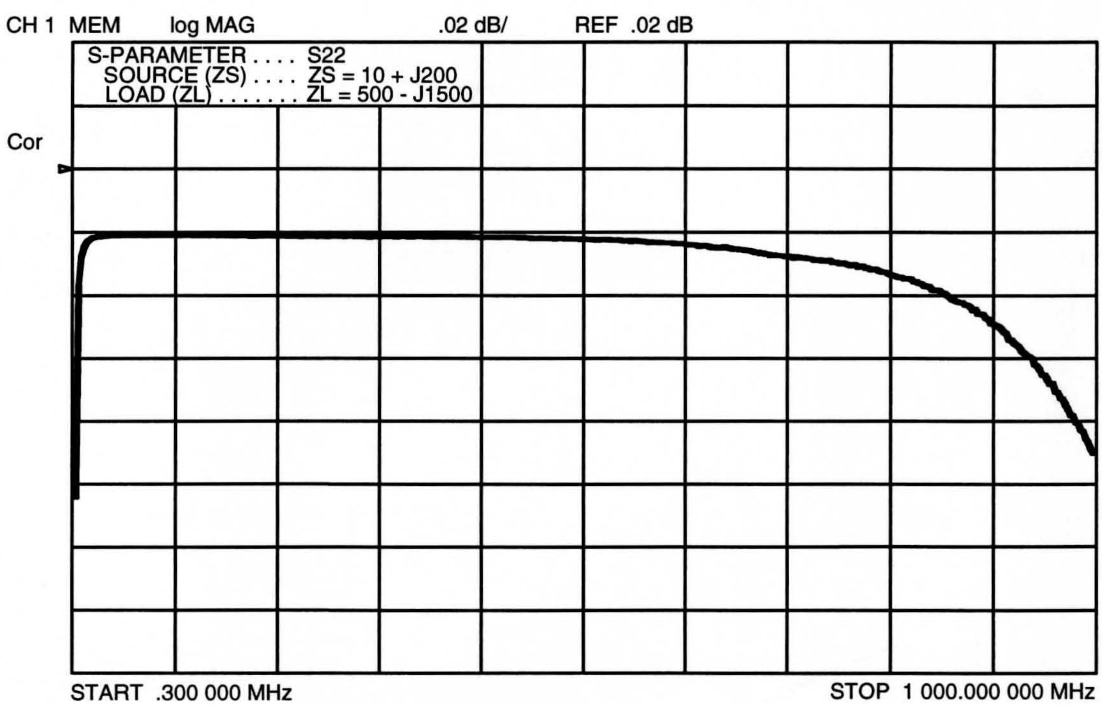

Figure 5.26 S_{22} Graph from CONVERTZ of 1000 pF Shunt Capacitor with $Z_S = 10 + j200\ \Omega$ and $Z_L = 500 - j1500\ \Omega$

information unlike MDS which includes a lot of extraneous information about data files and plot templates making the graphs difficult to read. Also notice that CONVERTZ includes the source and load impedance information that the DUT was characterized in while MDS requires the user to go back to the schematic drawing for this information. Although the above advantages may be considered trivial, one advantage remains which is by far the largest differentiating factor between the two programs. This advantage is the multiple source and load impedance types that CONVERTZ provides for. MDS allows the user to change the impedance values in real / imaginary format only which is equivalent to the Constant Impedance Type of CONVERTZ. Recall in addition to the Constant Impedance Types, CONVERTZ allows for variable impedances as well as measured impedance values which are stored in data files. As far as Delphi Packard is concerned, the flexible impedance types are what makes CONVERTZ a valuable tool for predicting a device's performance in the automotive environment.

CHAPTER VI

SAMPLE APPLICATIONS FOR CONVERTZ

6.1 Resistive Attenuator Design Example

Now that CONVERTZ has been developed and a high confidence level in its results has been obtained, a couple examples illustrating the usefulness of the program will next be discussed. The first example deals with a resistive attenuator design and verification using a network analyzer. A resistive attenuator is designed to absorb a specified amount of power while presenting a defined impedance at both the input (Z_{IN}) and output (Z_{OUT}). There are two common topologies for resistive attenuators which include the Pi and Tee where the Pi configuration, shown in Figure 6.1, will be used for this example. The equations for computing the element values for this attenuator are

$$\frac{1}{R_1} = \frac{1}{Z_{IN}}\left(\frac{A+1}{A-1}\right) - \frac{1}{R_3} \tag{62}$$

$$\frac{1}{R_2} = \frac{1}{Z_{OUT}}\left(\frac{A+1}{A-1}\right) - \frac{1}{R_3} \tag{63}$$

$$\frac{1}{R_3} = \frac{2}{(A-1)}\sqrt{\frac{A}{Z_{IN} Z_{OUT}}} \tag{64}$$

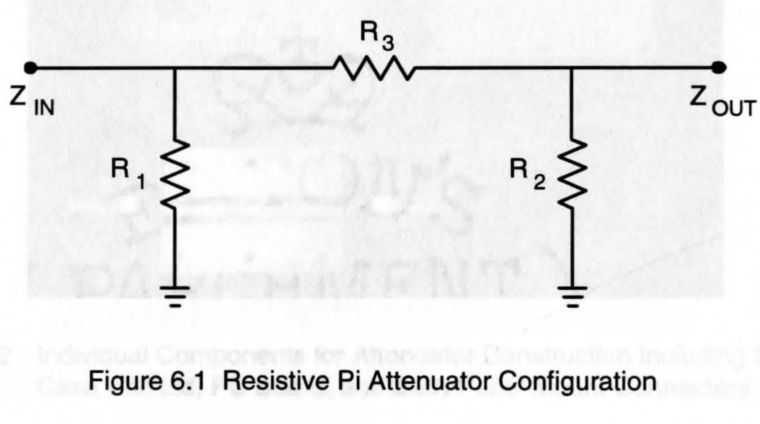

Figure 6.1 Resistive Pi Attenuator Configuration

where Z_{IN} and Z_{OUT} are the input and output impedance respectively and A is the desired attenuation as a numerical ratio of input to output power.[7] It should be noted that attenuation in dB equals 10 * $\log_{10}(A)$.

Lets first design a 10 dB attenuator for a 50 Ω system. According to equations (62), (63), and (64), if A is 10 and Z_{IN} and Z_{OUT} are 50, the theoretical values for R_1, R_2 and R_3 are 96 Ω, 96 Ω, and 71 Ω respectively. This attenuator was constructed using surface mount resistors with actual values of 100 Ω, 100 Ω, and 68 Ω since they were the closest values available. These resistors were soldered to a printed circuit (PC) board and placed in an electromagnetically shielded enclosure to minimize stray fields and extraneous body capacitance from interfering with the measurements. Figure 6.2 is a photo of the individual components used to construct the attenuator excluding the resistors. The shielded enclosure is milled from a single block of aluminum as well as the lid to minimize the number of gaps that allow RF energy to penetrate. The PC board is double-sided with a single microstrip line designed to maintain a 50 Ω characteristic impedance along its entire length. The two connectors are SMA panel mount type which are used to interface the 50 Ω coaxial test cables of the network analyzer to the 50 Ω microstrip on the PC board.

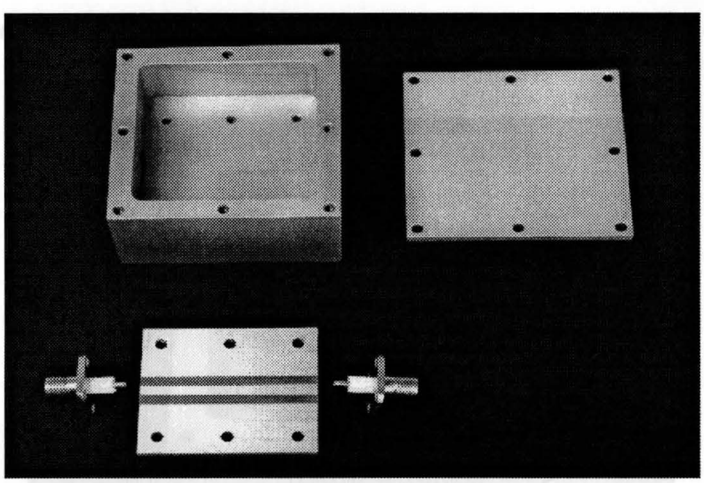

Figure 6.2 Individual Components for Attenuator Construction Including Shielded Case with Lid, PC Board, and SMA Panel Mount Connectors

The finished attenuator assembly with the lid removed is shown in Figure 6.3. This device was connected to the network analyzer test system after a two-port calibration was performed from 300 kHz to 500 MHz. The forward S-parameters, s_{11} and s_{21}, were then measured in the instrument's standard 50 Ω environment. The reverse S-parameters were not measured since this device is symmetrical, therefore $s_{11} = s_{22}$ and $s_{21} = s_{12}$. Figure 6.4 contains the s_{11} graph which shows the DUT has an excellent return loss meaning that the input impedance of the DUT matches very well to the 50 Ω source impedance of the analyzer. Good numbers for return loss range from 20 to 30 dB (or more). Figure 6.5 contains the s_{21} graph for the attenuator which shows slightly less than the 10 dB attenuation originally designed for. This can be explained by the small variance in actual resistor values from the theoretical ones derived from the equations. For clarity purposes, it should be explained that the s_{11} and s_{21} parameters shown in Figures 6.4 and 6.5 are actually negative values. However, when referring to s_{11} as return loss and s_{21} as attenuation, the negative sign is implied resulting in the positive values as used above.

As can be seen from the figures just described, a 10 dB attenuator designed for a 50 Ω system produces results very close to theory when measured with 50 Ω test equipment. For further illustration, however, another 10 dB attenuator was constructed except this time for a 10 Ω system rather than a 50 Ω. Using the equations in (62), (63), and (64), when A is 10 and Z_{IN} and Z_{OUT} are also 10, the theoretical values for R_1, R_2, and R_3 are 19 Ω, 19 Ω, and 14 Ω respectively.

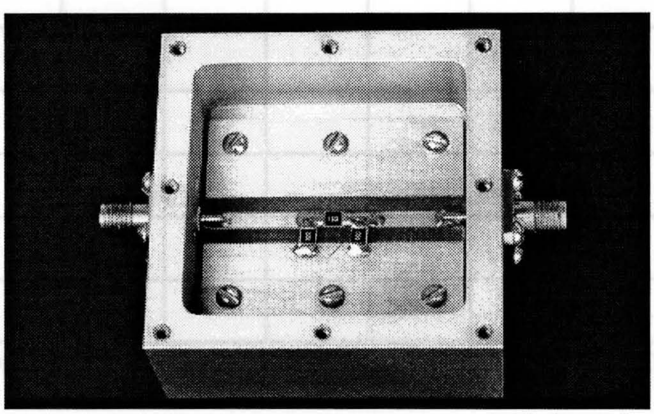

Figure 6.3 Assembled 10 dB Attenuator for CONVERTZ Experiment

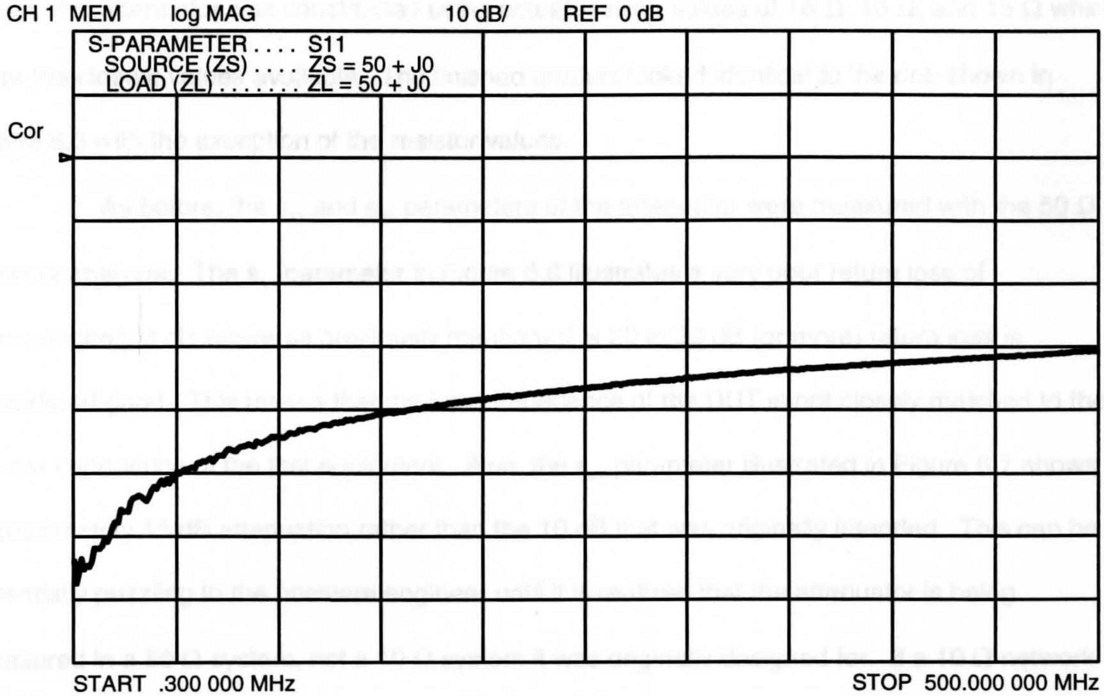

Figure 6.4 S_{11} of 10 dB Attenuator Designed for 50 Ω System and Measured with 50 Ω Test Equipment

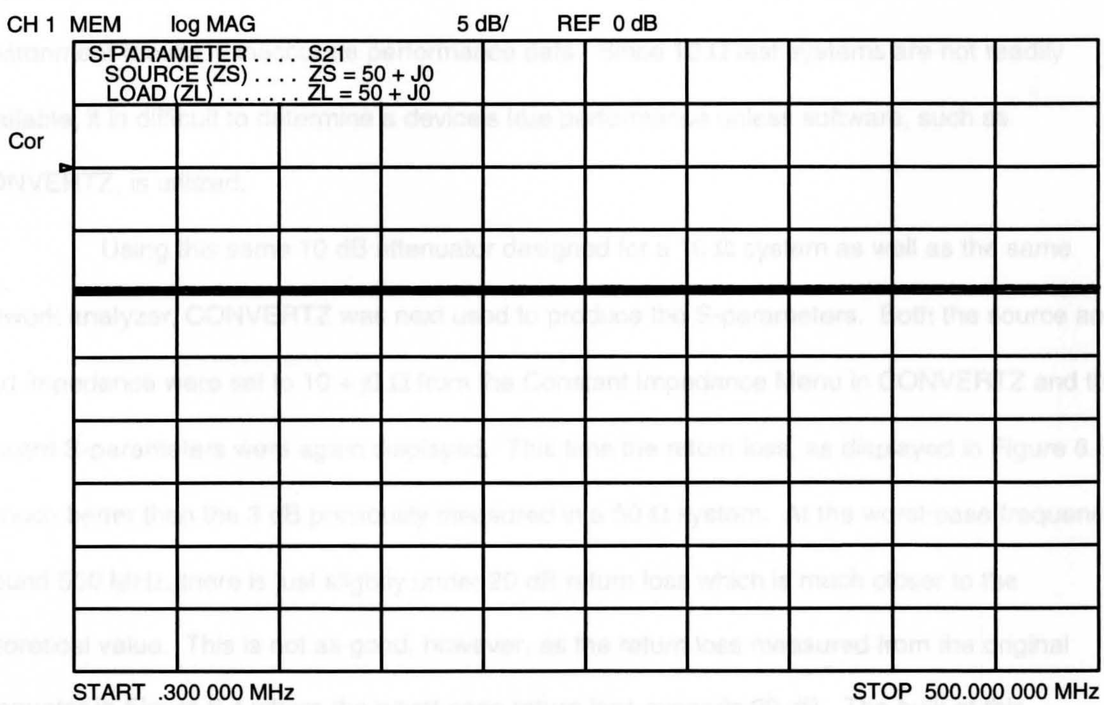

Figure 6.5 S_{21} of 10 dB Attenuator Designed for 50 Ω System and Measured with 50 Ω Test Equipment

Again, this attenuator was constructed using actual resistor values of 18 Ω, 18 Ω, and 15 Ω which were the closest values available. The finished product looked identical to the one shown in Figure 6.3 with the exception of the resistor values.

As before, the s_{11} and s_{21} parameters of the attenuator were measured with the 50 Ω network analyzer. The s_{11} parameter in Figure 6.6 illustrates a very poor return loss of approximately 3 dB where as previously mentioned, a 20 to 30 dB (or more) return loss is considered good. This means that the input impedance of the DUT is not closely matched to the source impedance of the test equipment. Also, the s_{21} parameter illustrated in Figure 6.7 shows approximately 15 dB attenuation rather than the 10 dB that was originally intended. This can be potentially puzzling to the unaware engineer until it is realized that the attenuator is being measured in a 50 Ω system, not a 10 Ω system it was originally designed for. If a 10 Ω network analyzer test system had been used to measure this attenuator, the s_{11} and s_{21} parameters would look much the same as the S-parameters for the first attenuator designed for a 50 Ω system. This helps illustrate the fact that a device measured in a test system not representative of the intended environment results in inaccurate performance data. Since 10 Ω test systems are not readily available, it is difficult to determine a device's true performance unless software, such as CONVERTZ, is utilized.

Using this same 10 dB attenuator designed for a 10 Ω system as well as the same network analyzer, CONVERTZ was next used to produce the S-parameters. Both the source and load impedance were set to 10 + j0 Ω from the Constant Impedance Menu in CONVERTZ and the forward S-parameters were again displayed. This time the return loss, as displayed in Figure 6.8, is much better than the 3 dB previously measured in a 50 Ω system. At the worst-case frequency around 500 MHz, there is just slightly under 20 dB return loss which is much closer to the theoretical value. This is not as good, however, as the return loss measured from the original attenuator in Figure 6.4 where the worst-case return loss exceeds 30 dB. The bulk of this discrepancy can be attributed to the 50 Ω characteristic impedance microstrip on the printed

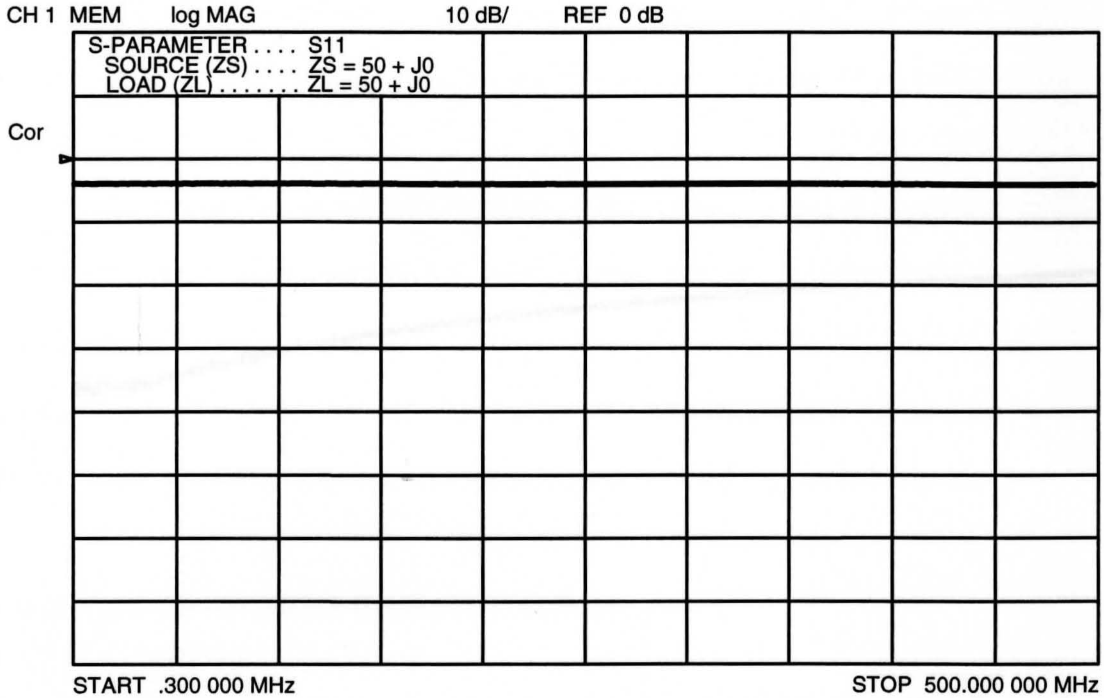

Figure 6.6 S_{11} of 10 dB Attenuator Designed for 10 Ω System and Measured with 50 Ω Test Equipment

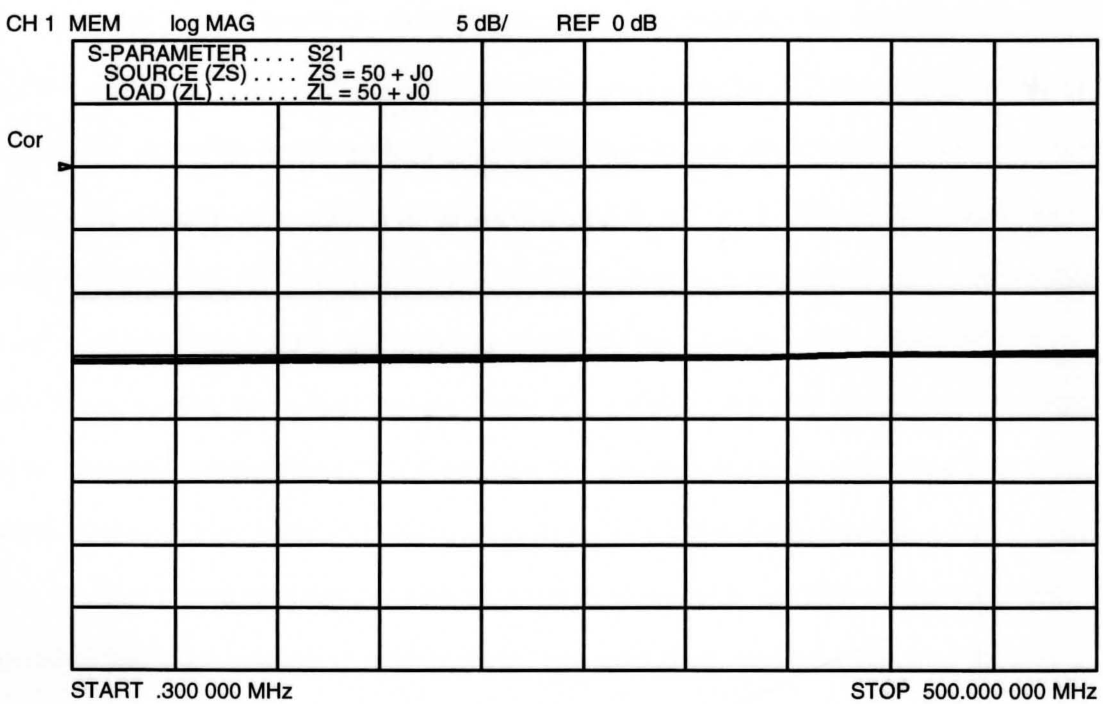

Figure 6.7 S_{21} of 10 dB Attenuator Designed for 10 Ω System and Measured with 50 Ω Test Equipment

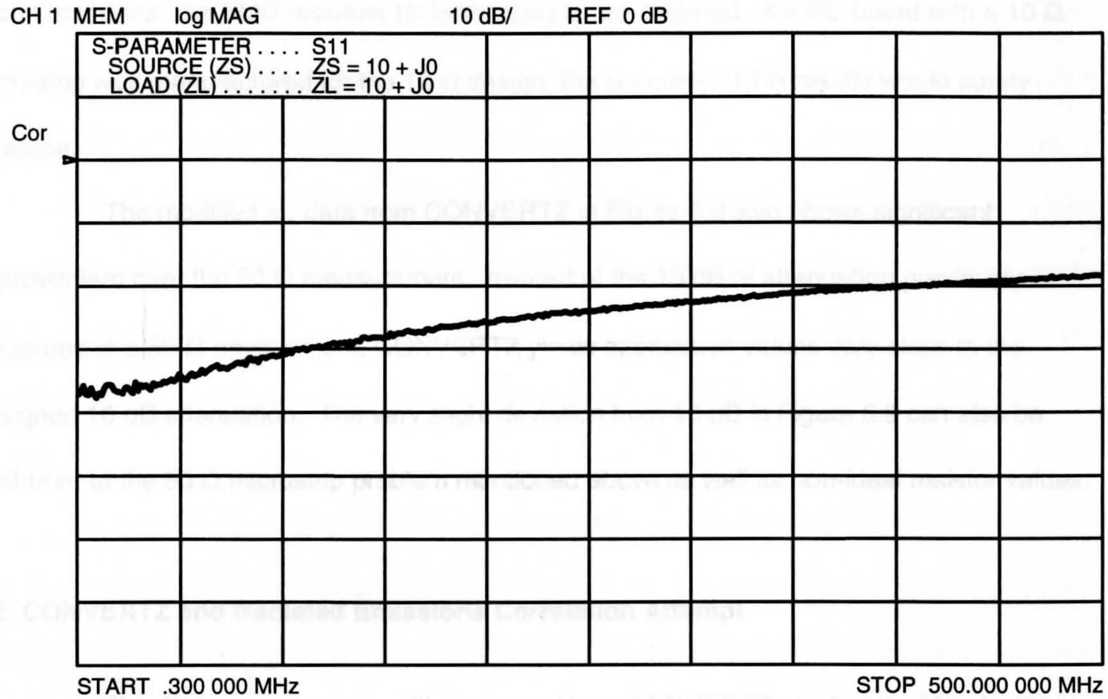

Figure 6.8 S_{11} from CONVERTZ of 10 dB Attenuator Designed for 10 Ω System and Measured with $Z_S = Z_L = 10 + j0\ \Omega$

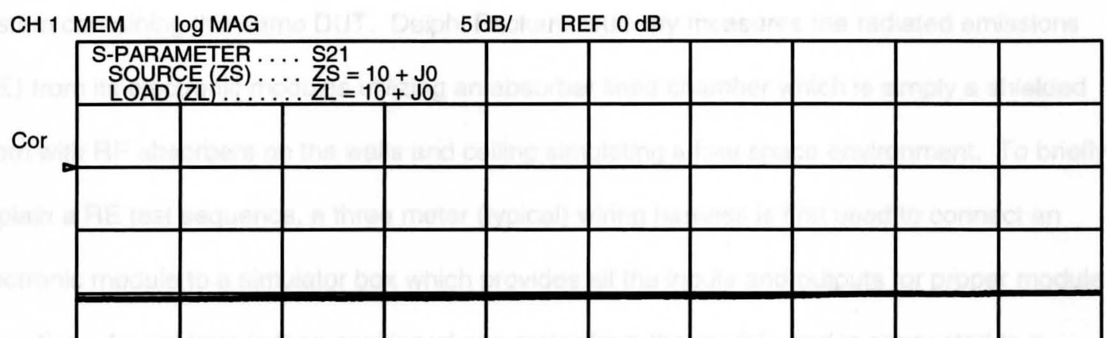

Figure 6.9 S_{21} from CONVERTZ of 10 dB Attenuator Designed for 10 Ω System and Measured with $Z_S = Z_L = 10 + j0\ \Omega$

circuit board that the SMD resistors for both designs are soldered. If a PC board with a 10 Ω microstrip was manufactured for the 10 Ω design, the accuracy of the results would surely increase.

The modified s_{21} data from CONVERTZ in Figure 6.9 also shows significant improvement over the 50 Ω measurement. Instead of the 15 dB of attenuation previously measured in a 50 Ω environment, CONVERTZ yields attenuation values very close to the designed 10 dB attenuation. The very slight deviation from 10 dB in Figure 6.9 can also be attributed to the 50 Ω microstrip problem mentioned above as well as non-ideal resistor values.

6.2 CONVERTZ and Radiated Emissions Correlation Attempt

This first example was an illustration of how CONVERTZ can be used for device design verification. A second application deals with the correlation that exists between a DUT's s_{21} measurement obtained from CONVERTZ and the electromagnetic energy radiated from a system containing this same DUT. Delphi Packard routinely measures the radiated emissions (RE) from its electronic modules utilizing an absorber lined chamber which is simply a shielded room with RF absorbers on the walls and ceiling simulating a free space environment. To briefly explain a RE test sequence, a three meter (typical) wiring harness is first used to connect an electronic module to a simulator box which provides all the inputs and outputs for proper module operation. An antenna is then positioned one meter from the module and is connected to a spectrum analyzer located outside the chamber. The spectrum analyzer quantifies the amount of energy, received by the antenna, that is radiated from the module and its associated wiring harness where the long length of harness tends to act as a transmitting antenna.

Due to multiple antenna locations, module orientations, and antenna polarizations required by the test specification, a thorough RE test can take a significant amount of time to setup and run. In order to possibly reduce the amount of RE testing done at Delphi Packard, it was hypothesized that a DUT's s_{21} data generated by CONVERTZ should correlate with a modified radiated emissions profile obtained by subtracting a baseline RE measurement (without the DUT in the system) from a RE measurement with the DUT inserted in the system. To clarify this statement with an example, it was anticipated that after a radiated emissions profile was obtained for an electronic module, a filter's s_{21} data (generated by CONVERTZ) could simply be subtracted from this data resulting in the same RE profile that would be obtained if the RE test was rerun with the filter connected to the module. This would eliminate the need to run multiple RE tests with various filter configurations until one results in a RE profile within corporate guidelines.

