Accessible Broadband Network Analysis

by

Esa Han Hsien Masood

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Electrical Science and Engineering

and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 10, 2002

Author ____
Department of Electrical Engineering and Computer Science
May 10, 2002

Certified by ____
Neil Gershenfeld
Thesis Supervisor

Accepted by ____
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Accessible Broadband Network Analysis

by

Esa Han Hsien Masood

Submitted to the
Department of Electrical Engineering and Computer Science

May 10, 2002

In Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Electrical Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

Abstract

Radio frequency network analysis is a powerful and versatile analysis tool. Due to the extremely high cost of vector network analyzers and their lack of portability, their use has been limited exclusively to advanced engineering and laboratory applications. Deploying such forms of instrumentation on a large scale would require the cost of such instruments to be significantly reduced, while retaining a useful subset of their current functionality. We have developed an instrument that has a parts cost of $125, and is capable of carrying out basic reflection measurements across a large frequency range. In addition, we have applied this instrument to several test applications: Chipless RFID tag reading, milk quality analysis and postal mail content analysis, and have demonstrated the usefulness of this instrument for these applications. Aside from these applications, a portable and low-cost network analyzer provides an accessible platform for new and interesting applications of network analysis to be developed.

Thesis Supervisor: Neil Gershenfeld
Title: Associate Professor of Media Arts and Sciences

Acknowledgements

In the name of God, most Compassionate, most Merciful.

I would like to express my sincere gratitude to the following people who have contributed tremendously to this thesis. My thesis advisor, Neil Gershenfeld for his direction and support for this project. Utmost thanks as well to everyone in the Physics and Media Group as well as alumni, for all the ideas, assistance and company in lab: Ben, Ben, Dan, Matt, Yael, Jason, Rich, Rehmi, Caroline, Susan, John, Femi and Ravi. Special thanks to Rich Fletcher, who recruited me as a UROP and made me part of the Physics and Media Group; Matt Reynolds for his ever willingness to help and his invaluable advice, especially when I first started off as a clueless UROP; and Yael Maguire for his brilliant Master's thesis which has served as a template for the writing up of this one.

In addition, I would like to thank my brothers, sisters as well as friends who have made the past few years of my education such a rewarding and memorable experience. I sincerely appreciate the moral and emotional support, especially in times when I'm faced with the seemingly impossible.

Finally, and most importantly, I would like to thank my parents for their years of untiring love, support and encouragement. They have given me so much, and to them I will always be indebted.

Table of Contents

1. Introduction

Network analyzers are used extensively in laboratories to characterize both linear and non-linear behavior of electrical networks. As they provide both magnitude and phase measurements over a frequency range, they allow a complete characterization of the behavior of these networks. In addition to characterizing electrical networks, the ability of network analyzers to provide complex impedance measurements allows the dielectric properties of materials to be inferred, and is increasingly becoming a useful instrument for materials characterization.

While the network analyzer is an extremely useful analysis tool, its use has been limited to the laboratory due to several factors. The most significant of these is the cost of such equipment: A network analyzer can cost within the range of $10,000 to $100,000, making it feasible only for industrial use or in research laboratories. In addition, network analyzers are bulky and are not suitable as portable analysis tools.

Furthermore, there is still much that is unknown about the full capabilities of network analysis. While other non-invasive probing techniques such as nuclear magnetic resonance (NMR), have been widely investigated and used in various medical imaging systems, frequency dependant complex impedance analysis remains a largely untapped resource in terms of non-invasive analysis of materials. We believe that the reason for this is two fold: Firstly, there is a lack of knowledge of how network analyzers can be applied in the real world to solve common problems. Secondly, the lack of portable and economically feasible network analyzers, that are able to perform dielectric analysis of materials, impedes efforts to find more widespread applications for network analyzers.

Thus, the aim of this thesis is to develop an instrument that possesses certain useful functions of network analyzers at a much lower cost than conventional models, and to demonstrate the usefulness of such a system on several applications. It is hoped that this investigation would illustrate new and useful applications of network analysis, adding to the existing knowledgebase of known practical applications. In addition, we also hope to provide a feasible hardware platform, which would enable the further investigation of new uses of broadband network analysis outside the laboratory.

There are currently several motivations for such an investigation. Firstly, considerable work has been done in our group in developing low-cost radio-frequency identification (RFID) tags as well as sensor tags, which are able to detect changes in the external environment. These tags essentially have resonant planar structures, and to date, network analyzers have been employed in reading these tags. Clearly, this is not a feasible method of tag reading, and thus more feasible broadband instrumentation is required in order for this RFID technology to be widely deployed.

A second motivation for this work would be the dielectric characterization of materials. In particular, through the collaboration of our group with Media Lab Asia (MLA), we have come to know that there is a demand for a feasible, non-invasive system that is able to provide a real-time analysis of milk quality. This is currently a problem in rural India, where farmers are paid based on the quality of their yield. One approach to this problem would be to analyze the complex dielectric permittivity of the milk samples, and use that to determine useful physical properties of the milk being analyzed. Thus, we are interested in a system that can carry out frequency dependent complex analysis of materials, and distinguish between milk samples of varying grades. As a test case, we will use this system to analyze various milk samples to evaluate its effectiveness.

A second test case, in which we are currently interested, is related to providing security in postal mail. The United States Postal Service processes large amounts of mail each day, and there are security concerns regarding the content of the mail being processed. Currently, scanning of postal mail is being considered, in order to weed out any hazardous material that may be sent via the postal service. Present methods of scanning, such as metal detectors, can be used for such a large volume of mail. However, these methods do not provide sufficient information about mail content as a considerable amount of hazardous material are non-metallic in nature (e.g. plastic explosives). X-ray scanning and other forms of non-invasive imaging techniques, on the other hand are economically not feasible as they are labor intensive to operate and the cost of the requisite equipment is high. Thus, we propose employing frequency-dependent complex impedance analysis for this, and we will investigate the usefulness of the instrument on such an application.

This thesis will be structured as follows: Chapter 2 explains some of the physical mechanisms involved that determine the dielectric constant of a material, in order to provide some background with regard to materials. Chapter 3 introduces the concept of network analysis, and describes the basic concepts involved as well as gives an overview of how commercial network analyzers are implemented. In Chapter 4, the design for the low-cost network analyzer board is presented. Also discussed are the way the different subsystems within the design work, and the design choices that were made. Chapter 5 describes the tag reading application undertaken with this instrument, as well as provides results and analysis of these results. Chapter 6 provides a similar treatment for the materials analysis application. Chapter 7 then sums up this thesis.

2. Dielectric Constant Theory[1]

Broadly speaking, materials can be grouped into several categories depending on their electrical properties. Materials in which the transport of electrons occur with ease, are termed conductors, while materials which allow the movement of electrons through their structures to a certain extent are termed semiconductors. Insulators (or dielectric materials) on the other hand do not permit the transport of electrons, and thus have low direct current (DC) conductivities.

2.1 Macroscopic View

Even though dielectrics do not have free electrons or mobile ions at the molecular level, interesting properties can nevertheless be observed in the presence of an electric field. The presence of large amounts of electric charge in a dielectric results in positive charge being pushed in one direction, and negative charge in the opposite direction, when an electric field is applied. The dielectric thus polarizes in the presence of an electric field, producing a net electric dipole moment per unit volume termed the polarization P. Polarization is a vector quantity, with units of Cm^{-2} and is conventionally defined as pointing from negative charge to positive charge.

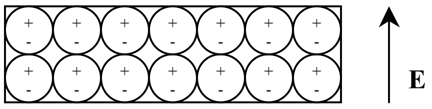


Figure 1. Polarization of a Dielectric in the Presence of an Electric Field

This transient charge motion when an electric field is applied, results in stored electrical energy within the dielectric material. The ability of dielectrics to store electrical energy is made use of in capacitors. Capacitors are used in numerous DC applications, such as in the flash unit of a camera, where charge storage and rapid release of the charge is necessary.

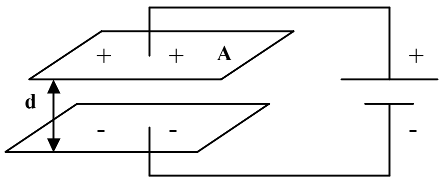


Figure 2. Parallel Plate Capacitor with Applied Voltage

[1] While a brief treatment to this subject is given here, more comprehensive coverage can be found in most materials science textbooks on dielectric materials. The references used for this chapter can be found in the bibliography.

Consider the case of a capacitor with two parallel conducting plates of area A separated by a distance d. (Figure 2) Assume that nothing initially exists between the plates. Under these conditions, applying a voltage V across the capacitor will produce a charge Q on each plate given by CV, where C is the capacitance in farads, and C is given by:

$$C = \frac{\varepsilon_o A}{d} \tag{2-1}$$

If a dielectric is now inserted between the plates, that completely fills the space, the electric field polarizes the dielectric and the resulting surface charge density P on the dielectric produces an internal electric field that opposes the applied field (Figure 2). This charge density also partially compensates the charge on the conducting plates. Maintaining a constant voltage across the plates results in more charge flowing onto the plates, and thus for the same voltage, there will be a greater charge Q′ on each plate. Correspondingly, there will be an increase in capacitance and stored energy on the plates as well. The new capacitance, C′ is given by:

$$C' = \frac{Q'}{V} = \frac{\varepsilon_o \varepsilon_r A}{d} = \frac{\varepsilon A}{d} \tag{2-2}$$

Where the increase in capacitance and charge is:

$$\frac{C'}{C} = \frac{Q'}{Q} = \varepsilon_r \tag{2-3}$$

which is the relative permittivity, or dielectric constant of the material. The dielectric constant of the material is unitless, whereas the permittivity of vacuum ε_o, and the permittivity of the dielectric, $\varepsilon = \varepsilon_o\varepsilon_r$, are expressed in units of farads per meter. Dielectric constant can be related directly to the polarization P, since the surface charge on the dielectric equals the increase in charge on the plates:

$$PA = Q' - Q = C'V - CV = CV(\varepsilon_r - 1) \tag{2-4}$$

Since $C = \varepsilon_o A / d$ and V / d is the electric field E between the plates of the capacitor,

$$P = \varepsilon_o E(\varepsilon_r - 1) \text{ and} \tag{2-5}$$

$$\varepsilon_r = 1 + \frac{P}{\varepsilon_o E} = 1 + \chi \tag{2-6}$$

where χ is the dielectric susceptibility.

2.2 Microscopic View

The effect of an electric field on a dielectric can also be explained in terms of atomic behavior, by considering the microscopic effects of an applied electric field. In dielectrics, five major microscopic mechanisms operate when polarization occurs.

Electronic Polarization.

Atoms within the dielectric consist of positively charged nuclei surrounded by electron clouds. In the absence of an electric field, the statistical centers of positive and negative charge coincide. When an electric field is applied, there is a shift in the charge centers, particularly of the electrons. For a separation of δ and a total charge of q, the atom has an induced dipole moment

$$p = q\delta \tag{2-7}$$

If there are N atoms per m^3, the total electric dipole moment per unit volume of the material, its polarization, P, is given by:

$$P = Np = Nqd \tag{2-8}$$

For moderate applied electric fields, the electric dipole moment of each atom is proportional to the field:

$$p = \alpha_e E \tag{2-9}$$

where the proportionality constant α_e is called the *electronic polarizability* (we assume here that the electric field felt by the atom is the same as the overall electric field applied to the dielectric). In purely covalent materials, such as diamond, this is usually the only source of polarization. From (2-6), (2-8) and (2-9), we get:

$$\varepsilon_r = \mathbf{1} + \frac{N\alpha_e}{\varepsilon_o} = 1 + \chi \tag{2-10}$$

This equation relates a macroscopic material property, the dielectric constant, to a microscopic property, the electronic polarizability. In general, larger atoms are more polarizable, since they have a larger electron cloud around their nucleus, which gets more easily distorted by an electric field.

Ionic Polarization

In ionic materials such as Sodium Chloride (NaCl), in addition to the distortion of the electron clouds of each electron, an electric field pushes negatively charged chlorine ions in one direction, and positively charged sodium ions in the opposite direction. This results in some Na-Cl bonds being stretched, and others being compressed. As a consequence of this, there is another contribution to the polarizability of the dielectric, α_i, in equation (2-10). However, this contribution to polarization and dielectric constant cannot respond as quickly to rapidly changing fields as electronic polarization, since it involves the displacement of entire ions. This leads to a pronounced frequency dependence of the dielectric constant, as will be discussed later.

Orientational (Dipolar) Polarization

Some molecules, like H_2O, have permanent dipole moments, and in the absence of a permanent electric field, they have no preferred direction of orientation. Applying an electric field causes these molecules to have a preferred orientation, and results in a further contribution to the polarization. In polar liquids such as water, this contribution is extremely significant at lower frequencies.

Interfacial (Space Charge) Polarization

In some dielectrics, electrons or ions can move more than interatomic distances, producing the build-up of layers of charge at internal interfaces such as grain boundaries or interphase boundaries. Such internal charge layers can make large contributions to the dielectric at low frequencies.

Ferroelectric Polarization

A few ionic crystals, called ferroelectrics, have symmetries that allow them to have a spontaneous ionic polarization in the absence of an electric field. In limited temperature ranges, some ferroelectrics, such as barium titanate and lead zirconium titanate (PZT), can have dielectric constants of several thousands.

2.3 Summary

The various microscopic mechanisms that have been described operate depending on the frequency at which the material is probed. Furthermore, different materials have different mechanisms that predominate depending on the physical nature of the material. This property allows us to infer information about the physical characteristics of a material by probing them over a wide range of frequencies. We can make use of this principle to perform electromagnetic characterization of materials, allowing us to perform non-destructive analysis of materials and infer information about the physical nature of the material. The topic of frequency dependence of the dielectric constant of materials will be elaborated on in chapter 6, where we discuss our investigation of the electromagnetic characterization of materials.

3. Network analyzer theory[2]

3.1 Characterizing Magnitude and Phase

Before approaching the subject of network analyzers, let us briefly address the more general topic of characterizing devices. That is, the kind of measurements we are interested in, the types of devices we can characterize and the instruments we can use for this.

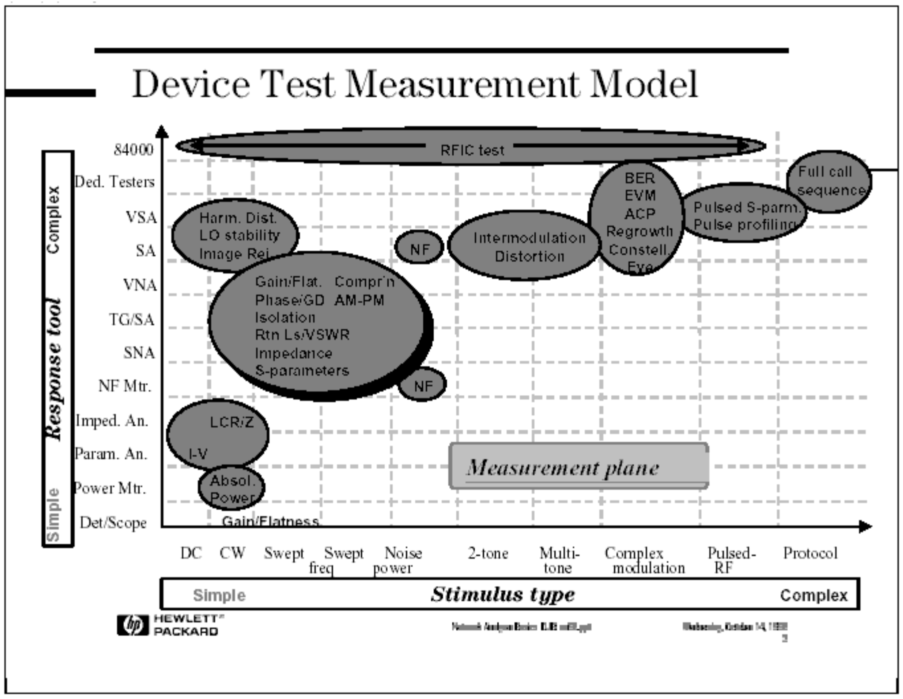
Typically, characterizing a device involves measuring its linear and non-linear behavior. Since the number of devices is huge, and there are many attributes of each device that we might be interested in, a large range of test equipment is required in order for all these attributes to be characterized. Figure 3 illustrates a model covering the wide range of measurements necessary for the complete linear and non-linear characterization of devices. Some instruments are optimized for only one test (e.g. Bit error rate), while others, such as network analyzers, are more general in nature. Network analyzers can measure both the linear and non-linear behavior of devices, although the measurement techniques are different.

Electrical devices that can be characterized using network analyzers include both passive and active devices. Examples of passive devices are resistors, capacitors and inductors, splitters, directional couplers and attenuators. Active devices would include transistors, oscillators, radio-frequency integrated circuits (RFICs) and modulators. Testing such devices is necessary, since they are often used as building blocks in larger systems, and we need to verify their specifications and performance before use.

In certain cases, magnitude-only measurements are sufficient. For example, the simple gain of an amplifier or stop-band rejection of a filter may be all the data we need. However, there are certainly other situations where phase information is critical. Thus, complete characterization of devices and networks involves measurement of phase as well as magnitude. Examples of this are: 1) The complex impedance of a device must be known for proper matching, 2) Models for circuit simulation require complex data, and 3) Time domain characterization requires magnitude and phase information to perform the inverse Fourier Transform.

Thus, we see that network analyzers are designed to be able to make complex impedance measurements of devices. Using appropriate probes, the complex impedance of materials can also be determined as will be elaborated on in Chapter 6.

[2] This chapter presents a summary of the theory of network analyzers. More detailed coverage can be found in the references used for this chapter, as given in the bibliography.



Here is a key to many of the abbreviations used above:

Response

84000	HP 84000 high-volume RFIC tester
Ded. Testers	Dedicated (usually one-box) testers
VSA	Vector signal analyzer
SA	Spectrum analyzer
VNA	Vector network analyzer
TG/SA	Tracking generator/spectrum analyzer
SNA	Scalar network analyzer
NF Mtr.	Noise-figure meter
Imped. An.	Impedance analyzer (LCR meter)
Power Mtr.	Power meter
Det./Scope	Diode detector/oscilloscope

Measurement

ACP	Adjacent channel power
AM-PM	AM to PM conversion
BER	Bit-error rate
Compr'n	Gain compression
Constell.	Constellation diagram
EVM	Error-vector magnitude
Eye	Eye diagram
GD	Group delay
Harm. Dist.	Harmonic distortion
NF	Noise figure
Regrowth	Spectral regrowth
Rtn Ls	Return loss
VSWR	Voltage standing wave ratio

Figure 3. Model for Device Characterization[3]

3.2 Transmission Line Basics

The most fundamental concept of high-frequency networks analysis involves incident, reflected and transmitted waves traveling along transmission lines. At low frequencies, where the wavelengths of signals on the wire are much larger than the length of the conductor, a simple wire is very useful in carrying

[3] Figure from "Network Analyzer Basics", by David Ballo, Hewlett-Packard Co, 1998.

power. Current travels down the wire easily, and voltage and current are the same everywhere along the wire.

At high frequencies however, the wavelengths of signals are comparable to or much smaller than the length of the conductor, and the measured envelope voltage depends on the position on the line. Under these conditions, there is a need to use transmission lines in order to enable the efficient transfer of RF power. A transmission line takes on a certain characteristic impedance (Z_o), and this is determined by the geometry of the line. For example, in low power applications such as cable TV, coaxial transmission lines are designed to have a characteristic impedance of 75 ohms for low loss. For RF and microwave applications, where the high power might be encountered, coaxial transmission lines are designed to have a characteristic impedance of 50 ohms, a compromise between maximum power handling (30 ohms) and low loss. In order to achieve maximum power transfer and low reflection, loads and sources have to be accurately matched to the characteristic impedance of the line.

Maximum Power Transfer

Consider a source with a source impedence R_s, connected to a load, R_l. We can show that, whether the source is a DC voltage or a sinusoid, maximum power transfer into the load occurs when $R_l = R_s$. Similarly for a transmission line, for the maximum transfer of energy into a transmission line from a source or from a transmission line to a load, the impedance of the source and load should match the characteristic impedance of the transmission line. Thus, in general, Z_o is the target input and output impedances when designing devices and networks. When the source impedance is not purely resistive, maximum power transfer occurs when the load impedance is equal to the complex conjugate of the source impedance.

Terminating Transmission Lines

Depending on the impedance of the load terminating the transmission line, we observe various effects along the transmission line depending on the magnitude of the reflected wave. A transmission line terminated by an impedance equal to Z_o results in maximum transfer of power to the load, and no reflected signal occurs (i.e. magnitude of the reflected wave is zero). This is the same as if the transmission line were infinitely long. In this case, the envelope of the RF signal along the length of the line would be constant (no standing wave pattern) since energy flow is in one direction only.

In the case where the transmission line is terminated by a short circuit, the absence of any resistive component in the load implies that no dissipation of power occurs. Since there is nowhere else for the energy to go, a reflected wave is produced which travels back down the line towards the source (i.e. in a direction opposite to the incident wave). For a short circuit, the boundary condition at the end of the line requires the voltage there to be zero, and for this to occur, the reflected wave must be equal in magnitude to the incident wave and 180^o out of phase with it.

In the case where the termination is by an open circuit, Ohm's law implies that the open circuit can support no current. Therefore, the reflected current wave must be 180^o out of phase with respect to the incident wave (the reflected voltage wave will be in phase with the incident wave). This guarantees that the current at the open will be zero, and as in the previous case, the reflected voltage waves will be identical in magnitude but traveling in the opposite direction. For both these the short and open circuit cases, a standing-wave pattern will be set up on the transmission line. The valleys of this standing-wave pattern will be zero and the peaks will be at twice the incident voltage level in both cases, with the positions of the peaks and valleys of the short and open being shifted along the line relative to each other.

In the case where the terminating impedance is a resistor that does not match the characteristic impedance of the line, some of the incident energy will be absorbed on the load, while the rest will be reflected down the transmission line. We find that the reflected voltage wave will have an amplitude 1/3 of that of the incident wave, and the two waves will be 180^o out of phase with the load. The phase relationship between the incident and reflected wave will change as a function of distance along the transmission line from the load, and the valleys of the standing wave pattern will no longer be zero and the peak will be less than that of the short/open case. The dependence of the standing wave pattern on the distance from the load, along the transmission line can be accounted for due to the variation of the effective impedance experienced by the traveling waves at different points on the transmission line. We can express the input impedance looking down the transmission line from the source as:

$$Z_{in} = Z_o \left(\frac{Z_l + jZ_o \tan\frac{2\pi l}{\lambda}}{Z_o + jZ_l \tan\frac{2\pi l}{\lambda}} \right) \qquad (3\text{-}1)$$

where λ is the wavelength of the traveling wave.

Reflection Parameters

We can define several parameters for a particular transmission line and termination, based on the amplitudes of the incident and reflected waves. The reflection coefficient, Γ, is defined as:

$$\Gamma = \frac{V_{reflected}}{V_{incident}} = \frac{Z_{load} - Z_o}{Z_{load} + Z_o} = \rho\angle\Phi \qquad (3\text{-}2)$$

and is the ratio of the reflected signal voltage to the incident signal voltage. The magnitude portion of Γ is termed ρ, which has a range of values from 0 to 1. It is often convenient to show reflection on a logarithmic display, and this is often expressed as return loss:

$$\text{Return Loss} = -20\log(\rho) \qquad (3\text{-}3)$$

Return loss is expressed in units of dB and is a scalar quantity. The definition of return loss includes a negative sign so that the return loss value is always a positive number. Return loss can be thought of as the number of dB that the reflected signal is below the incident signal, and this varies from infinity (for Z_o impedance) to 0 (for open or short circuit).

As we have already seen, two waves traveling in opposite directions on the same media cause a “standing wave”. This condition can be measured in terms of the voltage standing wave ratio (VSWR) and is defined as the maximum value of the RF envelope (peaks) over the minimum value of the envelope (valleys), and can also be expressed in terms of ρ :

$$\text{VSWR} = \frac{E_{\max}}{E_{\min}} = \frac{1+\rho}{1-\rho} \qquad \textbf{(3-4)}$$

VSWR can take on values between 1 and infinity.

The Smith Chart

By measuring the incident and reflected waves traveling down a transmission line, we see that a network analyzer can give us the complex reflection coefficient. However, we are typically interested in the impedance of the load terminating the transmission line (i.e. the device under test). It would be possible for us to calculate the complex impedance corresponding to each reflection coefficient value using (3-2), but this would give us a table of numbers which might be difficult to use. A simple graphical method to carry out this transformation was developed by Philip H. Smith in 1939. By mapping the impedance plane on to the polar plane, he devised the Smith Chart, which has become a ubiquitous in RF engineering design.

In this chart, the circles represent lines of constant resistance while arcs represent lines of constant reactance, with Z_o corresponding to the center of the chart. To use the Smith chart, we use the value of ρ from the network analyzer to draw a circle on the chart, and the angle of Γ to plot the appropriate radius on the chart. The complex impedance of the load would then correspond to the point of intersection of the circle and radius. In general, Smith Charts are normalized to Z_o, by dividing the impedance values by Z_o. This allows the chart to be used independent of the characteristic impedance of the transmission line, if normalized values are used.

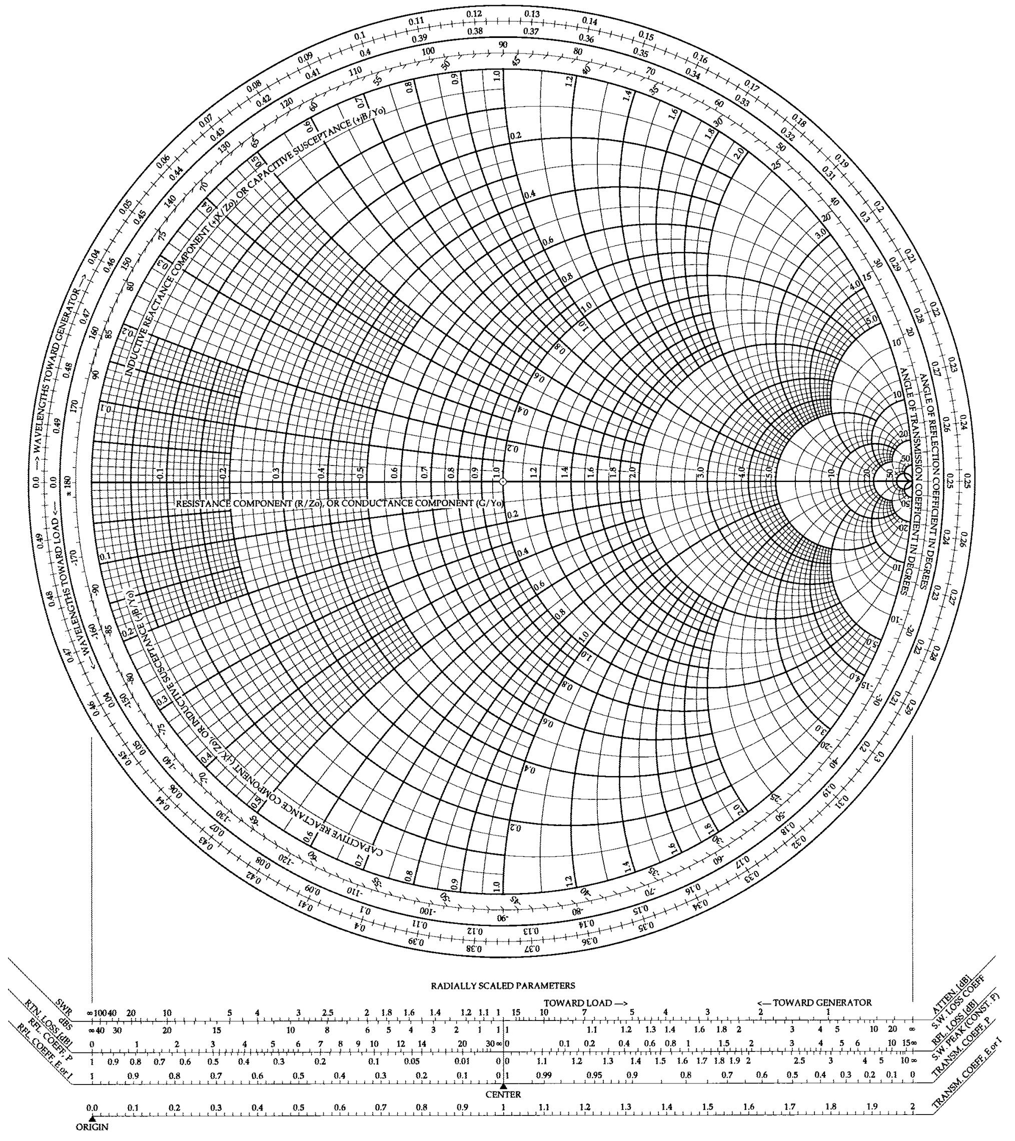


Figure 4. Smith Chart

3.3 Characterizing Unknown Devices

Network analysis is concerned with the accurate measurement of the ratio of the reflected signal to the incident signal and/or the transmitted signal to the incident signal for a particular signal transmitted into a system. More specifically, network analyzers characterize systems by measuring their scattering parameters (S-parameters). S-parameters are widely used for characterizing high-frequency networks and can be related to familiar measurements such as gain, loss and reflection coefficient.

A particular two-port device or network has four S-parameters (in general, an N-port device or network has N^2 parameters). These are determined by measuring the magnitude and phase of the transmitted and reflected signals for signals incident on each of the two ports, and calculating the appropriate ratios. For a particular device or network with ports 1 and 2, with incident signals a_1 and a_2 respectively, two output signals can be obtained, which we term b_1 and b_2. This is summarized in Figure 5.

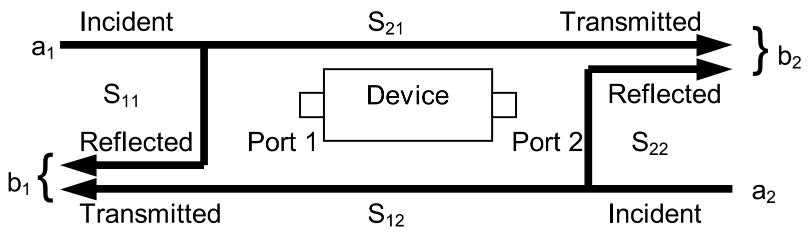


Figure 5. Diagram summarizing inputs into a 2-port device and corresponding outputs. The relation of S-parameters to the various input and output signals are also illustrated.

S_{11} and S_{21} can be determined by terminating Port 2 with a matched load, and transmitting only signal a_1. Where we have:

$$S_{11} = \frac{Reflected}{Incident} = \left.\frac{b_1}{a_1}\right|_{a_2 = 0} \tag{3-5}$$

and

$$S_{21} = \frac{Transmitted}{Incident} = \left.\frac{b_2}{a_1}\right|_{a_2 = 0} \tag{3-6}$$

S_{12} and S_{22} can be similarly determined. To summarize, the various S-parameters can be expressed as follows (with common measurement terms in brackets):

S_{11} = forward reflection coefficient (input match)
S_{22} = reverse reflection coefficient (output match)
S_{21} = forward transmission coefficient (gain or loss)
S_{12} = reverse transmission coefficient (isolation)

In terms of matrix notation, the S-parameters and the incident and reflected signals at the two ports can be expressed as:

$$\begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \tag{3-7}$$

S-parameters have several advantages over other parameters used to characterize networks (such as H, Y or Z parameters, not covered here). Firstly, they do not require the connection of undesirable loads to the device under test. Secondly, the measured S-parameters of multiple devices can be cascaded to predict overall system performance. Thirdly, H, Y and Z parameters can be derived from S-parameters if desired. For RF design in particular, S-parameters are particularly important, as they can be easily imported and used with electronic simulation tools.

3.4 Generalized Network Analyzer Implementation

In general, network analyzers consist of: 1) a source for stimulus, 2) signal separation devices, 3) a receiver that provides detection and 4) a processor/display for calculating and reviewing the results. These four sections are required in order to measure the incident, reflected and transmitted signals. Figure 5 summarizes this in the form of a block diagram.

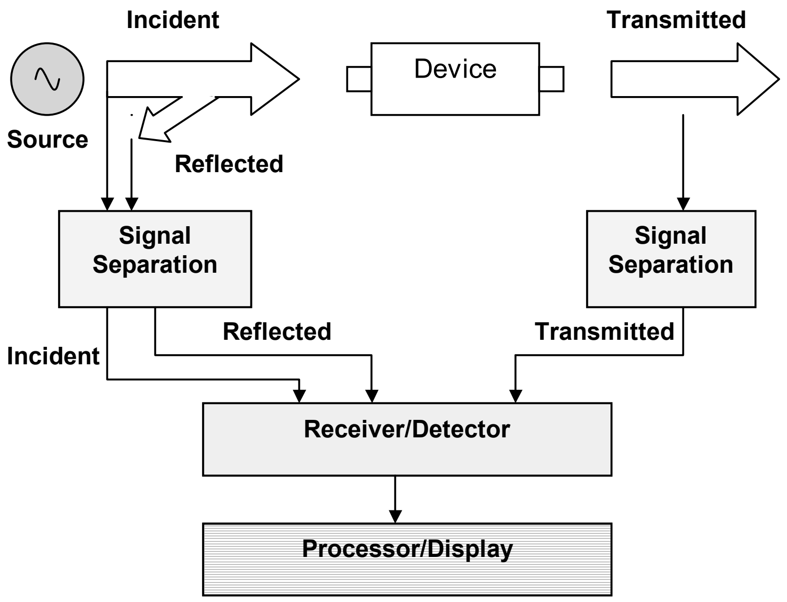


Figure 6. Generalized Block Diagram of a Network Analyzer.

Signal Source

The source provides the stimulus signal for the purposes of testing, and by sweeping the source frequency the response of the device under test at various frequencies can be determined. Typically, these sources are based on open-loop voltage control oscillators (VCOs), which are cheaper, or more synthesized sweepers, which provide higher performance especially for measuring narrowband components over small frequency spans.

Signal Separation

There are two functions of the signal separation devices. The first is to provide a reference signal that can be used to compare the measured signals. This can be done with splitters or directional couplers. Splitters are usually resistive, are non-directional devices and can be very broadband. The tradeoff is that they usually have 6 dB or more of loss in each arm. Directional couplers have low insertion loss and good isolation and directivity.

The second function of these devices is to, separate the incident (forward) and (reflected) reverse traveling waves at the input of the test device. This allows us to interpret the relevant parameters we wish to determine by comparing the incident and forward traveling waves. For this case, couplers are ideal as they are directional, have low loss and high reverse isolation. However, due to the difficulty of making broadband directional couplers, another method is often employed, which is to use a bridge.

Bridges are useful for measuring reflection as they can be made to operate over a broad range of frequencies. However, they are lossy and dissipate more of the transmitted signal compared to using directional couplers. Figure 7 shows a diagram of a resistive bridge.

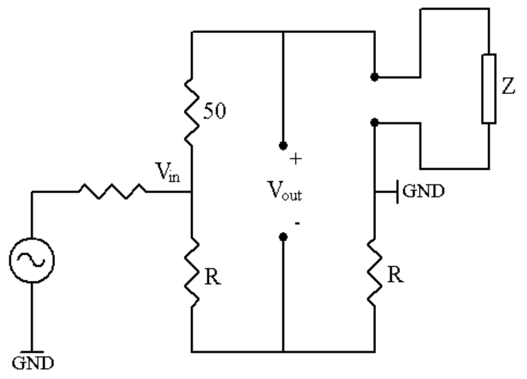


Figure 7. Diagram of a Resistive Bridge

From the diagram, we can obtain the following expression for V_{out},

$$V_{out} = \frac{V_{in} Z}{Z+50} - \frac{V_{in}}{2} \tag{3-8}$$

Solving for V_{out} / V_{in}, we find that it is proportional to the reflection coefficient,

$$2\frac{V_{out}}{V_{in}} = \frac{Z-50}{Z+50} = \Gamma \tag{3-9}$$

Thus, if we can obtain both the phase difference between V_{out} and V_{in} as well as their amplitudes, the complex reflection coefficient can be determined. This in turn allows us to tell the complex impedance of the load using (3-2). However, the resistive bridge is often used as a scalar device to measure only the magnitude of the reflection coefficient. For this, diode detectors are used to obtain the amplitudes of V_{in} and V_{out} . In network analyzers, the diode detector is built into the resistive network and the entire bridge structure can be very small, so as to minimize the parasitic reactances. These parasitics serve to unbalance the bridge, and have to be reduced to ensure accuracy as well as a flat measurement baseline.

Receiver

Signal detection in network analyzers is carried out in one of two ways: The first of these is to use a diode detector, which converts an RF signal level to a proportional DC level. In the case that the signal is AC modulated, the diode strips the RF carrier from the modulation. Diode detection is inherently scalar, so only magnitude information is preserved and phase information is lost.

In order to preserve phase information, modern network analyzers uses a tuned receiver, where the RF signal is mixed down to a lower "intermediate" frequency (IF) signal using a local oscillator (LO). The IF signal is then converted to a digital signal using an analog-to-digital converter and processed using digital signal processing (DSP) techniques. This allows both phase and magnitude information to be extracted from the IF signal.

3.5 Summary

To summarize, a network analyzer is a useful tool for determining the S-parameters associated with a particular object under test. Determining this requires knowing how a transmitted signal is affected by the test object, in terms of phase and magnitude, and the various components of the network analyzer that was described, allow us to determine this. The next chapter presents a system that has been implemented, which contains a useful portion of the functionality of commercial network analyzers. This instrument is capable of making reflection measurements, giving us some information about the S_{11} parameter of a particular object under test. This in turn will allow us to infer other properties of the object, depending on the probe that is used and the application we are interested in, as will be explained in the subsequent chapters.

4. System Design and Implementation

This chapter describes the final design of the low-cost network analyzer that was implemented, as well as explains some of the design choices made. Where appropriate, less effective methods of prior designs that were attempted will be mentioned as well. We will first elaborate on the design objectives for our low-cost network analyzer, and then describe how the system works as a whole, before proceeding to explain each subsystem in detail. To conclude this chapter, the system that was implemented will be evaluated and the results from various tests will be presented.

4.1 Design Objectives

In terms of implementation, the aim of this work was to develop a low-cost RF network analyzer that can make simple 1-port reflection measurements, providing spectral (phase and magnitude) information over a sufficiently broad range of frequencies. For this investigation, we are not concerned with precise reflection coefficient measurements or determining exact S_{11} values. In designing the instrument, several factors had to be considered. The first is the sweep range: this had to be sufficiently wide so that sufficient information could be gathered from the object under test. The second is sensitivity: The signal used to probe had to be of sufficient strength to allow small variations between different test objects to be detected. The third is cost: the parts cost had to be kept reasonably low (under $200 dollars). The fourth is related to the general ease of use of the system: The instrument had to be able to make quick measurements (i.e. be responsive to the user), output data in a usable form (such as to a PC where it could be displayed or stored) and have an appropriate software interface to support the hardware. In addition, having a user-friendly control interface would help make the instrument more accessible to a wider range of users who are unfamiliar with using laboratory measurement instruments.

4.2 Systems Level Overview

Figure 8. shows the block diagram of the system, with the various stages involved. The operation of the entire instrument is controlled by a PIC microcontroller, which also allows the instrument to communicate with a PC via an RS-232 interface. The signal generation stage of the instrument was implemented using a digital direct synthesis (DDS) chip, which is in turn controlled by the microcontroller. The DDS produces two signals: the in-phase signal (I) and a quadrature signal (Q) which is 90^{o} out of phase with I. These two signals are then low-pass filtered, amplified and transmitted onto an appropriate probe. Of these two signals, one acts as the active signal and is transmitted to a probe that is exposed to the tag or material being tested. The second signal, transmitted to a dummy probe, acts as a reference signal allowing us to detect changes in phase and magnitude of the active signal. A standing wave pattern is set up, on both the active and reference signal paths, with the amplitude and phase of the backward traveling wave dependant on the reflection coefficient of the load terminating that particular transmission line. Signal separation is achieved using a pair of directional-couplers to couple the backward traveling waves from the active and reference probes. These the two backward traveling waves are then compared by the receiver, which was implemented using a phase/magnitude detector chip. Comparing these two signals allows changes in the reflection coefficient to be detected, giving us information about changes in the physical environment at the probe. This chip converts differences in magnitude and phase of the two signals into a DC value, which is in turn read by the analog-to-digital converter (ADC) of the microcontroller, allowing the changes in phase and magnitude of the active signal at various points on the frequency spectrum to be converted into a digital value and read by the microcontroller. This in turn allows us to plot the phase and magnitude response spectra of a tag or object under test.

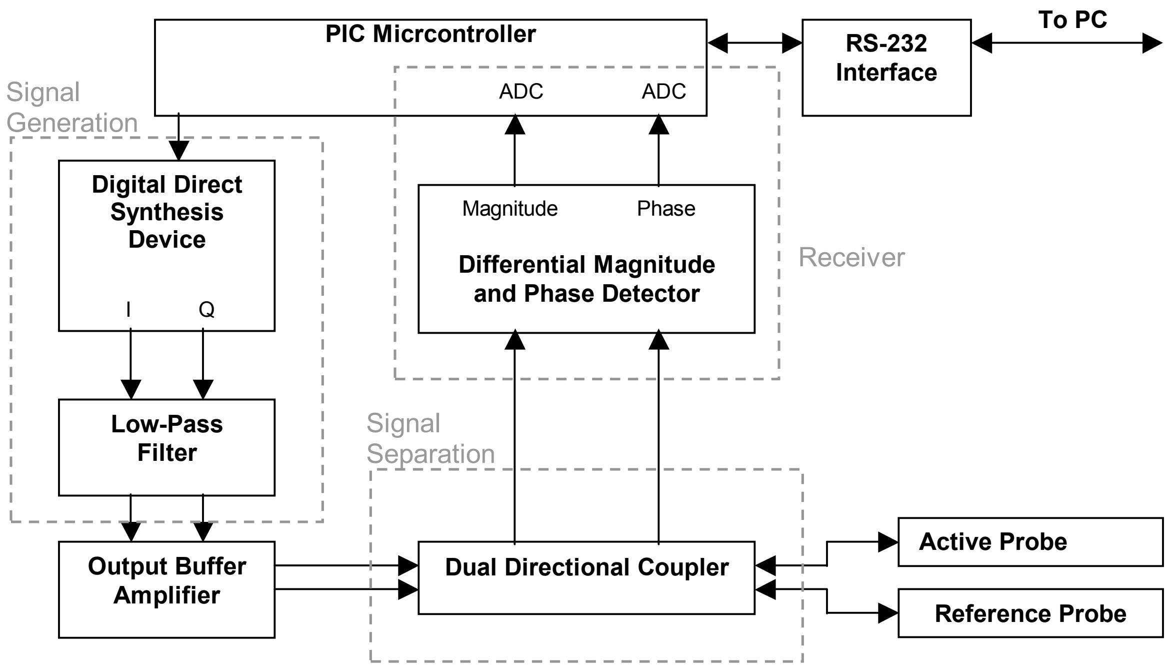


Figure 8. Block Diagram of Implemented System

The entire instrument was implemented on a single 4-layer, 0.062 inch printed circuit board, with the board size being 14.5 cm by 6.5 cm as shown in Figure 9.

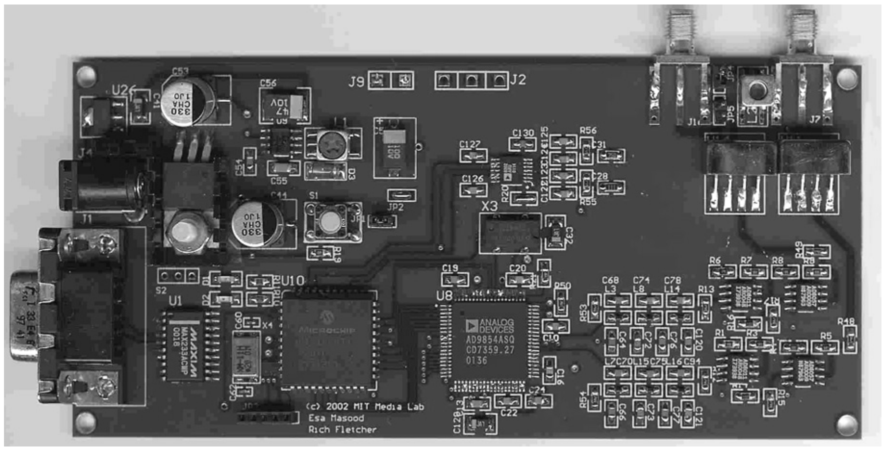


Figure 9. Picture of Instrument Hardware.

4.3 Signal Generation

Two basic methods exist for signal generation. Traditionally, this has been done using the analog Phase-Locked-Loop (PLL), but recent advances in digital direct synthesis (DDS) technology has resulted in single-chip complete-DDS products being available at a low cost. Compared to the analog-PLL, the DDS method has many advantages, and has long been recognized as a superior technology in generating highly accurate, frequency-agile (rapidly changeable frequency over a wide range) low-distortion output waveforms. In addition, using a DDS allows the output frequency and phase to be precisely and rapidly manipulated under digital control. Aside from these two methods, a hybrid method is sometimes used as well, which employs a DDS to control an analog PLL. This method allows one to generate signals in the intermediate frequency (IF) range, since the upper limit of high-end DDS chips stop at the hundreds of megahertz to gigahertz range.

For the instrument, signal generation was carried out using a single chip DDS because of the need to generate a signal with a rapidly changing frequency. In addition, using a DDS offered a solution to having a signal generation stage that is able to operate at a wide frequency range. From experience, most analog-PLL implementations, which operate in the DC-200 megahertz range, are able to produce output signals with a bandwidth in the order of tens of megahertz (for example the MC145170 used in an earlier design operated in this frequency range and had a sweep span of 40MHz), and using a PLL to generate a wide enough frequency sweep would necessitate operating the PLL at a much higher frequency (in the gigahertz range), and mixing that frequency down to the desired range. Another issue that has to be addressed when using a PLL is that the frequency of the signal being generated is not known, and most counters (used to detect the frequency of the output signal) do not operate in the tens/hundreds of megahertz range. Thus, a divider/prescalar would have to be used to convert the sweep signal to a lower frequency signal, before its frequency can be determined. All these factors would add more complexity to the design, and would have an impact on cost as well.

Of the DDS chips that are commercially available, the AD9854ASQ was finally selected. Previous versions were made using the AD9851, which has a smaller frequency range than the AD9854, but worked well within its specified frequency range. The AD9852, which produces a single output frequency signal (i.e. instead of quadrature outputs) was also used in previous designs, and the reference and active signals were obtained by splitting the signal up into active and reference signals using a power splitter. There were two disadvantages of this: Firstly, the splitter is lossy, thus attenuating of the output signal. Secondly, the reference and active signals were in phase, and this let to a clipping of the phase, due to the inability of the phase detector from distinguishing phase values of $\pm\Phi$ (this will be further explained in section 4.8 when the detector is described in more detail).

AD9854 digital-synthesizer operates at an internal clock rate up to 300 MHz, and contains two internal high-speed, high-performance quadrature 12-bit D/A converters, allowing the simultaneous generation of sine and cosine outputs. The AD9854's core provides 48-bit frequency resolution (1 microhertz tuning resolution with a 300 MHz system clock), and allows the generation of signals up to 150 MHz, which can be digitally tuned at the rate of 100 million new frequencies per second.

Other high-end DDS chips are also available, which provide a larger frequency sweep range than the AD9854, and were looked into. The top of the range DDS commercially available is the STEL-2375A, produced by ITT Industries. It operates at a clock frequency up to 1 GHz, and is capable of synthesizing waveforms between DC and 400 MHz, with a frequency resolution of 0.23 Hz. Another chip, the STEL-2373B operates at a clock frequency up to 800 MHz. It has 32-bit frequency resolution (0.186 Hz step size) and has a 0-320 MHz output frequency range. While it would have been desirable to have such a large swept frequency range, the cost of these chips greatly inhibited its use. The STEL-2373B for instance, costs $4,400 per chip (in quantities of 1-9) and thus would not be suitable for a low-cost design.

With regards to implementation, the operation of the AD9854 was controlled by a microcontroller (PIC16F876). The role of the microcontroller in controlling the DDS was firstly, to program several of the programming registers in the DDS chip. This set the clock frequency of operation, the internal clock multiplier of the DDS, output filter of the DDS, as well as powered down any unused stages in the DDS to reduce power consumption. This programming occurs when the system is powered up. The second function of the microcontroller is to control the frequency sweep of the DDS. The DDS was operated in the single-tone mode, and by specifying a particular frequency to the DDS the microcontroller causes the DDS to output a signal at that desired frequency. The frequency sweep was thus implemented by programming the PIC to specify a successively higher frequency to the DDS.

The AD9854 can be operated in either the serial or parallel programming mode. The parallel programming mode was opted for since it meant that that the DDS can be programmed at a much faster rate, allowing a faster frequency sweep. The trade-off for operating the DDS in the parallel programming mode was that it required more pins on the microcontroller, and hence a larger microcontroller. A total of

18 microcontroller pins were used to operate the DDS: 8 data input pins, 6 address input pins, 1 master reset pin, 1 serial/parallel mode select pin, 1 data write pin and 1 update clock pin.

In order to conserve power, unused stages in the DDS can be powered down. For this application, the comparator stage of the DDS was powered down by setting the appropriate bit in the programming register of the DDS. The comparator is used to convert the output sinusoidal waveform into a square wave, and thus is not needed since this application requires as pure a sinusoidal output as possible.

Another issue that had to be observed when implementing the DDS was that of heat dissipation. Due to the operation of the DDS at such a high clock frequency, it had a power dissipation of over 3 Watts, and precautions were taken in laying out the PCB and mounting the DDS chip. The 9854ASQ DDS chip is made with thermally enhanced packaging, with a heat sink at the bottom of the chip. Thus, the footprint of the chip had to have an exposed conducting surface, which was connected to the internal ground plane by vias. When mounting the chip, solder paste was applied to the conducting surface first and the PCB was heated using a hotplate, and then cooled with the chip in place to establish a bond between the heat sink of the chip and the exposed conducting surface of the PCB. This allowed for heat to be conducted away from the chip to the board, and through the ground and power planes in the board, preventing the chip from overheating during continuous operation.

The actual implementation of the DDS chip required several other external passive components. Majority of these were bypass capacitors (0.1 uF chip capacitors). In addition to these, an RC filter was also used to filter the signal of the internal PLL of the DDS. A 2 kΩ resistor was also used to specify the output current of the DDS: The DDS was operated slightly under the maximum output current rating in order to achieve maximum output signal strength without overloading the DDS. A 50 MHz crystal oscillator was also used as a reference clock for the internal clock. Together with an internal clock multiplier set at a factor of 6, the DDS was operated at an internal clock frequency of 300 MHz.

4.4 Signal Filtering

Low-pass filtering of both the output signals of the DDS is essential since several higher frequency components are produced along with the fundamental signal being generated. In addition to the desired signal of frequency f_o, being generated, the output signal of the DDS contains frequency components at f_c, the internal clock frequency of the DDS, as well as frequency components at f_c-f_o and f_c+f_o. At a clock frequency of 300 MHz, and producing an output of 125 MHz, unwanted frequency components occur at 175 MHz, 300 MHz and 425 MHz. Thus a low-pass filter with a sharp enough cut-off is necessary so as to ensure that the signal of interest is preserved, while sufficiently attenuating the higher frequency signals.

A three-stage L-C filter was used, as recommended by the datasheet for the AD9854, and this was found to sufficiently remove the higher frequency components of the DDS output signal. The magnitude

Bode plot of the filter used is shown in Figure 7, while the schematic for the output filter, along with component values, can be found in the Appendix.

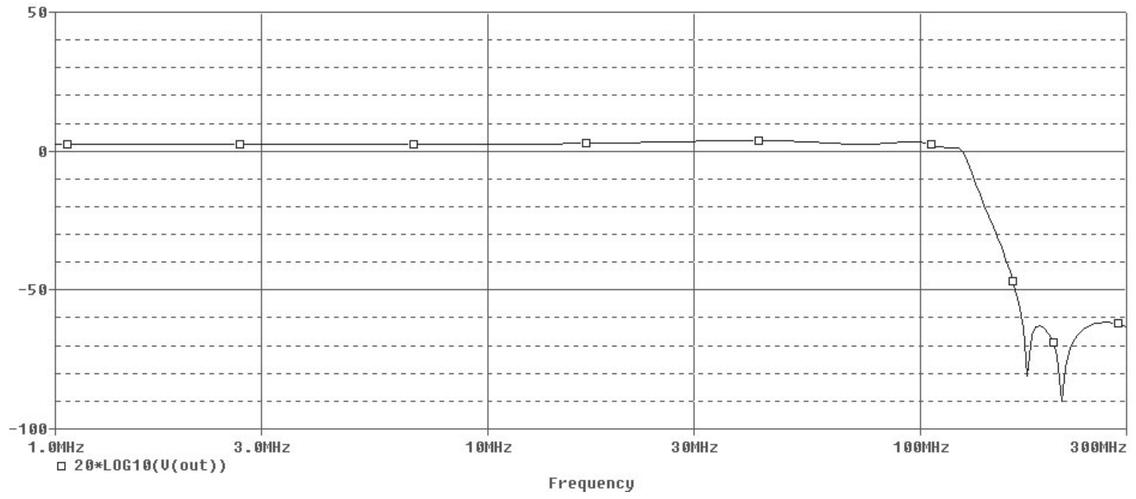


Figure 10. Bode Plot (Magnitude) for Low Pass Filter

4.5 Signal Amplification

The signal produced by the DDS varies depending on frequency, with the amplitude of the signal being in the range of 440 to 640 mV peak-to-peak. This signal is further attenuated by the low-pass filter stage, and thus has insufficient power to be used directly. In order to allow the instrument to have sufficient sensitivity, the signals in the two signal paths were amplified before being channeled into the antenna. While a larger signal would allow the instrument to have greater sensitivity, this comes at an expense of higher power consumption. In addition, for the method of amplification used, there exists a trade-off between the amount of gain possible and the operating bandwidth of the amplifier. This places a limit on the optimal amount of gain of the amplifier stage.

The biggest design challenge for our application is the need for the amplifier to be able to amplify input signals from DC up to about 150 megahertz, with relatively constant gain throughout the spectrum of frequencies. In addition, the output had to have a sufficiently large amplitude to be useful for the test applications.

Several amplifier designs were experimented with, and the final design employed a two-stage, positive-feedback operational amplifier configuration. The op-amps used were the AD8011 for the first amplification stage, and the AD8057 for the second amplification stage. Both amplifiers are high speed, wide bandwidth op-amps, and operate off a ±5 V supply voltage. A two-stage configuration was adopted, instead of a single-stage configuration, so that the gain on each individual amplification stage could be reduced to achieve the same total output gain. This allowed the amplifier stage would have a better frequency response. The AD8057 was cascaded after the AD8011, as it is a higher power amplifier compared to the AD8011. Other factors that help improve the frequency response characteristics of the amplifier stage include using a positive feedback configuration instead of a negative feedback configuration, minimizing the resistance of the feedback resistor, and using higher supply voltages for the op-amps.

Figure 11 shows the implementation of the two-stage design for a single signal path.

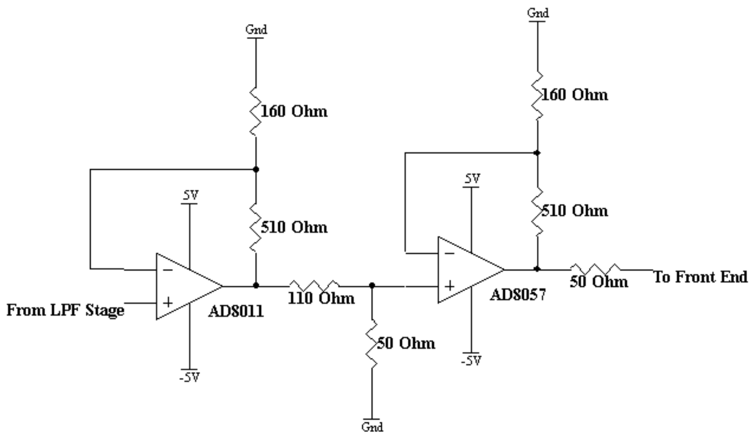


Figure 11. Schematic of Output Amplifier Stage

A previous design was attempted using a single output amplification stage using the AD8011. But this required the gain of the single stage to be very high, causing the gain of the amplification stage to fall off significantly at higher frequencies. This was thus abandoned for a two-stage output amplifier.

4.6 Front End Design

The role of the front end is to achieve signal separation between the forward and backward traveling waves on the transmission line. This would in turn allow us to determine the reflection coefficient, and from that, various properties of the load terminating the transmission line. Two methods are commonly used for signal separation, as mentioned in the previous chapter, namely using directional couplers and using a bridge. The advantage of directional couplers is that they are directional, have low loss and high reverse isolation. Compared to bridges, directional couplers are less broadband in nature, but for the frequency range the system is operating in, this does not pose as a problem. Bridges can work over a wider range of frequencies, but they exhibit greater losses and less power will be transmitted to the load under test. In addition, the parasitics associated with bridges would lead to inaccurate measurements.

Several methods of measuring reflection coefficient were investigated, and will be described briefly here. The first method would be to transmit the signal through two directional couplers arranged in series. The first directional coupler couples a portion of the forward traveling wave, and this acts as the reference signal for the detector. The second directional coupler couples a portion of the backward traveling wave. Thus by comparing the forward and backward traveling waves, the reflection coefficient at the end of the line can be determined. Figure 12 shows a diagram of such an implementation.

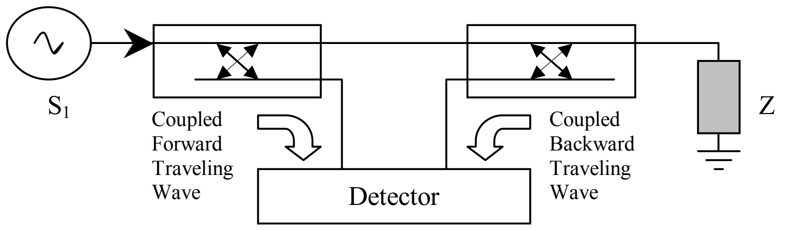

S1
Coupled Forward Traveling Wave
Detector
Coupled Backward Traveling Wave
Z

Figure 12. Series Directional Coupler Configuration for Measuring Reflection Coefficient

Such a method of measuring reflection coefficient has several advantages. Firstly, it requires only one signal to be generated. Secondly, since a directional coupler is used to obtain a reference from the forward traveling wave, most of the signal strength is channeled to the transmission line. However, the main problem that was faced with this approach was that it led to a very uneven measurement baseline due to wide fluctuations in the response of the probe (specifically the antenna for the tag reading application) to the signal. This was mainly due to the fact that a tuned antenna had to be used, which does not have a constant frequency response. Consequently, the design that was finally adopted was one that employed a differential configuration, allowing the natural response of the antenna to be subtracted, leading to a much flatter baseline.

A second method that was looked at was that of using a directional bridge. This was implemented by splitting a signal between two equivalent signal paths, measuring the phase and magnitude of the signal relative to in the two paths, and comparing that with the input signal (see discussion in Section 3.4). Introducing a load on one of the arms of the bridge, and not the other, effectively unbalances the bridge by causing the amplitude and phase of the signal in one side of the bridge to change. The main problem encountered with this method was that the bridge was also unbalanced by many other factors, such as parasitics and other factors in the environment. Consequently, this led to a very uneven measurement baseline.

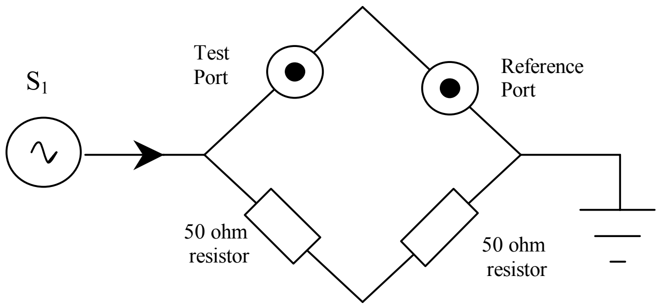


Figure 13. Diagram Showing Bridge Implementation

Since we do not require precise Γ values, an alternative method to the bridge was adopted. The front end design that was used in the final implementation is shown in Figure 14,

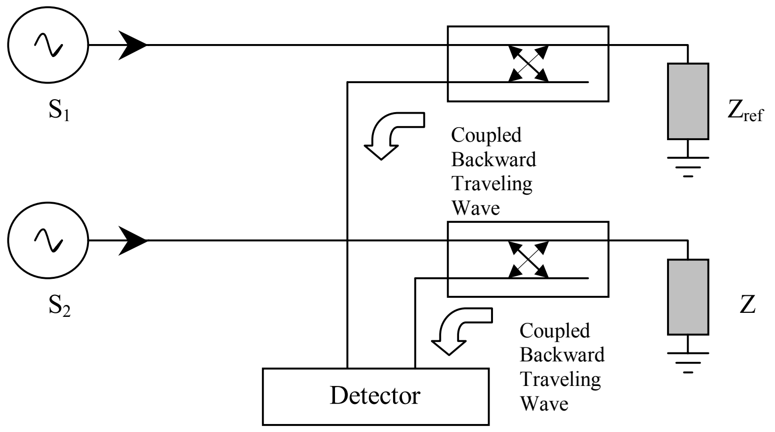

S1
S2
Coupled Backward Traveling Wave
Coupled Backward Traveling Wave
Detector
Zref
Z

Figure 14. Parallel Directional Coupler Configuration for Measuring Reflection Coefficient

This design employs a parallel configuration of two directional couplers. Two signals are transmitted through the couplers, one of which is transmitted to the active probe (used for taking measurements), while the other is transmitted to a dummy or reference probe. The directional coupler couples the backward traveling wave coming from each of the probe, and the detector uses both the coupled signals to determine magnitude/phase changes in the presence of a load. While this method does not give accurate reflection coefficient values, it allows one to measure changes in Γ. This provides us with sufficient useful information about the load terminating the transmission line, as will be illustrated in the next two chapters when applications are discussed. The dummy probe for a particular application is fixed, and its purpose is to generate a signal for the detector, which would allow the background variations in the system to be subtracted. This can arise due to many factors, such as variations in the generated signal strength, variation in the gain of the amplifier stage over the sweep frequency range, variations in reverse isolation of the directional couplers, etc. It was found that such a differential implementation was successful at removing the natural frequency dependence of the system, giving a much flatter measurement baseline. Such a baseline is extremely important since it acts as the reference when loads are measured. For the tags application in particular, we will see that peak detection is greatly enhanced with a flatter baseline as well.