As a first step to prove or disprove this theory, an experiment for a single conductor was developed and the test setup is shown in Figure 6.10. The receive antenna and spectrum analyzer are configured much the same as during a standard RE test. The electronic module and wiring harness are replaced, however, with a signal source, broadband power amplifier, 10 dB

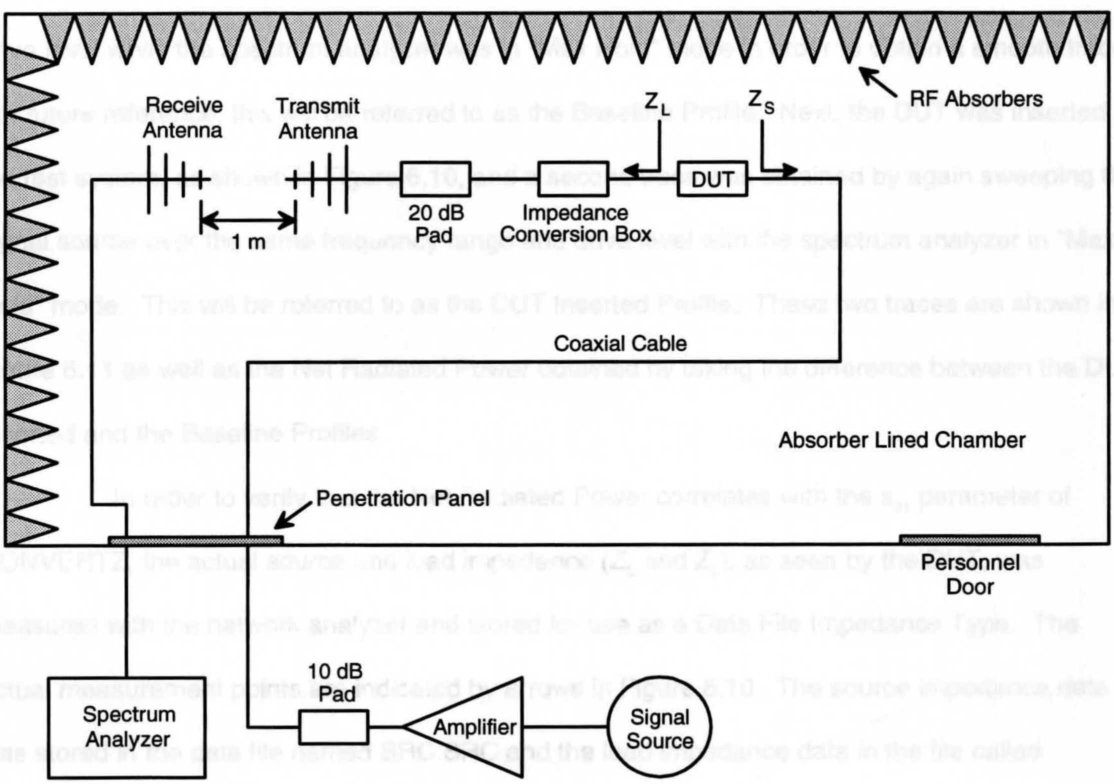

Figure 6.10 Test Setup for Radiated Emissions Experiment

attenuator (to protect the amplifier from any impedance mismatch), and a transmit antenna. The transmit antenna is used rather than a wiring harness since a harness is not an efficient radiating structure over a broad frequency range which is required for this experiment. The 20 dB attenuator (or pad as it is more commonly referred to) is used to "smooth" out the input impedance of the antenna to yield a constant 50 Ω impedance over frequency. The impedance conversion box is simply a shielded box utilizing the same basic components as the attenuator in Figure 6.3 except in this case, the PC board contains a single 680 Ω series resistor. This conversion box is used to adjust the input impedance of the antenna allowing flexibility in terms of the load impedance values that the DUT can be subjected to. Finally, the last component in the test setup is the actual DUT which is also constructed similar to the attenuator and impedance conversion box except the circuit configuration is a 10 Ω shunt resistor.

To perform the experiment, a baseline RE profile was first obtained with the DUT removed from the test system. The signal source was swept from 200 to 500 MHz at a constant drive level while the spectrum analyzer was in "Max Hold" mode in order to obtain a smooth trace. For future reference, this will be referred to as the Baseline Profile. Next, the DUT was inserted in the test system, as shown in Figure 6.10, and a second trace was obtained by again sweeping the signal source over the same frequency range and drive level with the spectrum analyzer in "Max Hold" mode. This will be referred to as the DUT Inserted Profile. These two traces are shown in Figure 6.11 as well as the Net Radiated Power obtained by taking the difference between the DUT Inserted and the Baseline Profiles.

In order to verify that the Net Radiated Power correlates with the s_{21} parameter of CONVERTZ, the actual source and load impedance (Z_S and Z_L), as seen by the DUT, was measured with the network analyzer and stored for use as a Data File Impedance Type. The actual measurement points are indicated by arrows in Figure 6.10. The source impedance data was stored in the data file named SRC.SRC and the load impedance data in the file called ZC02.LOD. The linear magnitude of these impedance traces are shown in Figure 6.12 where the

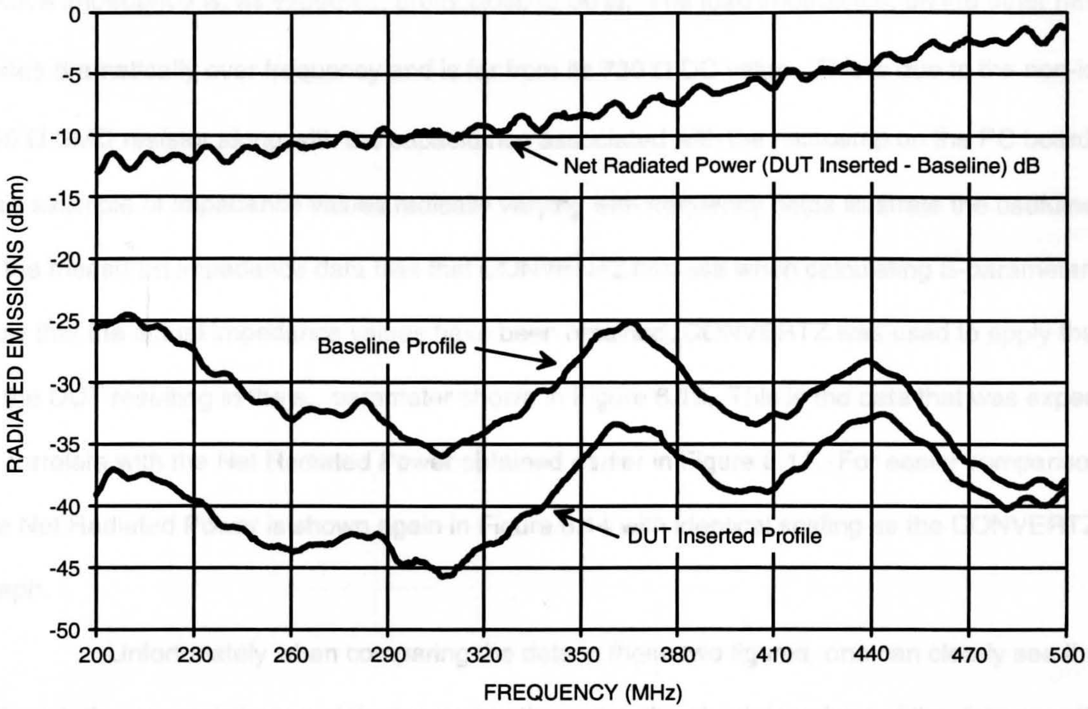

Figure 6.11 Radiated Emissions Data from Test Setup in Figure 6.10

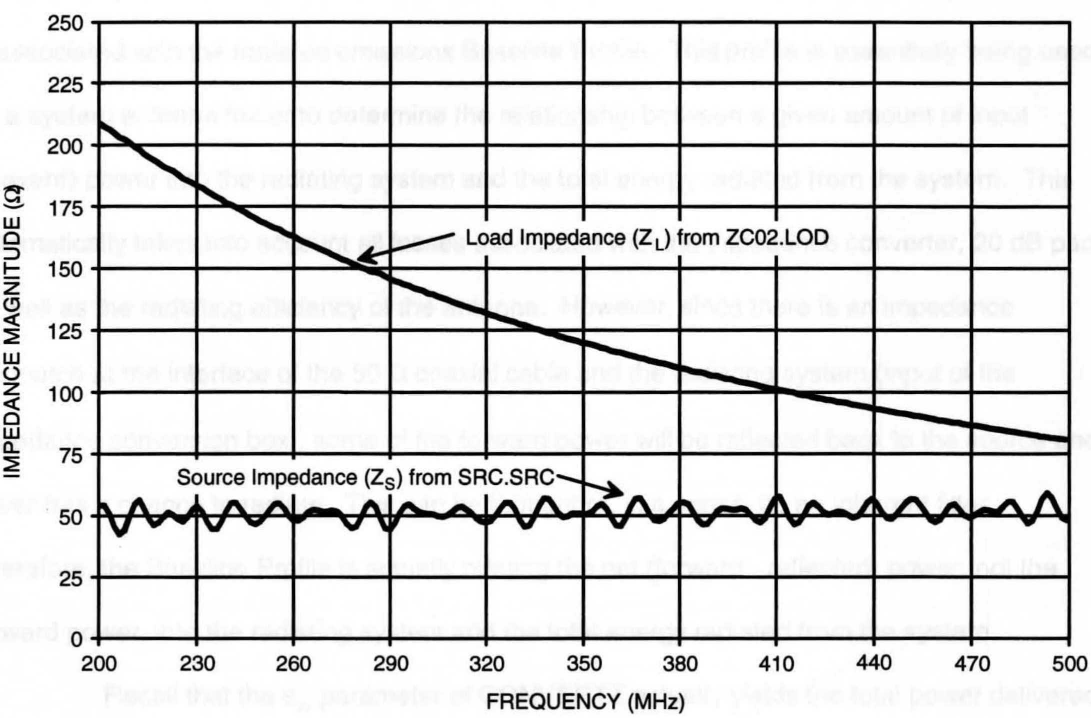

Figure 6.12 Measured Source and Load Impedance as seen by the DUT in Figure 6.10

source impedance is, as expected, pretty close to 50 Ω. The load impedance, on the other hand, varies dramatically over frequency and is far from its 730 Ω DC value. This is due to the non-ideal 680 Ω SMD resistor along with the capacitance associated with the microstrip on the PC board. This example of impedance values radically varying with frequency helps illustrate the usefulness of the measured impedance data files that CONVERTZ can use when calculating S-parameters. Now that the actual impedance values have been obtained, CONVERTZ was used to apply them to the DUT resulting in the s_{21} parameter shown in Figure 6.13. This is the data that was expected to correlate with the Net Radiated Power obtained earlier in Figure 6.11. For easier comparison, the Net Radiated Power is shown again in Figure 6.14 with identical scaling as the CONVERTZ graph.

Unfortunately when comparing the data in these two figures, one can clearly see that although the general shapes of the traces are the same, the absolute values of the data are off by greater than 6 dB. Therefore, the direct correlation that was originally intended does not appear to exist. After some thought, however, it was concluded that the reason for this lack of correlation is associated with the radiated emissions Baseline Profile. This profile is essentially being used as a system antenna factor to determine the relationship between a given amount of input (forward) power into the radiating system and the total energy radiated from the system. This automatically takes into account all losses associated with the impedance converter, 20 dB pad, as well as the radiating efficiency of the antenna. However, since there is an impedance mismatch at the interface of the 50 Ω coaxial cable and the radiating system (input of the impedance conversion box), some of the forward power will be reflected back to the source and never has a chance to radiate. This can be thought of, in a sense, as an inherent filter. Therefore, the Baseline Profile is actually relating the net (forward - reflected) power, not the forward power, into the radiating system and the total energy radiated from the system.

Recall that the s_{21} parameter of CONVERTZ actually yields the total power delivered to the load if the forward power incident on the DUT's input is known. In other words, any

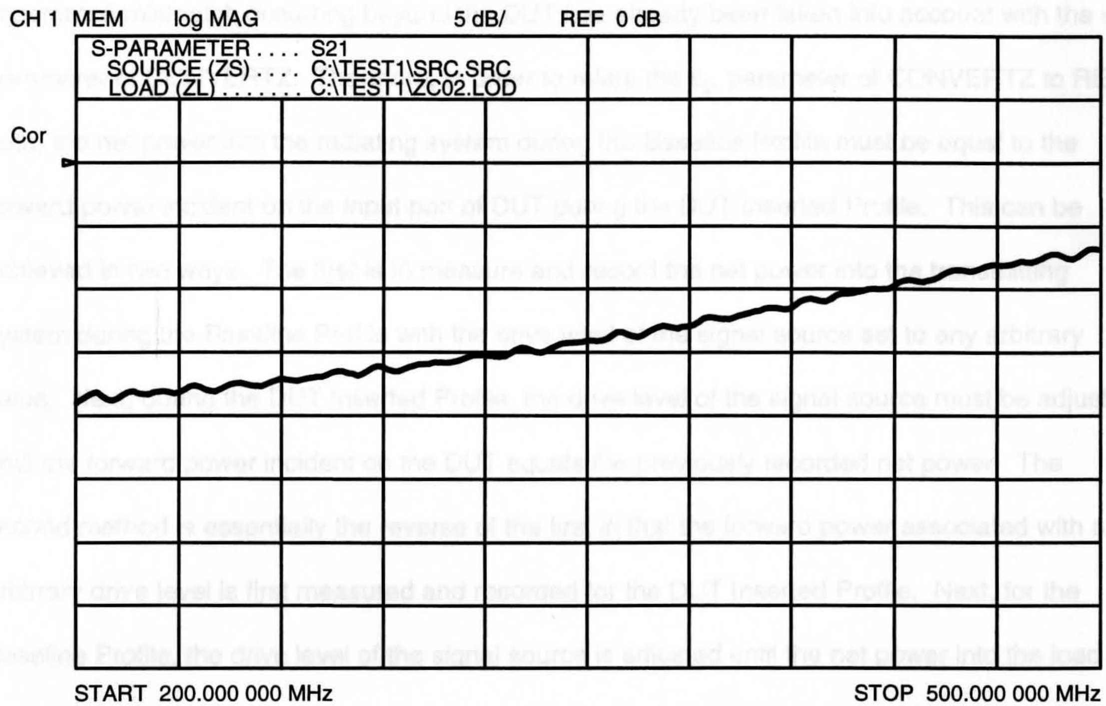

Figure 6.13 S_{21} Graph from CONVERTZ of 10 Ω Shunt Resistor with Z_S and Z_L from SRC.SRC and ZC02.LOD Data Files

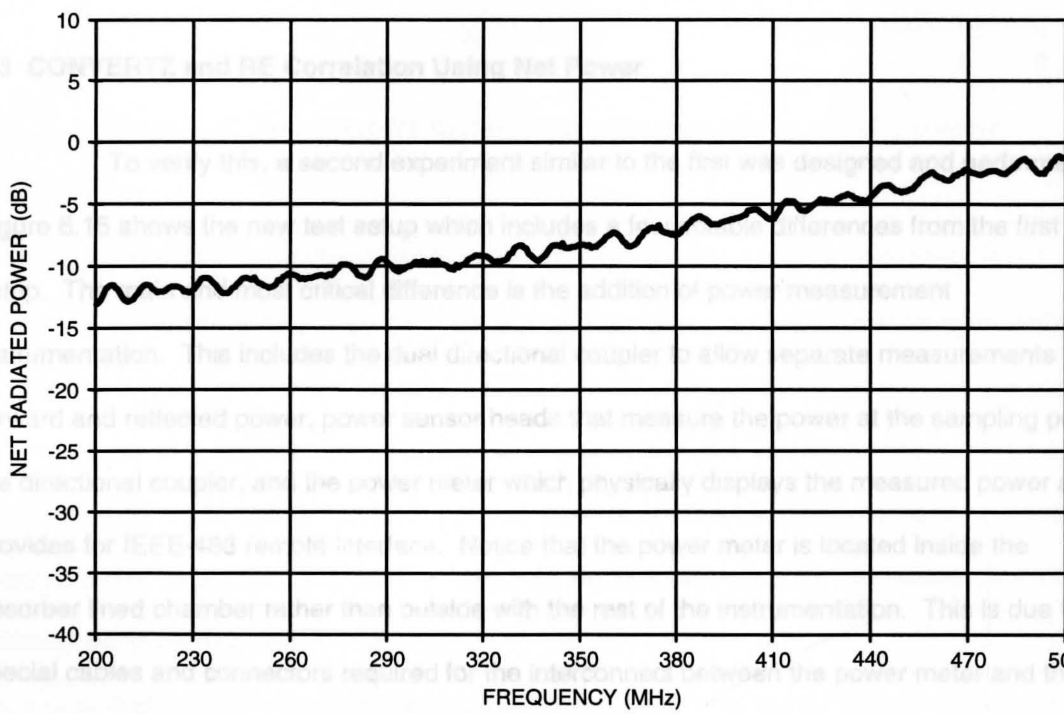

Figure 6.14 Net Radiated Power Associated with 10 Ω Shunt Resistor

impedance mismatch occurring beyond the DUT has already been taken into account with the s_{21} parameter of CONVERTZ. Therefore, in order to relate the s_{21} parameter of CONVERTZ to RE data, the net power into the radiating system during the Baseline Profile must be equal to the forward power incident on the input port of DUT during the DUT Inserted Profile. This can be achieved in two ways. The first is to measure and record the net power into the transmitting system during the Baseline Profile with the drive level of the signal source set to any arbitrary value. Next, during the DUT Inserted Profile, the drive level of the signal source must be adjusted until the forward power incident on the DUT equals the previously recorded net power. The second method is essentially the reverse of the first in that the forward power associated with an arbitrary drive level is first measured and recorded for the DUT Inserted Profile. Next, for the Baseline Profile, the drive level of the signal source is adjusted until the net power into the load is the same as the previously recorded forward power. Taking the difference between the two profiles for either of these methods yields the Net Radiated Power that should now correlate with the s_{21} parameter of CONVERTZ.

6.3 CONVERTZ and RE Correlation Using Net Power

To verify this, a second experiment similar to the first was designed and performed. Figure 6.15 shows the new test setup which includes a few notable differences from the first setup. The main and most critical difference is the addition of power measurement instrumentation. This includes the dual directional coupler to allow separate measurements of forward and reflected power, power sensor heads that measure the power at the sampling ports of the directional coupler, and the power meter which physically displays the measured power and provides for IEEE-488 remote interface. Notice that the power meter is located inside the absorber lined chamber rather than outside with the rest of the instrumentation. This is due to the special cables and connectors required for the interconnect between the power meter and the sensor heads. It was first verified that the power meter itself did not generate excessive

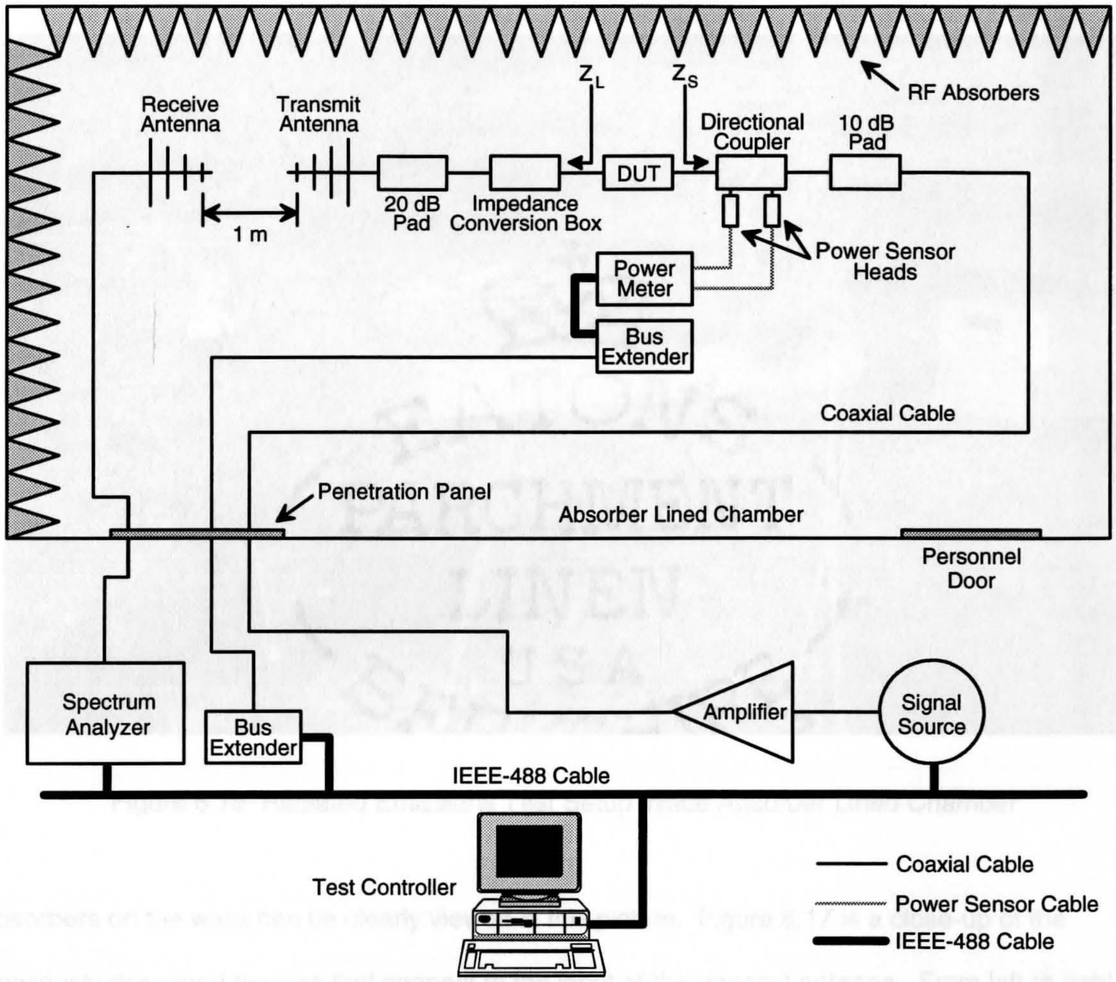

Figure 6.15 Test Setup for Radiated Emissions Experiment Using Net Power

emissions which could potentially interfere with the final results. A second difference is the addition of computer control since the drive level of the signal source needs to be constantly adjusted based on power meter readings. To do this manually for 100 or so frequency points per test would get quite tedious. The final difference is the addition of the two bus extenders which provide the IEEE-488 connection for the power meter inside the chamber. Since the penetration panel provides for coaxial cable connections only, the bus extenders are required to convert between the IEEE-488 and coaxial cables.

A photo of the complete test setup inside the absorber lined chamber is shown in Figure 6.16 illustrating both the receive antenna (left) and transmit antenna (right). Also, the RF

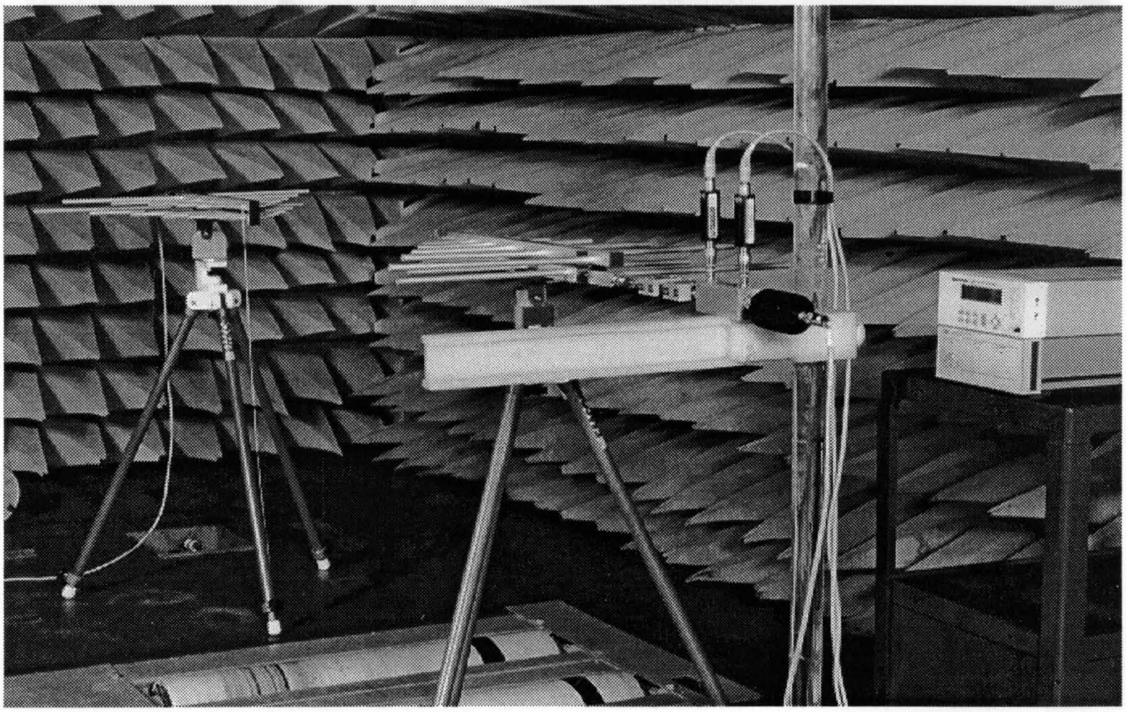

Figure 6.16 Radiated Emissions Test Setup Inside Absorber Lined Chamber

absorbers on the walls can be clearly viewed in this picture. Figure 6.17 is a close-up of the previously described devices that connect to the input of the transmit antenna. From left to right starting at the antenna input is a small 20 dB pad, impedance conversion box, DUT, dual directional coupler with forward and reflected power sensors attached, and a high power 10 dB pad. The power meter is shown sitting on top of the bus extender behind the 10 dB pad.

A computer program written specifically for this radiated emissions experiment, called RE_COMP, is included in Appendix B. The basic flow of the program will be discussed here although, as with CONVERTZ, the intricate details will not be included. First of all, the main menu of RE_COMP allows the user to set the start and stop frequency as well as the number of data points to acquire. After these parameters have been set, the program begins at the start frequency to obtain the Baseline Profile without the DUT inserted. This is done by first setting the drive level of the signal source to obtain between 26 dBm (400 mW) and 27 dBm (500 mW)

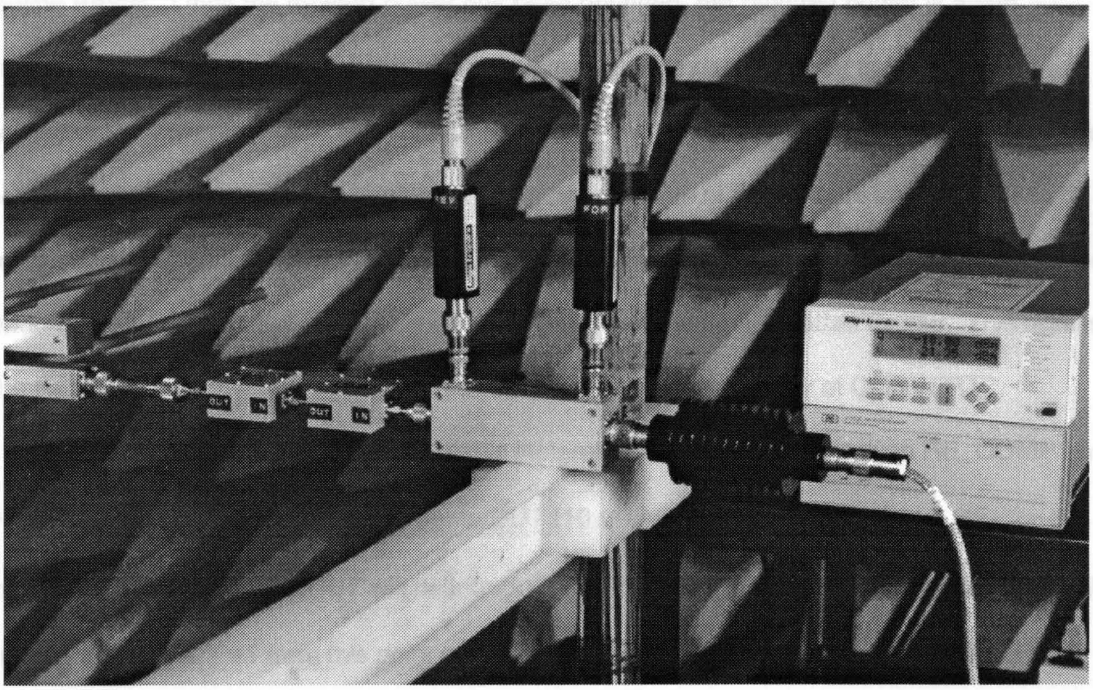

Figure 6.17 Transmit Section for Radiated Emissions Experiment Using Net Power

forward power. The reason for the upper limit is due to the power handling capabilities of the SMD components in both the impedance converter and the DUT which are 0.5 W. The lower limit is due to the need to radiate as much power as possible in order for the received signal to be well above the noise floor of the spectrum analyzer. Once this drive level is found, the program proceeds to measure the reflected power due to the impedance mismatch at the input of the radiating system. This is then used to calculate the net power by subtracting the reflected power from the forward power. Finally, the total radiated energy measured with the spectrum analyzer is read over the bus and recorded in an array along with the other data previously mentioned. This entire process is then repeated for each frequency point until the stop frequency is reached and the user is prompted to insert the DUT into the test setup.