For our system, the fact that the DDS is able to produce two output signals of similar magnitude effectively eliminates the need for the front end to produce a reference signal. This simplifies the front-end design considerably when the dual directional-coupler configuration is used. An alternative way of thinking of this is that since the reference probe is fixed, the frequency dependent reflection coefficient for the reference arm remains the same, thus the magnitude of the backward traveling wave from the reference probe is related to the forward traveling wave by the reflection coefficient. Thus, this method not only compensates for variations in signal strength (as was the case of the series directional-coupler configuration), it compensates for many other factors as well, increasing the quality of the measured data.

In terms of implementation, two TDC-10-1 directional couplers, manufactured by Mini-Circuits, were used. The output of the amplifier stage was connected to the ‘output’ port of the directional coupler while the ‘input’ port of the directional coupler was connected to the probe. Connecting the directional coupler as such couples the backward traveling wave on each line. The TDC-10-1 has a 1-400 MHz operating frequency range, and within the frequency range that the instrument was operated at, its mainline loss was a relatively constant function of frequency (varied from 0.91 dB at 1MHz to 0.88 dB at 130 MHz). Its coupling factor also remained relatively constant throughout the frequency range of operation: 10.47dB at 1MHz and 10.43dB at 130MHz. While a relatively high-end directional coupler was used, a different one could be considered in order to further reduce the cost of the instrument. The TDC-10-1 costs $13.30 each (in quantities of >1000), and is rated for high power applications. Thus, considering cheaper alternatives which have a smaller operating bandwidth and lower power rating, might help lower the board cost considerably.

4.7 Probe Compatibility

In order that the instrument can be used for a variety of applications, the system was designed to allow the probe to be easily detached from the main hardware (i.e. the PCB). Different applications often require application-specific probes, thus the two signal outputs (i.e. the reference and active outputs) of the PCB were terminated with 50 Ohm gold SMA connectors, allowing various probes to be attached to the board using a female-female SMA adapter. It was found that using a cable to connect the board to the probe decreased the sensitivity of the instrument, as the longer transmission line presents a higher total impedance, decreasing the strength of the signal that eventually reaches the probe.

The probes used were different for each application, for example, a loop antenna was used for tag reading, a coaxial probe was used for the milk quality analysis application and a parallel-plate capacitive probe was used to investigate the mail content analysis application. The design and analysis of each of these probes will be presented in the subsequent chapters where the applications of interest are described in more detail.

In addition, the board for the instrument was also designed to allow a variable inductor to be mounted and set up to terminate the reference signal, when the instrument was used for tag reading. This allows one to matching of the antenna by tuning the inductor, improving the measurement baseline for tag reading.

4.8 Signal Detection

Complete characterization requires both amplitude and phase information of the reflected signal. While it is relatively easy to detect the change in the amplitude of the reflected signal, by using a diode detector for instance, detecting changes in phase is somewhat more difficult. One approach, as mentioned in the previous chapter, would be to use a tuned receiver to mix the reflected signal down to a lower frequency, and then employing DSP techniques to extract the phase information. However, due to the cost associated with using a mixer and DSP chip, a simpler method of detection was sought.

The approach adopted for this design was to use an RF/IF gain and phase detector (AD8302) produced by Analog Devices. This chip compares two AC signals and gives two DC values as outputs, one of which corresponds to the log of the ratio of amplitudes of the two signals, and the second corresponds to the difference in phase of the two signals. This approach provided a much simpler solution to detecting both phase and magnitude information, reducing the cost and complexity of the system.

While a more comprehensive discussion can be found in the AD8302 datasheet, a brief summary of how the chip functions will be given here, in order to give a better idea of what the outputs of the AD8302 actually represent. The two input signals into the AD8302 (V_{INA} and V_{INB}) are first passed into a pair of identical logarithmic amplifiers. The amplifier provides a logarithmic compression of the input signal, allowing a large range of input signal to be converted to a compact decibel scaled output. The two outputs of the log amps are then subtracted, and since a subtraction in the logarithmic domain corresponds to a division in the linear domain, the resulting output (V_{MAG}) becomes,

$$V_{MAG} = V_{SLP} \log_{10}\left(\frac{V_{INA}}{V_{INB}}\right) \tag{4-1}$$

Where V_{SLP} is the slope, and can be expressed in units of "volts/decade"

For phase measurement, the outputs of the two log amps are used to drive an exclusive-OR style digital phase detector. Operating strictly on the relative zero-crossings of the two signals, the extracted phase difference is independent of the original input signal levels. The phase output has the general form,

$$V_{PHS} = V_{\Phi}[\Phi(V_{INA}) - \Phi(V_{INB})] \tag{4-2}$$

Where V_{Φ} is the phase slope in mV/degree and Φ is each signal's relative phase in degrees.

The chip is capable of utilizing ac-coupled input signals in the range from –60 dBm to 0 dBm in a 50 Ω system, from low frequencies up to 2.7 GHz. The outputs provide an accurate measurement or either gain or loss over a +/-30 dB range scaled to 30 mV/dB, and of phase over a 0^{o}-180^{o} range scaled to 10mV/degree. Both phase and gain measurement systems have output bandwidths of 30 MHz, which can be reduced by the addition of external filter capacitors.

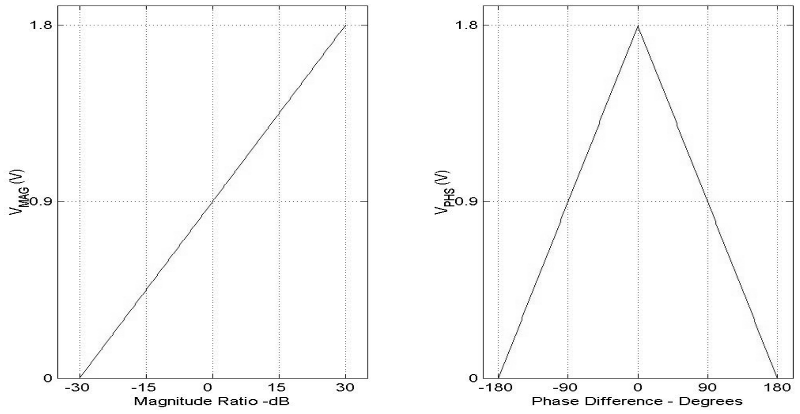


Figure 15. Idealized Transfer Characteristics of the Gain and Phase Detector

The AD8032 was operated on a +5 V voltage supply, and required several external passive components, for purposes including impedance matching with the source, and output low-pass filtering. The two signals used as inputs were the signals at the 'couple' ports of the two directional couplers to tap the backward traveling waves on the transmission lines. The output DC values representing the difference in magnitude and phase between the two signals, were read by 10-bit analog-to-digital converters on the microcontroller. These output values were in the range of 0-1.8 V, with 0 V corresponding to a –30 dB magnitude ratio between the two signals and 1.8V corresponding to a +30 dB magnitude ratio between the two signals. For the phase, 1.8 V corresponds to a phase difference of 0^o (i.e. both signals are in phase), and 0 V corresponds to a phase difference of 180^o (See Figure 15).

Note that by convention, the phase difference between two signals can take values in the range of $\pm 180^o$. Thus, this phase detector is unable to distinguish between $\pm 90^o$ for example, and as a result, the phase plot would be 'clipped' if the phase detector were operated about a phase-difference operating point of 0^o. To avoid this, the reference and active signals have to be 90^o out of phase with each other in the absence of a load, so that the phase detector would be operating around the middle of its continuous operating range. This led to the AD9854 being chosen over the AD9852, since the quadrature outputs of the AD9854 allowed one to be used as the reference signal, and the other as the active signal. This resolved the problem of the phase plot being 'clipped', which was a problem when the AD9852 was used and the reference signal obtained by using a splitter.

The complete schematic for this implementation can be found in the Appendix.

4.9 System Control and Operation

In this system, the microcontroller serves three main purposes. The first, and primary purpose is that it controls the operation of the DDS. This involves initializing the DDS chip by programming the appropriate registers. By doing so, it sets the mode of operation of the DDS, its internal clock frequency, and powers down unnecessary stages to conserve power. Once the DDS has been initialized, the microcontroller controls the frequency sweep of the DDS by programming the appropriate frequency into the frequency control registers in the DDS.

In addition to controlling the DDS, the microcontroller also acquires the phase and magnitude response of the signal from the detector. Two 10-bit analog-to-digital converters (ADCs) of the microcontroller are used for this. These two ADCs are connected to the phase and magnitude DC output values of the detector, allowing a digital value associated with the phase and magnitude response of the signal to be obtained. The 10-bit ADC converts a voltage level from 0 to 1.8 V into a digital value of 0 to 1023, giving a voltage resolution of 1.76 mV. Since the microcontroller was operated on a 3.3 V power supply and the phase and magnitude signals from the detector varied from 0-1.8 V, one of the ADCs of the microcontroller was used to set the upper limit (1.8 V) of the input range. This was done by connecting the reference ADC of the microcontroller (pin RA3) to the voltage reference output of the detector chip, and operating the ADCs in the appropriate mode (specified in the microcontroller program).

The third job of the microcontroller is to communicate with the external user interface. This involves transmitting the acquired information (i.e. the phase, magnitude and frequency) to the appropriate interface, as well receive external commands from the user interface. This communication was done using an RS-232 serial protocol.

Thus, the overall sequence of operations of the microcontroller can be summarized as follows: 1) Upon startup, the microcontroller initializes the DDS, and sets up the ADCs appropriately. 2) It begins a measuring phase and amplitude information, by first programming the DDS with an appropriate frequency, and then measuring the response from the detector. Sufficient delay is given between programming and measurement to allow the propagation of the signal to the front end, and to allow any transients to stablize. Once the phase and magnitude information at a particular frequency are acquired, the microcontroller transmits this to the PC interface for display. 3) The frequency at which the measurement is taken is incremented and the process is repeated, generating a frequency sweep, and acquiring phase and magnitude information across the entire frequency range. Figure 16 summarizes this sequence of operation carried out by the microcontroller. The full microcontroller code used to program the microcontroller can be found in the Appendix.

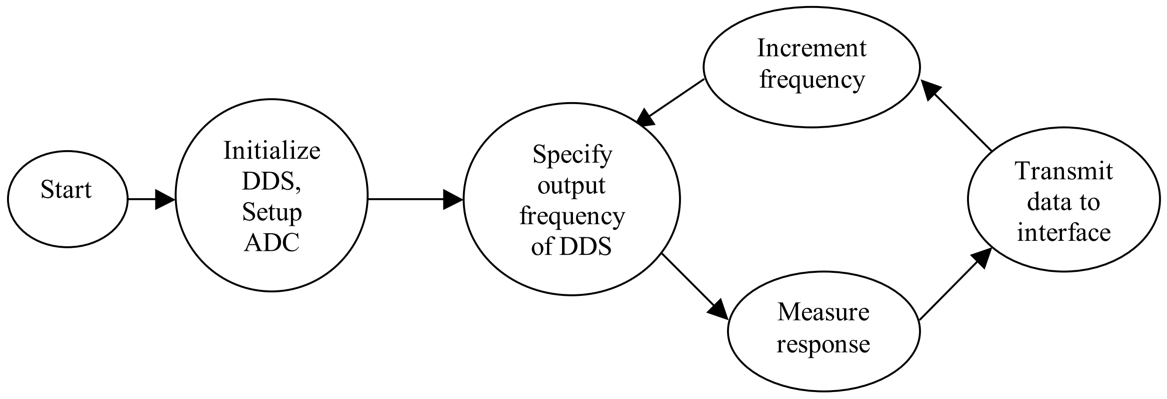

Start
Initialize DDS, Setup ADC
Specify output frequency of DDS
Increment frequency
Measure response
Transmit data to interface

Figure 16. Flow Chart Illustrating Simplified Sequence of Operation for Microcontroller.

The microcontroller that was used was the PIC16F877, produced by Microchip. This was selected as it had sufficient program memory, sufficient pins to operate the DDS in the parallel programming mode, and had built in ADCs. In addition, it operates on a sufficiently high clock speed (20 MHz) allowing rapid frequency sweeps to be generated and data to be transferred to the user interface quickly. The full schematic of the microcontroller can be found in the schematic, showing the various pin connections.

4.10 Power Supply

Since the various chips required varying voltage levels, several voltage conversions had to be carried out on the board. The instrument was powered externally using a 9 V, 1 A output wall adapter. The 9 V input was stepped down to 3.3 V using an IRU1030 linear voltage regulator made by International Rectifier. The only external components required for this regulator were two 330 uF electrolytic capacitors at the output and input terminals. A linear regulator was chosen, although it results in higher power dissipation, as it has a smaller output voltage ripple and lower electromagnetic interference (EMI) compared to a switching power supply. For a single board design like this one, power supply noise resulted in a degradation of the readings obtained by the instrument, as was experienced when the linear regulator was replaced with a switching regulator in a previous design. Because of the high power requirement of the DDS, the IRU1030 had to be mounted on a 0.5 inch, TO-220 finned heat sink, to ensure that the temperature of the device remained in its specified operating range. The 3.3V voltage level produced by this regulator was used to power the DDS and the microcontroller.

The 9 V input was also stepped down to 5 V using a second linear regulator (LM2937IMP-5 made by National Semiconductor). The 5V regulated output was also used to generate a -5V output using a

TPS6735 switching regulator, made by Texas Instruments. This switching regulator required quite a few external components (see schematics in Appendix) and is capable of supplying up to 200 mA of current. In addition, it has low output voltage ripple of approximately 5 mVpp. The ±5 V levels were used to power the output amplifier stage of the system. In addition, the 5 V level was used to power the MAX233 interface chip and the magnitude/phase detector. Figure 17 summarizes these voltage conversions.

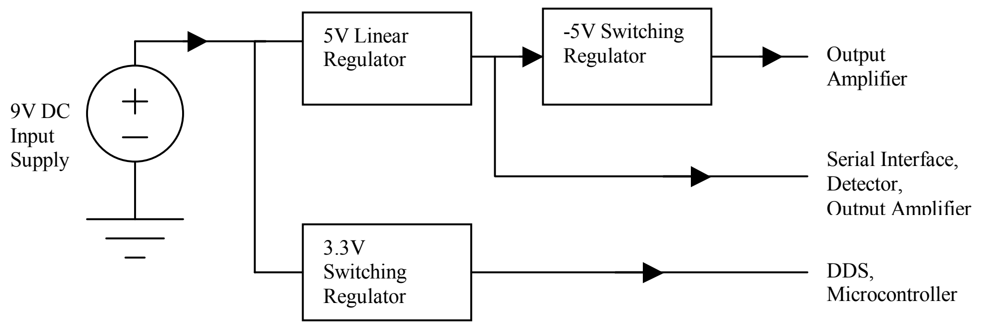


Figure 17. Power Conversion and Distribution Scheme for Instrument

One consequence of using linear regulators was the high power dissipation of the board. This would reduce battery life if this instrument were used for field use, which is undesirable. However, the attempt to replace the 3.3 V linear regulator with an equivalent switching regulator resulted in an increase in the amount of electromagnetic noise present in the circuit, due to the high switching speed, and this reduced the accuracy of the data acquired. Thus, one possible improvement to the design could be to employ switching regulators with sufficient RF isolation. Enclosing the switching regulators in a metallic enclosure, or have the power supply portion of the instrument on a separate board could help reduce the undesirable effects of EM noise while helping ensure a more efficient power supply.

4.11 Interface and Data Display

In order to display the data collected by the board, as well as to allow the operation of the board to be externally controlled, the microcontroller was connected to a PC by means of an RS-232 serial connection. For this, a MAX233 interface chip was used (produced by Maxim), allowing the microcontroller to be able to output character strings to the PC, and accept character commands from the PC.

Data collected by the board was sent to the PC in packets, with each packet consisting of the frequency of measurement, the magnitude data and the phase data. In addition, a termination packet was sent at the end of each frequency sweep, so as to allow the interface to detect when a sweep ends and move on to other functions such as processing and plotting the data. The data packets were organized as shown in

Figure 18, while the character string “88888888” was used for the termination packet. All packets were sent in ASCII format.

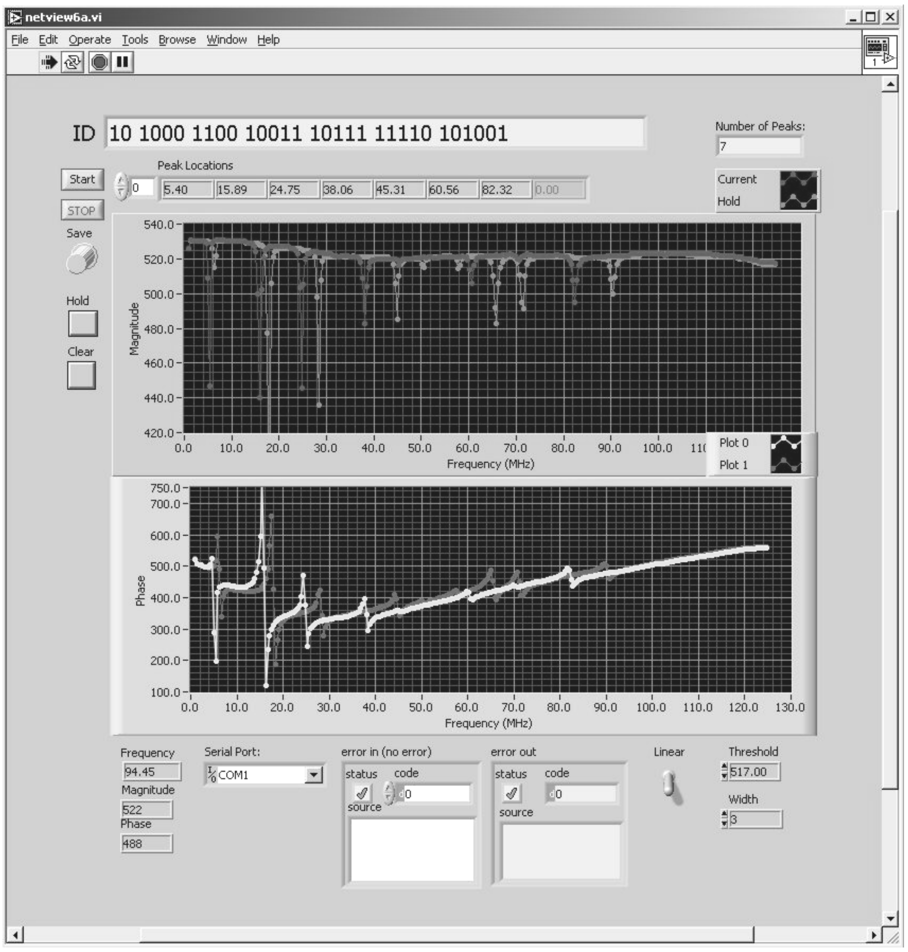
	Start	Frequency	Separation Character	Magnitude	Separation Character	Phase	End
Number of Characters:	1	8	1	4	1	4	2

Figure 18. Packet Format for Data Packet Sent by Board to PC

In order to process and display this data, and to provide a user-friendly way of controlling the board, an interface program was written for the board using Labview. Labview is an intuitive, industry-standard graphical development environment, and has a strong integration with a wide variety of measurement devices. It allows the rapid development of interface applications, especially those involving data flow, and thus is ideal for creating powerful applications to be used with this instrumentation. Functions and operations are invoked by passing certain inputs to a *virtual instrument*, which then returns outputs after performing the desired operation. Programming in Labview is greatly simplified as many virtual instruments are supplied with the software, allowing the programmer to work at a higher level of abstraction. Virtual instruments such as those for opening a serial connection, reading from the serial port, plotting data points and peak detection were used in the final interface program. In addition, Labview supports the execution of MATLAB scripts within the Labview program itself, and this proved to be an extremely valuable and versatile tool for carrying out various data operations.

The Labview program obtained and parsed data packets from the instrument and generated plots of magnitude and phase as a function of frequency. This gave a good visual representation of the frequency spectrum of the object under test. In addition, extra features were added to add more functionality: A function that holds traces was added to allow different materials or tags to be compared. Baseline subtraction could also be carried out, by clicking on a button on the control panel. This allowed for the frequency dependencies of the probe to be corrected for. A function was also added to save the plot data in a MATLAB data (*.mat) file, allowing analysis results to be stored and analyzed in MATLAB. In addition, time averaging of the collected data points was also done in order to improve amplitude resolution. For the tag reading application, it was also necessary for the program to be able to locate peaks on the magnitude frequency spectrum, and extract the information related to the peak (This will be explained in more detail in Chapter 5). Thus, a peak-detection routine was added to the program that took in inputs such as a threshold, and returned the frequencies at which peaks were located, the amplitude of the peaks, and the peak number of the particular spectrum. In addition, this peak data was processed and interpreted as bits of information.

The Labview program that was designed worked as follows: Upon execution, the program waits for the user to click on the ‘Start’ button on the console. This allows the user to ensure that the instrument is properly set up and powered up. In addition, the user can also properly configure various inputs to the console, such as specifying the correct COM port of the PC the instrument is connected to. Once the 'Start' button is clicked, the program opens the serial port and establishes communication with the instrument by sending a specific character to the board. Once this happens, the instrument responds by continuously transmitting data packets to the PC. The Labview program continuously loops, and reads the packets being output by the instrument. For every packet, it parses the data in to its frequency, magnitude and phase components. The value is rescaled from the representation used by the DDS (32-bit representation) to the corresponding frequency in megahertz. The program continuously reads from the serial port, and indexes the frequency, phase and magnitude data values into three separate matrices. When the interface program detects the termination string, it uses these matrices to generate phase and magnitude plots, as a function of frequency. Peak detection was carried out by passing the matrix containing the magnitude data for each sweep as an argument to a peak detection virtual instrument. Traces were held by storing the current trace as a global variable, and continuously displaying that variable on the graph. Data was saved by calling on



ID 10 1000 1100 10011 10111 11110 101001

Number of Peaks: 7

Peak Locations: 0 | 5.40 | 15.89 | 24.75 | 38.06 | 45.31 | 60.56 | 82.32 | 0.00

Frequency 94.45
Magnitude 522
Phase 488

Serial Port: COM1

error in (no error): status ✓ code 0 source

error out: status ✓ code 0 source

Threshold 517.00
Width 3

Figure 19. Screen Capture of the Labview Interface for the Instrument

the save command in MATLAB, which saves variables in the workspace as a .mat file. The full Labview program, netview.vi, is given in the Appendix, while Figure 19 shows a screen capture of the Labview interface while in operation.

It should be noted that, while the current interface has various features necessary for the applications that are being considered in this investigation, other applications that the low-cost network analyzer might be applied to might require the interface program to have different functions. However, the fact that Labview is a very powerful software and is one that is easy to program in, makes creating application-specific user interfaces extremely easy.

4.12 Performance and Specifications

The following are the performance specifications of the instrument, which were measured or calculated. Where measurements were taken, an Agilent E4403B spectrum analyzer was used.

1) Sweep range: 1 MHz to 125 MHz
2) Sweep speed (1 MHz to 125 MHz range): 700 ms
3) Frequency resolution (variable): nominal 0.2 MHz, as low as 1 microHertz (300 MHz DDS clock)
4) Magnitude resolution: 0.059 dB
5) Phase resolution: 0.18 degrees
6) Maximum baseline fluctuation: Magnitude: 0.35dB
   Phase: 35^{o}
7) Receiver dynamic range: ~30 dB
8) Output signal power: 1MHz: 8.3 dBm
   5MHz: 3.88 dBm
   40MHz: 0.316 dBm
   80MHz: -4.027 dBm
   120MHz: -9.10 dBm

9) Output signal-to-noise ratio (SNR): 39.06 dB (With signal at 60 MHz with a noise floor span of 40 MHz)
10) Power consumption 10.8 W
11) Parts cost (see Appendix for full price breakdown): $125 (in quantities of 1000)

The specifications of a commercially available network analyzer, that is cheaper than those made for laboratory use, is presented here as well to provide a comparison to what is currently available. The 250B network analyzer PCI card, produced by Saunders & Associates Inc. is sold as a test device used for the measurement of quartz crystals and ceramic resonators, and does precise measurements and characterization of crystals. This network analyzer card is installed into a personal computer, allowing the user to control the functions of the network analyzer via a custom program.

1) Frequency ranges: 10 KHz to 400 KHz, 500 KHz to 210 MHz
2) Output power: +18 dBm to –42 dBm (1-50 MHz), +15 to -42dBm (50 – 200 MHz)
3) Performs frequency and power sweeps
4) Supports various data display formats
5) 2 channel operation
6) Retail price of $5000 with OLE option (allows direct control of board using VB or visual C++ program)

4.13 Analysis of Design

In terms of cost, we see that the implemented system was built at a much lower cost than network analyzers used in the laboratory, which costs in the range of tens to a hundred thousand dollars. Compared to lower-end commercial network analyzers (as given in 4.12), the design presented here is still much cheaper. We see that even for the 250B PCI card network analyzer, it was designed to provide precise measurement values, and this would incur additional complexity in its design, adding to the extra cost. By having an instrument that is capable of more general measurements rather than precise ones, as in the case of the design presented here, allowed the cost to be significantly reduced. While our design does not support many of the features of commercial network analyzers, the subsequent chapters will illustrate that it is still a powerful and useful instrument, and has applications in RFID tag reading, as well as in materials analysis.

While the signal strength of the instrument varies with frequency, a reasonably flat mgnitude measurement baseline was still achieved. This can be accounted for by the differential front end design that was employed, which compensated for variations in the output signal.

To summarize, the network analyzer that was implemented was able to carry out basic reflection measurements across a broad range of frequencies (1-125 MHz). Examples of such reflection measurements will be further presented in the next two chapters. It is compatible with multiple probe designs (as will be seen in the next two chapters), and built into a compact form factor, and thus has the potential to be applied to a wide range of applications. The instrument also provides real time data and has a user-friendly interface, allowing users with little experience in operating RF test equipment to make use of the device. The combination of cost, ease of use and performance, makes the instrument an accessible tool for carrying out basic measurements, as well as a platform for investigating new applications of network analysis.

5. Tag Reader Application

5.1 Background and Motivation

At present, there is a fundamental need to wirelessly identify, sense and track arbitrary objects in our physical environment. One solution to this problem which is gaining commercial acceptance, is a technology known as electromagnetic tagging. This technology involves attaching a "tag" to a person or object, allowing a computer program or application to remotely track their position or sense their state. While other methods of tracking or sensing employ optical, acoustic methods, the method we are interested in here primarily involves the use of electromagnetic fields and the use of the electromagnetic response of the tag to encode information.

One current method of tagging employs tags in the form of radio transponders, and this has existed for the past 50 years. The simplest example of this is a radio beacon that emits a unique signal that can be detected by a radio receiver at a distance (several meters or more). Another variation of this consists of radio tags that only respond to a unique signal from a distant transmitter. Such tagging technology has been used in wild-animal tracking and military surveillance. Tags such as these comprise of a radio transmitter, which is powered by a battery and as a result, battery power increases along with the transmission distance.

Aside from long-range tagging, there is also a great interest in short-range identification using electromagnetic technology. This is commonly known as short-range RFID (Radio-Frequency Identification) and is widely employed today in applications such as automatic toll-collection on roads and security access cards. These tags make use of an electronic chip to communicate with the receiver over a short distance (~centimeters).

In an RFID system, the distance between the tag and the receiver (tag reader) is sufficiently small such that the signal between them is best characterized by the electromagnetic coupling between the tag antenna and the tag reader antenna. Parts of the tag reader are simply coupled together in a manner similar to transformer windings (inductive coupling) or as opposing plates in a capacitor (capacitive coupling). This coupling between tag and antenna serves two functions: Firstly, coupling is used as a means of supplying power to the tag. If the coupling is able to provide all the energy that is required by the electronics in the tag, then the tag needs no additional power source. Secondly, since the tag functions as an electrical load on the tag reader, the tag can communicate information to the reader simply by changing the value of its own impedance. The RFID tag accomplishes this task through the use of a small electronic chip, which functions as an active switch. As a result, the tag is not required to generate any transmitted signal, and the impedance-switching pattern is used to encode the information in the tag. The basic elements of an RFID tag are shown schematically in Figure 20. While this is not the only way RFID tags can function, it serves as a sample implementation.

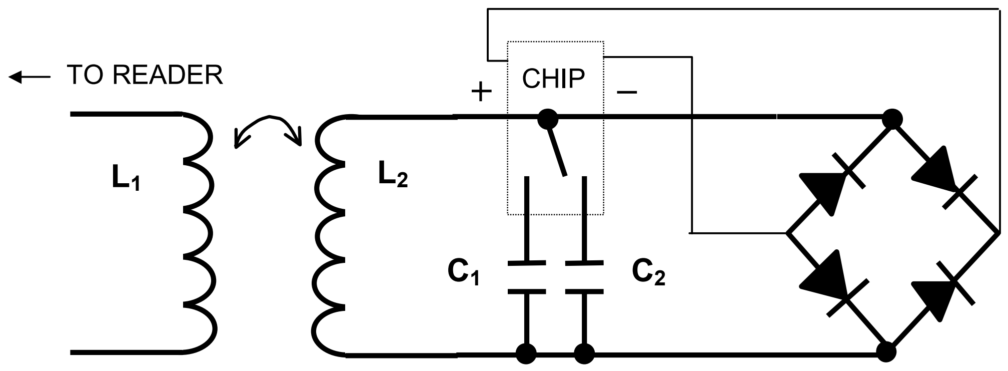


Figure 20. Diagram of a Tag Reader Antenna Inductively Coupled to a Batteryless RFID Tag

RFID tags can be made quite small and thin due to their relative simplicity, and thus can be embedded into security badges for instance, allowing automatic access to secure doors that are equipped with tag readers. Although the simplicity of the RFID circuitry provides a lower-cost alternative to radio tags, any amount of circuitry will result in the cost of the tag being more expensive in other wireless forms of identification, specifically the optical barcode. Optical barcodes can be easily printed on a variety of surfaces and thus, this technology is extremely cost-effective. However, such optical systems require a line-of-sight detection and some degree of alignment between the barcode and the reader, placing undesirable constraints on the user. Thus, there is a general desire to develop "electromagnetic barcode" technology, which does not require line-of-sight detection, allowing the reader to be hidden behind a wall panel or under a table. Also, the EM transfer of information is not affected by operation in dirty environments, allowing for higher reliability, and since orientation between the tag and reader is not required, it allows for a faster rate of scanning tags.

In an attempt to develop such RFID tags allowing its ubiquitous use, extensive work at the Physics and Media group has been undertaken by Rich Fletcher to develop a chipless RFID tag. Currently, the greatest obstacle preventing widespread use of RFID is its cost: although a chip for an RFID tag is considerably less complex than that of a Pentium chip, for example, the cost of handing and packaging these chips involve a cost in the order of $0.10. Given that there is little incentive for the semiconductor industry to push the cost of packaging considerably below this, one approach to low-cost tagging would be to eliminate the chip altogether.