Once the DUT has been inserted and the user presses "Continue", the DUT Inserted Profile is measured. At the start frequency, RE_COMP adjusts the drive level of the signal source to obtain the same forward power as the previously recorded net power ($\pm$ 1%) measured in the

Baseline Profile. Once the forward power has been obtained, the radiated power is again measured with the spectrum analyzer and stored in the same data array as the previously recorded data. This process is also repeated for each frequency point until the stop frequency is reached at which point the program dumps the entire contents of the array to an ASCII file in comma separated variable (CSV) format. Once the data is stored to a file, a spreadsheet can be used to open it and subtract the Baseline Profile data from the DUT Inserted Profile. This results in the Net Radiated Power which should correlate with the s_{21} parameter of CONVERTZ.

In order to check the validity of this statement, a comparison between RE_COMP and CONVERTZ was performed using the same DUT (10 Ω shunt resistor) as the initial RE experiment. Since a second DUT will be introduced a little later, the 10 Ω shunt resistor DUT will be referred to as DUT #1 from this point on. Figure 6.18 contains the s_{21} parameter of DUT #1 when measured in the standard 50 Ω network analyzer test system. This figure shows approximately 10 dB attenuation over the entire frequency range. For comparison purposes, an

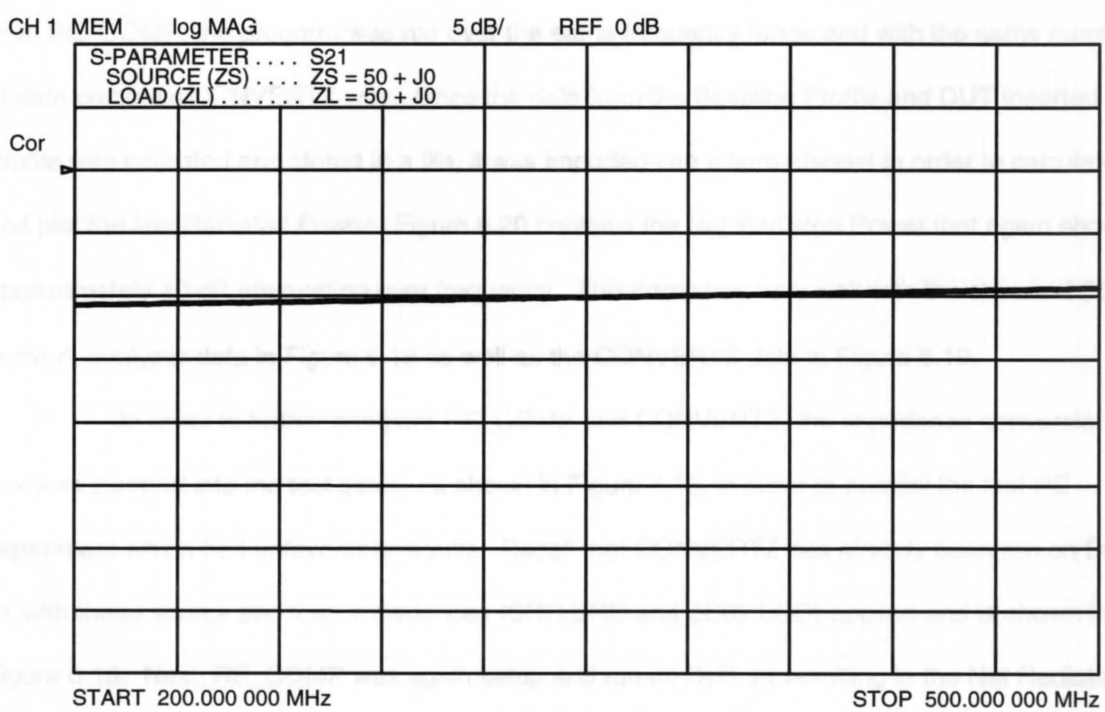

Figure 6.18 S_{21} of DUT #1 Measured with 50 Ω Network Analyzer Test System

experimental 50 Ω radiating system was implemented by simply removing the impedance converter in Figure 6.15, and using the 20 dB pad as the input to the radiating system (or load). This load impedance was measured with the network analyzer and stored in a file called MATCH.LOD to be used as a Data File Impedance Type by CONVERTZ. The source impedance was next measured at the output port of the directional coupler and stored in a file called SRC.SRC. Since both of these impedance files were very close to 50 + j0 Ω and look very similar to the source impedance in Figure 6.12, they are not shown here.

Running CONVERTZ on DUT #1 with the source and load impedances applied from the SRC.SRC and MATCH.LOD data files respectively, yields the s_{21} parameter shown in Figure 6.19. Since the measured source and load impedances of the radiating system are very close to 50 Ω, the graph looks very similar to the s_{21} graph from the 50 Ω network analyzer test system in Figure 6.18. The small wiggle across frequency in the graph from CONVERTZ is due to the actual source impedance which has a similar wiggle resulting from the long length of coaxial cable between the amplifier and 10 dB pad. Now to compare these graphs with radiated emissions data from RE_COMP, the program was run over the same frequency range and with the same number of data points as CONVERTZ was. Once the data from the Baseline Profile and DUT Inserted Profile was collected and stored in a file, it was imported into a spreadsheet in order to calculate and plot the Net Radiated Power. Figure 6.20 contains the Net Radiated Power that again shows approximately 10 dB attenuation over frequency. This compares very well with the standard 50 Ω network analyzer data in Figure 6.18 as well as the CONVERTZ data in Figure 6.19.

In order to further compare RE_COMP and CONVERTZ, the impedance conversion box was inserted into the test setup, as shown in Figure 6.15, in order to parallel the first RE experiment which had unfavorable results. Recall that CONVERTZ has already been run on DUT #1 with these source and load impedances (SRC.SRC and ZC02.LOD) applied and is shown in Figure 6.13. Next, RE_COMP was again setup and run on DUT #1 resulting in the Net Radiated Power shown in Figure 6.21. Notice the significant effect on radiated emissions that a change in

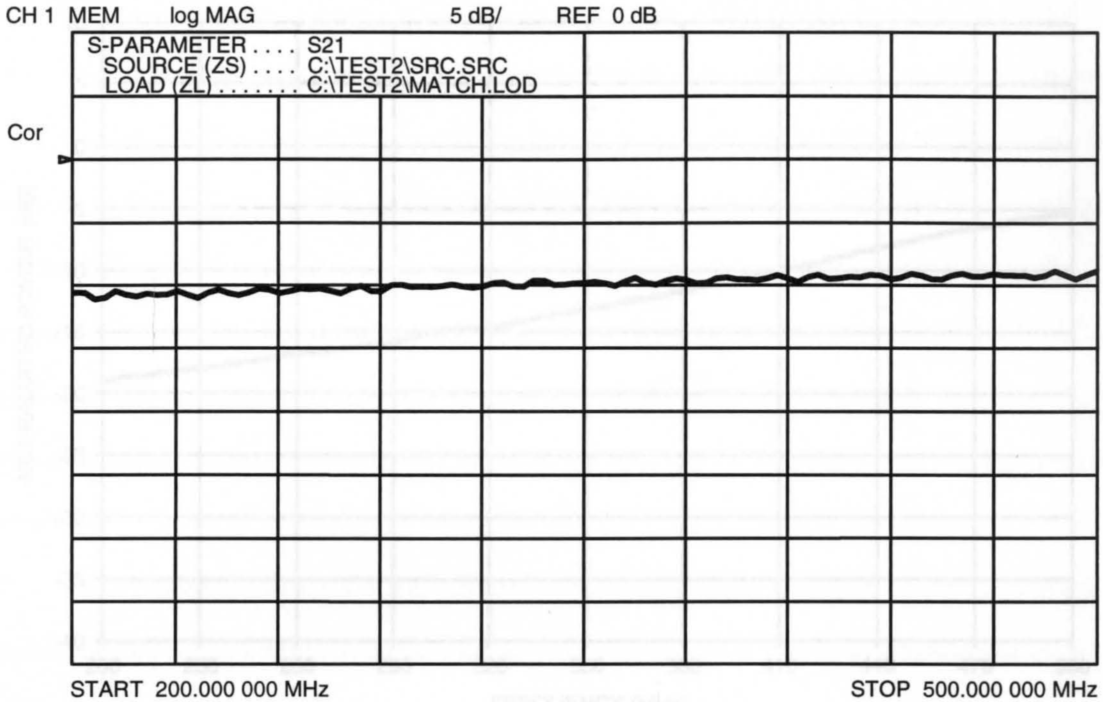

Figure 6.19 S_{21} Graph from CONVERTZ of DUT #1 with Z_S and Z_L from SRC.SRC and MATCH.LOD Data Files

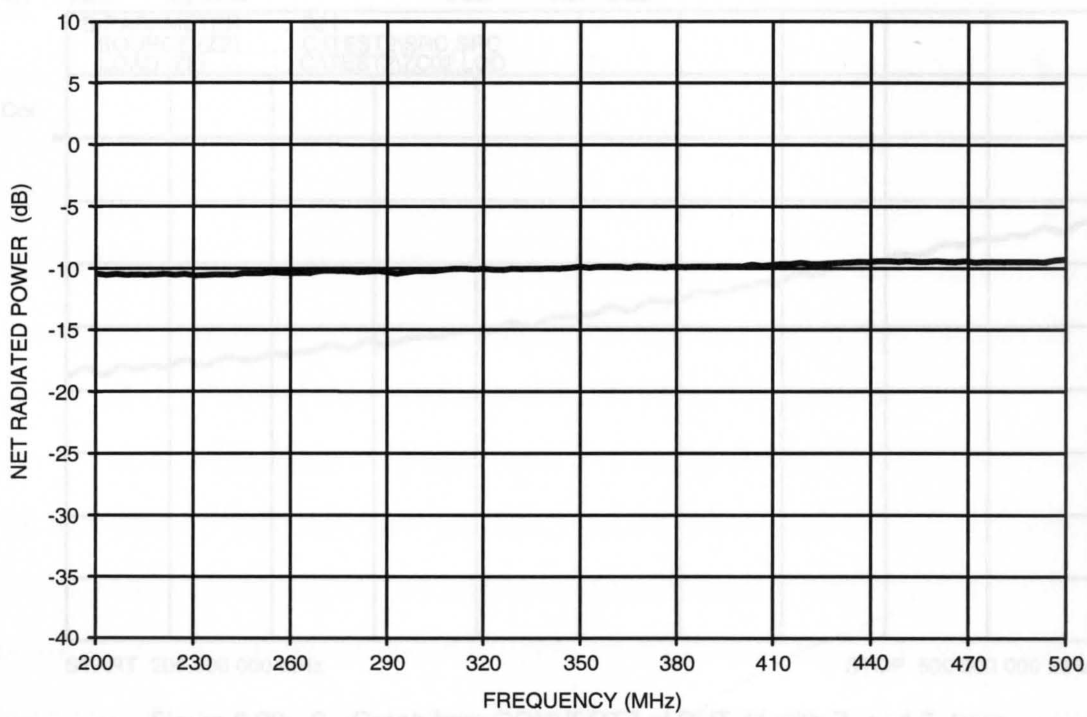

Figure 6.20 RE_COMP's Net Radiated Power for DUT #1 in 50 Ω System

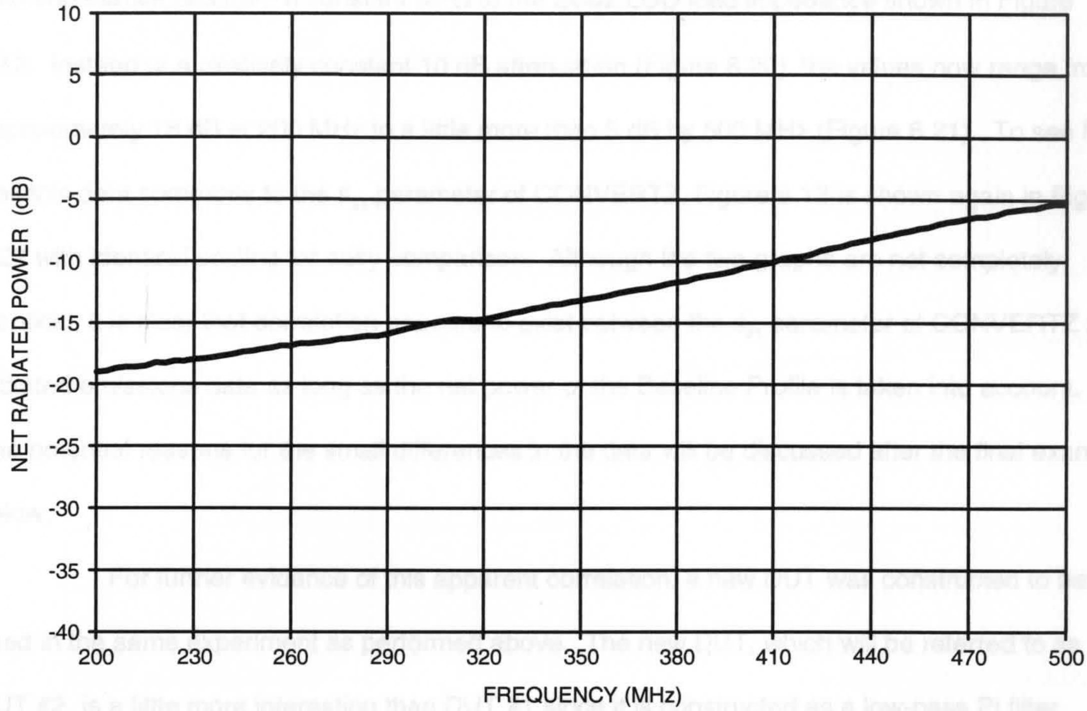

Figure 6.21 RE_COMP's Net Radiated Power for DUT #1 Connected to Impedance Converter (ZC02)

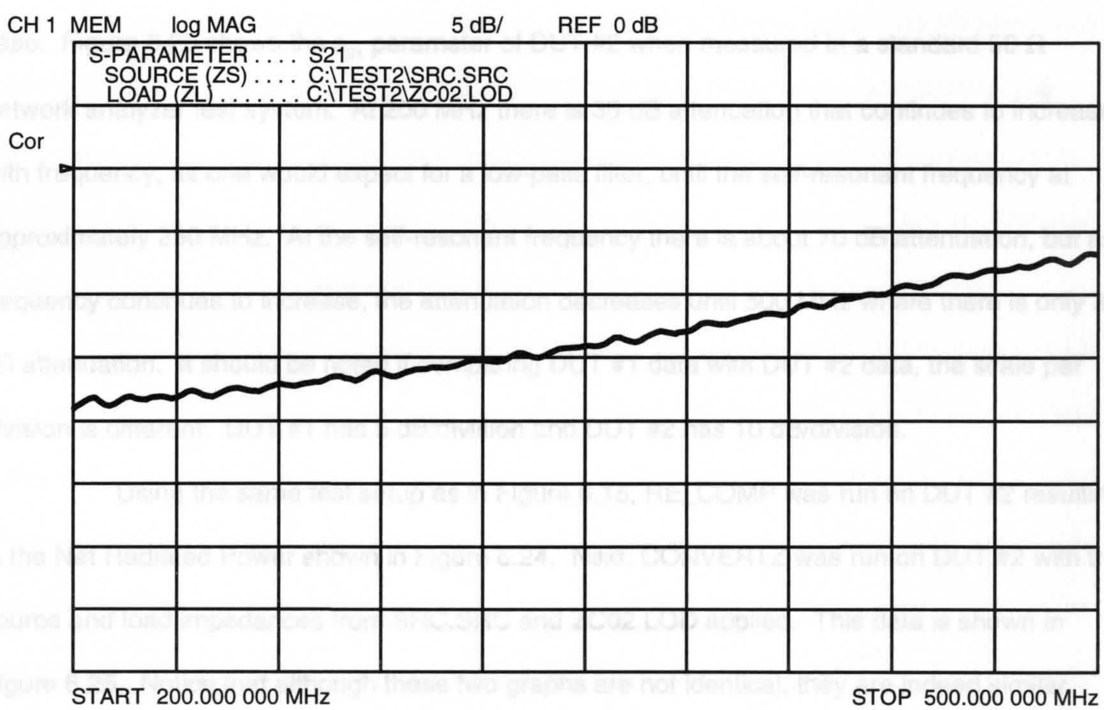

Figure 6.22 S_{21} Graph from CONVERTZ of DUT #1 with Z_S and Z_L from SRC.SRC and ZC02.LOD Data Files

load impedance has from a constant 50 Ω to the ZC02.LOD load impedance shown in Figure 6.12. Instead of a relatively constant 10 dB attenuation (Figure 6.20), the values now range from approximately 18 dB at 200 MHz to a little more than 5 dB by 500 MHz (Figure 6.21). To see how well this data compares to the s_{21} parameter of CONVERTZ, Figure 6.13 is shown again in Figure 6.22 with identical scaling for easy comparison. Although the two graphs are not completely identical, it is clear that correlation appears to exist between the s_{21} parameter of CONVERTZ and radiated emissions data as long as the net power of the Baseline Profile is taken into account. The potential reasons for the small differences in the data will be discussed after the final example below.

For further evidence of this apparent correlation, a new DUT was constructed to be used in the same experiment as performed above. The new DUT, which will be referred to as DUT #2, is a little more interesting than DUT #1 since it is constructed as a low-pass Pi filter containing a 33 pF shunt capacitor, 0.1 μH series inductor, and an 87 nF shunt capacitor. All three of these components are leaded devices, not surface mount devices as in the previous case. Figure 6.23 shows the s_{21} parameter of DUT #2 when measured in a standard 50 Ω network analyzer test system. At 200 MHz there is 35 dB attenuation that continues to increase with frequency, as one would expect for a low-pass filter, until the self-resonant frequency at the approximately 280 MHz. At the self-resonant frequency there is about 70 dB attenuation, but as frequency continues to increase, the attenuation decreases until 500 MHz where there is only 30 dB attenuation. It should be noted if comparing DUT #1 data with DUT #2 data, the scale per division is different. DUT #1 has 5 dB/division and DUT #2 has 10 dB/division.

Using the same test setup as in Figure 6.15, RE_COMP was run on DUT #2 resulting in the Net Radiated Power shown in Figure 6.24. Next, CONVERTZ was run on DUT #2 with the source and load impedances from SRC.SRC and ZC02.LOD applied. This data is shown in Figure 6.25. Notice that although these two graphs are not identical, they are indeed similar enough to reinforce the notion of correlation. It is again interesting to note the effect that the load

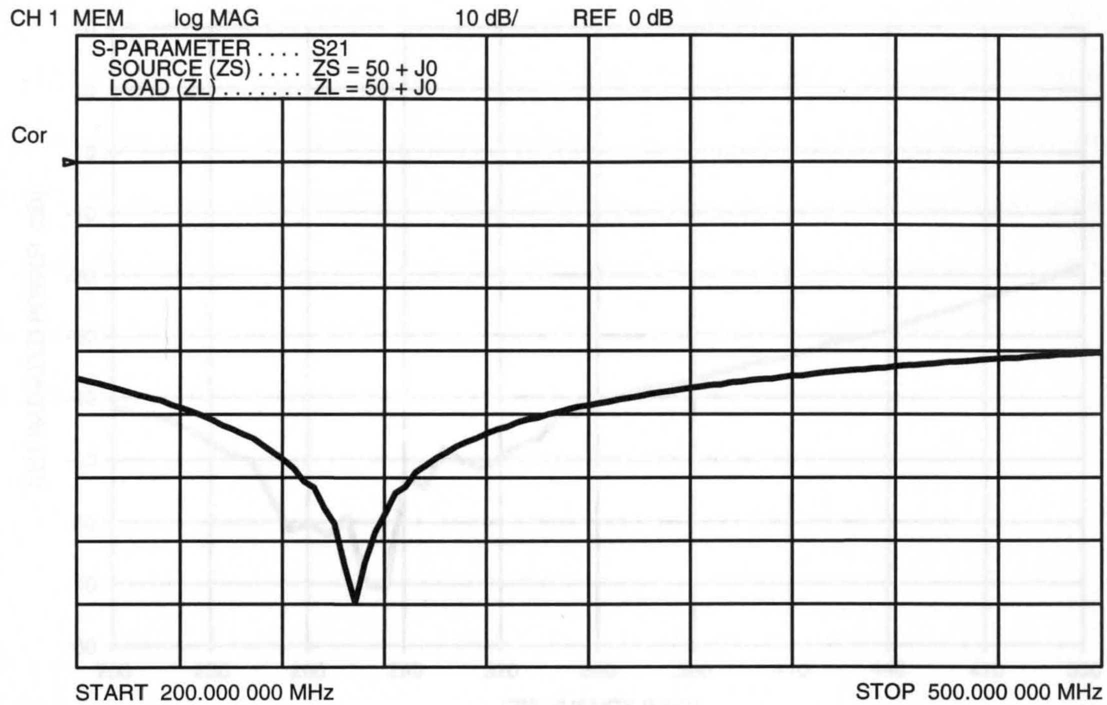

Figure 6.23 S_{21} of DUT #2 Measured with 50 Ω Network Analyzer Test System

impedance has on the radiated emissions data. At 200 MHz, there is around 35 dB attenuation with the 50 Ω load as shown in Figure 6.23. With the ZC02.LOD impedance, however, the attenuation is almost 45 dB as shown in Figures 6.24 and 6.25 which is an increase of 10 dB. Looking at 500 MHz, there is 30 dB attenuation with a 50 Ω load attached to DUT #2. With the ZC02.LOD impedance, the attenuation is around 20 dB which is a decrease of 10 dB.

The small differences found between the RE_COMP and CONVERTZ data for both DUT #1 and DUT #2 can be attributed to a number of factors. The first reason is the error that exists in the measured source and load impedance values used by CONVERTZ. Recall that the network analyzer is used to measure the source and load impedances for this test. Since the network analyzer is a reflection-based instrument, the further a device's impedance is from 50 Ω, the less accurate the impedance measurement is. Typically, above a few kΩ's, a reflection-based impedance measurement device is useless for precise measurements which are needed for

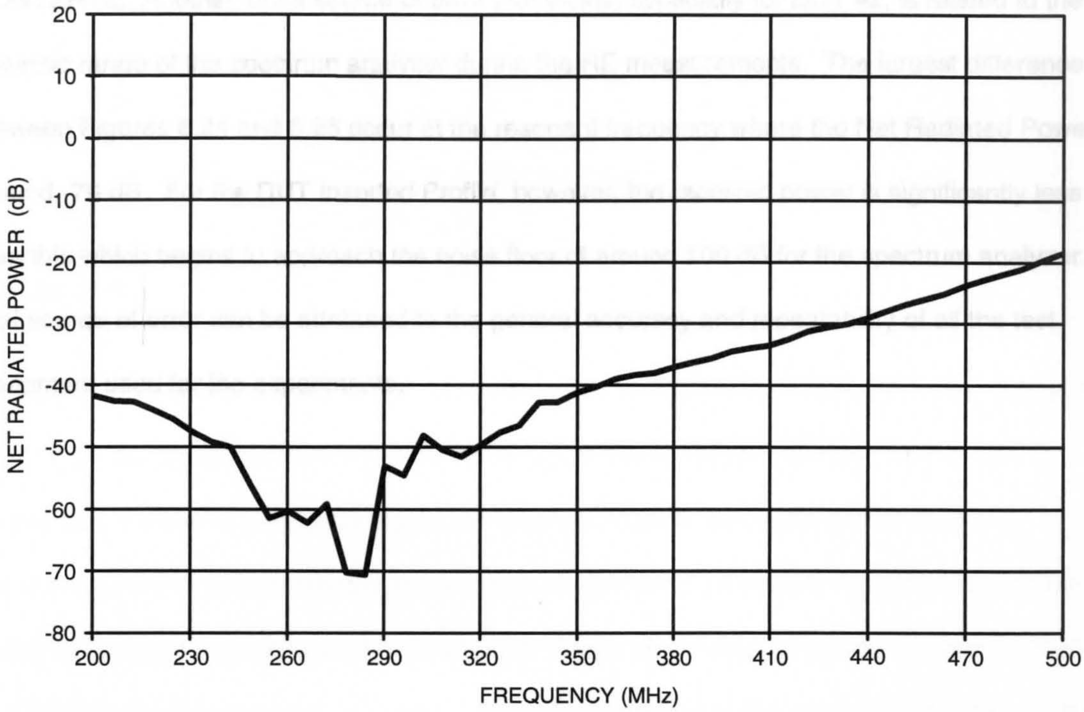

Figure 6.24 RE_COMP's Net Radiated Power for DUT #2 Connected to Impedance Converter (ZC02)

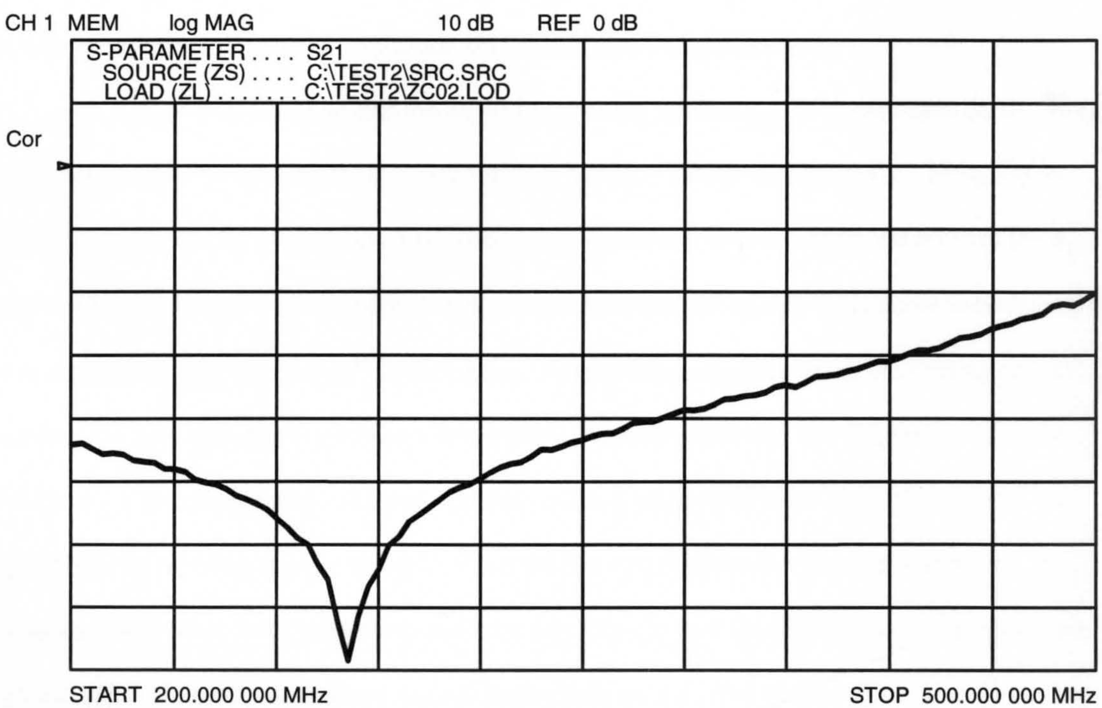

Figure 6.25 S_{21} Graph from CONVERTZ of DUT #2 with Z_S and Z_L from SRC.SRC and ZC02.LOD Data Files

CONVERTZ. Another small source of error that exists, especially for DUT #2, is related to the dynamic range of the spectrum analyzer during the RE measurements. The largest difference between Figures 6.24 and 6.25 occur at the resonant frequency where the Net Radiated Power is around -70 dB. For the DUT Inserted Profile, however, the received power is significantly less than this which begins to approach the noise floor of around 100 dB for the spectrum analyzer. A final source of error can be attributed to the general accuracy and repeatability of all the test equipment used for the experiments.

CHAPTER VII

CONCLUSION

7.1 Summary

This thesis has developed a technique that takes S-parameters, measured in a typical 50 Ω characteristic impedance test environment, and mathematically produces a new set of S-parameters based on any arbitrary source and load impedance. After a brief review of S-parameter theory was presented in Chapter II, a discussion on today's network analyzer test system was thoroughly discussed in Chapter III. A simplified block diagram was used to illustrate how the forward S-parameters, s_{11} and s_{21}, as well as the reverse S-parameters, s_{22} and s_{12}, are measured with a network analyzer coupled with an S-parameter test set. Various features of the network analyzer test system were also discussed such as the internal two-port full calibration routine which provides for very accurate and repeatable measurements.

Chapter IV emphasized the need for arbitrary impedance S-parameters by deriving the mathematical expression for the s_{21} parameter of the DUT shown in Figure 4.1. Plugging in different values for the source and load impedance variables shows a wide variation in the s_{21} results, thereby enforcing the notion that S-parameters measured in a 50 Ω environment may not be representative of "real world" performance. A couple approaches were next discussed to address the task of obtaining arbitrary impedance S-parameters with the decided method of choice being a mathematical concept. Based on the S-parameter work of K. Kurokawa,[3] an algorithm was developed for a two-port linear device which takes the four S-parameters, measured with 50 Ω test equipment, and converts them to four Z-parameters using equations (47) through (50). Since Z-parameters are not dependent on source and load impedance as S-parameters are, they can be employed in a new set of S-parameter equations, (57) through

(60), which contain source and load impedance variables. By using the four Z-parameters derived from the 50 Ω S-parameters as well as user-supplied source and load impedances, four new S-parameters can be obtained which are representative of a device's performance in its intended application.

Based on the algorithm just described, a computer program called CONVERTZ was written to automatically control Delphi Packard's HP 8753C network analyzer and create arbitrary impedance S-parameters. Chapter V explains the functionality of the program in detail where the computer first downloads the four 50 Ω S-parameters from the network analyzer and converts this data to four Z-parameters. The user then has a choice as to what type of impedance data to apply as a source and load including Constant, Variable, and Data File Impedance Types. Constant Impedance Type means the impedance value remains constant over the entire frequency range of the network analyzer. For instance, if the user wants to determine a DUT's performance with a source impedance of 5 Ω and a load impedance of 5000 Ω, this is the impedance type to select.

The Variable Impedance Type offers more flexibility than the Constant Impedance Type since these impedance values can change with frequency based on an inductor or capacitor supplied as a source/load. This is very useful in Delphi Packard's case since many of the filter devices produced there, for example, will ultimately be connected to a load with a long section of wiring harness. Since long wires are inductive by nature, the load impedance in CONVERTZ can be set to $100 + j\omega 1E\text{-}6$, for instance, to simulate a load impedance of 100 Ω connected to a 1 m length of wire (using the 1 μH/m rule of thumb). With frequency (ω) in the load term, CONVERTZ calculates the corresponding impedance value for each frequency test point that the network analyzer sweeps. For this example, the load impedance value used at 1 MHz will only be 100 Ω. As frequency increases, however, the load impedance will continue to increase and by 500 MHz, the applied load impedance will be 3.1 kΩ.

The Data File Impedance Type is also very flexible since this allows for measured impedance data to be read from a data file and then be applied to the S-parameter data at the corresponding test frequency. Therefore, if the actual source or load is available, CONVERTZ can utilize the impedance measurement capability of the network analyzer to create a file containing measured impedance values over frequency. This file can then be used to apply these "real life" impedances directly to a DUT therefore predicting its "real life" S-parameter performance. Once the impedance type and any associated values or data file names are supplied by the user, CONVERTZ can calculate the new S-parameters and upload the data to the network analyzer for viewing, formatting, or plotting.

Chapter VI contains sample applications for the CONVERTZ program with the first application consisting of the design and construction of resistive attenuators for both a 50 Ω and a 10 Ω system. The S-parameters of the attenuator designed for the 50 Ω system were measured with a standard 50 Ω network analyzer test system and yielded results very similar to the predicted values. The S-parameters of the attenuator designed for the 10 Ω system, however, were far from the expected results when measured with the same 50 Ω test equipment. The reason for this, of course, is due to the fact that the actual source and load impedance the attenuator was connected to was 50 Ω, not the 10 Ω impedance it was designed for. The CONVERTZ program was then used to measure/calculate the S-parameters of this attenuator where a 10 Ω source and load impedance was mathematically applied to the data. This resulted in S-parameter values very close to the expected results which shows what a valuable tool CONVERTZ can be for characterizing designs intended for non-50 Ω applications.

A second application for the program deals with the correlation that exists between a DUT's s_{21} measurement obtained from CONVERTZ and the electromagnetic energy radiated from a system containing this same DUT. It was originally hypothesized that a DUT's s_{21} data generated by CONVERTZ should have a direct correlation with the Net Radiated Power obtained by subtracting the radiated emissions Baseline Profile (without the DUT inserted in the system) from the DUT Inserted Profile. Conducting the original Radiated Emissions experiment, however, disproved this theory due the fact that the Baseline Profile has some inherent filtering in itself. This filtering is due to the impedance mismatch that occurs between the signal transmission line and the final load impedance of the radiating structure. Therefore, it is imperative that when determining the Baseline Profile, the relationship is established between net power into the radiating system vs. radiated power rather than forward power into the radiating system vs. radiated power. Once the net power was taken into account, as with the RE_COMP program, correlation was verified between the CONVERTZ s_{21} parameter and resultant radiated emissions data.

7.2 Discussion of Results

This thesis has resulted in a benchtop technique which can accurately predict a device's performance in a "real world" environment. More specifically, Delphi Packard Electric Systems now has a tool capable of accurately predicting their products performance in an automotive environment. The technique is comprised of generating arbitrary impedance S-parameters using a typical network analyzer test system along with the CONVERTZ software. The algorithms used in the software as well as their implementation have been proven technically correct throughout this thesis. This was done by successfully comparing results from CONVERTZ with hand calculations (section 6.1) and with results from Hewlett-Packard's MDS software (section 5.3). Although the MDS software does produce arbitrary impedance S-parameters, it is not tailored for automotive applications as is CONVERTZ due to its flexible source and load impedance types.

One of the main advantages of CONVERTZ is that it is easy to learn and use. Anyone with even the slightest familiarity with a network analyzer can be up and running with CONVERTZ in under 20 minutes. Due to this ease of use along with generating very fast results, CONVERTZ can be used for characterizing practically any two-port device that can be connected to the network analyzer. A specific example for Delphi Packard may be for various filter elements and configurations where a customer's requirement might be 30 dB attenuation, for example. In the past, if a certain filter configuration yielded only 20 dB in the 50 Ω test system, another filter element would be added to increase performance (and at the same time, cost). Now that it is possible to determine this filter's performance in its intended environment, it may be discovered that the 20 dB design (in 50 Ω) may actually produce 30 dB or more in the automotive environment, thereby saving the cost of the additional filter element.

Another Delphi Packard product that CONVERTZ may prove useful for is various cable constructions. In the past, it was not possible to accurately measure the attenuation/foot of cable constructions with a non-50 Ω characteristic impedance (Z_o). This was due to the impedance mismatch at the interface of the 50 Ω test port extension cable and the input of the cable under test (CUT). Unrealistically high attenuation values resulted from this mismatch which totally masked the true attenuation of the cable. By utilizing CONVERTZ in situations such as this, where Z_o = 135 Ω for example, allows the network analyzer's 50 Ω source and load impedance to be essentially converted to 135 Ω resulting in an accurate attenuation/foot measurement.

A final result of this thesis is the correlation that was proven between the s_{21} parameter of CONVERTZ and resultant effect on radiated emissions data. Basically it was shown that a filter's s_{21} measurement generated by CONVERTZ is truly representative of the reduction in radiated emissions when this filter is inserted. This will be very useful for determining the filter needed for an electronic module with a radiated emissions profile, 15 dB for instance, over the corporate guidelines. CONVERTZ can be used to find a filter solution that produces a minimum s_{21} measurement of 15 dB in the frequency band of interest by applying the source and load impedance values of the suspect line. There is one caution, however, which the engineer needs to be aware. This caution is that CONVERTZ yields the maximum reduction in the RE profile that can be achieved. As discussed in section 6.3, the reason for this is the Baseline RE Profile (without filter) already contains some inherent filtering due to internal impedance mismatches.

The data generated by CONVERTZ, on the other hand, already takes into account any impedance mismatch(es) and is represented in the s_{21} values. Therefore to be on the safe side, it is recommended that the engineer choose a filter with 5 to 10 dB additional attenuation over the needed value.

7.3 Suggestions for Future Work

Further work could be done in the future to enhance the CONVERTZ program by providing additional circuit configurations for source and load modeling. Presently, a resistor in series with a capacitor or inductor (Variable Impedance Type) is the only circuit configuration that has been implemented. Many other useful circuit configurations could be implemented as well including cascadable branches containing a resistance, capacitance, and inductance. This would allow for a resonant load condition which could dramatically affect a device's S-parameters. In terms of circuit configurations, however, this would probably be the most complex source/load configuration one would want to implement. For more advanced circuit configurations, it would probably be best to utilize a RF circuit simulator to calculate the equivalent impedance values over frequency. These values could then be dumped to a data file which can be read by CONVERTZ, similar to the Data File Impedance Type. Depending on the RF circuit simulator used, nonlinear elements could even be included in the impedance model.

Another improvement that could be implemented to this thesis involves the Data File Impedance Type. Recall that the impedance measurement capabilities of the network analyzer are employed to create the measured impedance data files used by CONVERTZ. As mentioned in Chapter VI, since the network analyzer utilizes a reflection-based measurement technique to determine impedance, the further a device's impedance is from 50 Ω, the less accurate the impedance measurement will be. Therefore, for impedance values exceeding a few kΩ's, it would be better to utilize a more accurate impedance measurement instrument, such as the HP 4291 impedance analyzer. This instrument can measure very low and very high impedances with extreme accuracy over frequency. These measured impedance values could then be stored to a data file using the same storage format as presently in use, and therefore read transparently by CONVERTZ requiring no software modification.

A final improvement that could be implemented in CONVERTZ would have application with both of the above suggestions. Presently, the Data File Impedance Type and RF circuit simulator (if implemented) requires an associated impedance value for the exact frequency of the arbitrary impedance S-parameter to be calculated. If this is not the case, CONVERTZ will return an error stating that stimulus parameters do not match. It would be useful to incorporate some type of interpolation algorithm that, within reason, utilizes the impedance data available to calculate arbitrary impedance S-parameters. Some type of warning should probably be displayed, however, stating the possibility of error due to impedance value uncertainties.