Work has been done to develop materials based tags, which encode information in the physical structures of the tags. The electromagnetic response of the tag varies depending on the material structure, thus by having tags of varying designs, tags of varying information content can be made. The approach that has been employed to encode information within the tag is to utilize the frequency domain. That is, by designing material structures that respond to particular frequencies, it is possible to distinguish a tag from its background noise. Of particular interest are structures that exhibit electromagnetic resonance, as this enables a detectable excitation of the tag with minimal energy. Furthermore, these resonances appear as peaks in the frequency spectrum, which can be used to decode the information associated with the tag.

The structures that have been developed can be best described as multiply-resonant planar resonators, which are essentially made up of copper foil cut into a branching tree structure (see Figure 21).

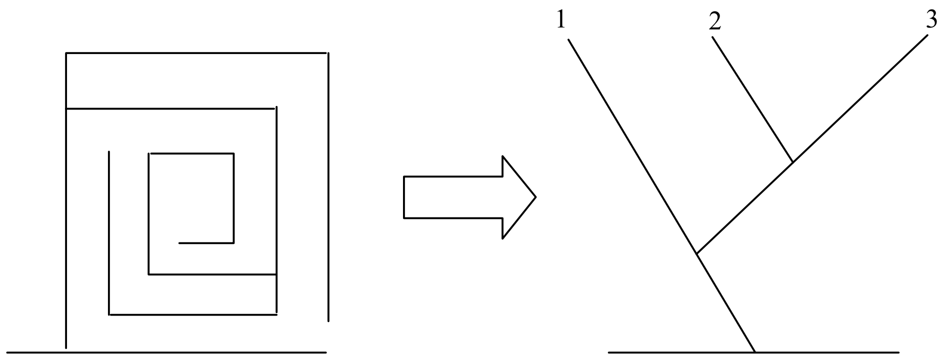


Figure 21. Diagram Illustrating Branching Antenna Structure of the Tag

This copper foil is folded and then laminated with a dielectric separating the folds, giving rise to a structure with multiple resonances. This multiplicity of resonances can be accounted for due to the presence of *multiple but non-equivalent current pathways* in the structure, and can be thought of as an L-C structure, where the inductance, L, is determined by the length of the branches of copper, and the capacitance, C, is determined by the area between corresponding branches separated by the dielectric. Figure 22 shows a picture of one of such tags that have been developed.

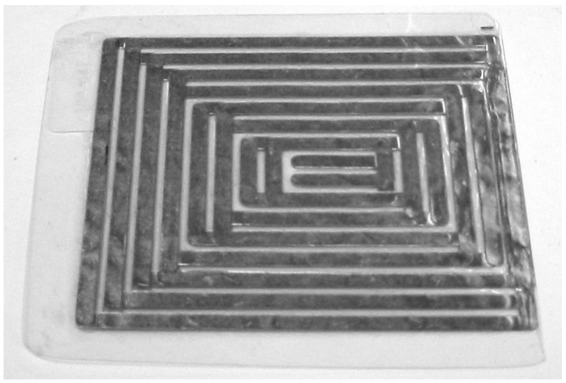


Figure 22. Picture of a Multi-Resonant Tag

We note that by varying the lengths of the branches, structures that have resonant frequencies at varying points on the frequency spectrum can be developed. By measuring the location of these resonant peaks, the information associated with the tag can be determined. Thus, interpreting the information content of such tags require a broadband instrument. This has been done using a HP vector network analyzer in the lab, although this is not cost effective for general use. In order that such tags can be widely deployed, a much more feasible system is required for tag reading.

This section of the thesis describes the work that has been done in implementing such a system using the hardware that was described in the previous section. How the generalized network analyzer board was used to allow it to perform tag reading will be described, and in addition, the suitable software interface that was developed to control the hardware, and to interpret the information content of the tag will be elaborated on. Together with this, we will also attempt to develop an algorithm of decoding the information associated with a tag based on its frequency spectrum.

It should be noted that the hardware that was developed, was initially designed solely for the purpose of tag reading. The earlier designs of the instrument were based on previous work that has been done in the Physics and Media Group with regards to low-cost RFID tag reading. [1] describes the previous generation tag reader that was implemented for reading chipless RFID tags, which operated in the 5-40 MHz range and is able to read the magnitude response spectrum of the tag. Later on, we realized that making the instrument able to read phase information, and allowing the design to support various probes would lead to a much more powerful instrument, with a wider range of applications. Thus, some of the design choices made in developing this instrument were made specifically with tag reading in mind. An example of this would be the frequency range the device operates: For the tags that were developed in our group, most of the resonant peaks are located in the 1 to 100 MHz range. Thus, the network analyzer board was designed to operate in this range in order that these peaks can be detected.

This part of the thesis describes the implementation of the network analyzer board as a tag reader, and evaluates its performance for this application.

5.2 System Description

To summarize, we require a tag reader that sweeps across the frequency spectrum of interest, and measures the magnitude response. In the presence of a tag, coupling between the tag and antenna will result in peaks at certain points in the magnitude response plot, whose locations correspond to the resonant frequencies of the tag. The tag reader must then detect the positions of these peaks and decode the information associated with them. Decoding involves converting the physical information of a tag into a particular identity, which often is a numerical value (ID number) associated with the tag.

The set-up used for this application consisted of the basic network analyzer board, described in Chapter 4, together with a single turn loop antenna used as the probe. The single turn loop antenna consisted of a 40 mil. trace printed trace on a PCB, connected to a male SMA connector. The antenna was then attached to the instrument using a female-female SMA adapter. For the reference probe, a surface mount coil inductor (Toko series, made by Panasonic), of similar inductance to the antenna coil, was used. The inductance of the antenna was measured using an HP LCR meter. The inductor used was tunable, and thus could be tuned to improve the matching between the reference and active probes. The hardware set-up is shown in Figure 23.

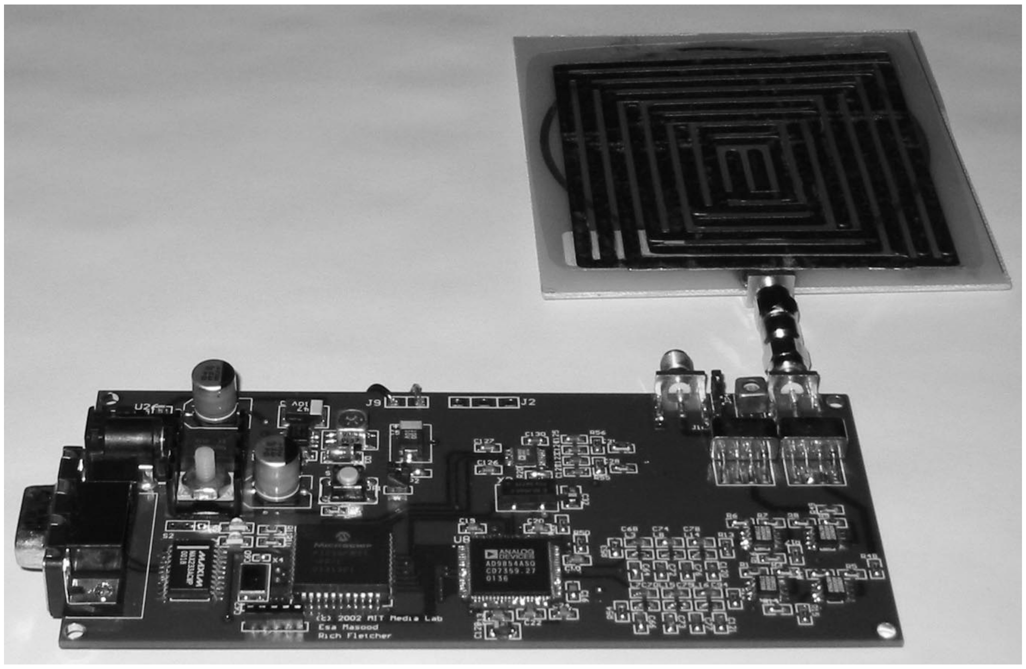


Figure 23. Picture of Hardware Setup Used for Tag Reading

The software interface used, as described previously, was designed in Labview and it generated phase and magnitude plots of the acquired data as a function of frequency. In addition, we used a peak detection function that is provided in Labview to locate the position of peaks in the spectrum. The peak positions were then used to associate a numeric ID with the tag, which will now be discussed.

While more complicated algorithms can be developed to interpret the spectral information associated with a tag, a relatively simple one was used here to associate a numeric ID with the spectral data. The peak-decoding algorithm was implemented as follows: The entire frequency spectrum was divided into 64 frequency bins (i.e. bins numbered from 0 to 63). If a peak were to be present within one of these bins, an associated ID would be associated with this peak, equal to the binary representation of that bin number (i.e. a 6 bit binary number is associated with each peak). The ID's of all the peaks in the spectrum were then appended to form the total ID of the tag. While the total number of bits of the final ID of the tag does not reflect the actual information content of the tag, this algorithm demonstrates a simple way of associating an ID with the spectral information of the tag. The actual information content of the tag would depend on our ability to control of the various peak positions and peak number in the design and fabrication of these tags.

5.3 Experimental Setup

The performance of the tag reader was evaluated by using it to read various tags. For this experiment, 6 different tags of varying structural designs were selected, and were labeled as Tag 1 through Tag 6. While elaborating about the intricacies of the structural design of the tags is out of the scope of this work, it would suffice to say that the different physical structures of the tags give rise to differing numbers of resonant peaks, varying positions of peaks within the spectrum as well as differences in peak amplitude and sharpness. Figure 24 shows the 6 tags that were used for this investigation.

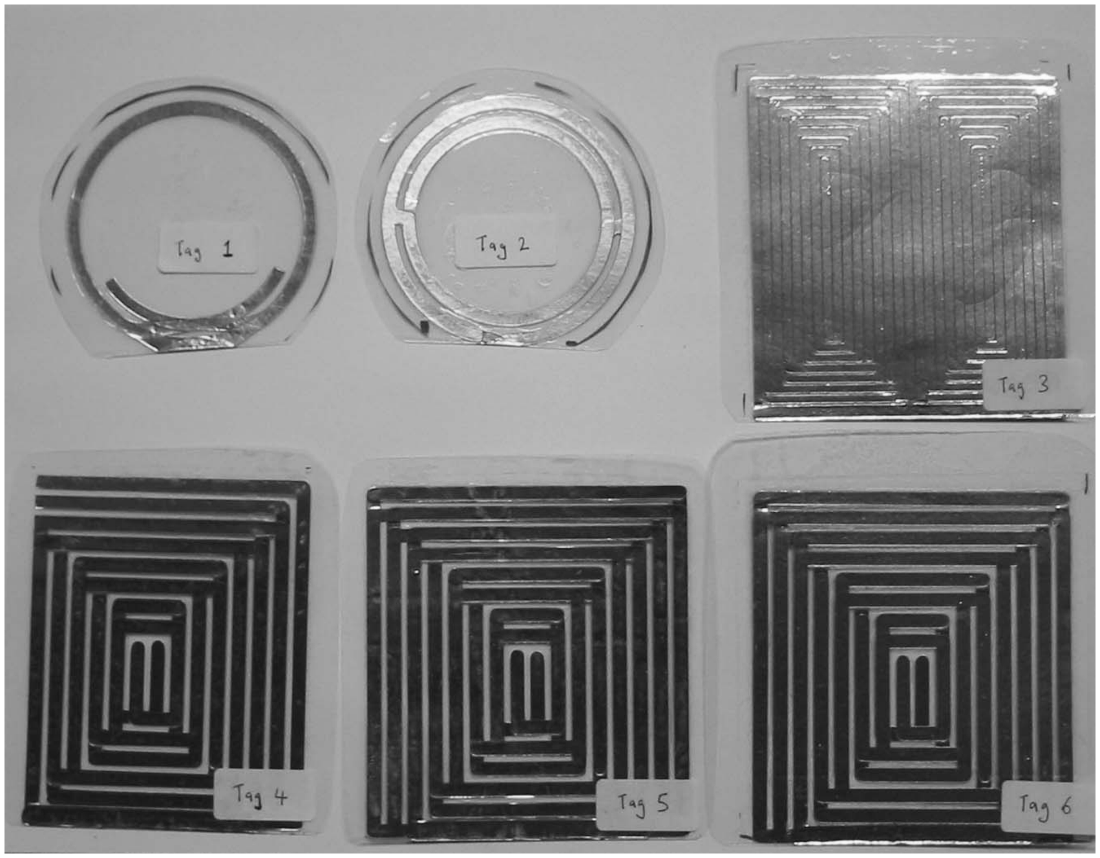


Figure 24. The 6 Chipless Materials-based Tags Used to Evaluate the Network Analyzer Board

In addition, using a particular tag, several different antennas were experimented with to evaluate the effect of antenna size on the performance of the tag reader (7, 8.5 and 10cm diameter circular antennas were used). Finally, for a specific antenna and tag, the tag was read from a range of distances (0mm, 5mm and 10mm) to evaluate the read range of the current tag reader design. The tag reader was run using the Labview interface described previously, and the results from the reads were compiled and are plotted using MATLAB, and will be presented in the next section. The numerical ID values associated with each tag, obtained using the algorithm described in the previous section, will also be presented.

5.4 Results

Figure 22 shows the complex spectrum of Tag 5, using an antenna with an 8.5cm diameter and at a 5mm read range. 7 peaks can be observed from the magnitude plot, indicating the positions of the resonant frequencies of the tag in the frequency spectrum. Peaks can be observed at: 6, 18, 28, 45, 58, 71 and 90 MHz (rounded off to the nearest MHz). The peaks are found to vary in amplitude, with the largest peak at 29 MHz (height of 5.2dB) and the smallest peak at 6 MHz (height of 0.7 dB) The corresponding phase plot shows a discontinuity in the phase at the points of resonance, corresponding to the peaks in the magnitude plot. For tag reading, we are only interested in the position of the peaks in the frequency spectrum, and we see that the magnitude spectrum alone is sufficient to allow us to identify the locations of these peaks.

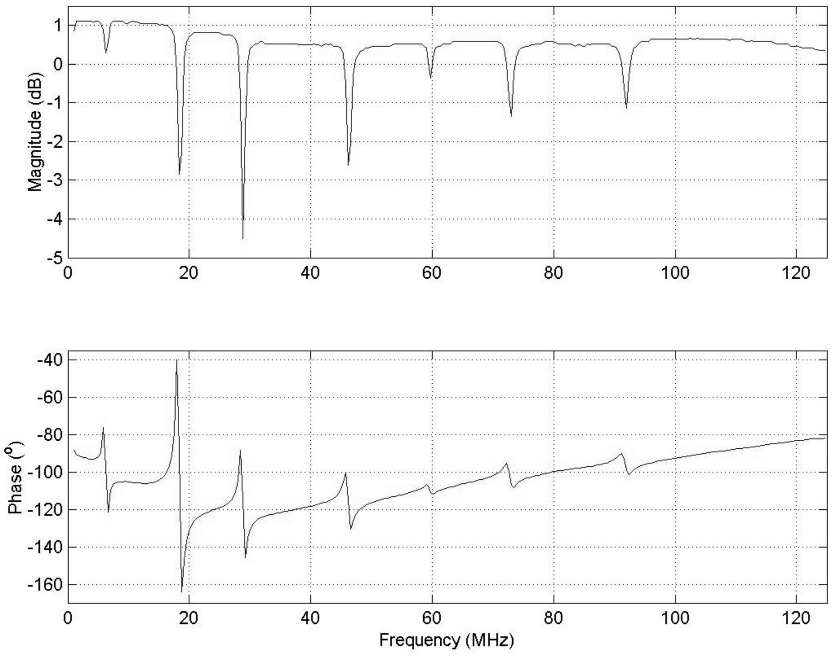


Figure 25. Graph Showing Complex Spectrum (Magnitude and Phase) of a Tag 5

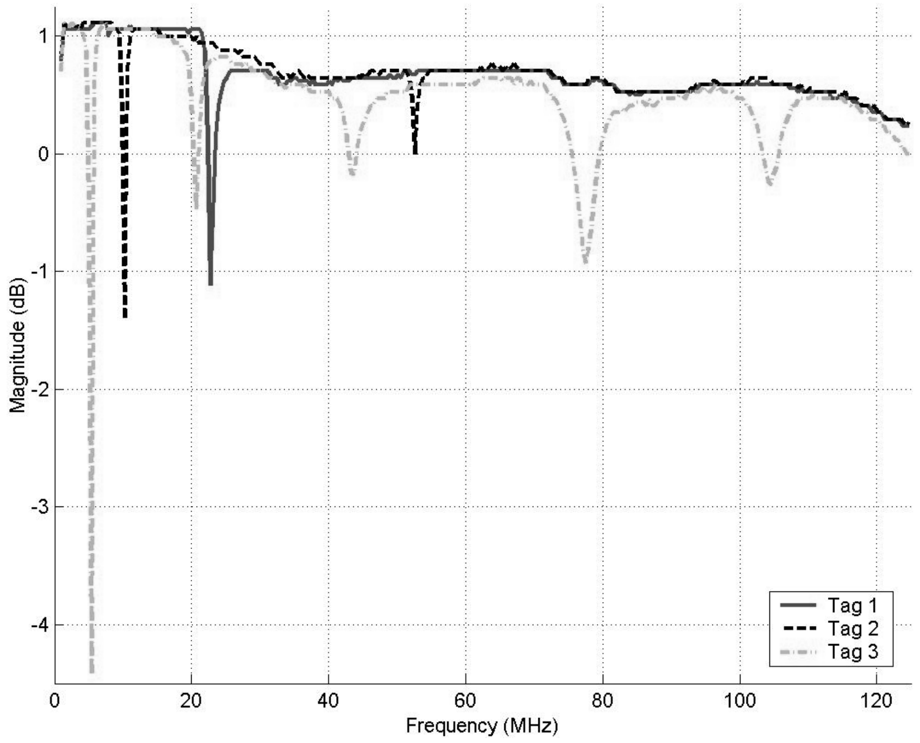


Figure 26. Graph Showing Magnitude Spectrum of Tags 1, 2 and 3

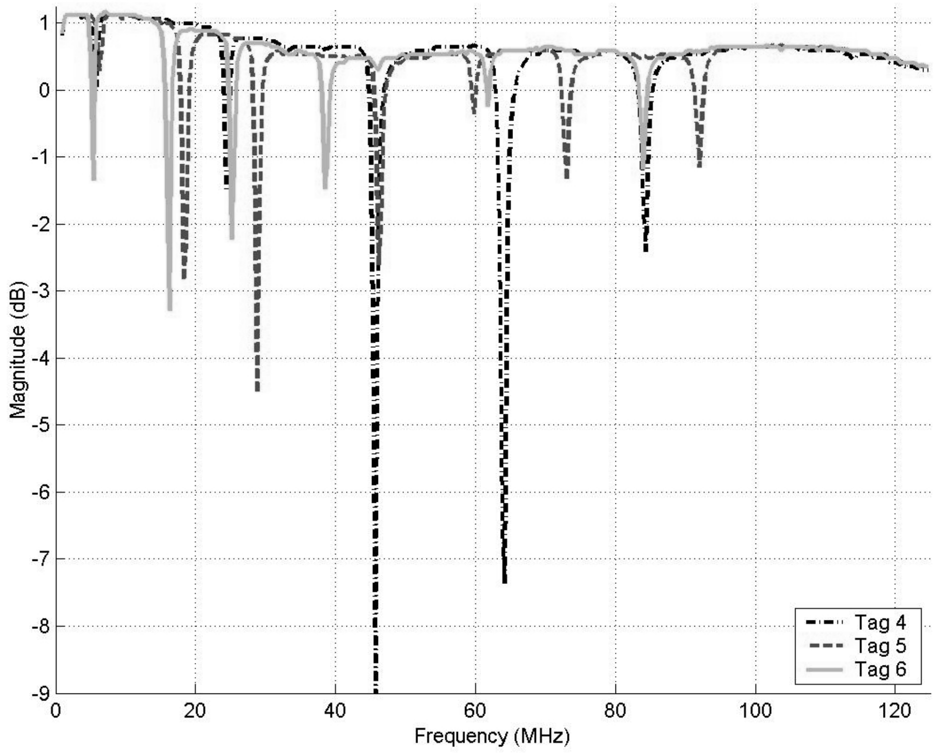


Figure 27. Graph Showing Magnitude Spectrum of Tags 4, 5 and 6.

Figures 26 and 27 illustrate the magnitude spectra of the 6 tags. We see that each tag has a unique frequency spectrum, with varying numbers of peaks, peak locations and peak amplitudes. The plots indicate that Tag 1 is a single peak tag, while Tag 2 has two resonance peaks. Tag 3, 4, 5 and 6 are multiple resonance tags, and have 5, 5, 7 and 5 peaks respectively. Table 1 shows the resonant peak locations and the ID associated with each tag.

Table 1. Resonant Peak Frequencies and ID Values Associated With Each Tag

Tag Number	Resonant Peak Frequencies (MHZ)	ID Number
1	22.1	001011
2	10.1, 51.3	000101011010
3	5.4, 20.4, 42.9, 76.7, 102.9	000010001010010101100111110100
4	5.8, 23.9, 44.4, 62.2, 81.8	000010001100010110011111101001
5	6.2, 17.9, 28.3, 44.8, 57.7, 71.0, 89.9	000011001001001110010110011101100100101101
6	5.4, 15.9, 24.7, 38.0, 82.1	000010001000001100010011101001

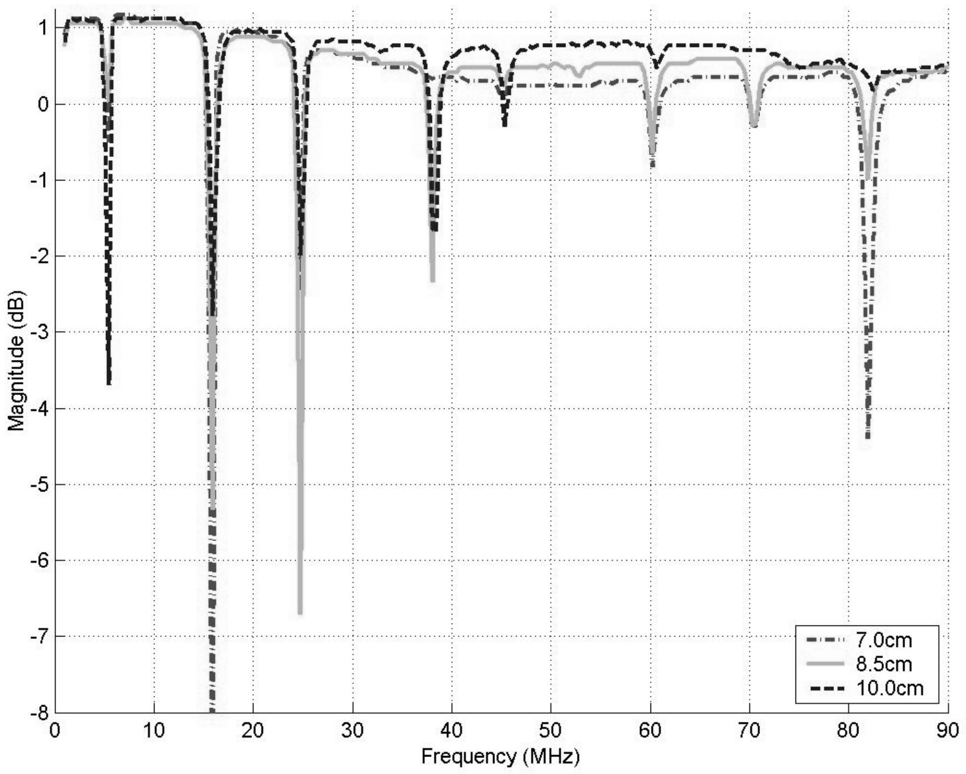


Figure 28. Plot of Magnitude Spectrum of Tag 6 Using Antennas of Different Diameters

Figure 26 shows the magnitude spectrum for Tag 6, using antennas of size 7.0cm, 8.5cm and 10.0cm, with a read range of 5mm. From the magnitude spectrum of the largest antenna, we see that the lower frequency peaks are much more evident than those at higher frequencies. The plot for the 7cm diameter antenna, however, shows large peaks at low and high frequencies, while some of the peaks in the 30 - 75 MHz range are absent. Of the three antennas, the peak heights across the entire frequency range are the most even for the 8.5 cm diameter antenna, with peaks across the entire spectrum being evident in the magnitude plot. Another observation is that the peak positions remain constant regardless of the size of the antenna used. Thus, the resonant frequencies are solely dependent on the tag properties, and not the antenna used.

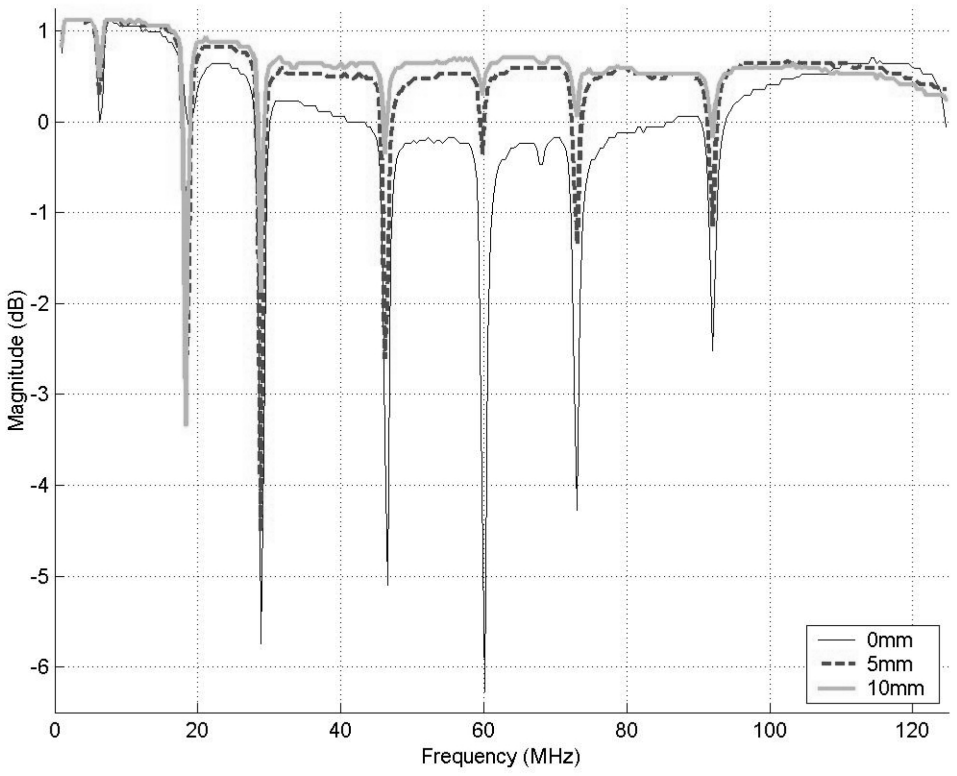


Figure 29. Plot of the Magnitude Spectrum of Tag 5 for Different Read Ranges

Figure 29 shows the magnitude spectrum of Tag 5 over 3 different read ranges, using the 7.5mm diameter antenna. The amplitudes of the peaks at most resonant frequencies are largest when the tag is closest to the antenna (read range of 0mm), and smallest when the tag is farthest away (read range of 10mm), with exception of the peak at 18 MHz, which is largest when the tag is 10mm from the antenna. An interesting observation is that as read range increases, the amplitudes of the peaks at higher frequencies drop much more significantly than those at lower frequencies. Thus, the read range for peaks at lower frequencies can be said to be much better than those at higher frequencies. We see that while the peak amplitudes vary with read range, the peak positions remain constant, and is independent on read range.

5.5 Analysis of Results

From the plots, we see that while the amplitudes associated with each tag changes with read range and antenna size, the peak positions are independent of the probe used and read range. This indicates that the best way to interpret the information content of the tag would be to utilize its peak positions. Utilizing peak amplitudes and sharpness would impose certain constraints to the read range, which might be impractical when deploying such a system of tagging.

With regards to the hardware, we see that the instrument can successfully obtain the magnitude spectrum associated with each tag, locate the peaks and decode the information associated with these peaks. In particular, the design of the network analyzer board provides a sufficiently flat measurement baseline across the entire frequency range, making peak detection much easier. From Table 1, we see that each tag has a different ID value, allowing the various tags to be distinguished from each other. While the algorithm used for this purpose is relatively simple, it nevertheless allows us to implement a system that would distinguish between several different tags. One disadvantage of the algorithm used is that the ID values of the different tags are not equal in length. One easy way to resolve this is to zero-pad the ID numbers, and restrict the ID numbers to a certain numbers of bits. Alternatively, a hashing algorithm could be used to compress an arbitrary length bit string into a fixed length one. This is possible since we know that there is some redundancy associated with the current bit string of a tag (e.g. not all the peaks of the tag can be the same frequency, and there are no repeated peaks). A third method is to employ a more sophisticated peak-decoding algorithm to convert the numbers to a fixed bit string. In order for this to be successfully implemented, the relation of the precise structure of the tag to its information content must be better understood. In addition, the degree of precision to which the tags can be manufactured must also be known in order to know the frequency resolution of the peaks in the spectrum.

For this investigation, we have operated the tag reader over a small read range (under 20mm). While this is sufficient for a near-field tag reading system, having a system that operates well over a longer read range (30-60mm) would definitely be desirable as this would make tag reading much more convenient for the user. Increasing the read range can be achieved by increasing the power of the output amplifiers. We have also observed that the read range for various peaks within the spectrum differ quite considerably. Figure 29 illustrates that peaks at the lower part of the frequency spectrum (< 40MHz) remain distinct as the read range increases, whereas the amplitudes of the peaks at higher frequencies fall off quickly as the read range is increased. This could be due to two factors: Firstly, the amplitude of the output signal of the instrument is smaller at higher frequencies than lower frequencies. This is due to the differences in gain at high and low frequencies for the amplifier stage, as fell as the low-pass filter. Secondly, the antenna structures in the tags responsible for the higher frequency peaks are smaller than that of the lower frequency peaks. This results in stronger coupling between the antenna and the tag at lower frequencies than at higher frequencies.

Thus, one way to have a system that has a longer read range is to utilize just the lower frequency range of the instrument. Although the decrease in spectral bandwidth would result in tags having fewer bits, increasing the frequency resolution of both the instrument as well as the tags can make up for this. In terms of manufacturing tags, it would mean making tags with sharp resonant peaks and being able to control the exact dimensions of the tag precisely in the manufacturing process. In terms of the instrument, it means decreasing the frequency step size at which it operates which can easily be done.

With regards to the antenna used, we note that the read range of the system does not vary monotonically with antenna size. For this analysis, we found that the medium sized antenna was best suited, as it allowed the peaks throughout the frequency spectrum to be detected. While a larger antenna increases the sensitivity of the probe, it also results in a smaller signal being transmitted through it due to the greater resistance and inductance of a larger antenna. Thus, choosing the optimum antenna size of the design is essential in maximizing its performance. The optimum antenna size would depend on several factors, including the size of the tag and the power of the signal being transmitted, and thus would differ depending on the specific characteristics of the system.