APPENDIX A

CONVERTZ Software Listing

```
10   !                          CONVERTZ
20   !          WRITTEN BY MICHAEL W. ALLENDER 10/18/94
30   !
40   ! This program downloads 2-port, error-corrected data from the HP 8753c network
50   ! analyzer with a 50 ohm test set (source = load = 50 ohms).
60   !
70   ! The data is then manipulated to represent data from a user-specified source and load
80   ! impedance and uploaded back to the hp 8753c.
90   !
100  ! The program also allows for impedance measurements to be used for source/load
110  ! impedances to apply to the DUT
120  !
130  COM /Orig_s_param/ COMPLEX S11(1:1601),S21(1:1601),S12(1:1601),S22(1:1601)
140  COM /Loads/ COMPLEX Zs(1:1601),Zl(1:1601)
150  COM /Device/@Hp8753c,@Ana_disp
160  COM /Var1/Text$[50],Zs$[100],Zl$[100],Rs$[15],Xs$[15],Rl$[15],Xl$[15],Ro$[5],Xo$[5],
     Load_type$[25],Source_type$[25],Cs$[15],Cl$[15],Ls$[15],Ll$[15]
170  COM /Var2/Start_freq,Stop_freq,Num_of_points,Choice
180  COM /Var3/Zs_label$[50],Zl_label$[50],Drive$[1],Subdir$[12],Source_path$[50],
     Load_path$[50],Sweep_type$[15]
190  COM /Arrays/Freq(1:1601)
200  !
210  CLEAR SCREEN
220  DISP "Initializing Program... Please Wait..."
230  Ro$="50"                     ! initialize real part of characteristic impedance (Ro)
240  Xo$="0"                      ! initialize imaginary part of characteristic impedance (Xo)
250  Rs$="50"                     ! initialize real part of source impedance (Rs)
260  Xs$="0"                      ! initialize imaginary part of source impedance (Xs)
270  Rl$="50"                     ! initialize real part of load impedance (Rl)
280  Xl$="0"                      ! initialize imaginary part of load impedance (Xl)
290  Cs$=""                       ! initialize source capacitance (Cs)
300  Cl$=""                       ! initialize load capacitance (Cl)
310  Ls$="0"                      ! initialize source inductance (Ls)
320  Ll$="0"                      ! initialize load inductance (Ll)
330  Load_type$="CONSTANT"        ! initialize load type
340  Source_type$="CONSTANT"      ! initialize source type
350  Drive$="C"                   ! initialize disk drive containing data file
360  Subdir$="LOADDATA"           ! initialize subdirectory containing data file
370  Zs_label$="(Zs = Rs + jXs)"  ! initialize source impedance label
380  Zl_label$="(Zl = Rl + jXl)"  ! initialize load impedance label
390  Zs$="Zs = "&Rs$&" + j"&Xs$   ! initialize source impedance for "CONSTANT"
400  Zl$="Zl = "&Rl$&" + j"&Xl$   ! initialize load impedance for "CONSTANT"
410  !
420  MAT Zs=(0)                   ! initialize all array elements to 0
430  MAT Zl=(0)                   ! initialize all array elements to 0
```

```
440    FOR I=1 TO 1601
450      Zs(I)=CMPLX(VAL(Rs$),VAL(Xs$))     ! fill source array with constant data
460      Zl(I)=CMPLX(VAL(Rl$),VAL(Xl$))     ! fill load array with constant data
470    NEXT I
480    CALL Main_menu                       ! this sub is the main driver
490    !
500    END                                  ! end of main program
510    !
520    !*******************************************************************************************
530    !                        Beginning of Subroutines
540    !*******************************************************************************************
550    !
560    !-------------------------------------------------------------------------------------------
570    SUB S11  ! calculates new s11 based on user-defined Zs and Zl
580      !
590      COM /Orig_s_param/ COMPLEX S11(*),S21(*),S12(*),S22(*)
600      COM /Loads/ COMPLEX Zs(*),Zl(*)
610      COM /Device/@Hp8753c,@Ana_disp
620      COM /Var1/Text$,Zs$,Zl$,Rs$,Xs$,Rl$,Xl$,Ro$,Xo$,Load_type$,Source_type$,Cs$,
         Cl$,Ls$,Ll$
630      COM /Var2/Start_freq,Stop_freq,Num_of_points,Choice
640      !
650      COMPLEX S_new(1:1601),Z11,Z12,Z21,Z22,Denom
660      MAT S_new=(0)                      ! initializing all array elements to 0
670      PRINT TABXY(24,3),"  CONVERTZ - Converting Data  "
680      DISP "Converting S11 Data... Please Wait..."
690      !
700      FOR I=1 TO Num_of_points           ! calculate new s11 data
710        Denom=(1-S11(I))*(1-S22(I))-S21(I)*S12(I)
720        Z11=50*((1-S22(I))*(1+S11(I))+S12(I)*S21(I))/Denom
730        Z12=100*S12(I)/Denom
740        Z21=100*S21(I)/Denom
750        Z22=50*((1-S11(I))*(1+S22(I))+S21(I)*S12(I))/Denom
760        S_new(I)=((Z11-CONJG(Zs(I)))*(Z22+Zl(I))-Z21*Z12)/((Z11+Zs(I))*(Z22+Zl(I))
           -Z21*Z12)
770      NEXT I
780      !
790      OUTPUT @Hp8753c;"CHAN1;DUACOFF;LOGM;DISPDATA;" ! chan 1, disp log mag
800      OUTPUT @Hp8753c;"FORM4;INPUDATA;"  ! send data in ASCII form
810      FOR I=1 TO Num_of_points
820        OUTPUT @Hp8753c;S_new(I)         ! output converted data to 8753c
830      NEXT I
840      OUTPUT @Hp8753c;"DATI;DISPMEMO;AUTO;CONT;" ! data->mem,autoscale,cont
         sweep
850      !
860      From_sub$="S11"
870      CALL Ana_graphics(From_sub$)
880      PRINT TABXY(26,3),"  CONVERTZ - Display Menu  "
890      DISP "Choose Desired Action..."
900    SUBEND
910    !
920    !-------------------------------------------------------------------------------------------
930    SUB S21  ! calculates new s21 based on user-defined Zs and Zl
940      !
```

```
950    COM /Orig_s_param/ COMPLEX S11(*),S21(*),S12(*),S22(*)
960    COM /Loads/ COMPLEX Zs(*),Zl(*)
970    COM /Device/@Hp8753c,@Ana_disp
980    COM /Var1/Text$,Zs$,Zl$,Rs$,Xs$,Rl$,Xl$,Ro$,Xo$,Load_type$,Source_type$,Cs$,
       Cl$,Ls$,Ll$
990    COM /Var2/Start_freq,Stop_freq,Num_of_points,Choice
1000   !
1010   COMPLEX S_new(1:1601),Z11,Z12,Z21,Z22,Denom,N,D
1020   MAT S_new=(0)                        ! initialize all array elements to 0
1030   PRINT TABXY(24,3),"  CONVERTZ - Converting Data  "
1040   DISP "Converting S21 Data... Please Wait..."
1050   !
1060   FOR I=1 TO Num_of_points             ! calculate new s21 data
1070     Denom=(1-S11(I))*(1-S22(I))-S21(I)*S12(I)
1080     Z11=50*((1-S22(I))*(1+S11(I))+S12(I)*S21(I))/Denom
1090     Z12=100*S12(I)/Denom
1100     Z21=100*S21(I)/Denom
1110     Z22=50*((1-S11(I))*(1+S22(I))+S21(I)*S12(I))/Denom
1120     N=SQR(ABS(REAL(Zs(I)))/ABS(REAL(Zl(I))))*2*REAL(Zl(I))*Z21
1130     D=(Z11+Zs(I))*(Z22+Zl(I))-Z21*Z12
1140     S_new(I)=N/D
1150   NEXT I
1160   !
1170   OUTPUT @Hp8753c;"CHAN1;DUACOFF;LOGM;DISPDATA;" ! chan 1, disp log mag
1180   OUTPUT @Hp8753c;"FORM4;INPUDATA;"    ! send data in ASCII form
1190   FOR I=1 TO Num_of_points
1200   OUTPUT @Hp8753c;S_new(I)             ! output converted data to 8753c
1210   NEXT I
1220   OUTPUT @Hp8753c;"DATI;DISPMEMO;AUTO;CONT;" ! data->mem,autoscale,cont
       sweep
1230   !
1240   From_sub$="S21"
1250   CALL Ana_graphics(From_sub$)
1260   PRINT TABXY(26,3),"  CONVERTZ - Display Menu  "
1270   DISP "Choose Desired Action..."
1280 SUBEND
1290 !
1300 !------------------------------------------------------------------------------------------------------
1310 SUB S12  ! calculates new s12 based on user-defined Zs and Zl
1320   !
1330   COM /Orig_s_param/ COMPLEX S11(*),S21(*),S12(*),S22(*)
1340   COM /Loads/ COMPLEX Zs(*),Zl(*)
1350   COM /Device/@Hp8753c,@Ana_disp
1360   COM /Var1/Text$,Zs$,Zl$,Rs$,Xs$,Rl$,Xl$,Ro$,Xo$,Load_type$,Source_type$,Cs$,
       Cl$,Ls$,Ll$
1370   COM /Var2/Start_freq,Stop_freq,Num_of_points,Choice
1380   !
1390   COMPLEX S_new(1:1601),Z11,Z12,Z21,Z22,Denom,N,D
1400   MAT S_new=(0)                        ! initializing all array elements to 0
1410   PRINT TABXY(24,3),"  CONVERTZ - Converting Data  "
1420   DISP "Converting S12 Data... Please Wait..."
1430   !
1440   FOR I=1 TO Num_of_points             ! calculating new s12 data
1450     Denom=(1-S11(I))*(1-S22(I))-S21(I)*S12(I)
```

```
1460       Z11=50*((1-S22(I))*(1+S11(I))+S12(I)*S21(I))/Denom
1470       Z12=100*S12(I)/Denom
1480       Z21=100*S21(I)/Denom
1490       Z22=50*((1-S11(I))*(1+S22(I))+S21(I)*S12(I))/Denom
1500       N=SQR(ABS(REAL(Zl(I)))/ABS(REAL(Zs(I))))*2*REAL(Zs(I))*Z12
1510       D=(Z11+Zs(I))*(Z22+Zl(I))-Z21*Z12
1520       S_new(I)=N/D
1530     NEXT I
1540     !
1550     OUTPUT @Hp8753c;"CHAN1;DUACOFF;LOGM;DISPDATA;" ! chan 1, disp log mag
1560     OUTPUT @Hp8753c;"FORM4;INPUDATA;"    ! send data in ASCII format
1570     FOR I=1 TO Num_of_points
1580       OUTPUT @Hp8753c;S_new(I)                 ! output converted data to 8753c
1590     NEXT I
1600     OUTPUT @Hp8753c;"DATI;DISPMEMO;AUTO;CONT;" ! data->mem,autoscale,cont
         sweep
1610     !
1620     From_sub$="S12"
1630     CALL Ana_graphics(From_sub$)
1640     PRINT TABXY(26,3),"   CONVERTZ - Display Menu   "
1650     DISP "Choose Desired Action..."
1660   SUBEND
1670   !
1680   !---------------------------------------------------------------------------------------------------------------
1690   SUB S22  ! calculates new s22 based on user-defined Zs and Zl
1700     !
1710     COM /Orig_s_param/ COMPLEX S11(*),S21(*),S12(*),S22(*)
1720     COM /Loads/ COMPLEX Zs(*),Zl(*)
1730     COM /Device/@Hp8753c,@Ana_disp
1740     COM /Var1/Text$,Zs$,Zl$,Rs$,Xs$,Rl$,Xl$,Ro$,Xo$,Load_type$,Source_type$,Cs$,
         Cl$,Ls$,Ll$
1750     COM /Var2/Start_freq,Stop_freq,Num_of_points,Choice
1760     !
1770     COMPLEX S_new(1:1601),Z11,Z12,Z21,Z22,Denom
1780     MAT S_new=(0)                              ! initializing all array elements to 0
1790     PRINT TABXY(24,3),"  CONVERTZ - Converting Data   "
1800     DISP "Converting S22 Data... Please Wait..."
1810     !
1820     FOR I=1 TO Num_of_points          ! calculate new s22 data
1830       Denom=(1-S11(I))*(1-S22(I))-S21(I)*S12(I)
1840       Z11=50*((1-S22(I))*(1+S11(I))+S12(I)*S21(I))/Denom
1850       Z12=100*S12(I)/Denom
1860       Z21=100*S21(I)/Denom
1870       Z22=50*((1-S11(I))*(1+S22(I))+S21(I)*S12(I))/Denom
1880       S_new(I)=((Z22-CONJG(Zl(I)))*(Z11+Zs(I))-Z12*Z21)/((Z11+Zs(I))*(Z22+Zl(I))
           -Z21*Z12)
1890     NEXT I
1900     !
1910     OUTPUT @Hp8753c;"CHAN1;DUACOFF;LOGM;DISPDATA;" ! chan 1, disp log mag
1920     OUTPUT @Hp8753c;"FORM4;INPUDATA;"    ! send data in ASCII form
1930     FOR I=1 TO Num_of_points
1940       OUTPUT @Hp8753c;S_new(I)                 ! output converted data to 8753c
1950     NEXT I
1960     OUTPUT @Hp8753c;"DATI;DISPMEMO;AUTO;CONT;" ! data->mem,autoscale,cont
```

```
        sweep
1970    !
1980    From_sub$="S22"
1990    CALL Ana_graphics(From_sub$)
2000    PRINT TABXY(26,3),"  CONVERTZ - Display Menu   "
2010    DISP "Choose Desired Action..."
2020  SUBEND
2030  !
2040  !----------------------------------------------------------------------------------------------------------------
2050  SUB Ana_settings        ! This sub verifies that a two-port calibration is active and then
2060                          ! downloads ANA settings such as start/stop freq, # of points,
2070                          ! and sweep type
2080    COM /Device/@Hp8753c,@Ana_disp
2090    COM /Var1/Text$,Zs$,Zl$,Rs$,Xs$,Rl$,Xl$,Ro$,Xo$,Load_type$,Source_type$,Cs$,
        Cl$,Ls$,Ll$
2100    COM /Var2/Start_freq,Stop_freq,Num_of_points,Choice
2110    COM /Var3/Zs_label$[50],Zl_label$[50],Drive$[1],Subdir$[12],Source_path$[50],
        Load_path$[50],Sweep_type$[15]
2120    COM /Arrays/Freq(1:1601)
2130    !
2140    CLEAR SCREEN
2150 Begin:                                   !
2160    SYSTEM PRIORITY 0                     ! reset system priority
2170    CALL Title_box                        ! prints blue title box at top of screen
2180    USER 1 KEYS
2190    PRINT TABXY(29,3),"CONVERTZ - Instructions"
2200    PRINT TABXY(5,9),"1)  Set the following parameters on the HP 8753C Network
        Analyzer:"
2210    PRINT TABXY(9,11),"* Start & Stop Frequency"
2220    PRINT TABXY(9,12),"* Number of Data Points"
2230    PRINT TABXY(9,13),"* Sweep Type"
2240    PRINT TABXY(9,14),"* IF Bandwidth"
2250    PRINT TABXY(5,17),"2)  Perform a Two-Port Full Calibration on the Test System"
2260    PRINT TABXY(5,20),"3)  Connect the Device Under Test (DUT) to S-Parameter Test
        Set"
2270    DISP "Press CONTINUE to proceed..."
2280    ON KEY 1 LABEL "Continue" GOTO Endloop1
2290    ON KEY 2 LABEL "" GOTO Loop1
2300    ON KEY 3 LABEL "" GOTO Loop1
2310    ON KEY 4 LABEL "" GOTO Loop1
2320    ON KEY 5 LABEL "" GOTO Loop1
2330    ON KEY 6 LABEL "" GOTO Loop1
2340    ON KEY 7 LABEL "" GOTO Loop1
2350    ON KEY 8 LABEL "  Main    Menu" CALL Main_menu
2360 Loop1: GOTO Loop1                        ! waiting for softkey to be pressed
2370 Endloop1:                                !
2380    MAT Freq=(0)                          ! initialize all array elements to 0
2390    REMOTE @Hp8753c
2400    OUTPUT @Hp8753c;"CALIFUL2?;"          ! verify user did a full 2-port cal.
2410    ENTER @Hp8753c;Ans                    ! get answer
2420    IF Ans<>1 THEN                        ! if Ans=0 then 2-port cal. is not active
2430      CLEAR SCREEN
2440      DISP "Full Two-Port Calibration Required... Press CONTINUE to Proceed..."
2450      SEND 7;UNL UNT             ! tell all instruments on bus to "UNLISTEN" & "UNTALK"
```

```
2460        LOCAL 7                       ! return all instruments on bus to "LOCAL" mode
2470        CALL Error_box
2480        GOTO Begin
2490      END IF
2500      PRINT TABXY(27,3),"CONVERTZ - Downloading Data"
2510      FOR I=9 TO 20                  ! clear instructions
2520        PRINT TABXY(5,I);"                                                    "
2530      NEXT I
2540      DISP "Downloading S-Parameters... Please Wait..."
2550      !
2560      OUTPUT @Hp8753c;"POIN?;"           ! request number of points from analyzer
2570      ENTER @Hp8753c;Num_of_points       ! get answer
2580      OUTPUT @Hp8753c;"STAR?;"           ! request start frequency from analyzer
2590      ENTER @Hp8753c;Start_freq          ! get answer
2600      OUTPUT @Hp8753c;"STOP?;"           ! request stop frequency from analyzer
2610      ENTER @Hp8753c;Stop_freq           ! get answer
2620      !
2630      OUTPUT @Hp8753c;"LOGFREQ?;"        ! ask if log frequency sweep
2640      ENTER @Hp8753c;Ans                 ! get answer
2650      IF Ans=1 THEN                      ! if Ans=1 then its a log sweep
2660        Sweep_type$="LOG SWEEP"
2670      ELSE                               ! if Ans<>1 then check for linear sweep
2680        OUTPUT @Hp8753c;"LINFREQ?;"      ! ask if linear sweep
2690        ENTER @Hp8753c;Ans               ! get answer
2700        IF Ans=1 THEN                    ! if Ans=1 then its a linear sweep
2710          Sweep_type$="LINEAR SWEEP"
2720        ELSE                             ! invalid sweep type for program
2730          DISP "Sweep Type Must be Log or Linear... Press CONTINUE to Proceed..."
2740          CALL Error_box
2750          CALL Main_menu
2760        END IF
2770      END IF
2780      !
2790      OUTPUT @Hp8753c;"OUTPLIML;"        ! request frequency points
2800      FOR I=1 TO Num_of_points
2810        ENTER @Hp8753c;Freq(I)           ! read each frequency
2820      NEXT I
2830      CALL Down_s_param                  ! download all four s-parameters
2840    SUBEND
2850    !
2860    !------------------------------------------------------------------------------------------------------------
2870    SUB Down_s_param
2880      ! This section of code downloads all four s-parameters from the
2890      ! network analyzer in real/imaginary format and stores the data in
2900      ! complex arrays.
2910      !
2920      COM /Orig_s_param/ COMPLEX S11(*),S21(*),S12(*),S22(*)
2930      COM /Device/@Hp8753c,@Ana_disp
2940      COM /Var1/Text$,Zs$,Zl$,Rs$,Xs$,Rl$,Xl$,Ro$,Xo$,Load_type$,Source_type$,Cs$,
          Cl$,Ls$,Ll$
2950      COM /Var2/Start_freq,Stop_freq,Num_of_points,Choice
2960      COM /Var3/Zs_label$,Zl_label$,Drive$,Subdir$,Source_path$,Load_path$,
          Sweep_type$
2970      !
```

```
2980    MAT S11=(0)                              ! initializing all array elements to 0
2990    MAT S12=(0)
3000    MAT S21=(0)
3010    MAT S22=(0)
3020    OUTPUT @Hp8753c;"CHAN1;TITL"""";CHAN2;TITL"""";"               ! erase titles
3030    OUTPUT @Hp8753c;"CHAN1;DISPDATA;S11;CONVOFF;REAL;DUACON;"
3040    OUTPUT @Hp8753c;"CHAN2;DISPDATA;S11;CONVOFF;IMAG;"    ! channel 2
3050    OUTPUT @Hp8753c;"SING;CHAN1;AUTO;CHAN2;AUTO;" ! single sweep/autoscale
3060    OUTPUT @Hp8753c;"FORM4;OUTPDATA;"
3070    FOR I=1 TO Num_of_points
3080      ENTER @Hp8753c;S11(I)
3090    NEXT I
3100    OUTPUT @Hp8753c;"CHAN1;S21;REAL;AUTO;CHAN2;S21;IMAG;AUTO;FORM4;
        OUTPDATA;"
3110    FOR I=1 TO Num_of_points
3120      ENTER @Hp8753c;S21(I)
3130    NEXT I
3140    OUTPUT @Hp8753c;"CHAN1;S12;REAL;AUTO;CHAN2;S12;IMAG;AUTO;FORM4;
        OUTPDATA;"
3150    FOR I=1 TO Num_of_points
3160      ENTER @Hp8753c;S12(I)
3170    NEXT I
3180    OUTPUT @Hp8753c;"CHAN1;S22;REAL;AUTO;CHAN2;S22;IMAG;AUTO;FORM4;
        OUTPDATA;"
3190    FOR I=1 TO Num_of_points
3200      ENTER @Hp8753c;S22(I)
3210    NEXT I
3220    OUTPUT @Hp8753c;"CONT;"
3230    !
3240    CALL Term_box                       ! print source/load termination display
3250    CALL Display_menu                   ! prompts user to display s-parameter or change
3260                                        ! source/load impedance
3270  SUBEND
3280  !
3290  !------------------------------------------------------------------------------------
3300  SUB Display_menu       ! This sub prompts user to display a s-parameter or
3310                         ! goto source/load impedance menu
3320    SYSTEM PRIORITY 0 ! reset system priority back to 0
3330    PRINT TABXY(26,3),"   CONVERTZ - Display Menu   "
3340    !
3350    USER 1 KEYS
3360    ON KEY 1 LABEL "Display   S11 " CALL S11
3370    ON KEY 2 LABEL "Display   S12 " CALL S12
3380    ON KEY 3 LABEL "Display   S21 " CALL S21
3390    ON KEY 4 LABEL "Display   S22 " CALL S22
3400    ON KEY 5 LABEL "  Zs/Zl   Menu" CALL Load_type_menu
3410    ON KEY 6 LABEL "  Scale   Menu" CALL Scale
3420    ON KEY 7 LABEL "  Plot    Menu" CALL Plot
3430    ON KEY 8 LABEL "  Main    Menu" CALL Main_menu
3440    DISP "Choose A Softkey..."
3450 Loop1: GOTO Loop1                           ! begin looping until softkey is pressed
3460  SUBEND
3470  !
3480  !------------------------------------------------------------------------------------
```

```
3490    SUB Constant_loads
3500      COM /Var1/Text$,Zs$,Zl$,Rs$,Xs$,Rl$,Xl$,Ro$,Xo$,Load_type$,Source_type$,Cs$,
          Cl$,Ls$,Ll$
3510      COM /Var2/Start_freq,Stop_freq,Num_of_points,Choice
3520      COM /Var3/Zs_label$,Zl_label$,Drive$,Subdir$,Source_path$,Load_path$,
          Sweep_type$
3530      COM /Loads/ COMPLEX Zs(*),Zl(*)
3540      !
3550 Begin:                                   !
3560      SYSTEM PRIORITY 0                   ! reset system priority
3570      SELECT Choice                       ! Choice is set in load_type_menu sub
3580        CASE 1                            ! change source impedance
3590          PRINT TABXY(24,3),"  CONVERTZ - Constant Source   "
3600          USER 1 KEYS
3610          ON KEY 1 LABEL " Change    Rs" GOSUB Rs
3620          ON KEY 2 LABEL " Change    Xs" GOSUB Xs
3630          ON KEY 3 LABEL "" GOTO Loop1
3640          ON KEY 4 LABEL "" GOTO Loop1
3650          ON KEY 5 LABEL "" GOTO Loop1
3660          ON KEY 6 LABEL "" GOTO Loop1
3670          ON KEY 7 LABEL "" GOTO Loop1
3680          ON KEY 8 LABEL "  Done" GOSUB Done_source
3690          DISP "Choose A Softkey..."
3700 Loop1: GOTO Loop1                        ! begin looping until softkey is pressed
3710          !
3720        CASE 2                            ! change load impedance
3730          PRINT TABXY(25,3),"  CONVERTZ - Constant Load   "
3740          USER 1 KEYS
3750          ON KEY 1 LABEL " Change    Rl" GOSUB Rl
3760          ON KEY 2 LABEL " Change    Xl" GOSUB Xl
3770          ON KEY 3 LABEL "" GOTO Loop2
3780          ON KEY 4 LABEL "" GOTO Loop2
3790          ON KEY 5 LABEL "" GOTO Loop2
3800          ON KEY 6 LABEL "" GOTO Loop2
3810          ON KEY 7 LABEL "" GOTO Loop2
3820          ON KEY 8 LABEL "  Done" GOSUB Done_load
3830          DISP "Choose A Softkey..."
3840 Loop2: GOTO Loop2                        ! begin looping until softkey is pressed
3850      END SELECT
3860      !
3870 Rs:                                      ! changing real part of source impedance
3880      PRINT TABXY(3,15),"  Source Impedance (Zs): CONSTANT   "
3890      INPUT "Enter Real Part of Source Impedance (Rs) in Ohms:",Rs
3900      Rs$=VAL$(Rs)
3910      IF LEN(Rs$)<1 THEN GOTO Rs
3920      IF VAL(Xs$)<0 THEN                  ! negative number
3930        Zs$="Zs = "&Rs$&" - j"&VAL$(ABS(VAL(Xs$)))
3940        Zs_label$="(Zs = Rs - jXs)"
3950      ELSE                                ! positive number
3960        Zs$="Zs = "&Rs$&" + j"&Xs$
3970        Zs_label$="(Zs = Rs + jXs)"
3980      END IF
3990      Source_type$="CONSTANT"
4000      PRINT TABXY(3,17),"                    "           ! clear line
```

```
4010      PRINT TABXY(21-LEN(Zs_label$)/2,17),Zs_label$
4020      PRINT TABXY(3,19),"                              "          ! clear line
4030      PRINT TABXY(21-LEN(Zs$)/2,19),Zs$
4040      DISP "Choose A Softkey..."
4050      RETURN
4060      !
4070 Xs:                                  ! changing imaginary part of source impedance
4080      PRINT TABXY(3,15),"  Source Impedance (Zs): CONSTANT  "
4090      INPUT "Enter Imaginary Part of Source Impedance (Xs) in Ohms:",Xs
4100      Xs$=VAL$(Xs)
4110      IF LEN(Xs$)<1 THEN GOTO Xs
4120      IF VAL(Xs$)<0 THEN                   ! negative number
4130        Zs$="Zs = "&Rs$&" - j"&VAL$(ABS(VAL(Xs$)))
4140        Zs_label$="(Zs = Rs - jXs)"
4150      ELSE                                 ! positive number
4160        Zs$="Zs = "&Rs$&" + j"&Xs$
4170        Zs_label$="(Zs = Rs + jXs)"
4180      END IF
4190      Source_type$="CONSTANT"
4200      PRINT TABXY(3,17),"                              "          ! clear line
4210      PRINT TABXY(21-LEN(Zs_label$)/2,17),Zs_label$
4220      PRINT TABXY(3,19),"                              "          ! clear line
4230      PRINT TABXY(21-LEN(Zs$)/2,19),Zs$
4240      DISP "Choose A Softkey..."
4250      RETURN
4260      !
4270 Done_source:          ! fills all source array elements with constant source data
4280      IF Source_type$="CONSTANT" THEN                        ! fill source array
4290        PRINT TABXY(23,3),"        Please Wait...          "
4300        DISP "Generating Source Data... Please Wait..."
4310        MAT Zs=(0)! RESET ALL ARRAY ELEMENTS TO 0
4320        FOR I=1 TO Num_of_points
4330          Zs(I)=CMPLX(VAL(Rs$),VAL(Xs$))
4340        NEXT I
4350      END IF
4360      CALL Load_type_menu
4370      RETURN
4380 Rl:                                  ! changing real part of load impedance
4390      PRINT TABXY(42,15),"  Load Impedance (Zl): CONSTANT  "
4400      INPUT "Enter Real Part of Load Impedance (Rl) in Ohms:",Rl
4410      Rl$=VAL$(Rl)
4420      IF LEN(Rl$)<1 THEN GOTO Rl
4430      IF VAL(Xl$)<0 THEN                   ! negative number
4440        Zl$="Zl = "&Rl$&" - j"&VAL$(ABS(VAL(Xl$)))
4450        Zl_label$="(Zl = Rl - jXl)"
4460      ELSE                                 ! positive number
4470        Zl$="Zl = "&Rl$&" + j"&Xl$
4480        Zl_label$="(Zl = Rl + jXl)"
4490      END IF
4500      Load_type$="CONSTANT"
4510      PRINT TABXY(41,17),"                              "          ! clear line
4520      PRINT TABXY(59-LEN(Zl_label$)/2,17),Zl_label$
4530      PRINT TABXY(41,19),"                              "          ! clear line
4540      PRINT TABXY(59-LEN(Zl$)/2,19),Zl$
```

```
4550      DISP "Choose A Softkey..."
4560      RETURN
4570      !
4580 Xl:                                    ! changing imaginary part of load impedance
4590      PRINT TABXY(42,15),"   Load Impedance (Zl): CONSTANT   "
4600      INPUT "Enter Imaginary Part of Load Impedance (Xl) in Ohms:",Xl
4610      Xl$=VAL$(Xl)
4620      IF LEN(Xl$)<1 THEN GOTO Xl
4630      IF VAL(Xl$)<0 THEN                 ! negative number
4640        Zl$="Zl = "&Rl$&" - j"&VAL$(ABS(VAL(Xl$)))
4650        Zl_label$="(Zl = Rl - jXl)"
4660      ELSE                               ! positive number
4670        Zl$="Zl = "&Rl$&" + j"&Xl$
4680        Zl_label$="(Zl = Rl + jXl)"
4690      END IF
4700      Load_type$="CONSTANT"
4710      PRINT TABXY(41,17),"                              "     ! clear line
4720      PRINT TABXY(59-LEN(Zl_label$)/2,17),Zl_label$
4730      PRINT TABXY(41,19),"                              "     ! clear line
4740      PRINT TABXY(59-LEN(Zl$)/2,19),Zl$
4750      DISP "Choose A Softkey..."
4760      RETURN
4770 Done_load:                  ! fills all load array elements with constant load data
4780      IF Load_type$="CONSTANT" THEN
4790        PRINT TABXY(23,3),"         Please Wait...          "
4800        DISP "Generating Load Data... Please Wait..."
4810        MAT Zl=(0)! reset all array elements to 0
4820        FOR I=1 TO Num_of_points
4830          Zl(I)=CMPLX(VAL(Rl$),VAL(Xl$))
4840        NEXT I
4850      END IF
4860      CALL Load_type_menu
4870      RETURN
4880    SUBEND
4890    !
4900    !------------------------------------------------------------------------------------------------------------
4910    SUB Plot
4920      COM /Device/@Hp8753c,@Ana_disp
4930      COM /Var1/Text$,Zs$,Zl$,Rs$,Xs$,Rl$,Xl$,Ro$,Xo$,Load_type$,Source_type$,Cs$,
          Cl$,Ls$,Ll$
4940      DIM Junk$[70]
4950      !
4960      SYSTEM PRIORITY 0                                    ! reset system priority
4970      OUTPUT @Hp8753c;"FULP;"                              ! default to full page plot
4980      PRINT TABXY(26,3),"    CONVERTZ - Plot Menu    "
4990      DISP "Plot Quadrant: [FULL]... Choose A Softkey..."
5000      ON KEY 1 LABEL " X -   - -" GOSUB Lu                 ! plot left upper quad
5010      ON KEY 2 LABEL " - X   - -" GOSUB Ru                 ! plot right upper quad
5020      ON KEY 3 LABEL " - -   X -" GOSUB Ll                 ! plot left lower quad
5030      ON KEY 4 LABEL " - -   - X" GOSUB Rl                 ! plot right lower quad
5040      ON KEY 5 LABEL " X X   X X" GOSUB Full               ! plot full page
5050      ON KEY 6 LABEL "  Plot    File" CALL Plot_file
5060      ON KEY 7 LABEL "  Plot" GOTO Endloop1                 !
5070      ON KEY 8 LABEL "Display   Menu" CALL Display_menu
```

```
5080 Loop1: GOTO Loop1                        ! waiting for softkey to be pressed