5.6 Summary

The design for the low-cost network analyzer board provides a viable solution for the implementation of a chipless materials-based RFID system. In terms of tag reading, readings made using this instrument indicate that sufficient spectral information about the tag can be obtained, allowing us to replace the commercial network analyzers that have been used for this purpose.

Nevertheless, we still need a better understanding the actual information of the frequency spectrum of the tag before such a system can be successfully implemented. This would allow better algorithms to be developed to decode the information associated with the spectrum of a tag. In terms of hardware, one possible improvement would be to increase the output power of the tag-readers so that the read range can be increased.

In addition to RFID tags, sensor tags have been previously developed in our group, whose physical properties are altered as a result of a change in the external environment (e.g. pressure or temperature change). This change in physical properties changes the resonant frequency of the tag. Thus, together with these sensor tags, another possible application for the network analyzer board would be in sensing changes in the external environment, or changes in the temperature or properties of objects. Since these sensor tags, which are materials based, can be very cheaply produced, such a sensing system is another possible avenue for a wide deployment of network analysis, in addition to RFID tag reading.

6. Materials Analysis Application

6.1 Background and Motivation

There has been considerable interest in the electromagnetic analysis of materials, specifically in the characterization of materials based on their frequency dependent complex permittivity. Such a technique of analysis offers a quick, non-destructive way of analyzing various materials, and has been employed in analyzing biological substances [2], various liquids [3] as well as for environmental applications [4]. Since various materials have different microscopic physical properties and thus respond differently when probed with an electric field, analyzing the complex-dielectric permittivity of materials is extremely useful and powerful, as it allows us to characterize and distinguish between a wide range of materials.

Aside from analyzing the materials based on their dielectric permittivity, other methods of RF materials exist and are commonly employed. One such method is nuclear magnetic resonance (NMR) spectroscopy, which is a powerful and versatile method for the investigation of the structure, dynamics and composition of both liquid and solid molecules. NMR uses the magnetic properties of molecules and their interaction with both an external magnetic field and electromagnetic waves to produce detailed images (as in magnetic resonance imaging, MRI), or to determine the chemical composition of a test material. The information obtained using NMR spectroscopy reflects the magnetic resonance properties of nuclear particles within the material, particularly of carbon and hydrogen.

For NMR, a strong magnetic field is required to perform measurements, and magnets used for this often are in the range of 1 to 5 Teslas. This is approximately 20,000 times stronger than the magnetic field of the earth. Because the requirements on the magnetic and electric fields, the instrumentation used for NMR is often bulky and/or requires high power to produce the necessary fields. This makes it difficult to design a low cost and lightweight NMR spectrometer that is cheap, portable and can be used for field use. However, measuring other RF properties, such as dielectric permittivity, are more amenable to the design of simple yet sensitive instruments [3], hence are of greater interest for large scale deployment.

Conventionally, the dielectric constant of materials is divided up into real and imaginary parts. This can be best understood by understanding the transport mechanisms that operate in a material in the presence of an electric field. A time varying electric field induces two different kinds of current in a material. Conduction current is produced by a net flow of free charges, and the bound charges generate a displacement current. The conduction current is related to the electric field density by Ohm's law as follows,

$$J_C = \sigma E \qquad \textbf{(6-1)}$$

where J_C is the conduction current density, and σ is the conductivity of the material. Displacement current density J_D is related to electric flux density by,

$$J_D = j\omega D \tag{6-2}$$

The total current density, J_T, is the sum of these two currents:

$$J_T = \sigma E + j\omega\varepsilon E \tag{6-3}$$

While conduction represents the loss of power, there exists another source of power loss in dielectric materials. When a time-harmonic electric field is applied, the polarization dipoles within the material flip constantly back and forth. Since the charge carriers have a finite mass, the field must do work to move them and the response cannot be instantaneous. Hence, the polarization vector lags behind the applied electric field, and this is noticeable especially at higher frequencies. In order to include this phenomenon, we can modify (6-3) as follows,

$$\begin{aligned} J_T &= \sigma E + \omega\kappa'' E + j\omega\varepsilon E \\ &= j\omega\left(\varepsilon - j\frac{\sigma + \omega\kappa''}{\omega}\right)E = j\omega\varepsilon^* E \end{aligned} \tag{6-4}$$

where ε^* is referred to as the complex dielectric constant. The usual notation is:

$$\varepsilon \equiv \varepsilon'\varepsilon_o \quad \text{and} \quad \frac{\sigma + \omega\kappa''}{\omega} \equiv \varepsilon''\varepsilon_o \tag{6-5}$$

where ε' and ε'' represent the real and imaginary parts of the relative complex dielectric permittivity of the material respectively. Another useful value that is often expressed is the loss tangent,

$$\tan\delta \equiv \frac{\varepsilon''}{\varepsilon'} \tag{6-6}$$

This value is termed as such because it represents the tangent of the angle between the displacement phasor and the total current, and it is close to zero for a low-loss material.

The complex dielectric constant of a material varies as a function of frequency, and this can be accounted for by the different polarization and transport mechanisms (as were mentioned in Chapter 2) that operate at different frequencies, due to the differences in the structural properties of the materials at a microscopic level. Physically, what we achieve by probing a material across a frequency range is that we obtain information about the material across varying length scales. This in turn tells us about the structural and chemical properties of the material. Another way of looking at this is to think of a dielectric as a lumped circuit. This method is often used at lower frequencies to represent a dielectric specimen as an equivalent electrical circuit at a given frequency. If we represent the specimen as a capacitance C_p in parallel with a resistance R_p, the total impedance of the specimen Z can then be given as,

$$\frac{1}{Z} = \frac{1}{R_p} + j\omega C_p \tag{6-7}$$

In terms of R_p and C_p, we can re-express the real and imaginary parts of the dielectric permittivity as,

$$\varepsilon' = \frac{C_p}{C_o} \tag{6-8}$$

$$\varepsilon'' = \frac{1}{R_p C_o \omega} \tag{6-9}$$

which allows us to calculate the dielectric permittivity of a material from the measured values of the equivalent parallel circuit components of the material. While this is a good model at low frequencies, the lumped circuit model breaks down at higher frequencies when the signal wavelength becomes comparable to the specimen dimensions, and a distributed-circuit model must instead be adopted. Thus, a dielectric would appear as different electrical networks across the frequency spectrum, and this can be measured using an electrical method.

The general behavior of the complex dielectric permittivity of a material across a large portion of the frequency spectrum is shown in Figure 30. At every frequency where ε' varies rapidly, there tends to be a peak of the ε'' curve. This is often analogous to resonance in a tuned circuit: molecules have a natural frequency due to their intermolecular interactions, and they will transfer maximum energy from an electromagnetic wave at these frequencies.

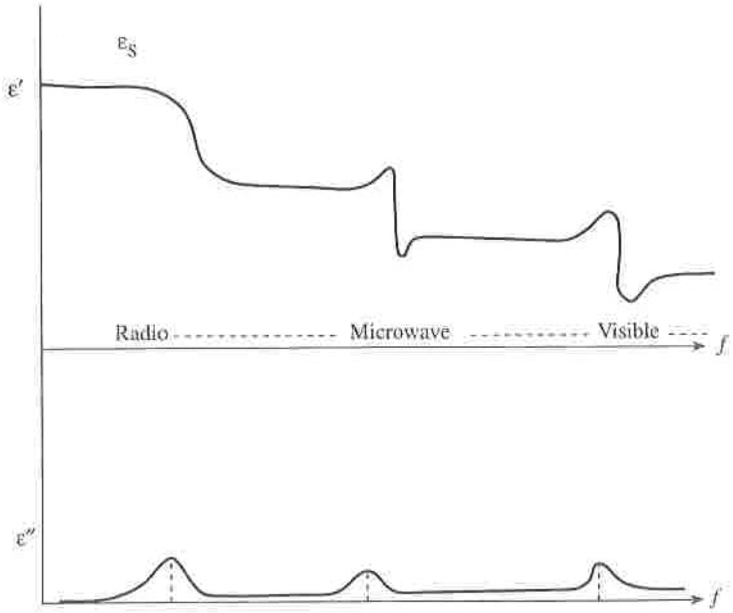


Figure 30. Typical Variation of ε' and ε'' with frequency.

This variation of complex dielectric permittivity with frequency has been described by several models, which have been proposed to model this behavior. One such model is the Debye model, which gives the complex dielectric permittivity of a material, ε , as,

$$\varepsilon = \varepsilon_\infty + \frac{\varepsilon_s - \varepsilon_\infty}{1 + jf/f_c} - j\frac{\sigma_o}{2\pi f \varepsilon_o} \qquad \textbf{(6-10)}$$

where ε_s and ε_∞ are the static and high frequency permittivities respectively, f_c is the critical or relaxation frequency, and σ_o is the ionic conductivity. A second model takes into account heterogeneity within materials, and expresses complex dielectric permittivity according to the Cole-Cole equation [5] as,

$$\varepsilon = \varepsilon_\infty + \frac{\varepsilon_s - \varepsilon_\infty}{1 + (jf/f_c)^{1-\alpha}} - j\frac{\sigma_o}{2\pi f \varepsilon_o} \qquad \textbf{(6-11)}$$

where α is the Cole-Cole spread parameter, which depends on the heterogeneity of the material. Both these models illustrate that dielectric permittivity varies as a function of frequency, and thus depends as well on the structural properties of the materials.

Several methods can be used to measure the complex permittivity of a particular material. These include methods employing free space, resonator, transmission or reflection measurements. The reflection method of determining dielectric permittivity has been widely investigated and is an extremely promising approach. The common way this is carried out is to attach an appropriate probe to a vector network analyzer (such as a HP8510 or a HP8753), and to introduce the material of interest to the probe. Accurate frequency dependent dielectric measurements have been taken this way, as will be elaborated on in the next section.

While work has been done in analyzing various biological and liquid substances in the lab, this method of analysis has not become widespread due to the inhibitive cost of instruments used to conduct such measurements. In addition, wideband relative dielectric constant measurements have not yet found much application in commercial material sensing applications. The main reason for this is cost: the reflection method described above relies on the use of an extremely expensive instrument, thus inhibiting its use outside the lab. Similarly, other methods of dielectric measurement require equally expensive and bulky instruments. Conceivably, we might wish to analyze biological substances in the field, or various substances or foods in the home. Thus, having a low-cost instrument that can carry out basic frequency dependent dielectric measurements would be extremely useful both in terms of implementing known applications, as well as to explore new applications of this form of analysis outside the lab.

Thus, the aim of this part of the thesis is to illustrate the usefulness of the low-cost network analyzer board in carrying out basic materials analysis. Here, we are interested in showing that a low cost instrument can be a useful tool in characterizing certain materials, as well as distinguishing between different material samples. We will illustrate the performance of the instrument using two test cases. The first is in analyzing milk quality, and the second is in analyzing postal mail content. Both these applications are extremely relevant today and provide good test cases to illustrate the usefulness of such non-destructive forms of materials analysis. It should be noted that, while it is desirable, we have not attempted to build an instrument that is capable of measuring accurate dielectric constant values, or acquire the precise complex permittivity spectra of materials. Instead, what we have aimed for is a low-cost instrument that provides a basic set of useful functionality. Relaxing the requirement on the instrument allows a great deal of complexity in its design to be eliminated, thus allowing it to be implemented in a form that is more affordable and accessible to a wider audience.

We will now elaborate on the test applications that were investigated. The first test case, which is the qualitative analysis of milk, has applications in India where determining the quality and chemical composition of milk remains a problem in many rural societies. In these societies, farmers are paid for their produce depending on the quality and fat content of the milk. This is necessary, since it is very conceivable that farmers might try to dilute their milk with water, or other liquids that are similar in color to the milk, in order to increase the amount of milk that they can sell. A typical transaction between farmers and wholesalers involves farmers transporting their produce to a certain collection point, where wholesalers determine the quality of the milk and decide how much to pay the farmer for that batch of milk. We can immediately see that for such a purpose, a system that is able to analyze samples quickly and non-destructively is extremely advantageous. Not only would efficiency be increased and less of the produce be lost due to testing, other indirect advantages would be that farmers would have to spend less time in queues, and wholesalers would have to spend less time collecting milk. This would in turn reduce the amount of time milk is spent unrefrigerated in the open, thus reducing the spoilage of milk. Thus, we propose employing complex dielectric permittivity analysis for analyzing milk quality. This form of physical analysis has several advantages over traditional forms of chemical analysis used to determine fat content. Firstly, it is not destructive, eliminating wastage. Secondly, it can be carried out much more quickly.

For such an application, we would require that the instrument used be sufficiently sensitive to be able to distinguish between milk samples of varying fat content. Since water constitutes the bulk of a sample of milk, we would expect that the difference in fat content would only lead to a slight variation in dielectric permittivity. Previous work has been done in the area of dielectric analysis of milk, and results from those tests illustrate that such a method is promising. Laogun has shown that the dielectric permittivity of milk from various mammalian species have a better agreement with the Cole-Cole model than the Debye model, within the 0.1 to 100MHz range, due to the inhomogeneous nature of milk, which is not accounted for by the Debye model [6]. Also observed, were different dielectric permittivities for the various milk samples, due to their differing compositions. In addition, using the single-mode cavity perturbation method, Kudra et. al. [7] have shown that varying ionic concentrations, milk fat percentage, and protein concentrations in milk lead to differences in the dielectric permittivity of milk. Thus, it has been well established that differences in the chemical composition in milk lead to differences in frequency dependant dielectric permittivity, and our aim for this application is to analyze the performance of the instrument with regard to this application.

The second test case in which we are interested is analysis of postal mail content, and this has implications in ensuring security in postal mail. The United States Postal Service (USPS) handles a large amount of mail daily, and a considerable amount of this is delivered across the country by air. However, almost none of this mail goes through any form of screening. This poses a serious security problem, since it is not unimaginable that someone could ship a package containing a bomb, or some other hazardous device, and have it detonated while in mid-flight. Through our collaboration here at the Media Lab with USPS, we have come to realize that this is an urgent and serious problem. Considering that the USPS has numerous postal collection points, for a system of screening mail to be implemented feasibly the per unit cost of the instrument used for analyzing mail would have to be sufficiently low. In addition, the method of analysis has to be non-intrusive (i.e. parcels and packages should remain intact and in their original forms) and fast, to cope with the large amount of mail. Thus, we propose using frequency dependent dielectric constant analysis for such an application. Specifically, we want to demonstrate that a low cost network analyzer can be used as an instrument for analyzing mail content, and distinguishing between different materials found in postal mail. Such a form of analysis is able to provide more information about a particular package than traditional metal detectors, and is much cheaper than other imaging systems that employ X-rays or nuclear magnetic resonance.

For the low-cost network analyzer board to be successfully implemented, it must be able to distinguish between various different materials such as paper, air, metal, and plastic. For this part of the thesis, we analyzed several different materials that could be found in postal mail to evaluate the performance of this instrument in distinguishing between these materials. While the implementation of a complete system to analyze postal mail and identify possibly hazardous packages is not within the scope of this thesis, illustrating that this instrument can perform basic materials analysis is definitely a step in the right direction in the development of such a system.

6.2 System Description

Appropriate probe selection is key in the successful analysis of materials. Due the different nature of the materials we are dealing with (liquid samples, in the case of milk analysis, and solid materials in the case of the mail application), different probes were selected for each application. However, the method employed for both, in determining the frequency dependent dielectric permittivity is the same; that is the reflection method was used in both cases. The reflection method of measuring dielectric permittivity involves sensing the changes in reflection coefficient at the end of the transmission line. The reflection coefficient is a function of the impedance (Z) terminating the transmission line, as given by (3-2), and for a particular probe, Z will be determined both by aspects of the measurement system, as well as the dielectric constant and loss tangent of the sample probe is exposed to. Thus, with prior measurements about particular the system and probe, one can accurately determine the complex permittivity of a particular material under test.

For our applications however, we do not wish to know the specific dielectric constants of the materials we are testing, but rather, we desire a system that allows us to infer properties of the materials and to distinguish between different samples. For such purposes, we require only knowledge of the relative differences between samples, and this can be done by directly comparing their phase and magnitude spectra, since these give us information of their complex dielectric permittivity.

For the measurement of dielectric constants of liquids, considerable work has been done with regard to the use of open-ended coaxial probes for accurate dielectric constant measurements. Boughriet et. al. [8] have has shown that measurements made using this type of probe agrees well with known dielectric constant values of various liquid solutions. In addition, the use of a coaxial probe has been proven to be a successful technique in taking measurements such as these, from RF frequencies up to 1 GHz [9]. Thus, the approach taken for the milk quality analysis investigation was to use the network analyzer board together with a 0.25 inch coaxial probe.

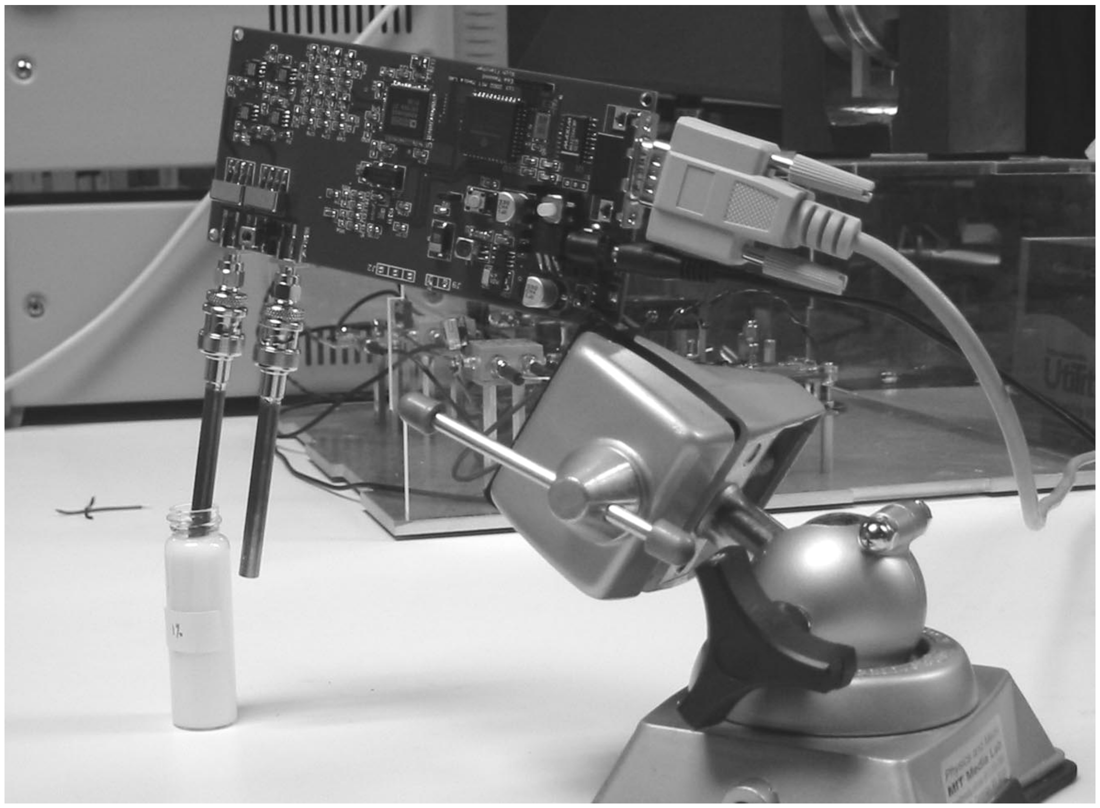


Figure 31. Taking Readings Using an Open-Ended Coaxial Probe

Measurements were made using this setup, by introducing the open end of the coaxial probe to a liquid sample (see Figure 31). As a function of the frequency dependent complex dielectric permittivity, $\varepsilon(\omega)$, the admittance of the load on the end of a coaxial line is given by,

$$Y = \frac{1}{Z} = \frac{j2\omega I_1}{[\ln(b/a)]^2}\varepsilon(\omega) - \frac{j\omega^3\mu_o I_2}{[\ln(b/a)]^2}\varepsilon^2(\omega) + \frac{\pi\omega^4\mu_o^{3/2}}{12}\left[\frac{b^2-a^2}{\ln(b/a)}\right]^2 \varepsilon^{5/2}(\omega) \qquad \textbf{(6-12)}$$

where ω is the operating angular frequency and a and b are the inner and outer radii of the coaxial line respectively. I_1 and I_2 are integration constants dependent only on the geometry of the aperture. This formula is referred to as a three-term approximation with calculation coefficients [10], and it has been used to obtain approximate readings for frequency-dependent dielectric constant [11]. From this equation we see that, at a particular frequency, the load terminating the coaxial probe is a function of $\varepsilon(\omega)$, allowing us to infer differences in $\varepsilon(\omega)$ by comparing the phase and magnitude spectra of different materials. Another identical probe was used to terminate the reference signal in order to match and compensate for the frequency dependence of the active probe.

While an open-ended coaxial probe is suitable for measuring liquid samples, it cannot be similarly applied to our second application, where test objects vary in terms of size and geometry. Thus, a different probe was selected for this application. A simple capacitive probe was chosen, comprising of two parallel conducting plates, separated by a distance of 3.8cm (1.5 inches), as shown in Figure 32.

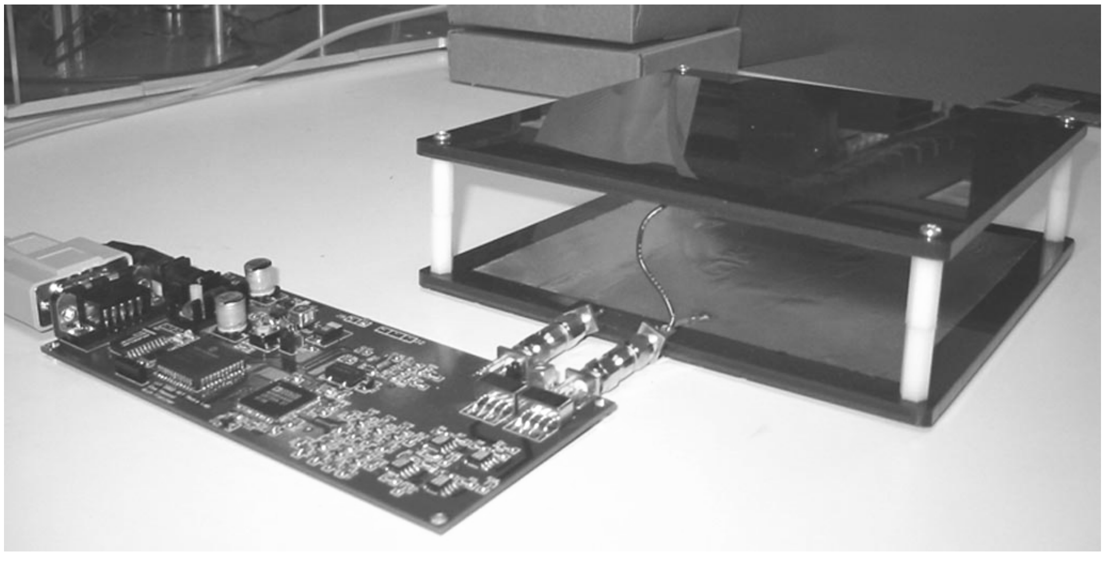


Figure 32. Capacitive Probe Used for Mail Content Analysis

The probe was made using two adhesive 14.0cm x 14.0cm copper sheets, and these were each attached to a 16.4cm x 16.4cm piece of hard plexi-glass. The sheets where then separated using four pairs of 0.75 inch vinyl standoffs. The active signal output of the network analyzer board was connected to one plate, while the other plate was grounded. This provided a capacitive termination for the transmission line.

For such a set up, making an approximation by assuming the fringing fields between the plates to be negligible (this is a poor approximation, but we will make this assumption for this qualitative treatment of the probe end dependencies), we see that the impedance terminating the transmission line can be expressed as:

$$Z = \frac{1}{j\omega C} = \frac{d}{j\omega A\varepsilon} \qquad \textbf{(6-13)}$$

Where C, d, A and ε are the capacitance of the probe, distance between the plates, area of the plates and dielectric constant of the material between the plates respectively, and are related as given in (2-2). Again, we see that Z at a particular frequency varies depending on the ε value of the material under test.

To obtain measurements, materials of interest were placed between the parallel plates, and the phase and magnitude spectra of the samples were recorded. Since the load terminating the output is dependent on the dielectric property of the material between the plates, the measured spectra changes depending on the material that is placed between the plates of the probe.

The capacitance of the probe was measured using a HP LCR meter, and the reference probe was terminated with a similar value chip capacitor (8.2pF), to provide proper matching.

6.3 Experimental Setup

The performance of the network analyzer board for this application was evaluated by using it to analyze several samples of milk of varying fat content. Light cream, whole milk, 2% milk and skim milk and water were analyzed by inserting the coaxial probe of the instrument under the surface of the sample. The spectra were then allowed to stabilize and the plots were recorded using the Labview interface program. The various complex spectra recorded will be presented in the next section. Table 2 gives the composition of the various types of milk used, as specified by the retailer.

Table 2. Composition of the Various Liquid Samples Used per 236ml Serving

Sample	Fat (g)	Cholesterol (mg)	Sodium (mg)	Sugars (g)	Protein (g)
Light Cream	47	157	157	15	0
Whole Milk	8	35	125	12	8
2% Milk	5	20	125	12	8
Skim Milk	0	5	125	12	8

For the mail analysis application, identical cardboard boxes (of dimensions 13.3 cm x 12 cm x 3 cm) were filled with samples of paper, acrylic, glass and metal, and were then placed between the plates of the capacitive sensor. An empty cardboard box was also analyzed as a control. The complex spectra of these materials will also be presented in the next section.

6.4 Results

Milk Quality Analysis Application

Figure 33 shows the magnitude plots for the various liquids analyzed. Also plotted is the magnitude spectrum for air, which is used as the baseline for reference. We observe that the various liquids give rise to distinct traces. In terms of magnitude, a large difference could be observed between the various milk samples and water. The differences between the traces for the milk samples were smaller, but were still distinct. It can be observed that a decrease in the fat content of the sample resulted in a gradually lower magnitude plot.

The phase plot of the various samples is shown in Figure 34. Here, we observe that the traces for air and the other substances are spaced quite widely apart, whereas those of the other 5 liquids are spaced more closely together. All of the traces are distinct except for whole milk and 2% milk, which overlap over the lower frequencies of the spectrum. We observe that the traces gradually shift downwards as the fat content of the samples decrease.

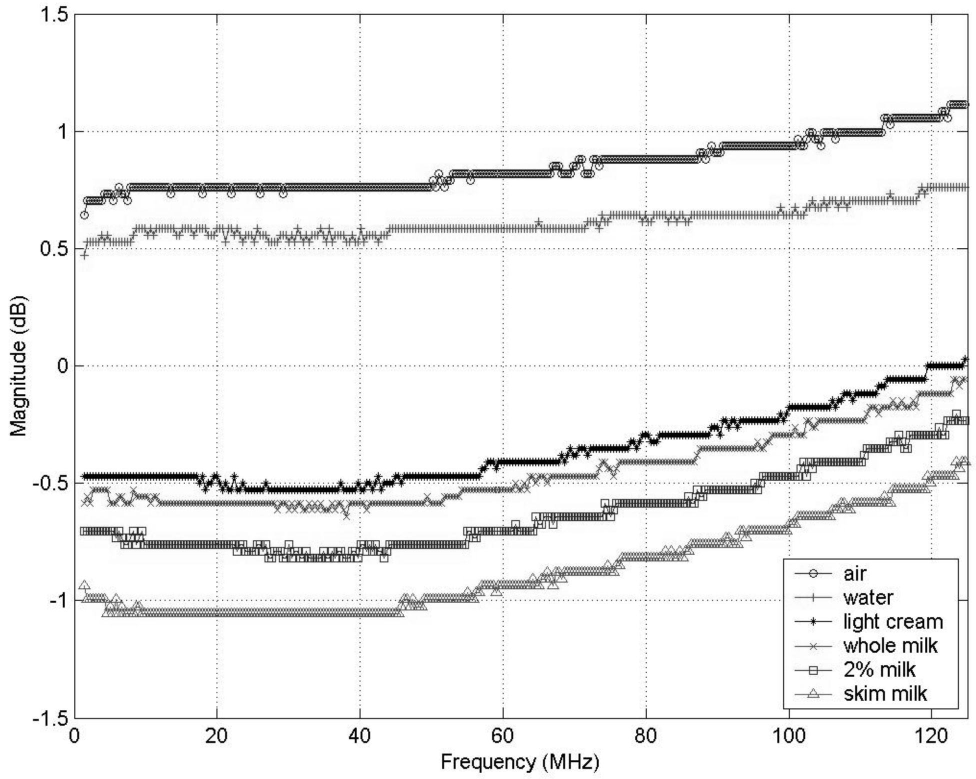


Figure 33. Magnitude Spectra for Various Liquids Analyzed

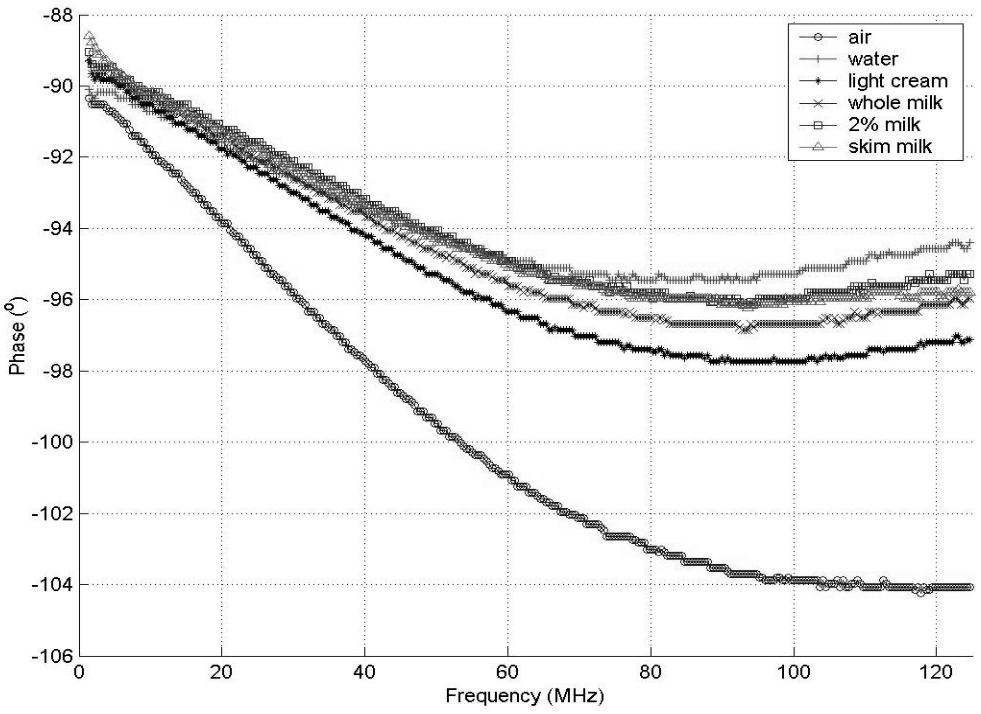


Figure 34. Phase Spectrum for Various Liquids Analyzed

Postal Mail Analysis Application

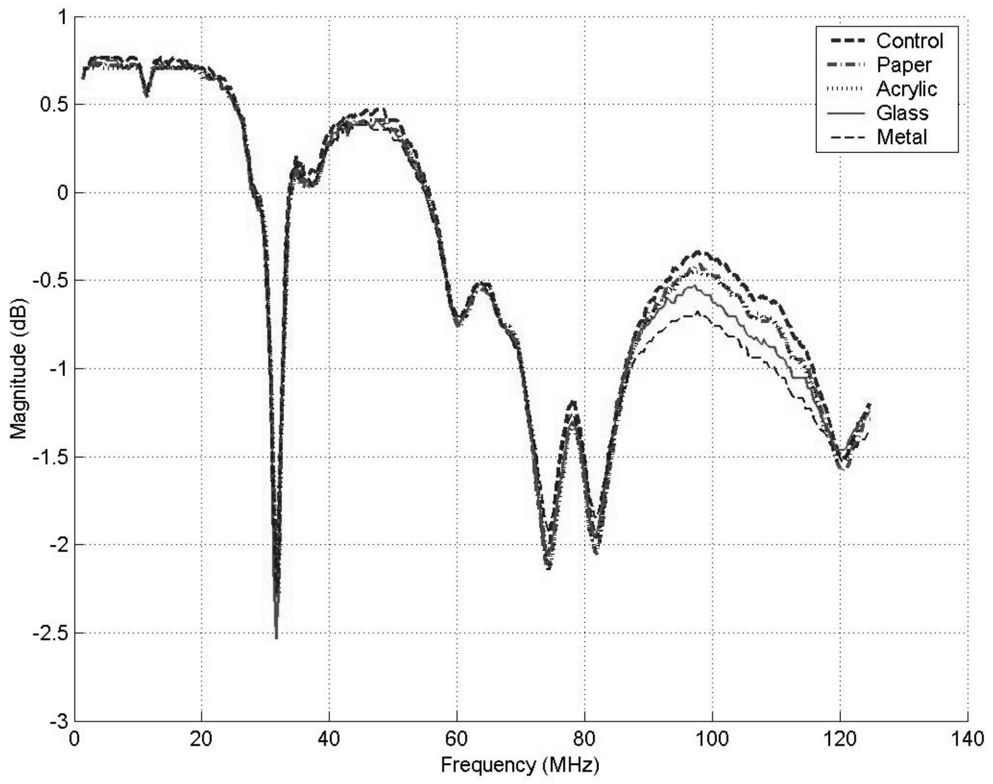


Figure 35. Magnitude Spectrum for Various Mail Materials Analyzed

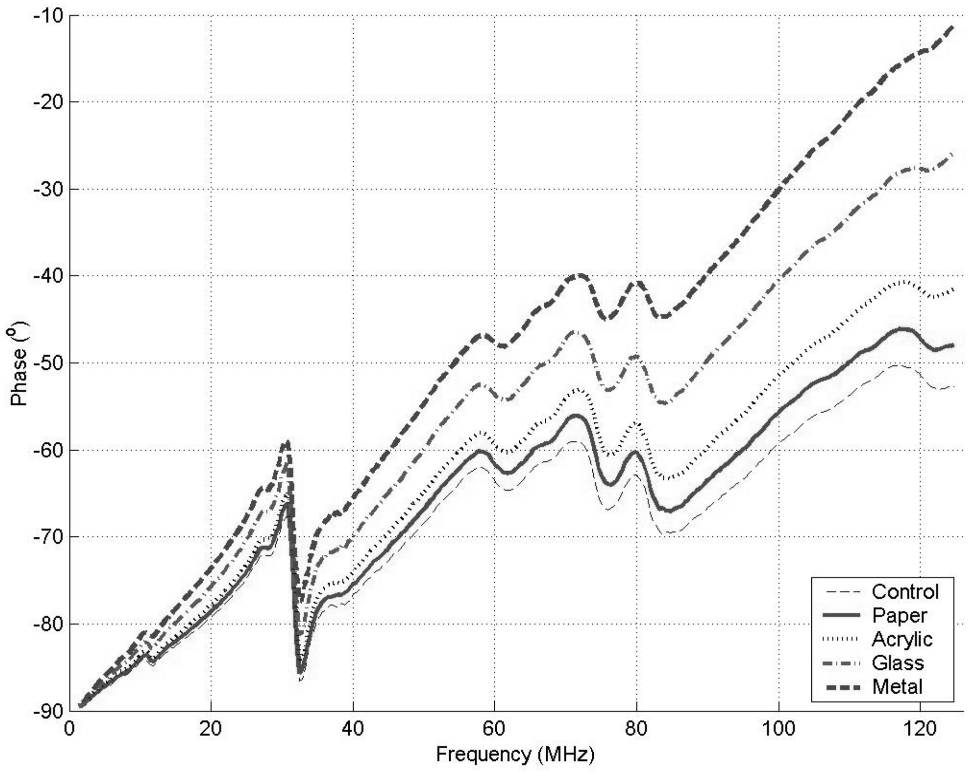


Figure 36. Phase Spectrum for Various Mail Materials Analyzed

Figures 35 and 36 show, respectively, the magnitude and phase plots of the various materials analyzed. We observe that there is hardly any difference in the magnitude spectrum of the various materials (except for a small region of the spectrum from 85 to 115 MHz), whereas for the phase spectrum, the curves are more distinct. This observation is consistent with what we would expect in theory, and we will elaborate on this in the next section. For the phase spectrum plot, while the measurement baseline is not perfectly even, we observe that the various materials give rise to distinct traces. Another observation is that the differences in dielectric properties of the materials are more pronounced at higher frequencies, as can be observed by the greater separation of the traces at the higher frequency portion of the spectrum.

Figures 37 and 38 show the baseline corrected magnitude and phase plots for the various mail materials analyzed. These plots compensate for the frequency variation of the probe, and reflect only the frequency dependence of the materials. From these two pots, we again observe that the magnitude plots for the various materials are very similar, and are have a constant value across the frequency range. The phase plots however, show significant differences in slopes and are generally linear, except towards the higher frequency end of the spectrum.

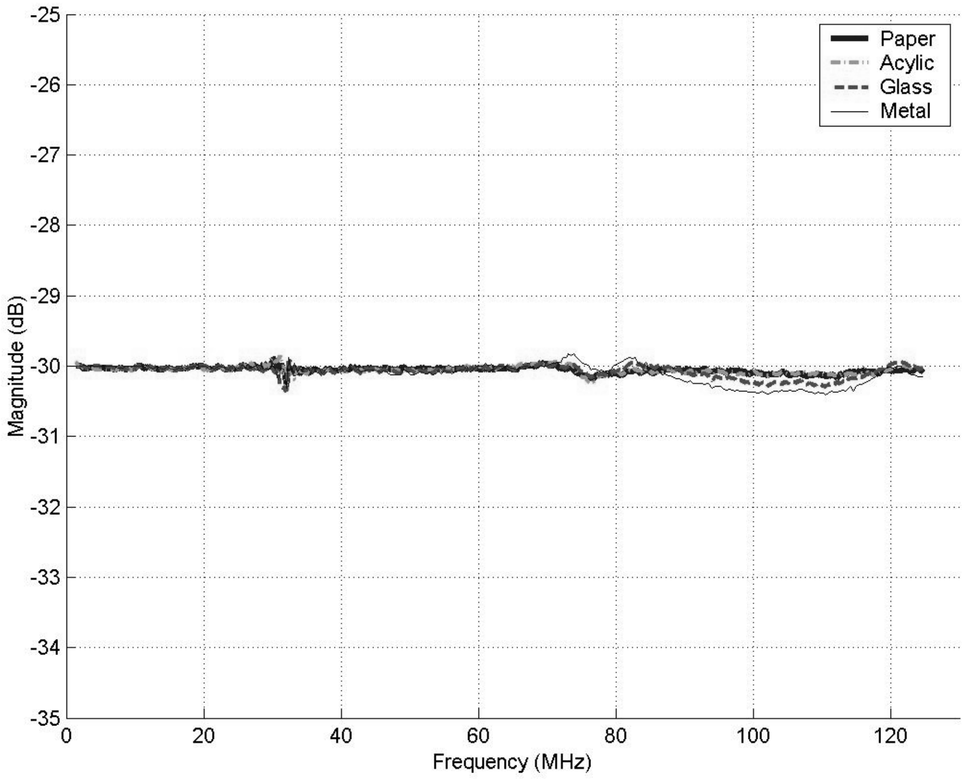


Figure 37 Baseline Corrected Magnitude Spectrum for Various Mail Materials Analyzed

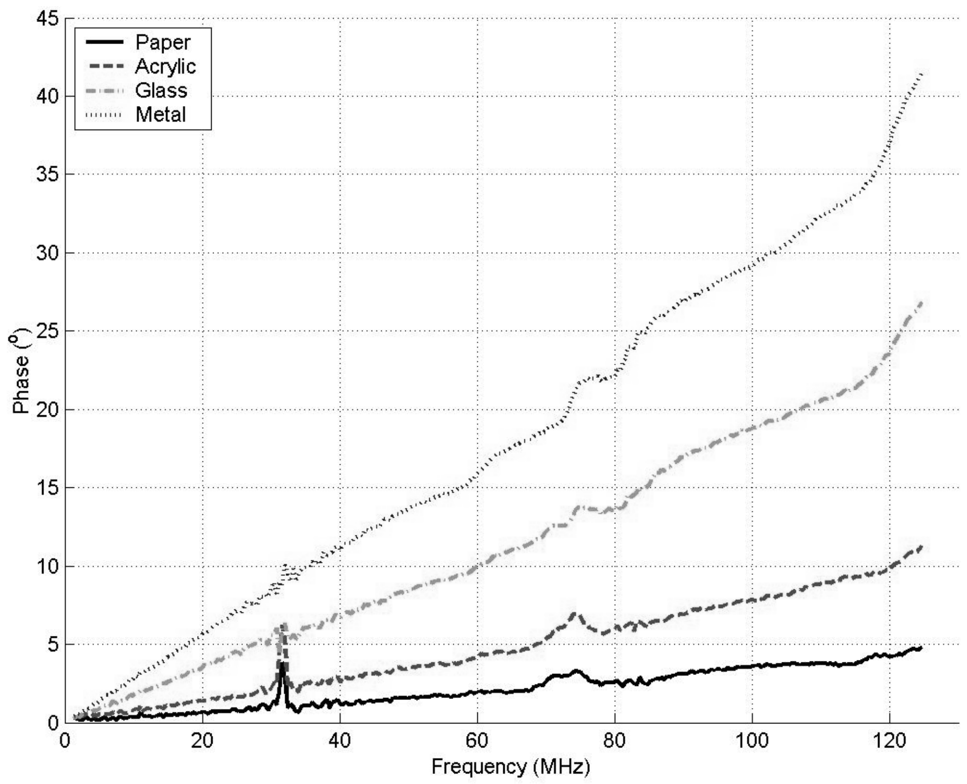


Figure 38. Baseline Corrected Phase Spectrum for Various Mail Materials Analyzed

6.5 Analysis of Results

Milk Quality Analysis Application

Differences were observed in both the phase and magnitude spectra for the different samples, and this is consistent with what we would expect in theory. For simplicity, we can express (6-9) as follows,

$$Y = c_1\omega\varepsilon - c_2\omega^3\varepsilon^2 + c_3\omega^4\varepsilon^{5/2} \tag{6-14}$$

Substituting (6-11) into the equation for the reflection coefficient (4-2), we can obtain the following expression,

$$\Gamma = \frac{-j(Z_oc_1\omega\varepsilon - Z_oc_2\omega^3\varepsilon^2) + 1 - Z_oc_3\omega^4\varepsilon^{5/2}}{j(Z_oc_1\omega\varepsilon - Z_oc_2\omega^3\varepsilon^2) + 1 + Z_oc_3\omega^4\varepsilon^{5/2}} \tag{6-15}$$

We see that both the phase and magnitude components of the reflection coefficient are functions of ε , thus accounting for the observed differences in both the phase and magnitude plots of the various milk samples. The theory and results indicate that information about a particular liquid, analyzed using an open-ended coaxial probe, is contained in both the phase and magnitude spectra. Thus, we can utilize both the spectra to characterize and distinguish between various milk samples. For example, while the phase spectra of 2% milk and skim milk overlap over a considerable portion of the frequency range, the two samples have distinct traces for their magnitude plots, allowing the two samples to be distinguished using a combination of both the plots. Similarly, could also be possible that two liquids might have the same magnitude plot and different phase spectrum plots.

The readings obtained indicate that the differences in chemical composition of the milk samples, as indicated in Table 2, result in the various samples having different phase and magnitude plots. This tells us that difference in the composition of milk affect its frequency dependent dielectric permittivity, and this agrees with the results obtained in previous work that has been done [6].

Postal Mail Analysis Application

We can obtain the approximate reflection coefficient for the end of the transmission line using equations (6-10) and (4-2). The refection coefficient for the capacitive probe can be expressed as,

$$\Gamma = \frac{d - j\omega\varepsilon A Z_o}{d + j\omega\varepsilon A Z_o} \qquad \text{(6-16)}$$

It can be shown that for this that,

$$|\Gamma| = 1 \qquad \text{(6-17)}$$

which is independent of ω and ε . This agrees with what we obtained for the baseline corrected magnitude spectrum (i.e. constant across the spectrum and same magnitude for the various materials). Looking at the phase component of the reflection coefficient we have,

$$\measuredangle\Gamma = \arctan\left(\frac{-2\omega\varepsilon A Z_o}{d^2 - Z_o{}^2\omega^2\varepsilon^2 A^2}\right) \qquad \text{(6-18)}$$

We see that $\measuredangle\Gamma$ varies as a function of ε which is consistent with what was observed, that is, the various materials, with differing dielectric constants, had distinct phase spectrum plots. From the theoretical values of the reflection coefficient for this probe, it can be inferred that the magnitude plot does not give us any information about a particular material being analyzed. Thus, information about the dielectric constant of a material is thus fully represented in the phase spectrum, and comparing phase spectra for different materials is sufficient for this application.

From the results obtained, we observe greatly differing slopes in the baseline corrected phase spectrum plots of the various materials. This suggests that comparing the phase spectra plots provides a viable way of characterizing and distinguishing between different materials.

6.6 Summary

The readings obtained by the low-cost network analyzer board indicate that dielectric analysis of materials is a powerful method of distinguishing and characterizing various materials. In addition, the instrument was successful in detecting slight differences in the dielectric constant, specifically in the milk analysis, indicating that it has the potential to be feasibly employed for various applications.

Thus, we conclude that this instrument provides a viable platform for the complex dielectric analysis of materials if we only require knowledge of the relative differences between samples and not their precise dielectric constant values. Indeed, most practical applications do not require such precise knowledge, allowing this instrument to have potentially diverse applications. The results obtained for the two applications that were studied, are extremely encouraging, and indicate that this instrument provides a feasible solution to distinguishing between milk samples of different quality, as well as various mail materials.

In addition to these two test cases, much broader applications exist for this instrument with regard to material analysis, and these are certainly worth exploring. Conceivably, frequency dependent dielectric constant analysis could be employed in a wide variety of commercial applications, in the home, and in the field for various biological and environmental applications. Specific examples of this would be detecting pollution in water bodies, and monitoring oil condition in motor vehicles. For applications such as these, deployment of such an instrument would first require characterizing the various liquids with the instrument. With this prior information, the instrument can then be calibrated to act as a method of sensing changes in the properties of materials, or as a tool for distinguishing between different materials.

While the employment of a real-time monitoring or testing system is beyond the scope of this thesis, the results obtained in the investigations indicate that such and implementation is possible. In addition, further refinements to the design of the instrument could potentially improve performance and enhance the ability of the instrument to distinguish between materials.

7. Conclusion

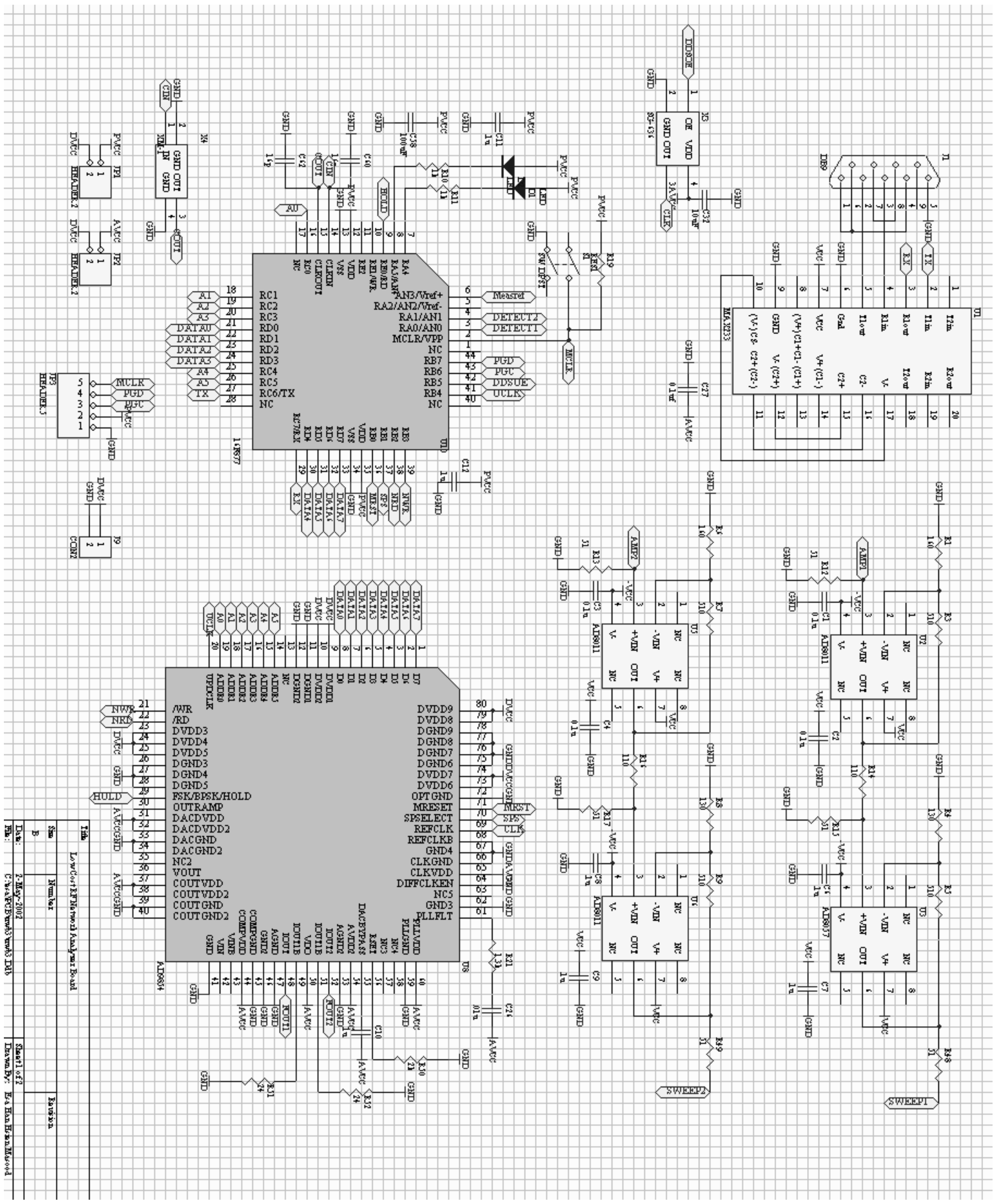
Traditionally, network analysis has been thought of as a tool for engineers to test and characterize electrical networks. While work has been done to illustrate the applicability of such a method of analysis to understanding things other than electrical devices, such as the dielectric properties of materials, biological samples, and environmental sensing, what is still lacking is a feasible way in which these applications can be carried out. Commercially available network analyzers are simply not suitable for wide deployment, due to the high cost and lack of portability of such instruments. This thesis has presented a viable solution for the wide deployment of network analyzers, providing a platform for wide deployment of network analyzers, making such a form of analysis accessible to people other than electrical engineers.

We have illustrated several applications that could lead to such instruments being deployed on a large scale. Low cost methods of analysis can certainly find numerous applications in wide ranging places, from rural villages in India, mail collection centers, and check out lanes in retail stores.

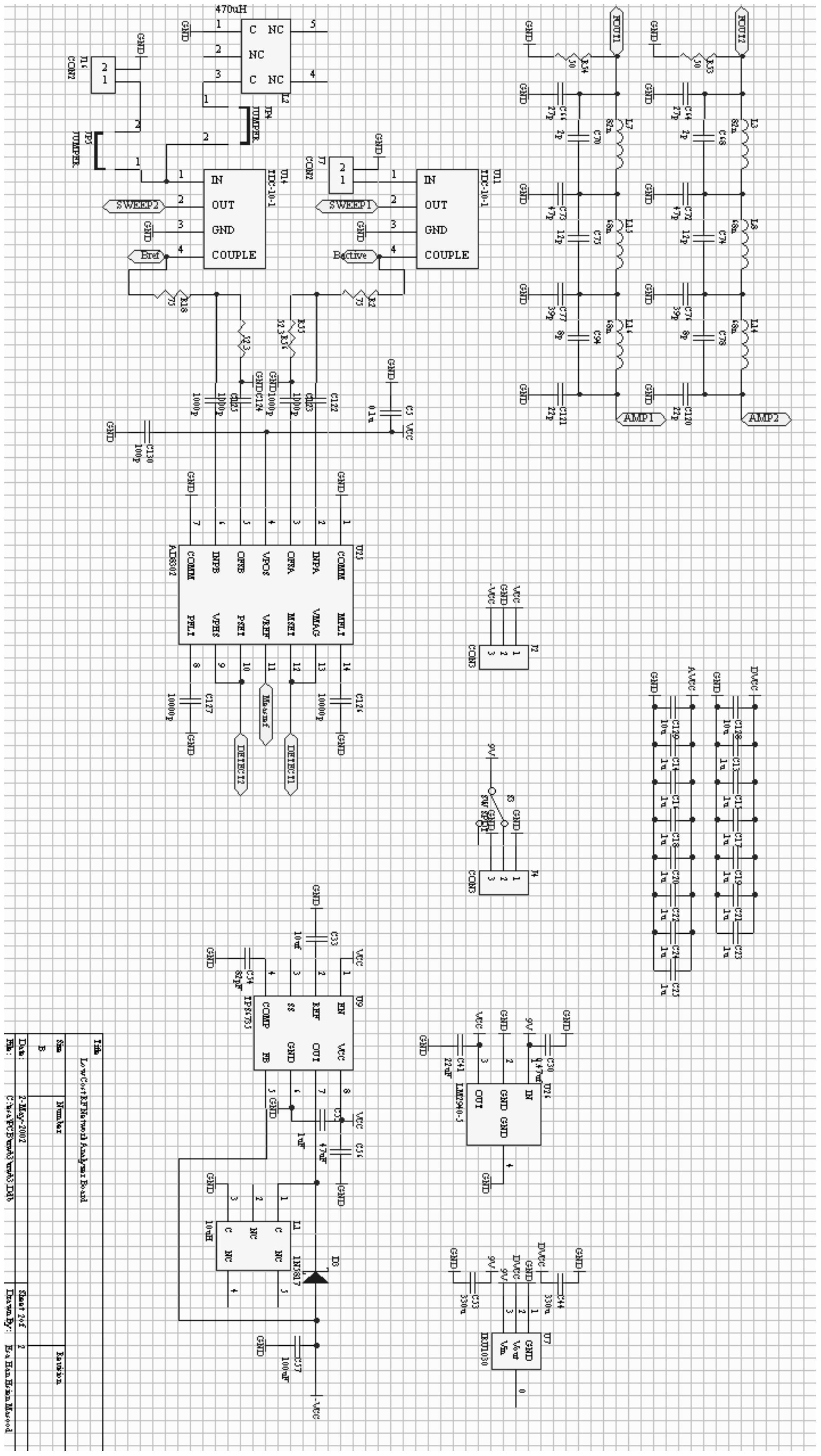
8. Appendix

8.1 Board Schematics

Sheet 1



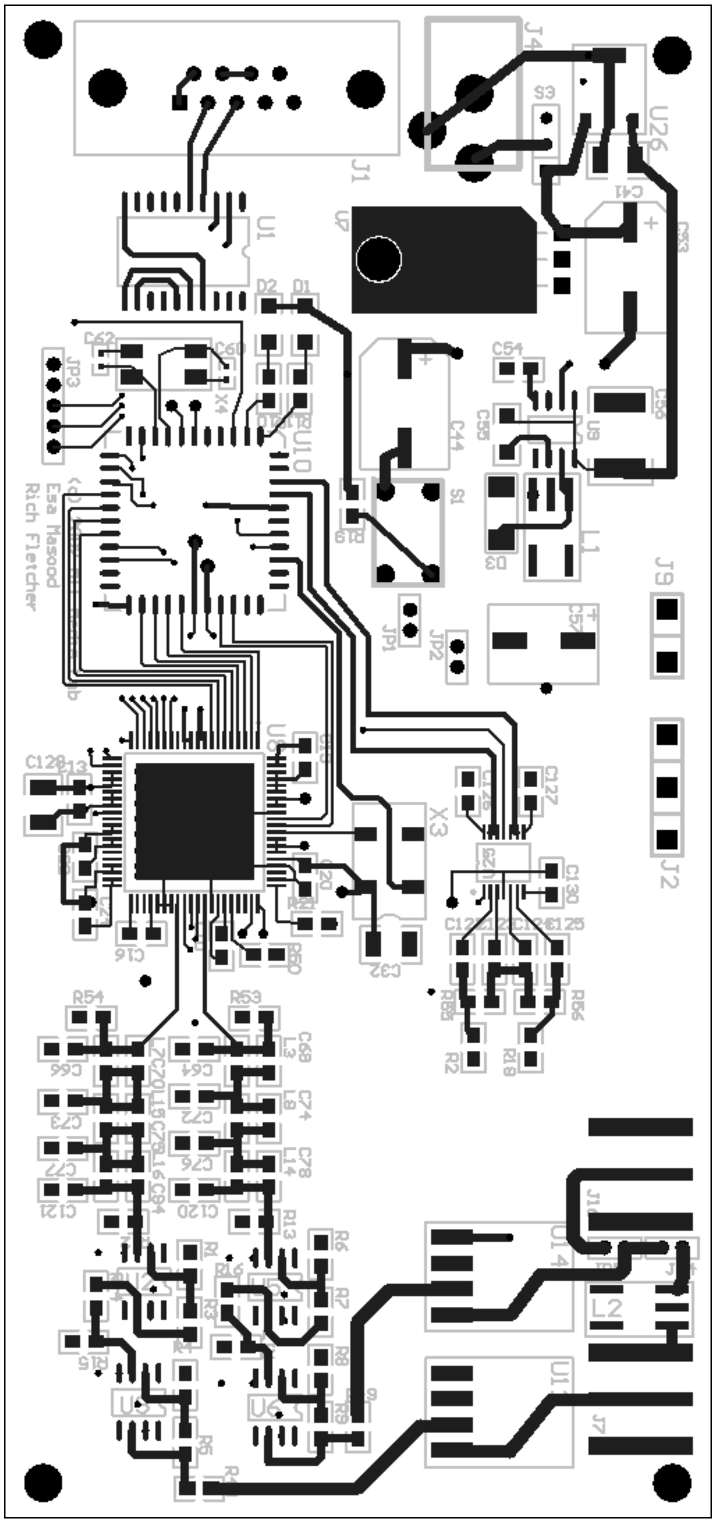
Sheet 2



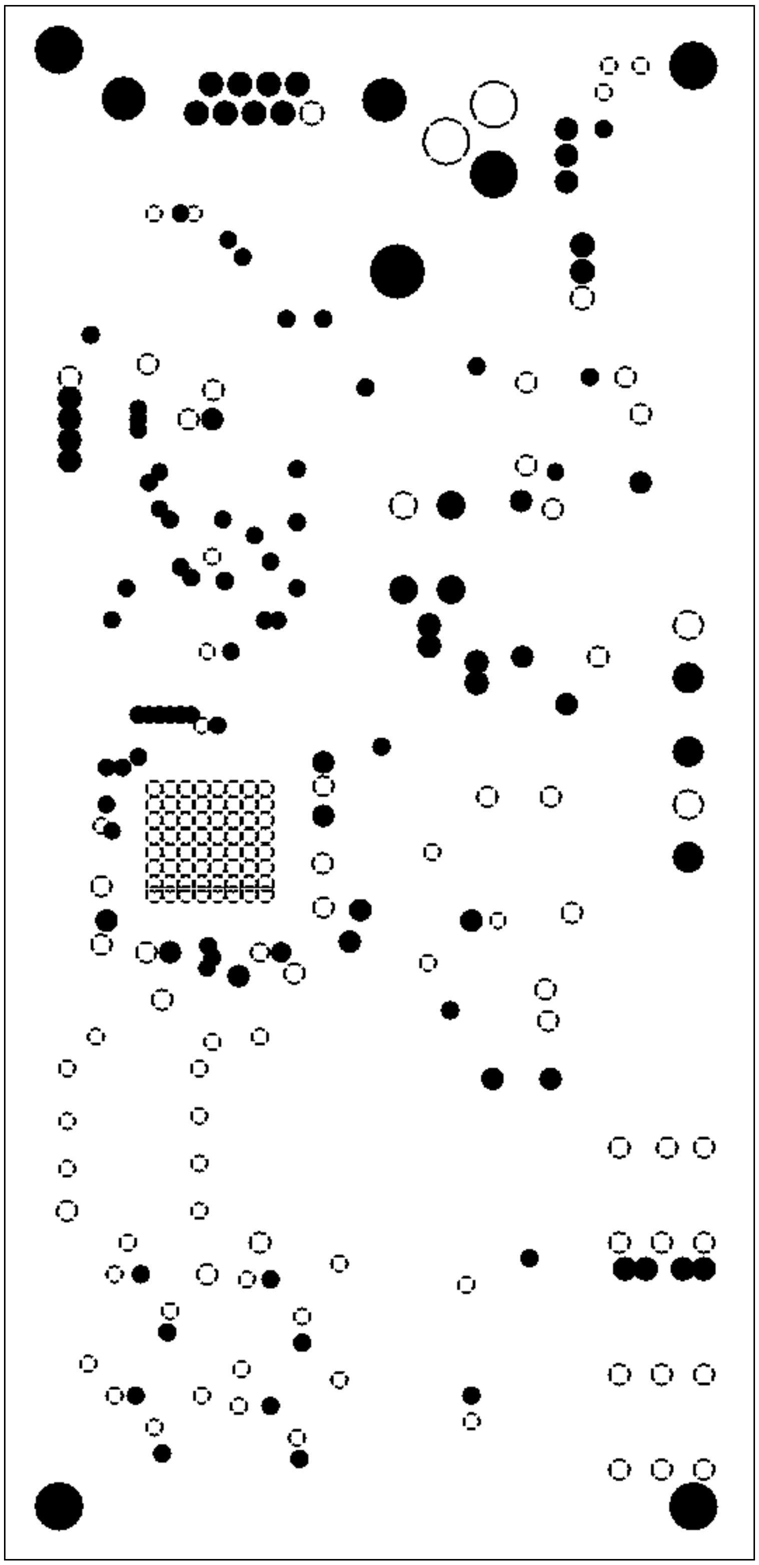
470uH
IN
OUT
GND
COUPLE
SWEEP1
SWEEP2
JUMPER
AD8302
TPS6735
LM2940-5
1N5817
AMP1
AMP2
VCC
GND
9V
DVCC
AVCC