5090 Endloop1:                                ! begin plot routine
5100      DISP "Plotting... Please Wait..."
5110      PRINT TABXY(24,3),"    CONVERTZ - Plotting...    "
5120      LOOP                                ! ensure all errors are cleared out of the buffer
5130        OUTPUT @Hp8753c;"OUTPERRO;"
5140        ENTER @Hp8753c;Err,Junk$
5150        EXIT IF Err=0
5160      END LOOP
5170      !
5180      OUTPUT @Hp8753c;"CLES;ESE 2;"
5190      OUTPUT @Hp8753c;"USEPASC;PLOT;"
5200      Stat=SPOLL(@Hp8753c)
5210      IF NOT BIT(Stat,5) THEN GOTO 5200
5220      SEND 7;TALK 16 CMD 9
5230      STATUS 7,6;Hpib
5240      WAIT .5                             ! this seems to fix the windows lockup problem
5250      IF NOT BIT(Hpib,6) THEN GOTO 5230
5260      REMOTE @Hp8753c
5270      OUTPUT @Hp8753c;"TALKLIST;"         ! put ana into talker/listener mode
5280      CALL Verify_plot
5290      CALL Display_menu
5300      !
5310 Lu:                                      ! plot left upper
5320      OUTPUT @Hp8753c;"LEFU;"
5330      DISP "Plot Quadrant: [UPPER LEFT]... Choose A Softkey..."
5340      RETURN
5350 Ll:                                      ! plot left lower
5360      OUTPUT @Hp8753c;"LEFL;"
5370      DISP "Plot Quadrant: [LOWER LEFT]...  Choose A Softkey..."
5380      RETURN
5390 Ru:                                      ! plot right upper
5400      OUTPUT @Hp8753c;"RIGU;"
5410      DISP "Plot Quadrant: [UPPER RIGHT]...  Choose A Softkey..."
5420      RETURN
5430 Rl:                                      ! plot right lower
5440      OUTPUT @Hp8753c;"RIGL;"
5450      DISP "Plot Quadrant: [LOWER RIGHT]...  Choose A Softkey..."
5460      RETURN
5470 Full:                                    ! plot full scale
5480      OUTPUT @Hp8753c;"FULP;"
5490      DISP "Plot Quadrant: [FULL]...  Choose A Softkey..."
5500      RETURN
5510    SUBEND
5520    !
5530    !----------------------------------------------------------------------------------------------------------------
5540    SUB Verify_plot
5550      COM /Device/@Hp8753c,@Ana_disp
5560      DIM Err$[70]
5570      OUTPUT @Hp8753c;"OUTPERRO;"
5580      ENTER @Hp8753c;Err,Err$
5590      IF Err<>0 THEN
5600        CLEAR SCREEN
5610        DISP Err$&"... "&"CONTINUE to Proceed..."
```

```
5620        CALL Error_box
5630        CALL Error_box
5630        CLEAR @Hp8753c
5640        OUTPUT @Hp8753c;"ENTO;"
5650        CALL Title_box
5660        CALL Term_box
5670      END IF
5680    SUBEND
5690    !
5700    !----------------------------------------------------------------------------------------------------------------
5710    SUB Main_menu
5720      COM /Device/@Hp8753c,@Ana_disp
5730      ASSIGN @Hp8753c TO 716
5740      ASSIGN @Ana_disp TO 717
5750      CLEAR @Hp8753c
5760      OUTPUT @Hp8753c USING "#"        ! address device @ 716 to listen
5770      STATUS 7,7;Bstatus               ! read status reg. 7
5780      IF BIT(Bstatus,13)=1 THEN        ! bit(13)=1 if instrument listened
5790        DIM Id$[50]
5800        OUTPUT @Hp8753c;"IDN?;"        ! ask instrument to identify itself
5810        ENTER @Hp8753c;Id$
5820        IF Id$[1,21]<>"HEWLETT PACKARD,8753C" THEN
5830          DISP "HP 8753C Network Analyzer NOT Found at Address 16"
5840          STOP
5850        END IF
5860      ELSE
5870        DISP "HP 8753C Network Analyzer NOT Found at Address 16"
5880        STOP
5890      END IF
5900      !
5910      OUTPUT @Ana_disp;"AF;"           ! clears 8753c graphic display
5920      SYSTEM PRIORITY 0                ! reset system priority
5930      SEND 7;UNL UNT                   ! all instruments on bus "UNLISTEN" & "UNTALK"
5940      LOCAL 7                          ! return all instruments on bus to "LOCAL" mode
5950      !
5960      CLEAR SCREEN
5970      ALPHA PEN 1
5980      KEY LABELS PEN 3
5990      PRINT CHR$(128)
6000      PRINT TABXY(22,1),CHR$(129)&CHR$(141)&"                                   "
6010      PRINT TABXY(22,7),"                                   "
6020      FOR I=2 TO 6
6030        PRINT TABXY(22,I)," "
6040        PRINT TABXY(57,I)," "
6050      NEXT I
6060      PRINT CHR$(128)&CHR$(136)
6070      PRINT TABXY(30,3),"CONVERTZ - Main Menu"
6080      PRINT TABXY(25,5),"Written by Michael W. Allender"
6090      PRINT TABXY(17,13),"F1   Convert 50 Ohm S-Parameter Data to"
6100      PRINT TABXY(22,14),"ARBITRARY Impedance S-Parameter Data"
6110      PRINT TABXY(17,16),"F5   Create a File Containing Measured"
6120      PRINT TABXY(22,17),"Impedance Values to be Used as Source"
6130      PRINT TABXY(22,18),"and/or Load Data for S-Parameter"
6140      PRINT TABXY(22,19),"Conversion"
6150      PRINT TABXY(17,13),CHR$(138)&CHR$(129)&"F1"
```

```
6160        PRINT TABXY(17,16),"F5"&CHR$(128)&CHR$(136)
6170        !
6180        USER 1 KEYS
6190        DISP "Choose A Softkey..."
6200        ON KEY 1 LABEL " Start  Program" CALL Ana_settings
6210        ON KEY 2 LABEL "" GOTO Loop4
6220        ON KEY 3 LABEL "" GOTO Loop4
6230        ON KEY 4 LABEL "" GOTO Loop4
6240        ON KEY 5 LABEL "Measure  Zs/Zl" CALL Load_file
6250        ON KEY 6 LABEL "" GOTO Loop4
6260        ON KEY 7 LABEL "" GOTO Loop4
6270        ON KEY 8 LABEL "  Exit  CONVERTZ" GOTO Exit_program
6280 Loop4: GOTO Loop4                  ! waiting for softkey to be pressed
6290 Exit_program:                      ! exit program
6300        CLEAR SCREEN
6310        SEND 7;UNL UNT              ! tell all instruments on bus to "UNLISTEN" & "UNTALK"
6320        LOCAL 7                     ! return all instruments on bus to "LOCAL" mode
6330        DISP "Program Complete!"
6340        QUIT
6350     SUBEND
6360     !
6370     !----------------------------------------------------------------------------------------------------------------------------------
6380     SUB Load_type_menu
6390        COM /Var2/Start_freq,Stop_freq,Num_of_points,Choice
6400        SYSTEM PRIORITY 0 ! reset system priority
6410        PRINT TABXY(26,3),"  CONVERTZ - Impedance Menu   "
6420        DISP "Choose Source or Load Impedance Type..."
6430        ON KEY 1 LABEL "Constant  Zs" GOSUB Constant_zs     ! choice #1
6440        ON KEY 2 LABEL "Constant  Zl" GOSUB Constant_zl     ! choice #2
6450        ON KEY 3 LABEL "Variable  Zs" GOSUB Variable_zs     ! choice #3
6460        ON KEY 4 LABEL "Variable  Zl" GOSUB Variable_zl     ! choice #4
6470        ON KEY 5 LABEL "DataFile  Zs" GOSUB Datafile_zs     ! choice #5
6480        ON KEY 6 LABEL "DataFile  Zl" GOSUB Datafile_zl     ! choice #6
6490        ON KEY 7 LABEL "" GOTO Loop1
6500        ON KEY 8 LABEL "Display  Menu" CALL Display_menu
6510 Loop1: GOTO Loop1                              ! waiting for softkey to be pressed
6520 Constant_zs:                                   !
6530        Choice=1
6540        CALL Constant_loads                     ! this sub changes Rs & Xs
6550        RETURN
6560 Constant_zl:                                   !
6570        Choice=2
6580        CALL Constant_loads                     ! this sub changes Rl & Xl
6590        RETURN
6600 Variable_zs:                                   !
6610        Choice=3
6620        CALL Variable_loads                     ! this sub changes Rs, Cs, & Ls
6630        RETURN
6640 Variable_zl:                                   !
6650        Choice=4
6660        CALL Variable_loads                     ! this sub changes Rl, Cl, & Ll
6670        RETURN
6680 Datafile_zs:                                   !
6690        Choice=5
```

```
6700     CALL Datafile_loads                  ! this sub selects the data file for Zs
6710     RETURN
6720 Datafile_zl:                              !
6730     Choice=6
6740     CALL Datafile_loads                  ! this sub selects the data file for Zl
6750     RETURN
6760   SUBEND
6770   !
6780   !----------------------------------------------------------------------------------------------------
6790   SUB Variable_loads
6800     ! This sub obtains the user-specified variable load and source
6810     ! impedances to characterize the DUT in.
6820     !
6830     COM /Var1/Text$,Zs$,Zl$,Rs$,Xs$,Rl$,Xl$,Ro$,Xo$,Load_type$,Source_type$,Cs$,
         Cl$,Ls$,Ll$
6840     COM /Var2/Start_freq,Stop_freq,Num_of_points,Choice
6850     COM /Var3/Zs_label$,Zl_label$,Drive$,Subdir$,Source_path$,Load_path$,
         Sweep_type$
6860     COM /Loads/ COMPLEX Zs(*),Zl(*)
6870     COM /Arrays/Freq(1:1601)
6880     !
6890     SYSTEM PRIORITY 0                    ! reset system priority
6900     SELECT Choice                        ! Choice is set in load_type_menu sub
6910       CASE 3                             ! change source impedance
6920         PRINT TABXY(24,3),"   CONVERTZ - Variable Source   "
6930         USER 1 KEYS
6940         ON KEY 1 LABEL " Change    Rs" GOSUB Rs
6950         ON KEY 2 LABEL " Change    Ls" GOSUB Ls
6960         ON KEY 3 LABEL " Change    Cs" GOSUB Cs
6970         ON KEY 4 LABEL "" GOTO Loop1
6980         ON KEY 5 LABEL "" GOTO Loop1
6990         ON KEY 6 LABEL "" GOTO Loop1
7000         ON KEY 7 LABEL "" GOTO Loop1
7010         ON KEY 8 LABEL "  Done" GOSUB Done_source
7020         DISP "Choose A Softkey..."
7030 Loop1: GOTO Loop1                        ! begin looping until softkey is pressed
7040         !
7050       CASE 4                             ! change load impedance
7060         PRINT TABXY(25,3),"   CONVERTZ - Variable Load   "
7070         USER 1 KEYS
7080         ON KEY 1 LABEL " Change    Rl" GOSUB Rl
7090         ON KEY 2 LABEL " Change    Ll" GOSUB Ll
7100         ON KEY 3 LABEL " Change    Cl" GOSUB Cl
7110         ON KEY 4 LABEL "" GOTO Loop2
7120         ON KEY 5 LABEL "" GOTO Loop2
7130         ON KEY 6 LABEL "" GOTO Loop2
7140         ON KEY 7 LABEL "" GOTO Loop2
7150         ON KEY 8 LABEL "  Done" GOSUB Done_load
7160         DISP "Choose A Softkey..."
7170 Loop2: GOTO Loop2                        ! begin looping until softkey is pressed
7180     END SELECT
7190     !
7200 Rs:                                      ! changing real part of source impedance
7210     PRINT TABXY(3,15),"  Source Impedance (Zs): VARIABLE   "
```

```
7220        INPUT "Enter Real Part of Source Impedance (Rs) in Ohms:",Rs
7230        Rs$=VAL$(Rs)
7240        IF LEN(Rs$)<1 THEN GOTO Rs
7250        IF Cs$="" THEN                          ! using inductance (Ls$)
7260          Zs$="Zs = "&Rs$&" + jw"&Ls$
7270          Zs_label$="(Zs = Rs + jwLs)"
7280        ELSE                                    ! using capacitance (Cs$)
7290          Zs$="Zs = "&Rs$&" - j/w"&Cs$
7300          Zs_label$="(Zs = Rs - j/wCs)"
7310        END IF
7320        Source_type$="VARIABLE"
7330        PRINT TABXY(3,17),"                              "        ! clear line
7340        PRINT TABXY(21-LEN(Zs_label$)/2,17),Zs_label$
7350        PRINT TABXY(3,19),"                              "        ! clear line
7360        PRINT TABXY(21-LEN(Zs$)/2,19),Zs$
7370        DISP "Choose A Softkey..."
7380        RETURN
7390        !
7400 Ls:                                            ! changing source inductance
7410        PRINT TABXY(3,15),"  Source Impedance (Zs): VARIABLE  "
7420        INPUT "Enter Source Inductance (Ls) in Henrys:",Ls
7430        Ls$=VAL$(Ls)
7440        IF LEN(Ls$)<1 OR VAL(Ls$)<0 THEN GOTO Ls
7450        Cs$=""
7460        Zs$="Zs = "&Rs$&" + jw"&Ls$
7470        Zs_label$="(Zs = Rs + jwLs)"
7480        Source_type$="VARIABLE"
7490        PRINT TABXY(3,17),"                              "        ! clear line
7500        PRINT TABXY(21-LEN(Zs_label$)/2,17),Zs_label$
7510        PRINT TABXY(3,19),"                              "        ! clear line
7520        PRINT TABXY(21-LEN(Zs$)/2,19),Zs$
7530        DISP "Choose A Softkey..."
7540        RETURN
7550        !
7560 Cs:                                            ! changing source capacitance
7570        PRINT TABXY(3,15),"  Source Impedance (Zs): VARIABLE  "
7580        INPUT "Enter Source Capacitance (Cs) in Farads:",Cs
7590        Cs$=VAL$(Cs)
7600        IF LEN(Cs$)<1 OR VAL(Cs$)<0 THEN GOTO Cs
7610        Ls$=""
7620        Zs$="Zs = "&Rs$&" - j/w"&Cs$
7630        Zs_label$="(Zs = Rs - j/wCs)"
7640        Source_type$="VARIABLE"
7650        PRINT TABXY(3,17),"                              "        ! clear line
7660        PRINT TABXY(21-LEN(Zs_label$)/2,17),Zs_label$
7670        PRINT TABXY(3,19),"                              "        ! clear line
7680        PRINT TABXY(21-LEN(Zs$)/2,19),Zs$
7690        DISP "Choose A Softkey..."
7700        RETURN
7710        !
7720 Done_source:                     ! fills all source array elements with variable source data
7730        IF Source_type$="VARIABLE" THEN
7740          PRINT TABXY(23,3),"    Please Wait...    "
7750          DISP "Generating Source Data... Please Wait..."
```

```
7760        MAT Zs=(0)                          ! reset all array elements to 0
7770        IF Cs$="" THEN                      ! using inductance value
7780          FOR I=1 TO Num_of_points
7790            Zs(I)=CMPLX(VAL(Rs$),2*PI*Freq(I)*VAL(Ls$))
7800          NEXT I
7810        ELSE                                ! using capacitive value
7820          FOR I=1 TO Num_of_points
7830            Zs(I)=CMPLX(VAL(Rs$),-1/(2*PI*Freq(I)*VAL(Cs$)))
7840          NEXT I
7850        END IF
7860      END IF
7870      CALL Load_type_menu
7880      RETURN
7890      !
7900 Rl:                                        ! changing real part of load impedance
7910      PRINT TABXY(42,15),"  Load Impedance (Zl): VARIABLE  "
7920      INPUT "Enter Real Part of Load Impedance (Rl) in Ohms:",Rl
7930      Rl$=VAL$(Rl)
7940      IF LEN(Rl$)<1 THEN GOTO Rl
7950      IF Cl$="" THEN                        ! using inductance (Ll$)
7960        Zl$="Zl = "&Rl$&" + jw"&Ll$
7970        Zl_label$="(Zl = Rl + jwLl)"
7980      ELSE                                  ! using capacitance (Cl$)
7990        Zl$="Zl = "&Rl$&" - j/w"&Cl$
8000        Zl_label$="(Zl = Rl - j/wCl)"
8010      END IF
8020      Load_type$="VARIABLE"
8030      PRINT TABXY(41,17),"                                 "     ! clear line
8040      PRINT TABXY(59-LEN(Zl_label$)/2,17),Zl_label$
8050      PRINT TABXY(41,19),"                                 "     ! clear line
8060      PRINT TABXY(59-LEN(Zl$)/2,19),Zl$
8070      DISP "Choose A Softkey..."
8080      RETURN
8090      !
8100 Ll:                                        ! changing load inductance
8110      PRINT TABXY(42,15),"  Load Impedance (Zl): VARIABLE  "
8120      INPUT "Enter Load Inductance (Ll) in Henrys:",Ll
8130      Ll$=VAL$(Ll)
8140      IF LEN(Ll$)<1 OR VAL(Ll$)<0 THEN GOTO Ll
8150      Cl$=""
8160      Zl$="Zl = "&Rl$&" + jw"&Ll$
8170      Zl_label$="(Zl = Rl + jwLl)"
8180      Load_type$="VARIABLE"
8190      PRINT TABXY(41,17),"                                 "     ! clear line
8200      PRINT TABXY(59-LEN(Zl_label$)/2,17),Zl_label$
8210      PRINT TABXY(41,19),"                                 "     ! clear line
8220      PRINT TABXY(59-LEN(Zl$)/2,19),Zl$
8230      DISP "Choose A Softkey..."
8240      RETURN
8250      !
8260 Cl:                                        ! changing load capacitance
8270      PRINT TABXY(42,15),"  Load Impedance (Zl): VARIABLE  "
8280      INPUT "Enter Load Capacitance (Cl) in Farads:",Cl
8290      Cl$=VAL$(Cl)
```

```
8300     IF LEN(Cl$)<1 OR VAL(Cl$)<0 THEN GOTO Cl
8310     Ll$=""
8320     Zl$="Zl = "&Rl$&" - j/w"&Cl$
8330     Zl_label$="(Zl = Rl - j/wCl)"
8340     Load_type$="VARIABLE"
8350     PRINT TABXY(41,17),"                              "        ! clear line
8360     PRINT TABXY(59-LEN(Zl_label$)/2,17),Zl_label$
8370     PRINT TABXY(41,19),"                              "        ! clear line
8380     PRINT TABXY(59-LEN(Zl$)/2,19),Zl$
8390     DISP "Choose A Softkey..."
8400     RETURN
8410     !
8420 Done_load:                    ! fills all load array elements with variable load data
8430     IF Load_type$="VARIABLE" THEN
8440       PRINT TABXY(23,3),"      Please Wait...   "
8450       DISP "Generating Load Data... Please Wait..."
8460       MAT Zl=(0)! RESET ALL ARRAY ELEMENTS TO 0
8470       IF Cl$="" THEN                    ! using inductance value
8480         FOR I=1 TO Num_of_points
8490           Zl(I)=CMPLX(VAL(Rl$),2*PI*Freq(I)*VAL(Ll$))
8500         NEXT I
8510       ELSE                              ! using capacitive value
8520         FOR I=1 TO Num_of_points
8530           Zl(I)=CMPLX(VAL(Rl$),-1/(2*PI*Freq(I)*VAL(Cl$)))
8540         NEXT I
8550       END IF
8560     END IF
8570     CALL Load_type_menu
8580     RETURN
8590   SUBEND
8600   !
8610   !----------------------------------------------------------------------------------------------------------------
8620   SUB Load_file
8630     COM /Device/@Hp8753c,@Ana_disp
8640     CLEAR SCREEN
8650     Print_instruct:                     !
8660     SYSTEM PRIORITY 0                   ! reset system priority
8670     USER 1 KEYS
8680     CALL Title_box                      ! prints blue box at top of screen
8690     PRINT TABXY(26,3),"DOWNLOAD Zs/Zl - Instructions"
8700     PRINT TABXY(5,11),"1)  Make a Calibrated S11 Measurment of the Desired
         Source/Load on"
8710     PRINT TABXY(9,12),"Channel 1 of the HP 8753C Network Analyzer"
8720     PRINT TABXY(5,15),"2)  Store the Data in Channel 1 Memory"
8730     DISP "Press CONTINUE to proceed..."
8740     ON KEY 1 LABEL "Continue" GOTO Endloop1            !<--
8750     ON KEY 2 LABEL "" GOTO Loop1                       !  |
8760     ON KEY 3 LABEL "" GOTO Loop1                       !  |
8770     ON KEY 4 LABEL "" GOTO Loop1                       !  | softkeys for
8780     ON KEY 5 LABEL "" GOTO Loop1                       !  | "continue"
8790     ON KEY 6 LABEL "" GOTO Loop1                       !  |
8800     ON KEY 7 LABEL "" GOTO Loop1                       !  |
8810     ON KEY 8 LABEL " Main   Menu" CALL Main_menu       !<--
8820 Loop1: GOTO Loop1                                      ! waiting for softkey to be
```

```
pressed
8830 Endloop1:                                  !
8840      REMOTE @Hp8753c                       ! put network analyzer in remote mode
8850      FOR I=11 TO 15                        ! clear instructions
8860        PRINT TABXY(1,I),"                                                      "
8870      NEXT I
8880      OUTPUT @Hp8753c;"CHAN1;DISPMEMO;"  ! display memory on channel 1
8890      OUTPUT @Hp8753c;"DISPMEMO?;"       ! is there a valid memory trace?
8900      ENTER @Hp8753c;Ans                 ! Ans=1 if valid memory trace
8910      !
8920      IF Ans=0 THEN                      ! no valid memory trace
8930        SEND 7;UNL UNT         ! tell all instruments on bus to "UNLISTEN" & "UNTALK"
8940        LOCAL 7                ! return all instruments on bus to "LOCAL" mode
8950        DISP "No Data in Channel 1 Memory... Press CONTINUE to Proceed..."
8960        CALL Error_box         ! print red box with "!!!! error !!!!!" & continue
8970        GOTO Print_instruct    ! reprint instr and wait for valid memory trace
8980      END IF
8990      DISP "Downloading Source/Load Impedance Data... Please Wait..."
9000      PRINT TABXY(25,3),"Downloading Data - Please Wait"
9010      !
9020      OUTPUT @Hp8753c;"STAR?;"              ! start frequency of 8753c?
9030      ENTER @Hp8753c;Start_freq             ! get answer
9040      OUTPUT @Hp8753c;"STOP?;"              ! stop frequency of 8753c?
9050      ENTER @Hp8753c;Stop_freq              ! get answer
9060      OUTPUT @Hp8753c;"POIN?;"              ! number of data points?
9070      ENTER @Hp8753c;Num_of_points          ! get answer
9080      OUTPUT @Hp8753c;"LOGFREQ?;"           ! sweep type=log?
9090      ENTER @Hp8753c;Ans                    ! get answer
9100      IF Ans=1 THEN                         ! Ans=1 if log sweep
9110        Sweep_type$="LOG SWEEP"
9120      ELSE                                  ! see if linear sweep
9130        OUTPUT @Hp8753c;"LINFREQ?"          ! sweep type=linear?
9140        ENTER @Hp8753c;Ans                  ! get answer
9150        IF Ans=1 THEN                       ! Ans=1 if linear sweep
9160          Sweep_type$="LINEAR SWEEP"
9170        ELSE                                ! error: neither log or linear sweep
9180          CLEAR SCREEN
9190          DISP "Sweep Type Must be LOG or LINEAR... Press CONTINUE to Proceed..."
9200          CALL Error_box
9210          GOTO Print_instruct
9220        END IF
9230      END IF
9240      !
9250      DIM Real_array(1601),Imag_array(1601)      ! dim arrays for largest
9260                                                 ! number of points possible
9270      MAT Real_array=(0)                    ! initialize all array elements to "0"
9280      MAT Imag_array=(0)                    ! initialize all array elements to "0"
9290      OUTPUT @Hp8753c;"DUACOFF;CONVZREF;REAL;AUTO;FORM4;OUTPFORM;"
9300      FOR I=1 TO Num_of_points
9310        ENTER @Hp8753c;Real_array(I)        ! get real values of load (RL)
9320      NEXT I
9330      OUTPUT @Hp8753c;"IMAG;AUTO;FORM4;OUTPFORM;"
9340      FOR I=1 TO Num_of_points
9350        ENTER @Hp8753c;Imag_array(I)        ! get imaginary values of load (XL)
```

```
9360       NEXT I
9370       !
9380       PRINT TABXY(23,3),"   Download Zs/Zl - Data Path    "
9390       Drive$="C"                          ! default drive
9400       Subdir$="LOADDATA"                  ! default subdirectory
9410       Filename$="DATA"                    ! default filename
9420       Ext$=".FIL"                         ! default file extension
9430       DIM Path_name$[50]                  ! variable for data path and filenamE
9440       ON ERROR GOSUB Ertrap               ! trap all errors
9450       Err$=""                             ! initialize error variable
9460       Path_name$=Drive$&":\"&Subdir$&"\"&Filename$&Ext$
9470       !
9480 Menu:                                     ! keys for changing path/filename for data
9490       SYSTEM PRIORITY 0                   ! reset system priority
9500       GOSUB Write_instr                   ! print instructions for softkeys
9510       DISP "Current Data Storage Path & Filename:  "&Path_name$
9520       ON KEY 1 LABEL " Drive" GOSUB Drive
9530       ON KEY 2 LABEL " Subdir" GOSUB Subdir
9540       ON KEY 3 LABEL "Filename" GOSUB Filename
9550       ON KEY 4 LABEL "  Ext" GOSUB Ext
9560       ON KEY 5 LABEL "  CAT" GOSUB Catalog
9570       ON KEY 6 LABEL "" GOTO Loop3
9580       ON KEY 7 LABEL "  Save    Data" GOTO Endloop3
9590       ON KEY 8 LABEL "  Main    Menu" CALL Main_menu
9600 Loop3: GOTO Loop3                         ! waiting for softkey to be pressed
9610 Endloop3:                                 !
9620       !
9630       GOSUB Create_file                   ! verify valid path/filename & create file
9640       DISP ""
9650       PRINT TABXY(23,3),"  Saving Data... Please Wait... "
9660       ASSIGN @Data_path TO Path_name$;FORMAT ON ! dos ASCII format
9670       OUTPUT @Data_path;Start_freq        ! write start frequency to file
9680       OUTPUT @Data_path;Stop_freq         ! write stop frequency to file
9690       OUTPUT @Data_path;Num_of_points     ! write number of points to file
9700       OUTPUT @Data_path;Sweep_type$       ! write sweep type to file
9710       FOR I=1 TO Num_of_points
9720         OUTPUT @Data_path;Real_array(I),Imag_array(I)    ! data values to file
9730       NEXT I
9740       ASSIGN @Data_path TO *              ! close file
9750       OFF ERROR                           ! turn error trapping off
9760       CALL Main_menu                      ! finished - return to main menu
9770       !=================================================================
9780       !          The following are the gosubs for this subprogram
9790       !=================================================================
9800 Drive:             ! **** change drive letter (GOSUB) ****
9810       GOSUB Clear_instr
9820       OUTPUT KBD;Drive$&"_<";
9830       LINPUT "Enter Drive Letter of Storage Media - [1] Char Max:",Temp_drive$
9840       Temp_drive$=UPC$(Temp_drive$)
9850       IF LEN(Temp_drive$)<1 OR LEN(Temp_drive$)>1 THEN GOTO Drive
9860       GOSUB Check_drive                   ! verify that drive choice is valid
9870       IF Err$="" THEN                     ! assign users drive choice if valid
9880         Drive$=Temp_drive$
9890         Subdir$=""                        ! default to root directory of new drive
```

```
9900       Path_name$=Drive$&":\"&Filename$&Ext$
9910     END IF
9920     DISP "Current Data Storage Path & Filename:  "&Path_name$
9930     GOSUB Write_instr
9940     RETURN
9950     !
9960 Subdir:        ! **** change subdirectory for data file (GOSUB) ****
9970     GOSUB Clear_instr                    ! clear softkey instructions
9980     OUTPUT KBD;Subdir$;
9990     LINPUT "Enter Subdirectory Name - No Slashes (\) - [12] Chars Max:",Temp_subdir$
10000    Temp_subdir$=UPC$(Temp_subdir$)
10010    IF LEN(Temp_subdir$)>12 THEN GOTO Subdir
10020    GOSUB Check_subdir                   ! ensure a valid choice for subdirectory
10030    IF Err$="" THEN Subdir$=Temp_subdir$
10040    IF Subdir$="" THEN                   ! root directory of current drive
10050      Path_name$=Drive$&":\"&Filename$&Ext$
10060    ELSE
10070      Path_name$=Drive$&":\"&Subdir$&"\"&Filename$&Ext$
10080    END IF
10090    DISP "Current Data Storage Path & Filename:  "&Path_name$
10100    GOSUB Write_instr
10110    RETURN
10120    !
10130 Ext:          ! **** change file extension of data file (GOSUB) ****
10140    GOSUB Clear_instr
10150    IF Ext$<>"" THEN OUTPUT KBD;Ext$[2];
10160    LINPUT "Enter File Extension - No Periods (.) - [3] Chars Max:",Ext$
10170    Ext$=UPC$(Ext$)
10180    IF Ext$<>"" THEN Ext$="."&Ext$
10190    IF LEN(Ext$)>4 THEN GOTO Ext
10200    IF Subdir$="" THEN
10210      Path_name$=Drive$&":\"&Filename$&Ext$
10220    ELSE
10230      Path_name$=Drive$&":\"&Subdir$&"\"&Filename$&Ext$
10240    END IF
10250    DISP "Current Data Storage Path & Filename:  "&Path_name$
10260    GOSUB Write_instr
10270    RETURN
10280    !
10290 Filename:       ! **** change filename of data file (GOSUB) ****
10300    GOSUB Clear_instr
10310    OUTPUT KBD;Filename$;
10320    LINPUT "Enter Filename - [8] Chars Max:",Filename$
10330    Filename$=UPC$(Filename$)
10340    IF LEN(Filename$)>8 OR LEN(Filename$)<1 THEN GOTO Filename