8.2 Printed Circuit Board Layout

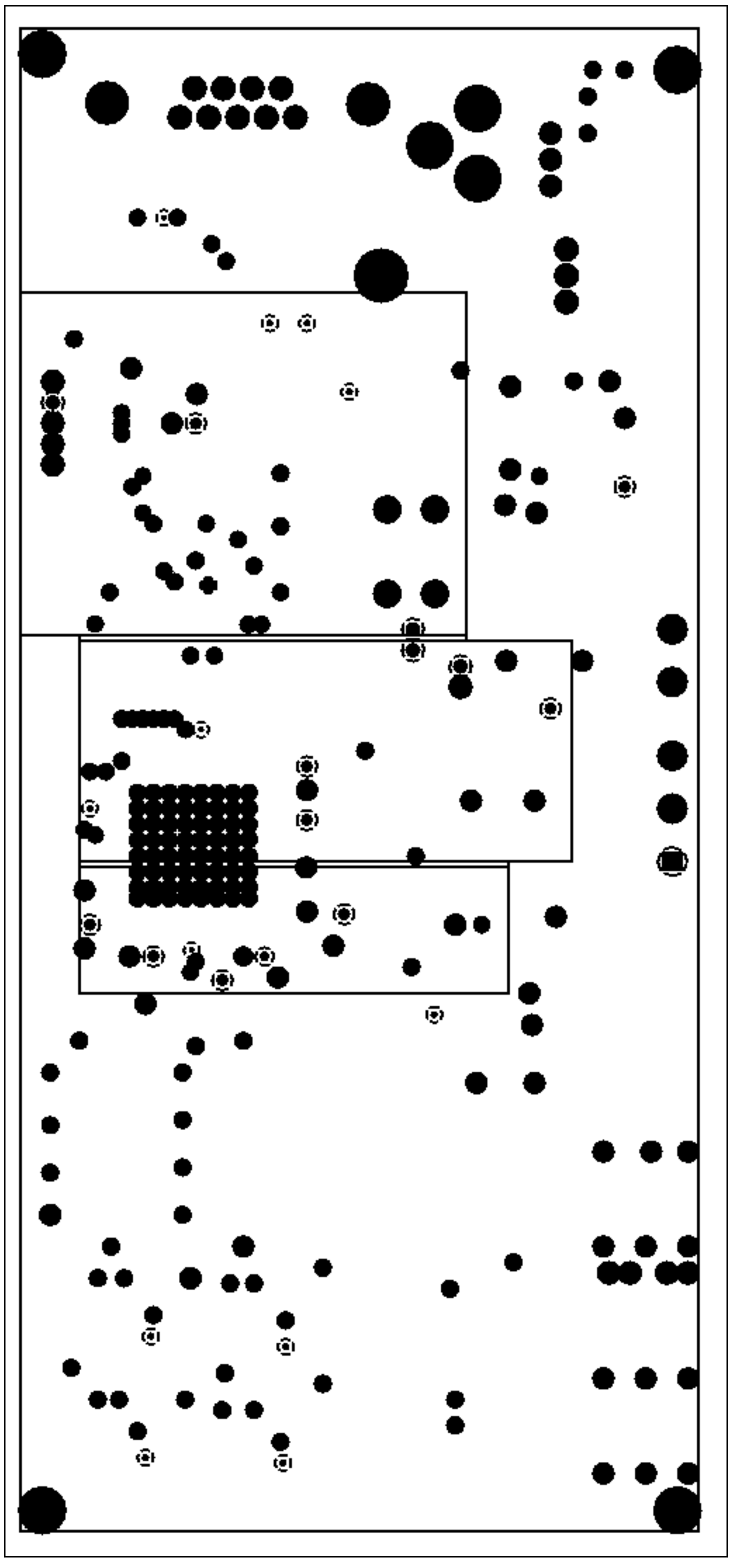
Top Layer



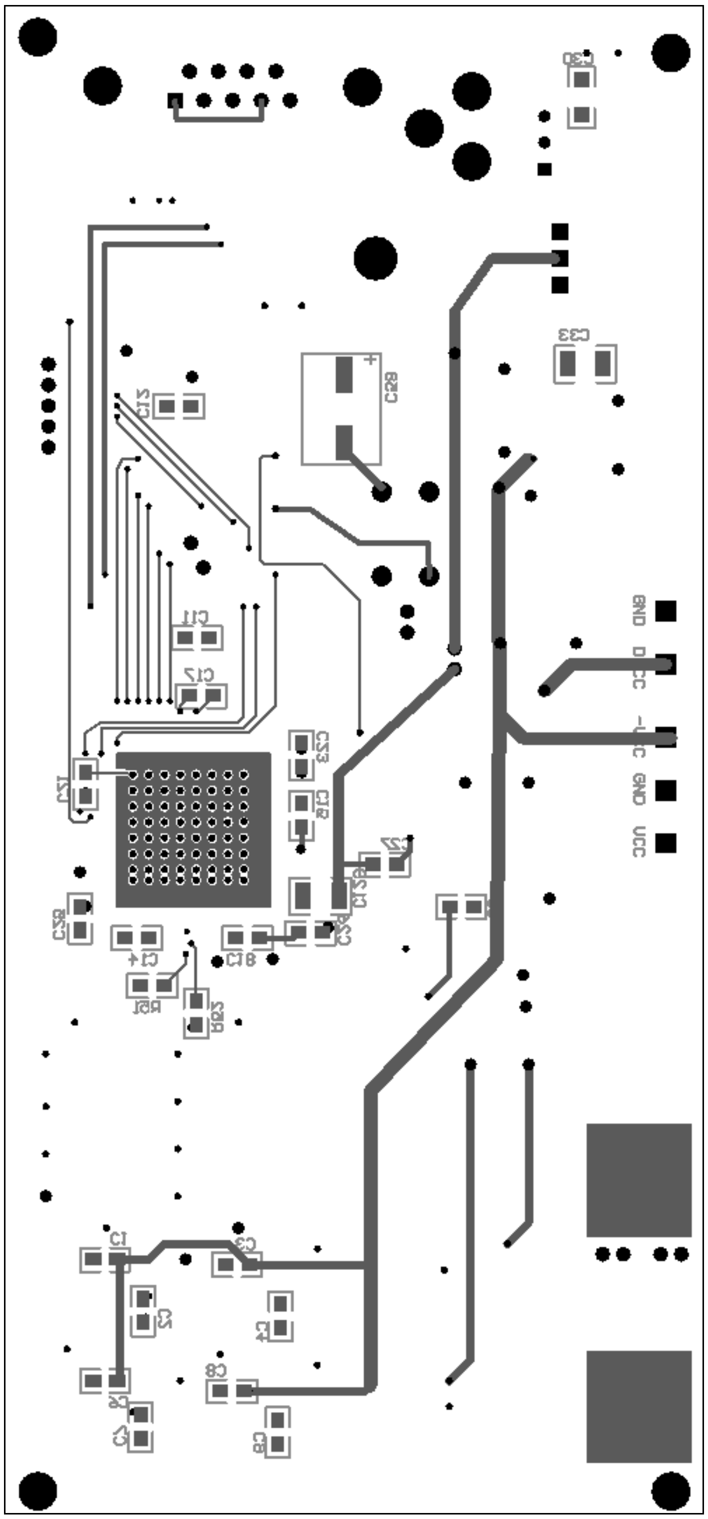
Internal Plane 1



Internal Plane 2



Bottom Layer



8.3 Microcontroller Code

The following is the C code used to program the PIC16F877 microcontroller.

```
#include "d:\progra~1\picc\examples\16f877.h"

#device *=16
#use delay (clock=20000000)
#fuses HS, NOWDT, PUT, NOPROTECT, NOBROWNOUT, NOLVP
#use rs232 (baud=115200, xmit=PIN_C6, rcv=PIN_C7)

/*
 * This is the parallel mode DDS program on the modified Standing Wave
Ratio Board.
 * It has been updated to work with filament-based communication.
 * Written by Olufemi Omojola, (c) 1999 MIT Media Laboratory.
 * August 1999.
 *
 * January 2000: Modified to comply with Tag Interface Specification,
Version 1.0
 *
 * September 2001: Modified by Esa Masood to work on broadband
tagreader board
 *
 * January 2002: Remodified by Esa Masood to work on network analyzer
board (ultrawideband v1.0)
 * current version is a working version for Netview1.1 (lab view based
interface program)
 *
*/


//New pin definitions

#define DDS_clk         PIN_B4
#define DDS_res         PIN_B0
#define DDS_sps         PIN_B1  /* Serial or parallel mode select -
high = parallel*/
#define DDS_nwr         PIN_B3  /* Write parallel data to registers */
#define DDS_nrd         PIN_B2  /* Read parallel data from registers */
#define DDS_hold  PIN_E0
#define DDS_oe          PIN_B5  /* unconnected pin in current rev,
KIV*/


#define LED_green       PIN_A4      /* LED goes on when driven low */
#define LED_red   PIN_A5

#define READPHASE PIN_A2


/* Constants for inlined assembler */
#define     STATUS           0x03
#byte DDS_DATA = 8           /* registers for D pins */
#byte   DDS_ADDR = 7         /* registers for C pins */
```

```
/* Constants for interrupt timing. */
#define INTS_PER_CYCLE  4     // (20000000/(4*256*256))
#define INTS_PER_RESET  0xFF

/* Constant for machine control synchronization */
#define CHUNK_SIZE      80    // Currently based on the packet size of
the filament.

/* Constants for tag reader state */
#define MACHINE_CONTROL_MODE  0x1
#define SWEEP_MODE            0x2
#define DATASTREAMING_MODE    0x4
#define POLL_MODE       0x8
#define PAUSE_MODE            0x10
#define ONBOARD_OUT_MODE      0x20
#define POSTPROCESS_MODE      0x40
#define POWER_DOWN_MODE       0x80

/* Prototypes */
void TimeDelay(long int delay);
long int str_to_int (char *str);
void newline();
//void Output(int f);
void calibrate();
void PrintVariables(void);
void init_freq(void);
separate void executeArgCommand(char p);
separate void Scan(void);
void checkArgCommand(char p);
void checkCommand(char p);
void wclk_pulse();
void fqud_pulse();
void DDS_send_parallel_freq(int *freq, int phase);
void reset_pulse();
void prog_pulse();
void DetermineTag();
char getState();
void sci_receive_isr();

void setEqualsReverse(int *source, int *dest);
void setEquals(int *source, int *dest);
long int str_to_int (char *str);
void print32hex(int *r);
void rotate_left32(int i);
void mul32(int bits);
void add32(void);
void str_to_int32 (char *str, int *result);
void div32(void);
void getPhase(int *freq, int *result);
void inc(int *source, int *increment);
int lessThan(int *arg1, int *arg2);

// Character used for command definition
char c, comm_char;

// Flags used for flow control in various parts of the program
```

```
static short int datastream = TRUE;

// Flag bits to store the operating parameters
static short int pollmode = FALSE, postprocess = FALSE, sscan = FALSE;
static short int sweep = TRUE, pause = FALSE, poll = FALSE;
static short int power = TRUE;

// Flag bit to deal with operator type
static short int mc = TRUE;

// Flag for onboard communication channels
static short int onboard_out = FALSE;

// Flag bits for command handling
static short int comm = FALSE;

/*
 * Variable Description:
 *
 * All variables ending with "startf" are the starting frequencies for
 * a sweep operation.
 * All variables ending with "stopf" are the ending frequencies for a
 * sweep operation.
 * All variables ending with "stepf" are the step or increment
frequencies
 * for a sweep operation.
 *
 * All variables ending with "singlef" are the frequencies used in a
 * single frequency operation.
 *
 * All variables ending with "thresh" are the threshold levels used
 * to detect a valid tag response (as distinct from background noise)
 * when the system is in real-imaginary mode.
 *
 * All variables ending in "mmin" are the variables used to store the
 * information about the frequency with the lowest magnitude.
 *
 * All variables ending in "_d" are the variables for use in
 * counting out the delays.
 */
byte int_count, reset_count;
static int d1, com_index, mval, mmin, thresh, tag_id, comm_step;
static int f_mmin[4];
static int startf[4], stopf[4], stepf[4], singlef[4];
static int rs1[4], rs2[4];
static int rd[8];
static long int t_delay_d;
static int tag_ids[26];
static int history[4];
static int id[4];
static char command_buf[12];

main()
{
  // Declarations and initialization code
  int i = 0;
  int mode = 0;
  //**test array
  int f1[6], addr;

  pause = FALSE;        // Set the stop variable

  c = (char)0;
  com_index = 0, comm_step = 0;


  // Initialize the start, stop, step, single, and base frequencies
  init_freq();

  // Initialize the A/D reference channel
  //setup_port_a(ANALOG_RA3_REF);  /* Used to be ANALOG_RA3_REF */


  setup_adc(ADC_CLOCK_INTERNAL);
  setup_adc_ports(ANALOG_RA3_REF);
 // setup_adc_ports(RA0_RA1_ANALOG_RA3_REF);

  // Turn the LED on
  output_low(LED_green);

  // Initialize the DDS module
  DDS_DATA = 0;
  DDS_ADDR = 0;
  output_high(DDS_nwr);  // active low pins
  output_high(DDS_nrd);

  // reset the DDS
  reset_pulse();

  //printf("Hello");

  // enable it for parallel mode operation
  output_high(DDS_sps);
  set_tris_d(0);
  set_tris_c(0xc0);


  // set clock multiplier (x6)
  DDS_ADDR = 0x1e;
  DDS_DATA = 0x46;    //multiplier set to X6, pll range >200mhz, pll
bypass bit = 0
  prog_pulse();

  // power down unused stages
  DDS_ADDR = 0x1d;
  DDS_DATA = 0x10;   // only comparator powered down
  prog_pulse();

  // Set dds to external update clock mode
  DDS_ADDR = 0x1f;
  DDS_DATA = 0x00;
  prog_pulse();

  // Disable shaped keying
  DDS_ADDR = 0x20;
  DDS_DATA = 0x00;
  prog_pulse();


  // Calibrate the sweep system and set the threshold value
  //calibrate();

  // Enable the crystal
  //output_high(DDS_oe); - Not enabled in current version

  // Initialize the id variables
  for(i=0;i<4;i++)
    history[i] = 0;
  tag_id = 0;

  int_count=INTS_PER_CYCLE;
  reset_count = INTS_PER_RESET;
  set_rtcc(0);
  setup_counters( RTCC_INTERNAL, RTCC_DIV_256);
  enable_interrupts(RTCC_ZERO);
  enable_interrupts(INT_RDA);
  enable_interrupts(GLOBAL);

  while(1){
    if(!mc){
      while(!pollmode || poll){
      // perform a single operation, either a sweep or a single
frequency point
      if(!pause) Scan();

      if (poll)
        poll = FALSE;
      }
    }
  }
}

// This function is called every time the RTCC (timer0) overflows (255-
>0).
// For this program this is apx 76 times per second. The count is 8, so
this
// fires a little under 10 times a second
#int_rtcc
clock_isr()
{

  if(--int_count==0) {
    sscan = TRUE;
    if(!pause) {
      if(!pollmode || poll)
      Scan();

      if(poll) poll = false;
    }
    sscan = FALSE;
    int_count=INTS_PER_CYCLE;
  }

  if(--reset_count==0){
    // reset the DDS
    reset_pulse();

    // enable it for parallel mode operation
    output_high(DDS_sps);

    reset_count=INTS_PER_RESET;
  }
  if(kbhit())
    sci_receive_isr();
}

#int_rda
sci_receive_isr()
{
  // Declarations
  char k;
  k = getc();

  if(mc) putc(2);

  if(k == '\r'){
    // Ignore this: terminal emulation program, probably
  }else if(k == 'P'){
    // This freezes the machine, and guarantees that
    // it is in machine control mode and paused for external control
    mc = true;
    pause = true;
    comm = false;
    com_index = 0;

    // This identifies it as a SWR board: the 2nd tag reader developed
    // at ML that works with the interface
    printf("ML02");
  }else if(!comm){
    // Check if it is the start of a new command
    if(k == 'g') c = k;
    else checkCommand(k);
    // check if it is a newline
  }else if(k == '\n'){
    if(com_index < 11)
      command_buf[com_index] = '\0';
    com_index = 0;
    executeArgCommand(comm_char);
    // It is an argument. store it in the command buffer
  }else{
    if(com_index < 11){
      command_buf[com_index] = k;
      ++com_index;
    }else{
      comm = false;
      com_index = 0;
    }
  }
  if(mc) putc(3);
}

//Pulse NRD pin on DDS to program chip
void prog_pulse()
{
  delay_us(1); //8ns delay, address setup time
  output_low(DDS_nwr);
  delay_us(1);
  output_high(DDS_nwr);
}

// Update the DDS frequency register
void fqud_pulse()   /* frequency update pulse for DDS3 */
{
  output_low(DDS_clk);
  output_high(DDS_clk);
  delay_us(1);
  output_low(DDS_clk);
}

// Pulse the reset line on the DDS
void reset_pulse()
{
  output_high(DDS_res);
  delay_ms(5);
  output_low(DDS_res);
}

// Sends the frequency word to the DDS
void DDS_send_parallel_freq(int *freq, int phase)
{
  // Declarations
  int i, j;
  int addr = 0;

  // Use default phase setting = 0 degrees
  // Send the phase word
  //if (phase==0) j = 0;
  //else if (phase==90) j = 64;

  //DDS_ADDR = addr;
  //DDS_DATA = j;
  //prog_pulse();

  addr = 0x04;
  DDS_ADDR = addr;

  // Send the bytes
  for (i=4; i>2; --i)
  {
    DDS_DATA = freq[i-1];
    prog_pulse();
    addr = ++addr;
    DDS_ADDR = addr;

  }
  fqud_pulse();

  DDS_DATA = 0;
  DDS_ADDR = 0;
}

/*
 * Function:
 *    checkCommand()
 *
 * Description:
 *    Performs the commands sent over the serial port.
 *
 * Arguments:
 *    char p: the character denoting the command to be executed.
 *
 * Return Value:
 *    None.
 *
 */
void checkCommand(char p)
{
  // Declarations
  char c2;
  int i;
  long int n = 0;
  short int temp;

  // Control loop: uses a switch statement to select the desired
  // command. Consider using a more intuitive user interface,
  // such as a command menu.
  switch(p){
    // These are the commands that don't have any arguments
  case 'd':
    datastream = !datastream;
    break;
  case 'o':
    onboard_out = !onboard_out;
    break;
  case 'e':
    pollmode = !pollmode;
    break;
  case 'p':
    calibrate();
    break;
  case 'w':
    sweep = !sweep;
    break;
  case 'x':
    pause = !pause;
    break;
  case 'z':
    poll = TRUE;
    break;
  case 's':
    // This queries the board for the values of the variables.
    // Put the sweep state on the screen
    if(mc){
      putc((byte)4);
      putc((byte)1);
      putc((byte)0);
      putc((byte)1);
      putc(getState());
    }else{
      printf("4101\r\n");
      printf("sweep: ");
      if(sweep) putc('1');
      else putc('0');
      newline();

      printf("pollmode: ");
      if(pollmode) putc('1');
      else putc('0');
      newline();

      printf("data stream: ");
      if(datastream) putc('1');
      else putc('0');
      newline();

      printf("postprocess: ");
      if(postprocess) putc('1');
      else putc('0');
      newline();

      printf("machinecontrol: ");
      if(mc) putc('1');
      else putc('0');
      newline();

      printf("paused: ");
      if(pause) putc('1');
      else putc('0');
      newline();

      printf("powerdown: ");
      if(power) putc('1');
      else putc('0');
      newline();
    }

    // put the properties out of the port
    PrintVariables();
  case 'n':
    // Returns the 32-bit identifier
    putc(id[0]);
    putc(id[1]);
    putc(id[2]);
    putc(id[3]);
    break;
  case 'y':
    init_freq();
    break;
  case 'M':
    mc = !mc;
    if(mc) enable_interrupts(RTCC_ZERO);
    else disable_interrupts(RTCC_ZERO);
    break;
  case 'v':
    power = !power;
    if(power)
      output_high(DDS_oe);
    else output_low(DDS_oe);
    break;
  case 'D':
    putc((byte)4);
    putc((byte)1);
    putc((byte)0);
    putc((byte)1);
    break;
  default:
    // These are commands that take arguments
    checkArgCommand(p);
    return;
  }
}

void checkArgCommand(char p)
{
  // Declarations

  // Switch on the defined commands
  switch(p){
  case 'k':
    // This sets the amount of wait time after switching frequency but
before sampling to allow
    // the DDS output to stabilize. The input should be the value in
units of 10-microsecond
    // intervals.
    break;
  case 't':
    // 't' sets the threshold value
    break;
  case 'a':
    // 'a' sets the resonant start frequency.
    break;
  case 'b':
    // 'b' sets the resonant stop frequency.
    break;
  case 'c':
    // 'c' sets the resonant step frequency.
    break;
  case 'f':
    // 'f' sets the resonant single frequency.
    break;
  case 'u':
    // 'u' sets the unique identifier.
    break;
  default:
    return;
  }
  comm_char = p;
  com_index = 0;
  comm = TRUE;
}

separate void executeArgCommand(char p)
{
  // Declarations
  int f[4];
  int v1, v2;

  // The following are the commands that are expecting arguments.
  switch(p){
  case 'k':
    // This sets the amount of wait time after switching frequency but
before sampling to allow
    // the DDS output to stabilize. The input should be the value in
units of 10-microsecond
    // intervals.
    t_delay_d = str_to_int(command_buf);
    break;
  case 't':
    // 't' sets the threshold value
    thresh = str_to_int(command_buf);
    break;
  case 'u':
    // 'u' sets the unique identifier
    for(d1=0;d1<4;d1++)
      id[d1] = command_buf[d1];
    break;
  default:
    str_to_int32(command_buf, f);
    switch(p){
    case 'a':
      // 'a' sets the resonant start frequency.
      getPhase(f, startf);
      break;
    case 'b':
      // 'b' sets the resonant stop frequency.
      getPhase(f, stopf);
      break;
    case 'c':
      // 'c' sets the resonant step frequency.
      getPhase(f, stepf);
      break;
    case 'f':
      // 'f' sets the resonant single frequency.
      getPhase(f, singlef);
      break;
    default:
      break;
    }
  }
  comm_step = 0;
  comm = FALSE;
}
```

```
char getState()
{
  // Declarations
  char ans = 0;

  if(mc) ans |= MACHINE_CONTROL_MODE;
  if(postprocess) ans |= POSTPROCESS_MODE;
  if(sweep) ans |= SWEEP_MODE;
  if(datastream) ans |= DATASTREAMING_MODE;
  if(pollmode) ans |= POLL_MODE;
  if(pause) ans |= PAUSE_MODE;
  if(onboard_out) ans |= ONBOARD_OUT_MODE;
  //  if(power) ans |= POWER_ON_MODE;

  return ans;
}

/*
 * Function:
 *    TimeDelay()
 *
 * Description:
 *    Executes a delay equal to the (argument * 10) microseconds.
 *
 * Arguments:
 *    long int delay: the number of 10-microsecond intervals to
 *                      delay for.
 *
 * Return Value:
 *      None.
 *
 */
void TimeDelay(long int delay)
{
  long int i = 0;

  for(;i < delay;i++)
    delay_us(5);
}

/* input = string;
   output = long integer; (something like atoi())  */
long int str_to_int (char *str)
{
  long int nc = 0; int i = 0;
  do {
    if (*(str+i) == '\0') return nc;
    else nc *= 10;
    nc += *(str+i) - '0';
    ++i;
  } while (TRUE);
}

/*
 * Function:
 *    newline()
 *
 * Description:
 *    Prints out of the serial port a carriage return and a
 *      line feed.
 *
 * Arguments:
 *    None.
 *
 * Return Values:
 *      None.
 *
 */
void newline()
{
  printf("\r\n");
}

/*
 * Function:
 *    calibrate()
 *
 * Description:
 *    Sets the threshhold value to above the ambient noise
 *      level.
 *
 * Arguments:
 *    None.
 *
 * Return Value:
 *    None.
 */
void calibrate()
{
  // Declarations
  long int d1max = 0;
  int f[4];
  int mval = 0;
  int b;

  // Does a sweep and keeps a record of the minimum ambient signal.
       // At the end, the threshold value is set to the lowest
       // recorded ambient response.
  thresh = 0;
  mmin = 0xFF;


  // Initialize the loop
  setEquals(startf, f);

  // Wait for the frequency to stabilize
  while(lessThan(f, stopf)){
    // set the frequency of the DDS
    DDS_send_parallel_freq(f, 0);

    // Wait for the signal to propagate
    TimeDelay(t_delay_d);

    // Measure the ambient response
    mval = read_adc();

    // Check for the minimum magnitude
    if (mval < mmin){
      mmin = mval;
      setEquals(f, f_mmin);
    }

    inc(f, stepf);
  }


  // Set the threshold
  thresh = mmin - 1;

  //use mmin-1 is for 5V reference
  //use mmin-5 for smaller reference voltages


  // print the threshold if necessary
  if(!mc){
    printf("version 072499 - %x\r\n", thresh);
    if(postprocess) printf("tr0");
  }
}

/*
 * Function:
 *    PrintVariables()
 *
 * Description:
 *    Prints out to the user the values of the system operating
parameters.
 *
 * Arguments:
 *    None.
 *
 * Return Value:
 *    None.
 *
 */
void PrintVariables(void)
{
  // Declarations

  // Print out: spits the data out as
  // start stop step single threshold
  newline();
  printf("startf: "); print32hex(startf); newline();
  printf("stopf: "); print32hex(stopf); newline();
  printf("stepf: "); print32hex(stepf); newline();
  printf("singlef: "); print32hex(singlef); newline();
  printf("thresh: %x", thresh); newline();
  printf("mmin: %x at", mmin); newline();
  printf("f_mmin: "); print32hex(f_mmin); newline();
  printf("t_delay_d: %4lx\r\n", t_delay_d);
  newline();
```

```
}

/*
 * Function:
 *    init_freq()
 *
 * Description:
 *    Initializes all the frequency variables. The relevant variables
 *      are: startf, stopf, stepf, singlef, thresh.
 *
 * Arguments:
 *    None.
 *
 * Return Value:
 *    None.
 */
void init_freq(void)
{
  // No declarations. This just resets all the above variables to known
  // values. The variables are initialized in hex. The respective
values
  // chosen are listed below in LSB-first format, in Hex:
  // startf = 0xB851eb80000 (5.4 MHz), stopf = 0x1C5f92c50000 (13.3
MHz),
  // singlef = 0x144444440000 (9.5 MHz), stepf = 0x369D030000 (100 kHz)
  // thresh = 0, mmin = 0xFF0000
  // clear the thresh holds
  thresh = 0;
  mmin = 0;
#asm
  clrf f_mmin[0]
    clrf f_mmin[1]
    clrf f_mmin[2]
    clrf f_mmin[3]
    clrf stepf[3]
#endasm
    /* conservative defaults are delay = 20 us, pwm period = 0x3F */
    t_delay_d = 20;

  // Set the values
  // Set the explicit byte values
// for 1 MHz
  startf[0] = 0x00;
  startf[1] = 0x00;
  startf[2] = 0xda;
  startf[3] = 0x00;

//for 60 Mhz
  singlef[0] = 0x00;
  singlef[1] = 0x00;
  singlef[2] = 0x33;
  singlef[3] = 0x33;

//for 125 MHz
  stopf[0] = 0x00;
  stopf[1] = 0x00;
  stopf[2] = 0xab;
  stopf[3] = 0x6a;

  //for 0.4 MHz
  stepf[0] = 0x00;
  stepf[1] = 0x00;
  stepf[2] = 0x58;
  stepf[3] = 0;


//for .5 MHz
//  stepf[0] = 0x0e;
//  stepf[1] = 0x74;
//  stepf[2] = 0xda;
//  stepf[3] = 0x00;


  // Bin assignments
  tag_ids[0] = 1;
  tag_ids[1] = 4;
  tag_ids[2] = 6;
  tag_ids[3] = 8;
  tag_ids[4] = 10;
  tag_ids[5] = 12;
  tag_ids[6] = 14;
  tag_ids[7] = 16;
  tag_ids[8] = 18;
  tag_ids[9] = 20;
  tag_ids[10] = 22;
  tag_ids[11] = 24;
  tag_ids[12] = 26;
  tag_ids[13] = 28;
  tag_ids[14] = 30;
  tag_ids[15] = 32;
  tag_ids[16] = 35;
  tag_ids[17] = 37;
  tag_ids[18] = 40;
  tag_ids[19] = 45;
  tag_ids[20] = 47;
  tag_ids[21] = 50;
  tag_ids[22] = 52;
  tag_ids[23] = 56;
  tag_ids[24] = 66;
  tag_ids[25] = 255;
}


/*
 * Function:
 *    Scan()
 *
 * Description:
 *    Performs the tag location function.
 *
 * Arguments:
 *    None.
 *
 * Return Value:
```

```
 *    None.
 *
 */
separate void Scan(void)
{
  // Declarations
  long int f[4], cnt = 0;
  int cur_id = 1, found = 0, nt;
  long mval, mval1, mval2;
  int b;

  // handle the case where we aren't in sweep mode
  if(!sweep){
    setEquals(singlef, f);

    // set the frequency of VCO1: This is done by setting the duty
cycle of PWM 1
    DDS_send_parallel_freq(f, 0);

    // Wait for the signal to propagate
    TimeDelay(t_delay_d);

    // Check for the tag response

    mval = read_adc();

    // Write out to the serial port if a tag was found
    if(mval < nt || datastream){
      if(mc) putc(mval);
      else{
      putc('*');
      print32hex(f);
      putc(' ');

      // Put output modifiers here
      printf("%x\r\n", mval);
      //printf("%u %x\r\n", mval, cnt, nt);
      }
    }
  }else{
    // This is the code for a sweep
    // Initialize the loop
    // Set the frequency to be the current startf frequency word.
    setEquals(startf, f);

    // set the frequency of the DDS
    DDS_send_parallel_freq(f, 0);

    // Do the sweep loop
    while(lessThan(f, stopf)){
      if(mc && cnt == 0){
      putc(0);
      cnt++;
      }

      // set the frequency of the DDS
      DDS_send_parallel_freq(f, 0);
```

```
   // Wait for the signal to propagate
//   TimeDelay(t_delay_d);

   // Check for the tag response
   set_adc_channel(0);     // set pic to read phase difference
   delay_us(20);
   mval1 = read_adc();
   set_adc_channel(1);     // set pic to read mad difference
   delay_us(20);
   mval2 = read_adc();

   // Note if a response was obtained
   nt = thresh + (0.1 * (71 - cnt));
   if(!postprocess){
   if(mval < nt || datastream){
     if(mc) putc(mval);
     else{
       putc('*');
       print32hex(f);
       putc(' ');

       // Put output modifiers here
       printf("%Lx %Lx\r\n", mval1, mval2);
       //printf("%u %x\r\n", mval, cnt, nt);
     }
   }
   }else if(mval < nt)
   if(cnt == tag_ids[cur_id-1])
     found = cur_id;

   if(c == 'g'){
   c = (char)0;
   setEquals(startf, f);
   continue;
   }

   // Do these checks to avoid waiting till the end of big loops
   if(!sscan){
   if(lessThan(f, startf))
     setEquals(startf, f);
   if(lessThan(stopf, f))
     setEquals(stopf, f);
   }

   // If the command switched to no sweep mode
   if (!sweep) break;

   // If we switched to pollmode, stop
   if(pollmode && !poll) break;

   if(mc) ++cnt;
   else{
   ++cnt;

   // handle increment for regular mode: in the event
   // of output modifiers
```

```
      }

      if(postprocess){
      if(cnt > tag_ids[cur_id-1])
        cur_id++;
      }else if(cnt == 79){
      if(mc){
        putc(0);
        mval = 0;
      }
      delay_ms(1);
      cnt = 0;
      }

      // Increment the frequency
      inc(f, stepf);
    }
    // end sweep loop
    printf("88888888\r\n");
    delay_ms(2);

    if(postprocess){
      // Update the history vector
      history[3] = history[2];
      history[2] = history[1];
      history[1] = history[0];
      history[0] = found;

      DetermineTag();
    }

    if(mc) putc(1);
  }
}

void DetermineTag()
{
  int i;

  for(i=0;i<2;i++)
    if(history[i] != history[i+1])
      return;

  if(tag_id != history[0])
    {
      tag_id = history[0];
      printf("tr%u", tag_id);
    }
}

//void Output(int f)
//{
//  // Declarations
//  long int i, count;
//  int t1, t2, swap, period, j, k;
//
//  // Put out the audio signal if it's enabled
//  // Initializations
//  i = 0;
//  j = 0;
//  k = 0;
//  swap = 0;

//  // map the number of periods to something between 2000 and 3024
//  count = (750 + f * 6) / 32;
// //count = (1000 + f * 4) / 32;

//    // map the period to something between 250 and 165
//  period = 334 - (int)(((long int)f * 6) / 5);
// //period = 250 - (int)(((long int)f * 3) / 2);

//  for(k=0;k<4;k++){
//    t1 = f;
//    for(j=0;j<8;j++){
//      t2 = f;
//     for(i=0;i<count;i++){
//    if((0x01 & t1) & (0x01 & t2)) output_high(AUDIO);
//    else output_low(AUDIO);
//    delay_us(period);
//    output_low(AUDIO);
//    delay_us(period);
//    swap++;
//    if(swap == 8){
//      t2 = f;
//      swap = 0;
//    }else t2 = t2 >> 1;
//     }
//      t1 = t1 >> 1;
//    }
//  }
//}


/*
 * Function:
 *    setEquals()
 *
 * Description:
 *    Sets the 4 bytes in the destination array to be equal to
 *      the 4 bytes in source array.
 *
 * Arguments:
 *    int *source: a pointer to the source array.
 *    int   *dest: a pointer to the destination array.
 *
 * Return Value: None.
 */
void setEquals(int *source, int *dest)
{

  *dest = *source;
  *(dest + 1) = *(source + 1);
  *(dest + 2) = *(source + 2);
  *(dest + 3) = *(source + 3);
```

```
}

/*
 * Function:
 *    setEqualsReverse()
 *
 * Description:
 *    Sets the 4 bytes in the destination array to be equal to
 *      the 4 bytes in source array, but in the opposite format
 *      (i.e. an MSB first source gets copied into an LSB first
 *      destination).
 *
 * Arguments:
 *    int *source: a pointer to the source array.
 *    int   *dest: a pointer to the destination array.
 *
 * Return Value: None.
 */
void setEqualsReverse(int *source, int *dest)
{
  *dest = *(source + 3);
  *(dest + 1) = *(source + 2);
  *(dest + 2) = *(source + 1);
  *(dest + 3) = *source;
}

/*
 * Function:
 *    print32hex()
 *
 * Description:
 *    Prints out of the serial port the hex representation of
 *      the 32-bit number in the 4-byte array pointed to by r.
 *
 * Arguments:
 *    int *r: a pointer to the array holding the 32-bit number
 *              to print.
 *
 * Return Values:
 *      None.
 *
 */
void print32hex(int *r)
{
  /*
   * Prints out in hex the 32-bit integer whose LSB is
   * *r (i.e. the low byte).
       */
  printf("%x%x%x%x",*(r+3), *(r+2),*(r+1), *r);
}

/*
 * Function:
 *    lessThan()
 *
 * Description:
 *    Returns 1 if the first 4-byte argument is less than the second,
 *      otherwise returns 0. The arguments are in LSB-first format.
 *
 * Arguments:
 *    int *arg1: a pointer to the first 4-byte argument array.
 *    int *arg2: a pointer to the second 4-byte argument array.
 *
 * Return Value:
 *      1: arg1 < arg2
 *    0: arg1 !< arg2
 *
 */
int lessThan(int *arg1, int *arg2)
{
  // Declarations
  int flag = 1;

  // Copy the 2 arguments (which are LSB-first) into rs1 and rs2 (which
  // are MSB-first).
  setEqualsReverse(arg1, rs1);
  setEqualsReverse(arg2, rs2);

#asm
  // Clear the Carry bit in the status register.
  // Do a subtraction at each byte, and check if the status bit is set
  // to zero at each step. If the status bit is ever zero, then arg1 is
  // not less than arg2.
  movf      rs1[0],0
    subwf   rs2[0],1
    btfss   STATUS,0
    goto    nlt_flag
    btfss   STATUS,2
    goto    nlt_end
    movf    rs1[1],0
    subwf   rs2[1],1
    btfss   STATUS,0
    goto    nlt_flag
    btfss   STATUS,2
    goto    nlt_end
    movf    rs1[2],0
    subwf   rs2[2],1
    btfss   STATUS,0
    goto    nlt_flag
    btfss   STATUS,2
    goto    nlt_end
    movf    rs1[3],0
    subwf   rs2[3],1
    btfsc   STATUS,0
    goto    nlt_end

    nlt_flag:
  movlw     0x00
    movwf   flag
    nlt_end:
#endasm

  return flag;
}
```

```
/*
 * Function:
 *    inc()
 *
 * Description:
 *    Increments the value in the 4-byte source array by the value in
 *      the 4-byte increment array. The arguments are in LSB-first
format.
 *
 * Arguments:
 *    int    *source: a pointer to the source array.
 *    int *increment: a pointer to the destination array.
 *
 * Return Value: None.
 */
void inc(int *source, int *increment)
{
  // Copy the 2 arguments (which are LSB-first) into rs1 and rs2
  // (which are MSB-first).
  setEqualsReverse(source, rs1);
  setEqualsReverse(increment, rs2);

      // Perform the 32-bit addition
  add32();

  // Copy the result back
  setEqualsReverse(rd, source);
}

/* Rotates the value in rs1 left by the specified number of bits */
void rotate_left32(int i)
{
  int count;

  count = i+1;
#asm
rl32:
  decfsz    count,1
    goto    shift_l32
    goto    end_rl32

    shift_l32:
  bcf STATUS,0
    rlf     rs1[3],1
    rlf     rs1[2],1
    rlf     rs1[1],1
    rlf     rs1[0],1
    goto    rl32

    end_rl32:
#endasm
}

void mul32(int bits)
{
  /* Setup: Initialize the loop, and the result register. */
  /* Declarations */
  int i = 0;
  int count = 33;
  int flag = 0;

  for(;i<4;i++)
    rd[i] = 0;
  for(;i<8;i++)
    rd[i] = rs1[i-4];

#asm
  /*
    MUL32:
    Assumes the arguments are in rs1 and rs2 with rs1[0] and rs2[0]
    containing the MSB, and returns the 64-bit result in rd, with the
    MSB in rd[0]. The lower 4 bytes of the result are in rd[4] (MSB),
    rd[5], rd[6] and rd[7]. No information whether or not there is
    an overflow on the top byte is returned. Assumes unsigned numbers.

    ; The main multiplication loop
    ; Loop as many times as specified in count.
    ; (one for each significant bit in rs1)
    */
mul_loop:
  decfsz    count,1
    goto mul_step
    goto mult_end

    /* Check if the lowest bit of rd[7] is 1: if it is, add rs2
       to temp, else do the bit shift and then loop again. */
    mul_step:
  btfsc     rd[7],0
    call    mul_add
    call    mb_shift
    bcf     STATUS,0
    goto    mul_loop

    mul_add:
  /* Add the 32-bit contents of rs2 to temp
     Setup: clear the flag variable. */
  clrf      flag

    /* Performing the add of the low byte. */
    movf    rs2[3],0
    addwf rd[3],1

    /* Perform the increment in the event there was an overflow
       from the lower byte. */
    movf    rs2[2],0
    btfsc   STATUS,0
    addlw   0x1
    btfsc   STATUS,0
    bsf     flag,0

    /* Add the 2 next lowest bytes. */
    addwf   rd[2],1
    /* Perform an increment if there was any overflow from the
       middle 2 bytes. */
    movf    rd[1],0
    btfsc   STATUS,0
    addlw   0x1
    movwf   rd[1]

    /* Save the STATUS bit of this add. */
    btfsc   STATUS,0
    bsf     flag,1

    /* Perform an increment if there was any overflow from the
       addition of the first increment: this WILL not happen if
       the previous addition overflows. */
    movf    rs2[1],0
    btfsc   flag,0
    addlw   0x1

    /* Save the STATUS bit of this add. */
    btfsc   STATUS,0
    bsf     flag,1


    /* Complete the addition of the 3rd bytes. */
    addwf   rd[1],1

    /* Perform the last addition: don't use the overflow bit this
       time (but it is set in case the caller cares). */
    movf    rd[0],0
    btfsc   STATUS,0
    addlw   0x1
    btfsc   STATUS,0
    bsf     flag,2
    movwf   rd[0]
    movf    rs2[0],0
    btfsc   flag,1
    addlw   0x1
    btfsc   STATUS,0
    bsf     flag,2
    addwf   rd[0],1

    /* Set the carry flag if necessary. */
    btfsc   flag,2
    bsf     STATUS,0

    /* Addition has been completed: temp now has the updated value. */
    return

    /* This code performs the bit shifting of temp and rd
       The shifting is performed in the order rd[0], rd[1], rd[2],
rd[3],
       rd[4], rd[5], rd[6], rd[7]. */
    mb_shift:
  rrf   rd[0],1
    rrf   rd[1],1
    rrf   rd[2],1
    rrf   rd[3],1
    rrf   rd[4],1
    rrf   rd[5],1
    rrf   rd[6],1
    rrf   rd[7],1
    return

    mult_end:
#endasm
}

void add32(void)
{
  int flag = 0;
#asm
  /*
    ; ADD_32:
    ; Assumes the arguments are in rs1 and rs2, with rs1[0] and rs2[0]
    ; being the MSBs, and return the results in rd[0] to rd[3], with
    ; rd[0] as the MSB, and the carry bit set if there is an overflow
    ; on the top byte.

    ; Timing information:
    ; Total Length: 32 instructions.
    ; Maximum Time: 32 cycles.
    ; Minimum Time: 32 cycles.
    */
  /* Performing the add of the low byte. */
  movf      rs1[3],0
    movwf   rd[3]
    movf    rs2[3],0
    addwf   rd[3],1

    /* Perform the increment in the event there was an overflow from
       the lower byte. */
    movf    rs1[2],0
    movwf   rd[2]
    movf    rs2[2],0
    btfsc   STATUS,0
    addlw   0x1
    btfsc   STATUS,0
    bsf     flag,0

    /* Add the 2 next lowest bytes. */
    addwf   rd[2],1

    /* Perform an increment if there was any overflow from the middle
       2 bytes. */
    movf    rs1[1],0
    btfsc   STATUS,0
    addlw   0x1
    movwf   rd[1]

    /* Save the STATUS bit of this add. */
    btfsc   STATUS,0
    bsf     flag,1
    /* Perform an increment if there was any overflow from the addition
       of the first increment: this WILL not happen if the previous
       addition overflows. */
    movf    rs2[1],0
    btfsc   flag,0
    addlw   0x1

    /* Save the STATUS bit of this add. */
    btfsc   STATUS,0
    bsf     flag,1


    /* Complete the addition of the 3rd bytes. */
    addwf   rd[1],1

    /* Perform the last addition: don't use the overflow bit this time
       (but it is set in case the caller cares).  */
    movf    rs1[0],0
    btfsc   STATUS,0
    addlw   0x1
    btfsc   STATUS,0
    bsf     flag,2
    movwf   rd[0]
    movf    rs2[0],0
    btfsc   flag,1
    addlw   0x1
    btfsc   STATUS,0
    bsf     flag,2
    addwf   rd[0],1

    /* Set the carry flag if necessary. */
    btfsc   flag,2
    bsf     STATUS,0
#endasm
    }

/* input = string;
   output = 32-bit integer with MSB in result[0]; (something like
   atoi())  */

void str_to_int32 (char *str, int *result)
{
  /* Declarations */
  int i, j;
  char str_f0[6] = "0000", str_f1[6] = "0000";
  long int f1 = 0, f0 = 0;

  /* Break the string into the 2 pieces */
  /* Get the length of the string */
  for(i=0;i<8;i++)
    if(*(str+i) == '\0')
      break;

  /* Put the lower half into str_f0 */
  for(j=3;i!=0;i--){
    str_f0[j] = *(str+i-1);
    if(j == 0){
```

```
    i--;
    break;
  }
  j--;
}

/* Put the upper half into str_f1 */
for(j=3;i!=0;i--)
  {
    str_f1[j] = *(str+i-1);
    if(j == 0)
    break;
    j--;
  }

/* Calculate the values f1 and f0 */
f1 = str_to_int(str_f1);
f0 = str_to_int(str_f0);

/* Algorithm: 32-bit number = f1 * 625 * 2^4 + f0 */
/* Perform the multiplication: f1 * 625 */
/* Setup for mul32 */
rs2[0] = 0;
rs2[1] = 0;
rs2[2] = 0x2;
rs2[3] = 0x71;        /* 625 = 0x271 */
rs1[0] = 0;
rs1[1] = 0;
rs1[2] = (int)(f1 >> 8);
rs1[3] = (int)f1;

mul32(32);             /* Call mul32() to do the multiplication */

/* Call rotate_left32(). This multiplies the result of the */
/* previous call by 2^4. Setup for rotate_left32(). Copy
   the results of the mul32() into rs1. */
rs1[0] = rd[4];
rs1[1] = rd[5];
rs1[2] = rd[6];
rs1[3] = rd[7];

rotate_left32(4);

/* Call add32 to do the addition */
/* Setup for the add32() */
rs2[0] = 0;
rs2[1] = 0;
rs2[2] = (int)(f0 >> 8);
rs2[3] = (int)f0;

add32();

/* The result is in rd[0]. Move it into the desired array. */
*result = rd[0];
*(result + 1) = rd[1];
*(result + 2) = rd[2];
*(result + 3) = rd[3];
```

```
}

void div32(void)
{
  int i = 0, count = 33, flag = 0;

  for(;i<4;i++)
    rd[i] = 0;
  for(;i<8;i++)
    rd[i] = rs1[i-4];

#asm

  /*
    DIV32:
    ; Assumes the numerator (the argument on top) is in rs1_0 (MSB),
    ; rs1_1, rs1_2, rs1_3 and the divisor is in rs2_0 (MSB), rs2_1,
    ; rs2_2, rs2_3. Returns the results in rd_0 (MSB), rd_1, rd_2,
    ; rd_3, (the lower 4 bytes of the result) with no information
    ; whether or not there is an underflow on the top byte. Assumes
    ; unsigned numbers.

    ; The main division loop
    ; Loop 32 times (one for each bit in rs1)
    */
div_loop:   decfsz      count,1
              goto div_step
              goto div_end

              div_step:
/*
  ; Shift the bits of the register pair temp and rd.
  ; Do the subtraction of rs2 from temp, and check if the status bit
  ; is set (indicating a negative result). If it is clear, the low
  ; order bit of rd, add the contents of rs2 to temp and loop.
  ; Otherwise set the low order bit of rd and loop.
  */
call  div_bit_shift
              call      div_sub
              btfss     STATUS,0
              goto      neg_bit
              goto      pos_bit

              neg_bit:  bcf   rd[7],0
              call      div_add
              goto      div_loop

              pos_bit:  bsf   rd[7],0
              goto      div_loop


              div_add:
/*
  ; Add the 32-bit contents of rs2 to rd[0] - rd[3]
  ; Setup: clear the flag variable
  */
clrf  flag
```

```
// Performing the add of the low byte
movf      rs2[3],0
addwf     rd[3],1

// Perform the increment in the event there was
// an overflow from the lower byte
movf      rs2[2],0
btfsc     STATUS,0
addlw     0x1
btfsc     STATUS,0
bsf flag,0

// Add the 2 next lowest bytes
addwf     rd[2],1

// Perform an increment if there was any overflow
// from the middle 2 bytes
movf      rd[1],0
btfsc     STATUS,0
addlw     0x1
movwf     rd[1]

// Save the STATUS bit of this add
btfsc     STATUS,0
bsf flag,1
/*
  Perform an increment if there was any overflow
  from the addition of the first increment: this
  WILL not happen if the previous addition
  overflows
  */
movf      rs2[1],0
btfsc     flag,0
addlw     0x1

// Save the STATUS bit of this add
btfsc     STATUS,0
bsf flag,1


// Complete the addition of the 3rd bytes
addwf     rd[1],1
/*
  Perform the last addition: don't use the
  overflow bit this time (but it is set in
  case the caller cares).
  */
movf      rd[0],0
btfsc     STATUS,0
addlw     0x1
btfsc     STATUS,0
bsf flag,2
movwf     rd[0]
movf      rs2[0],0
btfsc     flag,1
addlw     0x1
```

```
              btfsc     STATUS,0
              bsf flag,2
              addwf     rd[0],1

              // Set the carry flag if necessary
              btfsc     flag,2
              bsf STATUS,0

              // addition has been completed: rd[0] - rd[3]
              // now has the updated value
              return


              div_sub:
/*
  ; Subtract the 32-bit contents of rs2 to temp
  ; Setup: clear the flag variable
  ; Setup: clear the flag variable
  */
clrf  flag

              // Perform the subtraction on the lower byte
              movf      rs2[3],0
              subwf     rd[3],1

              // Perform the subtraction on the next higher
              // byte
              movf      rs2[2],0
              btfss     STATUS,0
              incfsz    rs2[2],0
              subwf     rd[2],1

              // Perform the subtraction on the next higher
              // byte
              movf      rs2[1],0
              btfss     STATUS,0
              incfsz    rs2[1],0
              subwf     rd[1],1

              // Perform the subtraction on the highest byte
              movf      rs2[0],0
              btfss     STATUS,0
              incfsz    rs2[0],0
              subwf     rd[0],1

              // Subtraction has been done
              return

              div_bit_shift:
/*
  ; Shift the contents of the adjoining registers temp and rs1 left by
  ; a bit, shifting zero into the empty spot
  ;; Setup: Clear the flag, clear the C bit of the STATUS register
  ;; initialize the fsr register and the s_counter
  */
bcf   STATUS,0
              rlf   rd[7],1
              rlf   rd[6],1
              rlf   rd[5],1
              rlf   rd[4],1
              rlf   rd[3],1
              rlf   rd[2],1
              rlf   rd[1],1
              rlf   rd[0],1
              return

              // Shifting has been done. Return
              return

              div_end:
/* Division has been done: answer is in rd[4] - rd[7] */
#endasm
}

void getPhase(int *freq, int *result)
{
  /* Calculates the phase word for DDS clocked at a frequency */
  /* of 120 MHz based on the frequency present in the 32-bit word */
  /* MSB is in *freq. The result is returned in the 4-byte array  */
  /* pointed to by result, with the LSB in *result.               */

  /* Since the phase word = 2^32 * freq / CLOCK, */
  /* For a 120 MHZ clock, */
  /* 2^32 / 120 MHz = 2^23 / 234375 */
  /* 2^23 / 234375 = (2^12.2^6.2^5 / 125.125.15) which is the */
  /* calculation implemented below. */

  /* For the 30 MHz clock (multiplied up by 6 to 180 MHz), */
  /* 2^32 / 180 MHz = 2^25 / 1406250 */
  /* 2^25 / 1406250 = 2^12.2^6.2^7 / 125.125.90 which is the other
calculation implemented. */

  /* if frequency is less than 2^20 then we can do an extra  */
  /* 2^5 multiply first.  Otherwise, leave it for last.  */
  /* The variable z, is a flag which is zero if frequency is <24 */
  /* bits, 1 otherwise. */
  int z = 1;

  /* do an 12-bit rotate left if freq is small, 2 otherwise*/
  /* Setup for the rotate left */
  rs1[0] = *freq;
  rs1[1] = *(freq+1);
  rs1[2] = *(freq+2);
  rs1[3] = *(freq+3);

  if ((rs1[0]==0)&&(rs1[1]<16)) {
    z=0;
    rotate_left32(12);
  }else rotate_left32(2);

  /* Divide by 125 (0x7D). */
  rs2[0] = 0;
  rs2[1] = 0;
  rs2[2] = 0;
  rs2[3] = 0x7D;

  div32();

  /* Rotate the result left by 7 again. */
  rs1[0] = rd[4];
  rs1[1] = rd[5];
  rs1[2] = rd[6];
  rs1[3] = rd[7];

  rotate_left32(6);

  /* Divide by 125 again. */
  div32();

  rs1[0] = rd[4];
  rs1[1] = rd[5];
  rs1[2] = rd[6];
  rs1[3] = rd[7];


  /* Rotate the result by 7 this time. */
  rotate_left32(7);

  /* Divide by 90 (0x5A). */
  rs2[0] = 0;
  rs2[1] = 0;
  rs2[2] = 0;
  rs2[3] = 0x5A;

  div32();

  /* if freq is large (i.e. z=1) then we still have one more */
  /* multiplication to do (rotate left) */
  if (z==1) {
    rs1[0] = rd[4];
    rs1[1] = rd[5];
    rs1[2] = rd[6];
    rs1[3] = rd[7];
    rotate_left32(10);
    rd[4] = rs1[0];
    rd[5] = rs1[1];
    rd[6] = rs1[2];
    rd[7] = rs1[3];
  }

  /* The result is now in top 4 bytes of rd. Move it into the result
   * array. notice that the order of the bytes have been reversed */
  /* rd[4] is MSB and rd[7] is LSB, but right_shift routine */
  /* takes the lowest byte in array as the LSB, so we must */
  /* make result[0] the LSB. */
  *(result + 3) = rd[4];
  *(result + 2) = rd[5];
  *(result + 1) = rd[6];
  *result = rd[7];
}
```