10350    IF Subdir$="" THEN
10360      Path_name$=Drive$&":\"&Filename$&Ext$
10370    ELSE
10380      Path_name$=Drive$&":\"&Subdir$&"\"&Filename$&Ext$
10390    END IF
10400    DISP "Current Data Storage Path & Filename:  "&Path_name$
10410    GOSUB Write_instr
10420    RETURN
10430    !
```

```
10440 Catalog:    ! **** display contents of current drive/directory (GOSUB) ****
10450     Temp_drive$=Drive$           ! Temp_drive$ is drive variable for gosub Check_drive
10460     GOSUB Check_drive                  ! check for valid drive
10470     IF Err$="YES" THEN GOTO Menu       ! if not a valid drive, goto menu
10480     Temp_subdir$=Subdir$         ! Temp_subdir$ is subdir var for gosub Check_subdir
10490     GOSUB Check_subdir                 ! check for valid subdirectory
10500     IF Err$="YES" THEN GOTO Menu       ! if not a valid subdir, goto menu
10510     CLEAR SCREEN
10520     CAT
10530     SYSTEM PRIORITY 0                  ! reset system priority
10540     DISP "Press CONTINUE to proceed..."
10550     ON KEY 1 LABEL "Continue" GOTO Endloop4
10560     ON KEY 2 LABEL "" GOTO Loop4
10570     ON KEY 3 LABEL "" GOTO Loop4
10580     ON KEY 4 LABEL "" GOTO Loop4
10590     ON KEY 5 LABEL "" GOTO Loop4
10600     ON KEY 6 LABEL "" GOTO Loop4
10610     ON KEY 7 LABEL "" GOTO Loop4
10620     ON KEY 8 LABEL "" GOTO Loop4
10630 Loop4: GOTO Loop4                      ! waiting for softkey to be pressed
10640 Endloop4:                              !
10650     CLEAR SCREEN
10660     CALL Title_box
10670     PRINT TABXY(23,3),"    Download Zs/Zl - Data Path     "
10680     GOTO Menu
10690     RETURN
10700     !
10710 Check_subdir:    ! **** see if subdirectory exists (GOSUB) ****
10720     Err$=""
10730     MASS STORAGE IS Drive$&":\"&Temp_subdir$&"\"
10740     IF Err$="YES" THEN                 ! subdirectory does not exist
10750       GOSUB Clear_instr
10760       OUTPUT KBD;"Y_<";
10770       LINPUT "Subdirectory Does Not Exist... Do You Want To Create? [Y or N]",A$
10780       IF UPC$(A$)="Y" THEN
10790         Err$=""                        ! initialize error variable
10800         CREATE DIR Drive$&":\"&Temp_subdir$   ! create subdirectory
10810         IF Err$="YES" THEN             ! invalid subdirectory name
10820           CALL Error_box
10830           CALL Title_box
10840           PRINT TABXY(23,3),"    Download Zs/Zl - Data Path     "
10850         END IF
10860         IF Err$="" THEN MASS STORAGE IS Drive$&":\"&Temp_subdir$&"\"
10870       END IF
10880     END IF
10890     RETURN
10900     !
10910 Check_drive:     ! **** ensure valid drive choice (GOSUB) ****
10920     Err$=""                            ! initialize error variable
10930     MASS STORAGE IS Temp_drive$&":\"
10940     IF Err$="YES" THEN
10950       GOSUB Clear_instr
10960       CALL Error_box
10970       CALL Title_box
```

```
10980     PRINT TABXY(23,3),"    Download Zs/Zl - Data Path    "
10990   END IF
11000   RETURN
11010   !
11020 Create_file:    ! **** checks path/filename & opens file (GOSUB) ****
11030   Temp_drive$=Drive$          ! Temp_drive$ is drive variable for GOSUB Check_drive
11040   GOSUB Check_drive           ! check for valid drive
11050   IF Err$="YES" THEN GOTO Menu    ! if not a valid drive, goto menu
11060   Temp_subdir$=Subdir$        ! Temp_subdir$ is subdir var for gosub Check_subdir
11070   GOSUB Check_subdir          ! check for valid subdirectory
11080   IF Err$="YES" THEN GOTO Menu    ! if not a valid subdir, goto menu
11090   Err$=""                     ! initialize error variable
11100   Overwrite$=""               ! initialize overwrite? variable
11110   CREATE Filename$&Ext$,0     ! create file to store load data in
11120   IF Err$="YES" THEN          !
11130     IF UPC$(Overwrite$)="N" THEN
11140       GOTO Menu
11150     END IF
11160     GOSUB Clear_instr
11170     CALL Error_box
11180     CALL Title_box
11190     PRINT TABXY(23,3),"    Download Zs/Zl - Data Path    "
11200     GOTO Menu
11210   END IF
11220   RETURN
11230   !
11240 Write_instr:    ! **** prints softkey instructions (GOSUB) ****
11250   PRINT TABXY(26,9),"F1   Change Drive Letter"
11260   PRINT TABXY(26,11),"F2   Change Subdirectory"
11270   PRINT TABXY(26,13),"F3   Change File Name"
11280   PRINT TABXY(26,15),"F4   Change File Extension"
11290   PRINT TABXY(26,17),"F5   Catalog Current Directory"
11300   PRINT TABXY(26,19),"F7   Save Data to Disk"
11310   PRINT TABXY(26,21),"F8   Return to Main Menu"
11320   PRINT TABXY(26,9),CHR$(138)&CHR$(129)&"F1"
11330   PRINT TABXY(26,11),"F2"
11340   PRINT TABXY(26,13),"F3"
11350   PRINT TABXY(26,15),"F4"
11360   PRINT TABXY(26,17),"F5"
11370   PRINT TABXY(26,19),"F7"
11380   PRINT TABXY(26,21),"F8"&CHR$(128)&CHR$(136)
11390   RETURN
11400   !
11410 Clear_instr:    ! **** clears softkey instructions (GOSUB) ****
11420   PRINT TABXY(26,9),"                              "
11430   PRINT TABXY(26,11),"                              "
11440   PRINT TABXY(26,13),"                              "
11450   PRINT TABXY(26,15),"                              "
11460   PRINT TABXY(26,17),"                              "
11470   PRINT TABXY(26,19),"                              "
11480   PRINT TABXY(26,21),"                              "
11490   RETURN
11500   !
11510 Ertrap:         ! **** error handler (on error) ****
```

```
11520     Err$="YES"                              ! set error variable to "YES"
11530     SELECT ERRN                             ! select error type
11540       CASE 54                               ! error - duplicate filename
11550         GOSUB Clear_instr
11560         OUTPUT KBD;"N_<";
11570         LINPUT "Duplicate FileName - OverWrite? [Y or N]:",Overwrite$
11580         IF UPC$(Overwrite$)<>"Y" THEN Overwrite$="N"
11590         IF UPC$(Overwrite$)="Y" THEN
11600           PURGE Filename$&Ext$
11610           CREATE Filename$&Ext$,0           ! create file to store load data in
11620           Err$=""
11630         END IF
11640       CASE 56                               ! error - invalid subdirectory name
11650         IF ERRL(11110) THEN
11660           DISP "Invalid FileName... Press CONTINUE to Proceed..."
11670         END IF
11680         IF ERRL(10930) THEN
11690           DISP "Invalid Drive Specification... Press CONTINUE to Proceed..."
11700         END IF
11710         IF ERRL(10800) THEN
11720           DISP "Invalid Subdirectory Name... Press CONTINUE to Proceed..."
11730         END IF
11740       CASE 196                              ! error - disk drive not ready
11750         DISP "Disk Drive Not Ready... Press CONTINUE to Proceed..."
11760       CASE ELSE                             ! error - bad news if program ends up here
11770         CLEAR SCREEN
11780         DISP "!!!!!!!!!! UNRECOVERABLE APPLICATION ERROR !!!!!!!!!!"
11790         STOP                              ! unaccounted for error condition... program stops
11800     END SELECT
11810     ERROR RETURN
11820   SUBEND
11830   !
11840   !-------------------------------------------------------------------------------------------------------
11850   SUB Error_box
11860     SYSTEM PRIORITY 0                       ! reset system priority
11870     PRINT CHR$(129)&CHR$(137)
11880     PRINT TABXY(22,1),"                        "
11890     PRINT TABXY(22,5),"                        "
11900     FOR I=2 TO 4
11910       PRINT TABXY(22,I)," "
11920       PRINT TABXY(57,I)," "
11930     NEXT I
11940     PRINT CHR$(128)&CHR$(136)
11950     PRINT TABXY(23,3),"       !!! ERROR !!!        "
11960     !
11970     ON KEY 1 LABEL "Continue" GOTO Endloop1
11980     ON KEY 2 LABEL "" GOTO Loop1
11990     ON KEY 3 LABEL "" GOTO Loop1
12000     ON KEY 4 LABEL "" GOTO Loop1
12010     ON KEY 5 LABEL "" GOTO Loop1
12020     ON KEY 6 LABEL "" GOTO Loop1
12030     ON KEY 7 LABEL "" GOTO Loop1
12040     ON KEY 8 LABEL "" GOTO Loop1
12050 Loop1: GOTO Loop1                           ! waiting for softkey to be pressed
```

```
12060 Endloop1:                              !
12070  SUBEND
12080  !
12090  !-------------------------------------------------------------------------------------------------------------
12100  SUB Title_box
12110    PRINT CHR$(129)&CHR$(141)
12120    PRINT TABXY(22,1),"                                    "
12130    PRINT TABXY(22,5),"                                    "
12140    FOR I=2 TO 4
12150      PRINT TABXY(22,I)," "
12160      PRINT TABXY(57,I)," "
12170    NEXT I
12180    PRINT CHR$(128)&CHR$(136)
12190  SUBEND
12200  !
12210  !-------------------------------------------------------------------------------------------------------------
12220  SUB Term_box
12230    COM /Var1/Text$,Zs$,Zl$,Rs$,Xs$,Rl$,Xl$,Ro$,Xo$,Load_type$,Source_type$,Cs$,
         Cl$,Ls$,Ll$
12240    COM /Var3/Zs_label$,Zl_label$,Drive$,Subdir$,Source_path$,Load_path$,
         Sweep_type$
12250    DIM Temp$[75]
12260    PRINT CHR$(129)&CHR$(141)
12270    PRINT TABXY(24,9),"                                "
12280    PRINT TABXY(2,13),"                                                                            "
12290    PRINT TABXY(2,21),"                                                                            "
12300    PRINT CHR$(136)
12310    FOR I=10 TO 12
12320      PRINT TABXY(24,I)," "
12330      PRINT TABXY(55,I)," "
12340    NEXT I
12350    FOR I=14 TO 20
12360      PRINT TABXY(2,I)," "
12370      PRINT TABXY(40,I)," "
12380      PRINT TABXY(78,I)," "
12390    NEXT I
12400    PRINT CHR$(128)&CHR$(136)
12410    PRINT TABXY(26,11),"Active Source/Load Impedance"
12420    Temp$="Source Impedance (Zs): "&Source_type$
12430    PRINT TABXY(21-LEN(Temp$)/2,15),Temp$
12440    Temp$="Load Impedance (Zl): "&Load_type$
12450    PRINT TABXY(59-LEN(Temp$)/2,15),Temp$
12460    PRINT TABXY(21-LEN(Zs_label$)/2,17),Zs_label$
12470    PRINT TABXY(59-LEN(Zl_label$)/2,17),Zl_label$
12480    PRINT TABXY(21-LEN(Zs$)/2,19),Zs$
12490    PRINT TABXY(59-LEN(Zl$)/2,19),Zl$
12500  SUBEND
12510  !
12520  !-------------------------------------------------------------------------------------------------------------
12530  SUB Datafile_loads
12540    COM /Var1/Text$,Zs$,Zl$,Rs$,Xs$,Rl$,Xl$,Ro$,Xo$,Load_type$,Source_type$,Cs$,
         Cl$,Ls$,Ll$
12550    COM /Var2/Start_freq,Stop_freq,Num_of_points,Choice
12560    COM /Var3/Zs_label$,Zl_label$,Drive$,Subdir$,Source_path$,Load_path$,
```

```
       Sweep_type$
12570  COM /Loads/ COMPLEX Zs(*),Zl(*)
12580  DIM Title$[75]
12590  !
12600  SELECT Choice                      ! Choice is set in Load_type_menu sub
12610    CASE 5                           ! change source impedance
12620      Title$=" CONVERTZ - Data File Source "
12630    CASE 6                           ! change load impedance
12640      Title$=" CONVERTZ - Data File Load  "
12650  END SELECT
12660  PRINT TABXY(40-LEN(Title$)/2,3),Title$
12670  !
12680  ON ERROR GOSUB Ertrap              ! trap all errors
12690 Menu:                               !
12700  SYSTEM PRIORITY 0                  ! reset system priority
12710  USER 1 KEYS
12720  !
12730  ON KEY 1 LABEL " File    Name" GOSUB Datafile
12740  ON KEY 2 LABEL "Change  Drive" GOSUB Drive
12750  ON KEY 3 LABEL " Change  Subdir" GOSUB Subdir
12760  ON KEY 4 LABEL "  CAT" GOSUB Catelog
12770  ON KEY 5 LABEL "" GOTO Loop1
12780  ON KEY 6 LABEL "" GOTO Loop1
12790  ON KEY 7 LABEL "" GOTO Loop1
12800  ON KEY 8 LABEL "  EXIT    Menu" GOSUB Done
12810  PRINT TABXY(1,24),"CURRENT DATA PATH: "&Drive$&":\"&Subdir$
12820  DISP "Choose A Softkey..."
12830 Loop1: GOTO Loop1                   ! begin looping until softkey is pressed
12840  !
12850 Drive:          ! change current data drive (GOSUB)
12860  OUTPUT KBD;Drive$&"_<";
12870  LINPUT "Enter Drive Letter Containing Data File:",Temp_drive$
12880  Temp_drive$=UPC$(Temp_drive$)
12890  IF LEN(Temp_drive$)<1 OR LEN(Temp_drive$)>1 THEN GOTO Drive
12900  GOSUB Check_drive                  ! verify that drive choice is valid
12910  IF Err$="" THEN                    ! if no error then...
12920    Drive$=Temp_drive$               ! assign new drive if valid
12930    Subdir$=""
12940  END IF
12950  PRINT TABXY(1,24),"                                            "
12960  PRINT TABXY(1,24),"CURRENT DATA PATH:  "&Drive$&":\"&Subdir$
12970  DISP "Choose A Softkey..."
12980  RETURN
12990  !
13000 Check_drive:   ! verifies that disk drive is valid and ready (GOSUB)
13010  Err$=""                            ! initialize error variable
13020  MASS STORAGE IS Temp_drive$&":\"
13030  IF Err$="YES" THEN
13040    CALL Error_box
13050    CALL Title_box
13060    CALL Term_box
13070    PRINT TABXY(40-LEN(Title$)/2,3),Title$
13080  END IF
13090  RETURN
```

```
13100    !
13110    RETURN
13120 Subdir:          ! change current subdirectory (GOSUB)
13130    OUTPUT KBD;Subdir$;
13140    LINPUT "Enter Subdirectory Containing Data File:",Temp_subdir$
13150    Temp_subdir$=UPC$(Temp_subdir$)
13160    IF LEN(Temp_subdir$)>12 THEN GOTO Subdir
13170    GOSUB Check_subdir                ! ensure a valid choice for subdirectory
13180    IF Err$="" THEN                   ! if no error then...
13190      Subdir$=Temp_subdir$            ! assigns new subdirectory if valid
13200    END IF
13210    PRINT TABXY(1,24),"                                    "
13220    PRINT TABXY(1,24),"CURRENT DATA PATH:  "&Drive$&":\"&Subdir$
13230    DISP "Choose A Softkey..."
13240    RETURN
13250 Check_subdir:          ! verify subdirectory exists (GOSUB)
13260    Err$=""
13270    MASS STORAGE IS Drive$&":\"&Temp_subdir$
13280    IF Err$="YES" THEN                ! subdirectory does not exist
13290      CALL Error_box
13300      CALL Title_box
13310      CALL Term_box
13320      PRINT TABXY(40-LEN(Title$)/2,3),Title$
13330    END IF
13340    RETURN
13350 Catelog:         ! print directory of current MSI (GOSUB)
13360    Temp_drive$=Drive$       ! Temp_drive$ is drive variable for GOSUB Check_drive
13370    GOSUB Check_drive                 ! check for valid drive
13380    IF Err$="YES" THEN GOTO Menu      ! if not a valid drive, goto menu
13390    Temp_subdir$=Subdir$     ! Temp_subdir$ is subdir var for GOSUB Check_subdir
13400    GOSUB Check_subdir                ! check for valid subdirectory
13410    IF Err$="YES" THEN GOTO Menu      ! if not a valid subdir, goto menu
13420    CLEAR SCREEN
13430    CAT
13440    SYSTEM PRIORITY 0                 ! reset system priority
13450    DISP "Press CONTINUE to proceed..."
13460    ON KEY 1 LABEL "Continue" GOTO Endloop2
13470    ON KEY 2 LABEL "" GOTO Loop2
13480    ON KEY 3 LABEL "" GOTO Loop2
13490    ON KEY 4 LABEL "" GOTO Loop2
13500    ON KEY 5 LABEL "" GOTO Loop2
13510    ON KEY 6 LABEL "" GOTO Loop2
13520    ON KEY 7 LABEL "" GOTO Loop2
13530    ON KEY 8 LABEL "" GOTO Loop2
13540 Loop2: GOTO Loop2                    ! waiting for softkey to be pressed
13550 Endloop2:                            !
13560    CLEAR SCREEN
13570    CALL Title_box
13580    CALL Term_box
13590    PRINT TABXY(40-LEN(Title$)/2,3),Title$
13600    GOTO Menu
13610    RETURN
13620 Datafile:          ! get filename.ext of data file (GOSUB)
13630    LINPUT "Enter FILENAME.EXT of Data File:",Datafile$
```

```
13640     IF LEN(Datafile$)=0 OR LEN(Datafile$)>12 THEN GOTO Datafile
13650     Datafile$=UPC$(Datafile$)
13660     Temp_drive$=Drive$          ! Temp_drive$ is drive variable for GOSUB Check_drive
13670     GOSUB Check_drive                        ! check for valid drive
13680     IF Err$="YES" THEN GOTO Menu        ! if not a valid drive, goto menu
13690     Temp_subdir$=Subdir$        ! Temp_subdir$ is subdir var for GOSUB Check_subdir
13700     GOSUB Check_subdir                       ! check for valid subdirectory
13710     IF Err$="YES" THEN GOTO Menu        ! if not a valid subdir, goto menu
13720     ASSIGN @Datafile TO Datafile$;FORMAT ON
13730     IF Err$="YES" THEN
13740       CALL Error_box
13750       CALL Title_box
13760       CALL Term_box
13770       PRINT TABXY(40-LEN(Title$)/2,3),Title$
13780       GOTO Menu
13790     END IF
13800     !
13810     ENTER @Datafile;Temp_start_freq;Temp_stop_freq;Temp_num_of_pts
13820                                              ! read start,stop & # of points
13830                                              ! from data file
13840     ENTER @Datafile;Temp_sweep_type$        ! read sweep type of data file
13850     ASSIGN @Datafile TO *                    ! close file
13860     Invalid_file$=""                         ! initialize error variable
13870     IF Temp_start_freq<>Start_freq OR Temp_stop_freq<>Stop_freq OR
          Temp_num_of_pts<>Num_of_points OR Temp_sweep_type$<>Sweep_type$ THEN
          Invalid_file$="YES"
13880     IF Invalid_file$="YES" THEN              ! files parameters did not match 8753c's
13890       CLEAR SCREEN
13900       DISP "HP 8753C Parameters Do Not Match Data File Parameters... CONTINUE To
            Proceed..."
13910       CALL Error_box                         ! print red error box at top of screen @ pause
13920       CALL Title_box                         ! print blue title box at top of screen
13930       CALL Term_box                          ! print source/load termination box
13940       PRINT TABXY(40-LEN(Title$)/2,3),Title$
13950       GOTO Menu                              ! return to prompt menu
13960     END IF
13970     SELECT Choice
13980       CASE 5                                 ! source impedance
13990         Source_type$="DATA FILE"
14000         Zs_label$="(Zs = Rs + jXs)"
14010         PRINT TABXY(3,15),"  Source Impedance (Zs): DATA FILE  "
14020         IF Subdir$<>"" THEN
14030           Zs$="Data File: "&Drive$&":\"&Subdir$&"\"&Datafile$
14040           Source_path$=Drive$&":\"&Subdir$&"\"&Datafile$
14050         ELSE
14060           Zs$="Data File: "&Drive$&":\"&Datafile$
14070           Source_path$=Drive$&":\"&Datafile$
14080         END IF
14090         PRINT TABXY(3,17),"                         "  ! clear line
14100         PRINT TABXY(21-LEN(Zs_label$)/2,17),Zs_label$
14110         PRINT TABXY(3,19),"                         "  ! clear line
14120         PRINT TABXY(21-LEN(Zs$)/2,19),Zs$
14130         ASSIGN @Datafile TO Datafile$;FORMAT ON
14140         PRINT TABXY(23,3),"        Please Wait...        "
```

```
14150        DISP "Reading Data File... Please Wait..."
14160        MAT Zs=(0)! RESET ALL ARRAY ELEMENTS TO 0
14170        ENTER @Datafile;A;B;C             ! read first 3 lines in data file
14180        ENTER @Datafile;Junk$             ! read fourth line in data file
14190        FOR I=1 TO Num_of_points
14200          ENTER @Datafile;Zs(I)           ! read source data from data file
14210        NEXT I
14220        ASSIGN @Datafile TO *             ! close file
14230      CASE 6                              ! load impedance
14240        Load_type$="DATA FILE"
14250        PRINT TABXY(42,15)," Load Impedance (Zl): DATA FILE  "
14260        Zl_label$="(Zl = Rl + jXl)"
14270        IF Subdir$<>"" THEN
14280          Zl$="Data File: "&Drive$&":\"&Subdir$&"\"&Datafile$
14290          Load_path$=Drive$&":\"&Subdir$&"\"&Datafile$
14300        ELSE
14310          Zl$="Data File: "&Drive$&":\"&Datafile$
14320          Load_path$=Drive$&":\"&Datafile$
14330        END IF
14340        PRINT TABXY(41,17),"                          "   ! clear line
14350        PRINT TABXY(59-LEN(Zl_label$)/2,17),Zl_label$
14360        PRINT TABXY(41,19),"                          "   ! clear line
14370        PRINT TABXY(59-LEN(Zl$)/2,19),Zl$
14380        ASSIGN @Datafile TO Datafile$;FORMAT ON
14390        PRINT TABXY(23,3),"        Please Wait...      "
14400        DISP "Reading Data File... Please Wait..."
14410        MAT Zl=(0)                        ! reset all array elements to 0
14420        ENTER @Datafile;A;B;C             ! read first three lines in data file
14430        ENTER @Datafile;Junk$             ! read fourth line in data file
14440        FOR I=1 TO Num_of_points
14450          ENTER @Datafile;Zl(I)           ! read load data from data file
14460        NEXT I
14470        ASSIGN @Datafile TO *             ! close file
14480    END SELECT
14490    PRINT TABXY(1,24),"                                                  "
14500    OFF ERROR
14510    CALL Load_type_menu
14520    RETURN
14530 Done:                                  !
14540    OFF ERROR
14550    PRINT TABXY(1,24),"                                                  "
14560    CALL Load_type_menu
14570    RETURN
14580 Ertrap:                      ! **** error handler (on error) ****
14590    Err$="YES"                            ! set error variable to "YES"
14600    ! PRINT ERRN                          ! for trouble shooting
14610    ! PAUSE                               ! for trouble shooting
14620    SELECT ERRN                           ! select error type
14630      CASE 56                             ! error - file or subdirectory not found
14640        IF ERRL(14130) THEN
14650          CLEAR SCREEN
14660          DISP "Invalid Path... Press CONTINUE to Proceed..."
14670        END IF
14680        IF ERRL(13020) THEN
```

```
14690          CLEAR SCREEN
14700          DISP "Invalid Drive Specification... Press CONTINUE to Proceed..."
14710        END IF
14720        IF ERRL(13270) THEN
14730          CLEAR SCREEN
14740          DISP "Subdirectory Does Not Exist... Press CONTINUE to Proceed..."
14750        END IF
14760        IF ERRL(13720) THEN
14770          CLEAR SCREEN
14780          DISP "File Not Found... Press CONTINUE to Proceed..."
14790        END IF
14800      CASE 183
14810        CLEAR SCREEN
14820        DISP "Invalid Path... Press CONTINUE to Proceed..."
14830      CASE 196                           ! error - disk drive not ready
14840        CLEAR SCREEN
14850        DISP "Disk Drive Not Ready... Press CONTINUE to Proceed..."
14860      CASE ELSE                          ! error - bad news if program ends up here
14870        CLEAR SCREEN
14880        DISP "!!!!!!!!!! UNRECOVERABLE APPLICATION ERROR !!!!!!!!!!"
14890        STOP ! unaccounted for error condition... program stops
14900    END SELECT
14910    ERROR RETURN
14920  SUBEND
14930  !
14940  !-----------------------------------------------------------------------------------
14950  SUB Ana_graphics(From_sub$)            ! puts 8753c in graphics mode & prints source
14960                                         ! and load information on analyzers' screen
14970    COM /Device/@Hp8753c,@Ana_disp
14980    COM /Var1/Text$[50],Zs$[100],Zl$[100],Rs$[15],Xs$[15],Rl$[15],Xl$[15],Ro$[5],Xo$[5],
Load_type$[25],Source_type$[25],Cs$[15],Cl$[15],Ls$[15],Ll$[15]
14990    COM /Var3/Zs_label$[50],Zl_label$[50],Drive$[1],Subdir$[12],Source_path$[50],
         Load_path$[50],Sweep_type$[15]
15000    DIM Line1$[50],Line2$[50],Line3$[50],Line4$[50],Line5$[50]
15010    !
15020    Line1$="S-Parameter.... "&From_sub$
15030    Line2$="Source (Zs)... "&Zs$
15040    Line3$="Load (Zl)....."&Zl$
15050    IF Source_type$="DATA FILE" THEN Line2$="Source (Zs)... "&Zs$[12]
15060    IF Load_type$="DATA FILE" THEN Line3$="Load (Zl)..... "&Zl$[12]
15070    !
15080    OUTPUT @Ana_disp;"AF;"               ! clears 8753c graphic display
15090    OUTPUT @Ana_disp;"PU;PA 400,3850;LB"&Line1$&CHR$(3)&";"
15100    OUTPUT @Ana_disp;"PA 485,3725;LB"&Line2$&CHR$(3)&";"
15110    OUTPUT @Ana_disp;"PA 485,3600;LB"&Line3$&CHR$(3)&";"
15120    !
15130  SUBEND
15140  !
15150  !-----------------------------------------------------------------------------------
15160  SUB Plot_file
15170    COM /Device/@Hp8753c,@Ana_disp
15180    COM /Var1/Text$,Zs$,Zl$,Rs$,Xs$,Rl$,Xl$,Ro$,Xo$,Load_type$,Source_type$,Cs$,
         Cl$,Ls$,Ll$
15190    DIM Junk$[70]
```

```
15200     !
15210     PRINT TABXY(24,3),"    CONVERTZ - Print Menu     "
15220     !
15230     LINPUT "Enter Filename (NO Extension): ",Name$
15240     IF LEN(Name$)>8 THEN GOTO 15220
15250     PRINT TABXY(24,3),"    CONVERTZ - Printing...    "
15260     DISP "Writing Data File... Please Wait..."
15270     Name$="c:\"&Name$&".plt"
15280     CREATE Name$,0
15290     ASSIGN @Datafile TO Name$;FORMAT ON
15300     DIM A$[32000]
15310     OUTPUT @Hp8753c;"FORM4;OUTPPLOT;"
15320     FOR I=1 TO 5
15330       ENTER @Hp8753c;A$
15340       OUTPUT @Datafile;A$
15350     NEXT I
15360     ASSIGN @Datafile TO *                     ! close file
15370     CALL Display_menu
15380   SUBEND
15390   !
15400   !-----------------------------------------------------------------------------------------------------------
15410   SUB Scale
15420     COM /Device/@Hp8753c,@Ana_disp
15430     !
15440     SYSTEM PRIORITY 0                         ! reset system priority
15450     PRINT TABXY(26,3),"   CONVERTZ - Scale Menu     "
15460     DISP "Choose A Softkey..."
15470     ON KEY 1 LABEL "  Auto   Scale" GOSUB Auto            !
15480     ON KEY 2 LABEL "  Scale   /Div" GOSUB Div             !
15490     ON KEY 3 LABEL "  Ref    Value" GOSUB Ref_val         !
15500     ON KEY 4 LABEL "  Ref   Position" GOSUB Ref_pos     !
15510     ON KEY 5 LABEL "" GOTO Loop1
15520     ON KEY 6 LABEL "" GOTO Loop1
15530     ON KEY 7 LABEL "" GOTO Loop1
15540     ON KEY 8 LABEL "Display   Menu" CALL Display_menu
15550 Loop1: GOTO Loop1                             ! waiting for softkey to be pressed
15560 Auto:                                         ! autoscale the current display
15570     OUTPUT @Hp8753c;"AUTO;"
15580     RETURN
15590 Div:                                      ! change # of units per division on current display
15600     LINPUT "Enter UNITS/DIV:",Scale$
15610     Temp$="SCAL "&Scale$&";"
15620     OUTPUT @Hp8753c;Temp$
15630     DISP "Choose A Softkey..."
15640     RETURN
15650 Ref_val:                                  ! change the reference value of the current display
15660     LINPUT "Enter Value Of Reference Line:",Refval$
15670     Temp$="REFV "&Refval$&";"
15680     OUTPUT @Hp8753c;Temp$
15690     DISP "Choose A Softkey..."
15700     RETURN
15710 Ref_pos:                              ! change the reference position of the current display
15720     LINPUT "Enter Position Of Reference Line (0=Bottom, 10=Top):",Refpos$
15730     Temp$="REFP "&Refpos$&";"
```

```
15740     OUTPUT @Hp8753c;Temp$
15750     DISP "Choose A Softkey..."
15760     RETURN
15770  SUBEND
15780  !
15790  !------------------------------------------------------------------------------------------------------------
```

APPENDIX B

RE_COMP Software Listing

```
10     !                    Radiated Emissions Experiment
20     !                  Written by Michael W. Allender 1/11/95
30     !
40     !
50     ! This program requires a hp 8591a spectrum analyzer, gigatronics 8542 power meter,
60     ! hp 8645a signal generator, and dc 3001 directional coupler (pe# 1419)
70     !
80     COM /Device/@Hp8645a,@Gig8542,@Hp8591a
90     COM /Var2/Start_freq,Stop_freq,Num_of_points,Current_freq,Drive_level,Ref_lvl
100    !
110    CLEAR SCREEN
120    ! initialize all variables
130    Start_freq=200
140    Stop_freq=500
150    Num_of_points=101
160    Ref_lvl=0
170    Drive_level=-35
180    Drive$="C"
190    Subdir$="LOADDATA"
200    !
210    CALL Title_box
220    CALL Disp_box
230    !
240    CALL Main_menu
250    !
260    END                                         ! end of main program
270    !
280    !*****************************************************************************************
290    !                         Beginning of Subroutines
300    !*****************************************************************************************
310    !
320    !-----------------------------------------------------------------------------------------
330    SUB Main_menu
340      COM /Device/@Hp8645a,@Gig8542,@Hp8591a
350      COM /Var2/Start_freq,Stop_freq,Num_of_points,Current_freq,Drive_level,Ref_lvl
360      DIM Temp$[30],Temp2$[40]
370      !
380      ASSIGN @Hp8645a TO 720              ! signal generator
390      ASSIGN @Hp8591a TO 718              ! spectrum analyzer
400      ASSIGN @Gig8542 TO 713              ! power meter
410      ABORT 7                             ! halt all bus activity
420      CLEAR 7                             ! reset hpib interface
430      !
440      SYSTEM PRIORITY 0                   ! reset system priority
450      !
```

```
460       USER 1 KEYS
470       DISP "Choose A Softkey..."
480       ON KEY 1 LABEL " Start  Freq" GOTO Start_freq
490       ON KEY 2 LABEL " Stop  Freq" GOTO Stop_freq
500       ON KEY 3 LABEL " Num of  Points" GOTO Num_poin
510       ON KEY 4 LABEL "" GOTO Loop
520       ON KEY 5 LABEL " Run   Program" CALL Main_program
530       ON KEY 6 LABEL "" GOTO Loop
540       ON KEY 7 LABEL "" GOTO Loop
550       ON KEY 8 LABEL "  Exit  Program" GOTO Exit_program
560 Loop: GOTO Loop                              ! waiting for softkey to be pressed
570 Start_freq:                                  ! setting start frequency
580       LINPUT "Enter Desired Start Frequency (MHz): ",Start$
590       Start_freq=VAL(Start$)                 ! set global variable
600       Temp$="Start Frequency.............."
610       Temp2$=Temp$[1,25-LEN(Start$)]&" "&Start$&" MHz"
620       PRINT TABXY(25,5),Temp2$
630       GOTO Loop
640 Stop_freq:                                   ! setting stop frequency
650       LINPUT "Enter Desired Stop Frequency (MHz): ",Stop$
660       Stop_freq=VAL(Stop$)                   ! set global variable
670       Temp$="Stop Frequency................"
680       Temp2$=Temp$[1,25-LEN(Stop$)]&" "&Stop$&" MHz"
690       PRINT TABXY(25,6),Temp2$
700       GOTO Loop
710 Num_poin:                                    ! setting number of points
720       LINPUT "Enter Desired Number Of Data Points: ",Num_poin$
730       Num_of_points=VAL(Num_poin$)
740       Temp$="Number of Points............."
750       Temp2$=Temp$[1,29-LEN(Num_poin$)]&" "&Num_poin$
760       PRINT TABXY(25,7),Temp2$
770       GOTO Loop
780 Exit_program:                                ! exit program
790       CLEAR SCREEN
800       SEND 7;UNL UNT               ! tell all instruments on bus to "UNLISTEN" & "UNTALK"
810       LOCAL 7                      ! return all instruments on bus to "LOCAL" mode
820       DISP "Program Complete!"
830       STOP
840     SUBEND
850     !
860     !-------------------------------------------------------------------------------------
870     SUB Disp_box
880       DIM Temp$[75]
890       PRINT CHR$(129)&CHR$(141)
900       PRINT TABXY(3,13),"                                                  "
910       PRINT TABXY(3,22),"                                                  "
920       PRINT CHR$(136)
930       FOR I=10 TO 12
940         PRINT TABXY(24,I)," "
950         PRINT TABXY(55,I)," "
960       NEXT I
970       FOR I=14 TO 21
980         PRINT TABXY(5,I)," "
990         PRINT TABXY(40,I)," "
```

```
1000        PRINT TABXY(75,I)," "
1010      NEXT I
1020      PRINT CHR$(129)&CHR$(136)
1030      PRINT TABXY(14,14)," Data Without DUT "
1040      PRINT TABXY(51,14)," Data With DUT "
1050      PRINT CHR$(128)&CHR$(136)     ! reset colors/attributes
1060      PRINT TABXY(25,11),"                              "
1070      PRINT TABXY(31,11),"Current Freq:  N/A"
1080      CALL Clear_values
1090      !
1100    SUBEND
1110    !
1120    !----------------------------------------------------------------------------------------------------
1130    SUB Title_box
1140      COM /Var2/Start_freq,Stop_freq,Num_of_points,Current_freq,Drive_level,Ref_lvl
1150      DIM Temp$[30],Temp2$[40]
1160      PRINT CHR$(129)&CHR$(141)
1170      PRINT TABXY(22,1),"                                   "
1180      PRINT TABXY(22,9),"                                   "
1190      FOR I=2 TO 8
1200        PRINT TABXY(22,I)," "
1210        PRINT TABXY(57,I)," "
1220      NEXT I
1230      PRINT CHR$(128)&CHR$(136)
1240      PRINT TABXY(31,3),"RADIATED EMISSIONS"
1250      Temp$="Start Frequency..............."
1260      Temp2$=Temp$[1,25-LEN(VAL$(Start_freq))]&" "&VAL$(Start_freq)&" MHz"
1270      PRINT TABXY(25,5),Temp2$
1280      Temp$="Stop Frequency................"
1290      Temp2$=Temp$[1,25-LEN(VAL$(Stop_freq))]&" "&VAL$(Stop_freq)&" MHz"
1300      PRINT TABXY(25,6),Temp2$
1310      Temp$="Number of Points............."
1320      Temp2$=Temp$[1,29-LEN(VAL$(Num_of_points))]&" "&VAL$(Num_of_points)
1330      PRINT TABXY(25,7),Temp2$
1340    SUBEND
1350    !
1360    !----------------------------------------------------------------------------------------------------
1370    SUB Main_program
1380      COM /Device/@Hp8645a,@Gig8542,@Hp8591A
1390      COM /Var2/Start_freq,Stop_freq,Num_of_points,Current_freq,Drive_level,Ref_lvl
1400      DIM Temp$[50],Temp2$[50]
1410      !
1420      OUTPUT @Hp8645a;"*RST;"           ! preset sig gen
1430      OUTPUT @Hp8591a;"IP;"             ! preset spectrum analyzer
1440      OUTPUT @Gig8542;"PR;"             ! preset power meter
1450      OUTPUT @Gig8542;"AE RE2EN;BE RE2EN;"                ! set display res XX.XX
1460      OUTPUT @Hp8645a;"AMPL -120DBM;AMPL:STATE ON;"  ! set sig gen to initial
1470                                                        ! low level amplitude
1480      OUTPUT @Hp8591a;"SP 10KHZ;"       ! set span of spec an to 10 kHz
1490      OUTPUT @Hp8591a;"RL"&VAL$(Ref_lvl)&"DB;"            ! set ref level of spec an
1500      DISP ""
1510      SYSTEM PRIORITY 0                 ! reset system priority
1520      USER 1 KEYS
1530      ON KEY 1 LABEL " PAUSE" GOSUB Temp_pause
```

```
1540    ON KEY 2 LABEL "" GOSUB Do_nothing
1550    ON KEY 3 LABEL "" GOSUB Do_nothing
1560    ON KEY 4 LABEL "" GOSUB Do_nothing
1570    ON KEY 5 LABEL "" GOSUB Do_nothing
1580    ON KEY 6 LABEL "" GOSUB Do_nothing
1590    ON KEY 7 LABEL "" GOSUB Do_nothing
1600    ON KEY 8 LABEL "" GOSUB Do_nothing
1610    ALLOCATE Array(Num_of_points-1,11)
1620    !                          Array format
1630    !  ____________________________________________________________________
1640    !  |       no dut            |            w/dut
1650    ! freq | drv | for pwr | ref p | net p | rad p| drv | for pwr | ref p | net p | rad p | flag
1660    !  0   |  1  |    2    |   3   |   4   |   5  |  6  |    7    |   8   |   9   |  10   | 11
1670    !------------------------------------------------------------------------------------------
1680    Incr=(Stop_freq-Start_freq)/(Num_of_points-1)
1690    Current_freq=Start_freq
1700    Array_count=0
1710    FOR I=1 TO Num_of_points
1720      Freq$=VAL$(Current_freq)
1730      !
1740      SELECT POS(Freq$,".")            ! round freq to 1/10000's
1750        CASE 0  ! no decimal point (integer) - do nothing
1760        CASE 2  ! 1 - 9 MHz
1770          Freq$=Freq$[1,6]
1780        CASE 3  ! 10 - 99 MHz
1790          Freq$=Freq$[1,7]
1800        CASE 4  ! 100 - 999 MHz
1810          Freq$=Freq$[1,8]
1820        CASE 5  ! 1000 - 9999 MHz
1830          Freq$=Freq$[1,9]
1840      END SELECT
1850      !
1860      Array(Array_count,0)=VAL(Freq$)    ! store freq in array
1870      Temp$="Current Freq:  "&Freq$&" MHz"
1880      PRINT TABXY(25,11),"                          "  ! clear line
1890      PRINT TABXY(40-LEN(Temp$)/2,11),Temp$
1900      OUTPUT @Hp8591a;"CF"&Freq$&"MHZ;"    ! set center freq of spec an
1910      Forward_power=0                      ! initialize forward power
1920      Current_drv_lvl=Drive_level          ! initialize current drive level
1930      CALL Set_power_meter                 ! enter offset values into pwr mtr
1940      OUTPUT @Hp8645a;"FREQ "&Freq$&"MHZ;"               ! set freq for sig gen
1950      !
1960      WHILE Forward_power<26.4 OR Forward_power>26.6       ! 400 to 500 mW
1970        OUTPUT @Hp8645a;"AMPL "&VAL$(Current_drv_lvl)&"DBM;"
1980        Temp2$="Drive Level................."
1990        PRINT TABXY(9,16),Temp2$[1,23-LEN(VAL$(Current_drv_lvl))]&"
"&VAL$(Current_drv_lvl)&" dBm"
2000        Array(Array_count,1)=Current_drv_lvl
2010        WAIT 2                             ! ensure time for power meter to respond
2020        OUTPUT @Gig8542;"AP;"              ! channel a of power meter
2030        !
2040        REPEAT
2050          All_clear=0
2060          WAIT .5
```

```
2070            ENTER @Gig8542;Power_1         ! read channel a value
2080            WAIT .5
2090            ENTER @Gig8542;Power_2         ! read channel a value
2100            WAIT .5
2110            ENTER @Gig8542;Power_3         ! read channel a value
2120            Up_bound=Power_1+.2
2130            Low_bound=Power_1-.2
2140            IF Power_2<Low_bound OR Power_2>Up_bound THEN
2150              All_clear=1
2160              PRINT TABXY(1,1);"POWER ERROR"
2170            END IF
2180            IF Power_3<Low_bound OR Power_3>Up_bound THEN
2190              All_clear=1
2200              PRINT TABXY(1,1);"POWER ERROR"
2210            END IF
2220          UNTIL All_clear=0
2230          Forward_power=Power_1
2240          !
2250          Temp2$="Forward Power.............."
2260          PRINT TABXY(9,17),Temp2$[1,23-LEN(VAL$(Forward_power))]&" "&
              VAL$(Forward_power)&" dBm"
2270          Array(Array_count,2)=Forward_power
2280          !--------------------------------------------------------------------------------------------------
2290          SELECT Forward_power
2300            CASE <26.4
2310              Current_drv_lvl=Current_drv_lvl+(26.5-Forward_power)
2320            CASE >26.6
2330              Current_drv_lvl=Current_drv_lvl-ABS(26.5-Forward_power)
2340            END SELECT
2350          IF Current_drv_lvl>-3 THEN
2360            DISP "WARNING... DRIVE LEVEL EXCEEDING -3 dBm... PROGRAM
                PAUSED..."
2370            PAUSE
2380          END IF
2390          !--------------------------------------------------------------------------------------------------
2400        END WHILE                          ! while for drive level
2410        !
2420        WAIT 5                             ! delay needed so spec an has time to respond
2430                                           ! to drive level change before it max holds
2440        OUTPUT @Hp8591a;"MXMH TRA;"        ! put spec an into max hold
2450        OUTPUT @Gig8542;"BP;"              ! channel b of power meter
2460        !
2470        REPEAT
2480          All_clear=0
2490          WAIT .5
2500          ENTER @Gig8542;Rev_power_1 ! read reflected power
2510          WAIT .5
2520          ENTER @Gig8542;Rev_power_2 ! read reflected power
2530          WAIT .5
2540          ENTER @Gig8542;Rev_power_3 ! read reflected power
2550          Up_bound=Rev_power_1+.2
2560          Low_bound=Rev_power_1-.2
2570          IF Rev_power_2<Low_bound OR Rev_power_2>Up_bound THEN
2580            All_clear=1
```

```
2590          PRINT TABXY(1,1);"REVERSE POWER ERROR"
2600        END IF
2610        IF Rev_power_3<Low_bound OR Rev_power_3>Up_bound THEN
2620          All_clear=1
2630          PRINT TABXY(1,1);"REVERSE POWER ERROR"
2640        END IF
2650      UNTIL All_clear=0
2660      Reverse_power=Rev_power_1
2670      !
2680      IF Reverse_power>=-25 THEN
2690      ELSE
2700        Reverse_power=0
2710      END IF
2720      Temp2$="Reverse Power..............."
2730      PRINT TABXY(9,18),Temp2$[1,23-LEN(VAL$(Reverse_power))]&" "&
          VAL$(Reverse_power)&" dBm"
2740      Array(Array_count,3)=Reverse_power
2750      IF Forward_power>Reverse_power THEN
2760        Net_power=10*LGT(10^(Forward_power/10)-10^(Reverse_power/10))
2770      ELSE                                 ! net power=0
2780        CLEAR SCREEN
2790        DISP "100% MISMATCH... RECONFIGURE SETUP AND TRY AGAIN..."
2800        PAUSE
2810        GOTO 2450
2820      END IF
2830      Net$=VAL$(Net_power)
2840      Net$=Net$[1,5]
2850      Array(Array_count,4)=VAL(Net$)
2860      Temp2$="Net Power.................."
2870      PRINT TABXY(9,19),Temp2$[1,23-LEN(Net$)]&" "&Net$&" dBm"
2880      WAIT 10                              ! get good peak reading on spec an.
2890      OUTPUT @Hp8591a;"MKPK;"              ! peak search
2900      ! WAIT 2
2910      OUTPUT @Hp8591a;"MKA?"               ! query for result
2920      ENTER @Hp8591a;Sa_power              ! get max peak value
2930      Temp2$="Radiated Power............."
2940      PRINT TABXY(9,20),Temp2$[1,23-LEN(VAL$(Sa_power))]&" "&VAL$(Sa_power)&
          " dBm"
2950      Array(Array_count,5)=Sa_power
2960      OUTPUT @Hp8645a;"AMPL -120DBM;"          ! turn off sig gen output power
2970      WAIT 1                               ! 1 sec delay for crt viewing
2980      OUTPUT @Hp8591a;"CLRW TRA;" ! put spec an into clr/wrt mode
2990      CALL Clear_values
3000      Array_count=Array_count+1
3010      Current_freq=Current_freq+Incr       ! increment frequency
3020    NEXT I
3030    !=================================================================
3040    LINPUT "Insert DUT into System... Press RETURN to Continue...",A$
3050    !=================================================================
3060    Array_count=0
3070    FOR I=1 TO Num_of_points
3080      Current_freq=Array(Array_count,0)
3090      Freq$=VAL$(Current_freq)
3100      Temp$="Current Freq:  "&Freq$&" MHz"
```

```
3110    PRINT TABXY(25,11),"                              " ! clear line
3120    PRINT TABXY(40-LEN(Temp$)/2,11),Temp$
3130    OUTPUT @Hp8591a;"CF"&Freq$&"MHZ;"   ! set center freq
3140                                        ! of spectrum analyzer
3150    Forward_power=0                  ! initialize forward power
3160    Current_drv_lvl=Drive_level      ! initialize current drive level
3170    CALL Set_power_meter             ! enter offset values into pwr mtr
3180    ! Display previous data
3190    Temp2$="Drive Level................."
3200    PRINT TABXY(9,16),Temp2$[1,23-LEN(VAL$(Array(Array_count,1)))]&" "
        &VAL$(Array(Array_count,1))&" dBm"
3210    Temp2$="Forward Power.............."
3220    PRINT TABXY(9,17),Temp2$[1,23-LEN(VAL$(Array(Array_count,2)))]&" "
        &VAL$(Array(Array_count,2))&" dBm"
3230    Temp2$="Reverse Power.............."
3240    PRINT TABXY(9,18),Temp2$[1,23-LEN(VAL$(Array(Array_count,3)))]&" "
        &VAL$(Array(Array_count,3))&" dBm"
3250    Temp2$="Net Power..................."
3260    PRINT TABXY(9,19),Temp2$[1,23-LEN(VAL$(Array(Array_count,4)))]&" "
        &VAL$(Array(Array_count,4))&" dBm"
3270    Temp2$="Radiated Power.............."
3280    PRINT TABXY(9,20),Temp2$[1,23-LEN(VAL$(Array(Array_count,5)))]&" "
        &VAL$(Array(Array_count,5))&" dBm"
3290    !
3300    OUTPUT @Hp8645a;"FREQ "&Freq$&"MHZ;"     ! set freq for sig gen
3310    Net_pwr_no_dut=Array(Array_count,4)
3320    Offset=0                         ! initialize offset variable
3330    WHILE Net_pwr_no_dut<-25         ! forward pwr <-25 is difficult for
3340                                     ! for pwr sensors to read.
3350      Net_pwr_no_dut=Net_pwr_no_dut+10
3360      Offset=Offset+10
3370    END WHILE
3380    Lower_bound=.99*Net_pwr_no_dut          ! tolerance within 1%
3390    Upper_bound=1.01*Net_pwr_no_dut         ! tolerance within 1%
3400    !
3410    WHILE Forward_power<Lower_bound OR Forward_power>Upper_bound
3420      OUTPUT @Hp8645a;"AMPL "&VAL$(Current_drv_lvl)&"DBM;"
3430      Temp2$="Drive Level................."
3440      PRINT TABXY(44,16),Temp2$[1,23-LEN(VAL$(Current_drv_lvl-Offset))]&" "
          &VAL$(Current_drv_lvl-Offset)&" dBm"
3450      Array(Array_count,6)=Current_drv_lvl-Offset
3460      WAIT 2                         ! settling time before forward power
3470      OUTPUT @Gig8542;"AP;"          ! channel a of power meter
3480      !
3490      REPEAT
3500        All_clear=0
3510        WAIT .5
3520        ENTER @Gig8542;Power_1       ! read forward power
3530        WAIT .5
3540        ENTER @Gig8542;Power_2       ! read forward power
3550        WAIT .5
3560        ENTER @Gig8542;Power_3       ! read forward power
3570        Up_bound_pfor=Power_1+.2
3580        Low_bound_pfor=Power_1-.2
```

```
3590          IF Power_2<Low_bound_pfor OR Power_2>Up_bound_pfor THEN
3600            All_clear=1
3610            PRINT TABXY(1,1);"POWER ERROR W/ DUT"
3620          END IF
3630          IF Power_3<Low_bound_pfor OR Power_3>Up_bound_pfor THEN
3640            All_clear=1
3650            PRINT TABXY(1,1);"POWER ERROR W/ DUT"
3660          END IF
3670        UNTIL All_clear=0
3680        Forward_power=Power_1
3690        !
3700        Temp2$="Forward Power..............."
3710        PRINT TABXY(44,17),Temp2$[1,23-LEN(VAL$(Forward_power-Offset))]&" "
            &VAL$(Forward_power-Offset)&" dBm"
3720        Array(Array_count,7)=Forward_power-Offset
3730        !
3740        SELECT Forward_power
3750          CASE <Lower_bound
3760            Current_drv_lvl=Current_drv_lvl+(Net_pwr_no_dut-Forward_power)
3770          CASE >Upper_bound
3780            Current_drv_lvl=Current_drv_lvl-ABS(Net_pwr_no_dut-Forward_power)
3790          END SELECT
3800      END WHILE
3810      !
3820      WAIT 5                                 ! delay needed so spec an has time to
3830                                             ! respond to drive level change before
3840                                             ! it max holds
3850      OUTPUT @Hp8591a;"MXMH TRA;" ! put spec an into max hold
3860      OUTPUT @Gig8542;"BP;"
3870      !
3880      REPEAT
3890        All_clear=0
3900        WAIT .5
3910        ENTER @Gig8542;Rev_power_1
3920        WAIT .5
3930        ENTER @Gig8542;Rev_power_2
3940        WAIT .5
3950        ENTER @Gig8542;Rev_power_3
3960        Up_bound_prev=Rev_power_1+.2
3970        Low_bound_prev=Rev_power_1-.2
3980        IF Rev_power_2<Low_bound_prev OR Rev_power_2>Up_bound_prev THEN
3990          All_clear=1
4000          PRINT TABXY(1,1);"REVERSE POWER ERROR W/ DUT"
4010        END IF
4020        IF Rev_power_3<Low_bound_prev OR Rev_power_3>Up_bound_prev THEN
4030          All_clear=1
4040          PRINT TABXY(1,1);"REVERSE POWER ERROR W/ DUT"
4050        END IF
4060      UNTIL All_clear=0
4070      Reverse_power=Rev_power_1
4080      !
4090      IF Reverse_power>=-25 THEN
4100      ELSE
4110        Reverse_power=0
```

```
4120      END IF
4130      Reverse_power=Reverse_power-Offset        ! subtract offset value
4140      Temp2$="Reverse Power.............."
4150      PRINT TABXY(44,18),Temp2$[1,23-LEN(VAL$(Reverse_power))]&" "
          &VAL$(Reverse_power)&" dBm"
4160      Array(Array_count,8)=Reverse_power
4170      Net_power=10*LGT(10^(Forward_power/10)-10^(Reverse_power/10))
4180      Net$=VAL$(Net_power)
4190      Net$=Net$[1,5]
4200      Array(Array_count,9)=VAL(Net$)
4210      Temp2$="Net Power.................."
4220      PRINT TABXY(44,19),Temp2$[1,23-LEN(Net$)]&" "&Net$&" dBm"
4230      WAIT 10                               ! get good peak reading
4240      OUTPUT @Hp8591a;"MKPK;"               ! peak search
4250      ! WAIT 2
4260      OUTPUT @Hp8591a;"MKA?"                ! query for result
4270      ENTER @Hp8591a;Sa_power               ! get max peak value
4280      IF Sa_power<-70 THEN                  ! increase res bw of spec anal
4290      OUTPUT @Hp8591a;"CLRW TRA;" ! put spec an into clr/wrt mode
4300      OUTPUT @Hp8591a;"RL -30 DB;"
4310      WAIT 7                                ! ensure screen update before max
4320                                            ! hold
4330      OUTPUT @Hp8591a;"MXMH TRA;" ! put spec anal into max hold
4340      WAIT 10
4350      OUTPUT @Hp8591a;"MKPK;"               ! peak search
4360      ! WAIT 2
4370      OUTPUT @Hp8591a;"MKA?"                ! query for result
4380      ENTER @Hp8591a;Sa_power               ! get max peak value
4390      OUTPUT @Hp8591a;"RL"&VAL$(Ref_lvl)&"DB;"
4400      END IF
4410      Temp2$="Radiated Power............."
4420      Sa_power=Sa_power-Offset
4430      PRINT TABXY(44,20),Temp2$[1,23-LEN(VAL$(Sa_power))]&" "&VAL$(Sa_power)
          &" dBm"
4440      Array(Array_count,10)=Sa_power
4450      OUTPUT @Hp8645a;"AMPL -120DBM;"             ! turn off sig gen output power
4460      OUTPUT @Hp8591a;"CLRW TRA;"                 ! put spec an into clr/wrt mode
4470      IF Offset<>0 THEN Array(Array_count,11)=1   ! flag for low for. power
4480      WAIT 1                                      ! delay for crt viewing of data
4490      Array_count=Array_count+1
4500      CALL Clear_values
4510    NEXT I
4520    CALL Get_filepath(@Data_path)
4530    FOR I=0 TO Num_of_points-1
4540      FOR J=0 TO 11
4550        OUTPUT @Data_path;Array(I,J),
4560      NEXT J
4570      OUTPUT @Data_path;""
4580    NEXT I
4590    ASSIGN @Data_path TO *                  ! close file
4600    CLEAR SCREEN
4610 Exit_early:                                ! if net power=0 program comes here
4620    CALL Title_box
4630    CALL Disp_box
```

```
4640      DEALLOCATE Array(*)                 ! free up memory
4650      SUBEXIT
4660 Temp_pause:                              ! entered when user hits pause softkey
4670      SYSTEM PRIORITY 0                   ! reset system priority
4680      ON KEY 1 LABEL "CONTINUE" GOTO Done
4690 Loop1:                                   !
4700      GOTO Loop1
4710 Done:                                    !
4720      ON KEY 1 LABEL " PAUSE" GOSUB Temp_pause
4730      RETURN
4740 Do_nothing:                      ! entered when user hits any other softkey but "pause"
4750      RETURN
4760    SUBEND
4770    !
4780    !------------------------------------------------------------------------------------------------
4790    SUB Set_power_meter
4800      COM /Device/@Hp8645a,@Gig8542,@Hp8591a
4810      COM /Var2/Start_freq,Stop_freq,Num_of_points,Current_freq,Drive_level,Ref_lvl
4820      DIM Temp$[60]
4830      !
4840      SELECT Current_freq                 ! calc forward coupling for directional coupler
4850        CASE <=148
4860          Fcoupl=-.0014286*Current_freq-40.3485714
4870        CASE <=553
4880          Fcoupl=.0021235*Current_freq-40.8742955
4890        CASE <=626
4900          Fcoupl=-.0016438*Current_freq-38.7909786
4910        CASE <=666
4920          Fcoupl=.002*Current_freq-41.072
4930        CASE <=764
4940          Fcoupl=-.0041837*Current_freq-36.9536532
4950        CASE <=811
4960          Fcoupl=.0023404*Current_freq-41.9380644
4970        CASE <=890
4980          Fcoupl=-.0048101*Current_freq-36.1390089
4990        CASE <=958
5000          Fcoupl=.0089706*Current_freq-48.403834
5010        CASE <=1000
5020          Fcoupl=-.0007143*Current_freq-39.1257006
5030      END SELECT
5040      !
5050      SELECT Current_freq                 ! calc reverse coupling for directional coupler
5060        CASE <=143
5070          Rcoupl=-.0009859*Current_freq-40.3690141
5080        CASE <=536
5090          Rcoupl=.0019593*Current_freq-40.7901799
5100        CASE <=609
5110          Rcoupl=.000274*Current_freq-39.88684
5120        CASE <=669
5130          Rcoupl=.0025*Current_freq-41.2425
5140        CASE <=758
5150          Rcoupl=-.0034832*Current_freq-37.2397753
5160        CASE <=807
5170          Rcoupl=.0008163*Current_freq-40.4987554
```

```
5180        CASE <=859
5190          Rcoupl=-.0057692*Current_freq-35.1842556
5200        CASE <=950
5210          Rcoupl=.0064835*Current_freq-45.7093265
5220        CASE <=1000
5230          Rcoupl=-.0086*Current_freq-31.38
5240      END SELECT
5250      !
5260      Temp$="AEOS"&VAL$(ABS(DROUND(Fcoupl,4)))&"EN;BEOS"&
          VAL$(ABS(DROUND(Rcoupl,4)))&"EN;"
5270      OUTPUT @Gig8542;Temp$      ! set offset values for directional coupler
5280      Temp$="AEFR"&VAL$(Current_freq)&"MZ;BEFR"&VAL$(Current_freq)&"MZ;"
5290      OUTPUT @Gig8542;Temp$          ! set current freq for internal cal. factors
5300    SUBEND
5310    !
5320    !------------------------------------------------------------------------------------------
5330    SUB Clear_values
5340      !
5350      PRINT TABXY(9,16),"Drive Level................"
5360      PRINT TABXY(9,17),"Forward Power.............."
5370      PRINT TABXY(9,18),"Reverse Power.............."
5380      PRINT TABXY(9,19),"Net Power.................."
5390      PRINT TABXY(9,20),"Radiated Power............."
5400      !
5410      PRINT TABXY(44,16),"Drive Level................"
5420      PRINT TABXY(44,17),"Forward Power.............."
5430      PRINT TABXY(44,18),"Reverse Power.............."
5440      PRINT TABXY(44,19),"Net Power.................."
5450      PRINT TABXY(44,20),"Radiated Power............."
5460      !
5470    SUBEND
5480    !
5490    !------------------------------------------------------------------------------------------
5500    SUB Get_filepath(@Data_path)
5510      !
5520      CLEAR SCREEN
5530      CALL Title2_box
5540      PRINT TABXY(23,3),"     Set Data Path    "
5550      Drive$="C"                     ! default drive
5560      Subdir$="LOADDATA"             ! default subdirectory
5570      Filename$="DATA"               ! default filename
5580      Ext$=".FIL"                    ! default file extension
5590      DIM Path_name$[50]             ! variable for data path and filename
5600      ON ERROR GOSUB Ertrap          ! trap all errors
5610      Err$=""                        ! initialize error variable
5620      Path_name$=Drive$&":\"&Subdir$&"\"&Filename$&Ext$
5630      !
5640 Menu:                               ! keys for changing path/filename for data
5650      SYSTEM PRIORITY 0              ! reset system priority
5660      GOSUB Write_instr              ! print instructions for softkeys
5670      DISP "Current Data Storage Path & Filename:  "&Path_name$
5680      ON KEY 1 LABEL " Drive" GOSUB Drive
5690      ON KEY 2 LABEL " Subdir" GOSUB Subdir
5700      ON KEY 3 LABEL "Filename" GOSUB Filename
```

```
5710        ON KEY 4 LABEL "  Ext" GOSUB Ext
5720        ON KEY 5 LABEL "  CAT" GOSUB Catalog
5730        ON KEY 6 LABEL "" GOTO Loop3
5740        ON KEY 7 LABEL "" GOTO Loop3
5750        ON KEY 8 LABEL "  Save    Data" GOTO Endloop3
5760 Loop3: GOTO Loop3                         ! waiting for softkey to be pressed
5770 Endloop3:                                 !
5780        !
5790        GOSUB Create_file                  ! verify valid path/filename & create file
5800        DISP ""
5810        PRINT TABXY(23,3),"   Saving Data... Please Wait... "
5820        ASSIGN @Data_path TO Path_name$;FORMAT ON   ! dos ASCII format
5830        ! OUTPUT @Data_path;Start_freq      ! write start frequency to file
5840        OFF ERROR                          ! turn error trapping off
5850        SUBEXIT
5860        !=====================================================================
5870        !       The following are the gosubs for this subprogram
5880        !=====================================================================
5890 Drive:               ! **** change drive letter (GOSUB) ****
5900        GOSUB Clear_instr
5910        Subdir$=""                         ! default to root directory of drive
5920        OUTPUT KBD;Drive$&"_<";
5930        LINPUT "Enter Drive Letter of Storage Media - [1] Char Max:",Temp_drive$
5940        Temp_drive$=UPC$(Temp_drive$)
5950        IF LEN(Temp_drive$)<1 OR LEN(Temp_drive$)>1 THEN GOTO Drive
5960        GOSUB Check_drive                  ! verify that drive choice is valid
5970        IF Err$="" THEN Drive$=Temp_drive$ ! assign users drive choice if valid
5980        Path_name$=Drive$&":\"&Filename$&Ext$
5990        DISP "Current Data Storage Path & Filename:  "&Path_name$
6000        GOSUB Write_instr
6010        RETURN
6020        !
6030 Subdir:           ! **** change subdirectory for data file (GOSUB) ****
6040        GOSUB Clear_instr                  ! clear softkey instructions
6050        OUTPUT KBD;Subdir$;
6060        LINPUT "Enter Subdirectory Name - No Slashes (\) - [12] Chars Max:",Temp_subdir$
6070        Temp_subdir$=UPC$(Temp_subdir$)
6080        IF LEN(Temp_subdir$)>12 THEN GOTO Subdir
6090        GOSUB Check_subdir                 ! ensure a valid choice for subdirectory
6100        IF Err$="" THEN Subdir$=Temp_subdir$
6110        IF Subdir$="" THEN                 ! root directory of current drive
6120          Path_name$=Drive$&":\"&Filename$&Ext$
6130        ELSE
6140          Path_name$=Drive$&":\"&Subdir$&"\"&Filename$&Ext$
6150        END IF
6160        DISP "Current Data Storage Path & Filename:  "&Path_name$
6170        GOSUB Write_instr
6180        RETURN
6190        !
6200 Ext:              ! **** change file extension of data file (GOSUB) ****
6210        GOSUB Clear_instr
6220        IF Ext$<>"" THEN OUTPUT KBD;Ext$[2];
6230        LINPUT "Enter File Extension - No Periods (.) - [3] Chars Max:",Ext$
6240        Ext$=UPC$(Ext$)
```

```
6250      IF Ext$<>"" THEN Ext$="."&Ext$
6260      IF LEN(Ext$)>4 THEN GOTO Ext
6270      IF Subdir$="" THEN
6280        Path_name$=Drive$&":\"&Filename$&Ext$
6290      ELSE
6300        Path_name$=Drive$&":\"&Subdir$&"\"&Filename$&Ext$
6310      END IF
6320      DISP "Current Data Storage Path & Filename:  "&Path_name$
6330      GOSUB Write_instr
6340      RETURN
6350      !
6360 Filename:        ! **** change filename of data file (GOSUB) ****
6370      GOSUB Clear_instr
6380      OUTPUT KBD;Filename$;
6390      LINPUT "Enter Filename - [8] Chars Max:",Filename$
6400      Filename$=UPC$(Filename$)
6410      IF LEN(Filename$)>8 OR LEN(Filename$)<1 THEN GOTO Filename
6420      IF Subdir$="" THEN
6430        Path_name$=Drive$&":\"&Filename$&Ext$
6440      ELSE
6450        Path_name$=Drive$&":\"&Subdir$&"\"&Filename$&Ext$
6460      END IF
6470      DISP "Current Data Storage Path & Filename:  "&Path_name$
6480      GOSUB Write_instr
6490      RETURN
6500      !
6510 Catalog:     ! **** display contents of current drive/directory (GOSUB) ****
6520      Temp_drive$=Drive$             ! Temp_drive$ is drive variable for GOSUB Check_drive
6530      GOSUB Check_drive                        ! check for valid drive
6540      IF Err$="YES" THEN GOTO Menu       ! if not a valid drive, goto menu
6550      Temp_subdir$=Subdir$          ! Temp_subdir$ is subdir var for GOSUB Check_subdir
6560      GOSUB Check_subdir                ! check for valid subdirectory
6570      IF Err$="YES" THEN GOTO Menu       ! if not a valid subdir, goto menu
6580      CLEAR SCREEN
6590      CAT
6600      SYSTEM PRIORITY 0                        ! reset system priority
6610      DISP "Press CONTINUE to proceed..."
6620      ON KEY 1 LABEL "Continue" GOTO Endloop4
6630      ON KEY 2 LABEL "" GOTO Loop4
6640      ON KEY 3 LABEL "" GOTO Loop4
6650      ON KEY 4 LABEL "" GOTO Loop4
6660      ON KEY 5 LABEL "" GOTO Loop4
6670      ON KEY 6 LABEL "" GOTO Loop4
6680      ON KEY 7 LABEL "" GOTO Loop4
6690      ON KEY 8 LABEL "" GOTO Loop4
6700 Loop4: GOTO Loop4                        ! waiting for softkey to be pressed
6710 Endloop4:                                !
6720      CLEAR SCREEN
6730      CALL Title2_box
6740      PRINT TABXY(23,3),"       Set Data Path          "
6750      GOTO Menu
6760      RETURN
6770      !
6780 Check_subdir:        ! **** see if subdirectory exists (GOSUB) ****
```

```
6790        Err$=""
6800        MASS STORAGE IS Drive$&":\"&Temp_subdir$&"\"
6810        IF Err$="YES" THEN                   ! subdirectory does not exist
6820          GOSUB Clear_instr
6830          OUTPUT KBD;"Y_<";
6840          LINPUT "Subdirectory Does Not Exist... Do You Want To Create? [Y or N]",A$
6850          IF UPC$(A$)="Y" THEN
6860            Err$=""                          ! initialize error variable
6870            CREATE DIR Drive$&":\"&Temp_subdir$   ! create subdirectory
6880            IF Err$="YES" THEN               ! invalid subdirectory name
6890              CALL Error_box
6900              CALL Title2_box
6910              PRINT TABXY(23,3),"       Set Data Path        "
6920            END IF
6930            IF Err$="" THEN MASS STORAGE IS Drive$&":\"&Temp_subdir$&"\"
6940          END IF
6950        END IF
6960        RETURN
6970        !
6980 Check_drive:        ! **** ensure valid drive choice (GOSUB) ****
6990        Err$=""                              ! initialize error variable
7000        MASS STORAGE IS Temp_drive$&":\"
7010        IF Err$="YES" THEN
7020          GOSUB Clear_instr
7030          CALL Error_box
7040          CALL Title2_box
7050          PRINT TABXY(23,3),"       Set Data Path        "
7060          END IF
7070        RETURN
7080        !
7090 Create_file:        ! **** checks path/filename & opens file (GOSUB) ****
7100        Temp_drive$=Drive$          ! Temp_drive$ is drive variable for GOSUB Check_drive
7110        GOSUB Check_drive                    ! check for valid drive
7120        IF Err$="YES" THEN GOTO Menu         ! if not a valid drive, goto menu
7130        Temp_subdir$=Subdir$        ! Temp_subdir$ is subdir var for GOSUB Check_subdir
7140        GOSUB Check_subdir                   ! check for valid subdirectory
7150        IF Err$="YES" THEN GOTO Menu         ! if not a valid subdir, goto menu
7160        Err$=""                              ! initialize error variable
7170        Overwrite$=""                        ! initialize overwrite? variable
7180        CREATE Filename$&Ext$,0              ! create file to store load data in
7190        IF Err$="YES" THEN                   !
7200          IF UPC$(Overwrite$)="N" THEN
7210            GOTO Menu
7220          END IF
7230          GOSUB Clear_instr
7240          CALL Error_box
7250          CALL Title2_box
7260          PRINT TABXY(23,3),"       Set Data Path        "
7270          GOTO Menu
7280        END IF
7290        RETURN
7300        !
7310 Write_instr:        ! **** prints softkey instructions (GOSUB) ****
7320        PRINT TABXY(26,9),"F1   Change Drive Letter"
```

```
7330      PRINT TABXY(26,11),"F2  Change Subdirectory"
7340      PRINT TABXY(26,13),"F3  Change File Name"
7350      PRINT TABXY(26,15),"F4  Change File Extension"
7360      PRINT TABXY(26,17),"F5  Catalog Current Directory"
7370      PRINT TABXY(26,19),"F8  Save Data to Disk"
7380      PRINT TABXY(26,9),CHR$(138)&CHR$(129)&"F1"
7390      PRINT TABXY(26,11),"F2"
7400      PRINT TABXY(26,13),"F3"
7410      PRINT TABXY(26,15),"F4"
7420      PRINT TABXY(26,17),"F5"
7430      PRINT TABXY(26,19),"F8"&CHR$(128)&CHR$(136)
7440      RETURN
7450      !
7460 Clear_instr:   ! **** clears softkey instructions (GOSUB) ****
7470      PRINT TABXY(26,9),"                              "
7480      PRINT TABXY(26,11),"                             "
7490      PRINT TABXY(26,13),"                             "
7500      PRINT TABXY(26,15),"                             "
7510      PRINT TABXY(26,17),"                             "
7520      PRINT TABXY(26,19),"                             "
7530      RETURN
7540      !
7550 Ertrap:        ! **** error handler (ON ERROR) ****
7560      Err$="YES"                          ! set error variable to "YES"
7570      SELECT ERRN                         ! select error type
7580        CASE 52                           ! error - invalid drive
7590          IF ERRL(7000) THEN
7600            DISP "Invalid Drive Specification... Press CONTINUE to Proceed..."
7610          END IF
7620          IF ERRL(6800) THEN              ! subdirectory does not exist
7630                                          ! return to program with Err$="Y"
7640          END IF
7650        CASE 53                           ! error - improper filename
7660          DISP "Invalid FileName... Press CONTINUE to Proceed..."
7670        CASE 54                           ! error - duplicate filename
7680          GOSUB Clear_instr
7690          OUTPUT KBD;"N_<";
7700          LINPUT "Duplicate FileName - OverWrite? [Y or N]:",Overwrite$
7710          IF UPC$(Overwrite$)<>"Y" THEN Overwrite$="N"
7720          IF UPC$(Overwrite$)="Y" THEN
7730            PURGE Filename$&Ext$
7740            CREATE Filename$&Ext$,0       ! create file to store load data in
7750            Err$=""
7760          END IF
7770        CASE 56                           ! error - invalid subdirectory name
7780          DISP "Invalid Subdirectory Name... Press CONTINUE to Proceed..."
7790        CASE 80                           ! error - disk drive not ready
7800          DISP "Disk Drive Not Ready... Press CONTINUE to Proceed..."
7810        CASE ELSE                         ! error - bad news if program ends up here
7820          CLEAR SCREEN
7830          DISP "!!!!!!!!!! UNRECOVERABLE APPLICATION ERROR !!!!!!!!!!"
7840          STOP                      ! unaccounted for error condition... program stops
7850      END SELECT
7860      ERROR RETURN
```

```
7870   SUBEND
7880   !
7890   !------------------------------------------------------------------------------------------------
7900   SUB Error_box
7910     SYSTEM PRIORITY 0                    ! reset system priority
7920     PRINT CHR$(129)&CHR$(137)
7930     PRINT TABXY(22,1),"                   "
7940     PRINT TABXY(22,5),"                   "
7950     FOR I=2 TO 4
7960       PRINT TABXY(22,I)," "
7970       PRINT TABXY(57,I)," "
7980     NEXT I
7990     PRINT CHR$(128)&CHR$(136)
8000     PRINT TABXY(23,3),"   !!! ERROR !!!"
8010     !
8020     ON KEY 1 LABEL "Continue" GOTO Endloop1
8030     ON KEY 2 LABEL "" GOTO Loop1
8040     ON KEY 3 LABEL "" GOTO Loop1
8050     ON KEY 4 LABEL "" GOTO Loop1
8060     ON KEY 5 LABEL "" GOTO Loop1
8070     ON KEY 6 LABEL "" GOTO Loop1
8080     ON KEY 7 LABEL "" GOTO Loop1
8090     ON KEY 8 LABEL "" GOTO Loop1
8100 Loop1: GOTO Loop1                        ! waiting for softkey to be pressed
8110 Endloop1:                                !
8120   SUBEND
8130   !
8140   !------------------------------------------------------------------------------------------------
8150   SUB Title2_box
8160     PRINT CHR$(129)&CHR$(141)
8170     PRINT TABXY(22,1),"                   "
8180     PRINT TABXY(22,5),"                   "
8190     FOR I=2 TO 4
8200     PRINT TABXY(22,I)," "
8210     PRINT TABXY(57,I)," "
8220     NEXT I
8230     PRINT CHR$(128)&CHR$(136)
8240   SUBEND
8250   !
8260   !------------------------------------------------------------------------------------------------
```

REFERENCES

[1] C. G. Masi, "Fun with S-Parameters," Test & Measurement World, May 1992, pp. 75-78.

[2] T. Grosch, "Introduction to S-Parameters," RF Design, September 1993, pp. 64-69.

[3] K. Kurokawa, "Power Waves and the Scattering Matrix," IEEE Transactions on Microwave Theory and Techniques, vol. MTT-13, March 1965, pp. 194-202.

[4] C. Slater, "Survey of Modern RF and Microwave Test Instruments," RF and Microwave Device Test for the '90s (Seminar Papers), 1993, pp. 1-29.

[5] T. Hillstrom, "Make S-Parameter Measurements in Mixed Impedances," Microwaves & RF, January 1992, pp. 111-118.

[6] "HP 85150B Microwave Design System," Hewlett-Packard Technical Data 5091-2295E, 1991, pp. 1-23.

[7] G. Breed, "Attenuator Basics," RF Design, February 1992, p. 77.