8.4 Microcontroller Control

The microcontroller control commands was developed by Olufemi Omojola and Rich Fletcher (Physics and Media Group, 2000) as part of an effort to develop a single interface for embedded tag reader control and interface programs. This was done based on the observation that a large portion of functionality was common to a wide range of tag readers. For example, in many tag readers, there is the ability to scan over a part of the tag reader's excitation frequency range with variable resolution. Such a standard software interface simplifies the development of subsequent tag readers by providing a common command set, a subset of which could be implemented in any particular tag reader, depending on the capabilities of its hardware. Separating the design of the tag reader's control structure from the hardware implementation allows us to develop a single robust control program (for a workstation) that would be able to control any tag reader and monitor its data visually. This approach has proven useful as applications have emerged that require multiple tag readers connected to a network without human intervention.

This microcontroller control set was adopted as well, for the network analyzer board. While a small subset of the commands that were available were implemented and used with the external Labview interface, the entire set of commands will be presented here to aid any further development of external interface programs (e.g. for custom applications) to control the network analyzer board, via the microcontroller.

The control interface differentiates between two types of external agents, human and machine. For human controllers (and the default control and communications channel, the RS-232 interface) the commands are ASCII (American Standard for Communications and Information Interchange) byte sequences and instrument responses as well as data output are also in ASCII format. For human control, the instrument inserts visual framing characters (such as carriage return characters for data output) into the instrument's output data stream. Also, reference information is inserted into the stream (such as text descriptors of returned data and text prompts for commands with interactive properties).

The machine control mode is designed for reliable control by software programs that need to either run specific sequences of commands or do reliable data plotting and collection. Framing bytes are inserted into the data stream between the control program and the instrument to simplify synchronization. No visual framing characters or reference information about returned data are inserted into the data stream. Data is not returned in ASCII format, but as raw byte values, adjusted to account for certain values reserved for framing and synchronization.

Output Data Format

The network analyzer board performs a scan of the RF spectrum and returns data values depending on the response in the environment at the probe. Aside from acquiring raw data, an arbitrary amount of post-processing can be performed on the data obtained from the probe before it is output through the main communications/control channel (arbitrary in the sense that the interface specification does not discuss what post-processing can or can not be performed). The interface provides for standard output formats for the case where no pre-processing is done on the raw data, since post-processing can be much more easily implemented in software (on a PC platform) rather than in firmware.

For the human control mode, the default output format is a sequence of lines, each line representing a single point in the data scan. Each line starts with an asterisk followed by an ASCII representation of the particular frequency point. Next, ASCII representations of the associated data values are output, separated from each other and from the frequency value by a single space. The line is terminated with a carriage return. In general, terminal emulation programs have been used to interface with and control the instrument, and the use of asterisks and carriage returns as framing characters are a reflection of this. Figure 1 below shows an example of the output of an instrument in human control mode that takes 2 measurements per frequency value (i.e. magnitude and phase readings), where each measurement is represented as a 16-bit value (obtained from a 10 bit analog-to-digital conveter), and the frequency representation is a 32-bit value. The values are transmitted in base-16 format (which is a more compact representation than the default decimal format, and as such minimizes the data transmit time), and <LF><CR> represents a line feed, carriage return pair.

```
*0021FEFC 0124 0234<LF><CR>
```

Figure 39. Sample instrument output

For the machine control mode, the default output format is simply a sequence of raw data values. These are arbitrarily sized (for multi-byte values, the MSB (most significant byte) comes first) and run from the data values from the frequency at the start of the scan to the frequency at the end of the scan, inclusive. The intent is that upon startup the control agent will synchronize with the instrument to discover the start and stop frequencies, as well as the size of the difference between frequency points. If necessary, the program can append these values to the appropriate data points by itself, reducing the bandwidth requirements on the data channel. A discovery command is implemented that will return the number of data values per frequency point and the size in bytes of each value. The data is output in a packet format, with the start of the data packet (and the start of the frequency scan) indicated by a byte value of '0'. The end of each packet is also marked with a '0' (for cases where the data from the scan extends over more than one data packet). The control agent uses these markers at the beginning and end of each packet for synchronization, and to accommodate channels that may impose arbitrarily small data packet sizes. The end of each frequency scan is indicated by a byte value of '1'. Figure 2 below shows the packet structure for an instrument in machine control mode that takes 1 measurement per frequency value, where each measurement is an 8-bit value, and the frequency representation is a 32-bit value. The channel used is limited to 6 byte packets, and the entire frequency scan is 6 points.

0	10	20	30	40	0

0	50	60	1

Figure 40. Sample packet structure

Commands

The commands consist of byte sequences sent by the external control agent. The instrument implements a passive control structure that is interrogated by the external agent, and some commands elicit a response from the instrument. Each command begins with a command byte, an arbitrary number of modifier bytes after the command byte and an arbitrary number of data bytes after the modifier bytes. Some commands allow or require interactive responses. The basic structure is shown in Figure ?? below. The first byte, 0x61 (in hexadecimal) is the ASCII character 'a', the command byte used for the set start frequency command in instruments with frequency scanning capability. The command has no modifiers, and is followed by an ASCII representation of the desired start frequency, (in this case, 5 bytes, all ASCII characters, representing the number 60000), which is translated into a numeric value within the instrument. The command is optionally terminated with the ASCII character '\n', 0x2E.

0x61	0x36	0x30	0x30	0x30	0x30	0x2E

Figure 41. Command byte structure

In machine control mode, the instrument responds to all commands with a data packet. The packet is started with a data value '2' and terminated with a data value '3', and may or not contain any data. This is in place of the '0' and '1' framing bytes outlined in section 3 above. The listing below gives a basic grouping of the commands, descriptions of each command and associated special properties. It lists the associated byte sequences for both directions of the data stream (to and from the instrument) and gives a reference for the data stream for both human control and machine control.

General Commands
These commands set general instrument properties, such as the output mode and format or the instrument operating mode, or initiate internal routines, such as self-calibration.

d; Set Data Streaming Mode: this toggles the data-streaming property of the instrument. When in data-streaming mode, the instrument continuously outputs the results of its frequency scan.
Command Byte: ASCII 'd'; Hexadecimal 0x64.
Command Modifiers: None.
Command Data: None.
Command Terminator: Not present.

e; Set Polling Mode : this toggles the polling property of the instrument. When in polling mode, the instrument waits for a trigger signal from the external control program, then performs a single search operation, returns the results of that search, then waits for the next trigger signal.
Command Byte: ASCII 'e'; Hexadecimal 0x65.
Command Modifiers: None.
Command Data: None.
Command Terminator: Not present.

z; Poll Instrument: for a instrument with the polling mode property set to true, this is the trigger signal that initiates a single search operation and returns the results. Otherwise, it is ignored.
Command Byte: ASCII 'z'; Hexadecimal 0x7A.
Command Modifiers: None.
Command Data: None.
Command Terminator: Not present.

o; Set Onboard Output Mode: this toggles the onboard output enable property of the instrument. For instruments with output channels on the reader itself (such as LED/LCD displays or speakers), when in onboard output mode, these channels are enabled. Otherwise, they are disabled.
Command Byte: ASCII 'o'; Hexadecimal 0x6F.
Command Modifiers: None.
Command Data: None.
Command Terminator: Not present.

h; Set Output Data Format: this command switches between multiple output formats. For instruments that support multiple formats, the command modifiers are used to signify which mode.
Command Byte: ASCII 'h'; Hexadecimal 0x68.
Command Modifiers: An optional ASCII representation of a number, starting at 0. When not present, changes output format to default. When present, it is instrument dependent.
Command Data: None.
Command Terminator: Present.

v; Set Power Saving Mode : this command toggles the power saving mode of the instrument. For suitably equipped instruments, in power saving mode the signal output chain is disabled.
Command Byte: ASCII 'v'; Hexadecimal 0x76.
Command Modifiers: None.
Command Data: None.
Command Terminator: Not present.

x; Pause Instrument: this command toggles the active state of the instrument. When paused, the instrument skips all software functions related to tag reading. It may or may not power down the tag reading signal chain. It however responds to external commands, and does not output any data.

Command Byte: ASCII ‘x’; Hexadecimal 0x78.
Command Modifiers: None.
Command Data: None.
Command Terminator: Not present.

M; Set Instrument Controller Mode : this command toggles the controller state of the instrument. The instrument may be in human control mode, for control by a human operator, or in machine control mode, for control by a software program. These states are as described in Section 2 above.
Command Byte: ASCII ‘M’; Hexadecimal 0x4D.
Command Modifiers: None.
Command Data: None.
Command Terminator: Not present.

P; Switch to Machine Control Mode and Pause: this command was added to allow software controllers that are connecting to a instrument place it in a deterministic state. Upon receiving this command, the instrument pauses and places itself in machine control mode, then awaits further instructions.
Command Byte: ASCII ‘P’; Hexadecimal 0x50.
Command Modifiers: None.
Command Data: None.
Command Terminator: Not present.

u; Set Identifier String: this sets the identifier kept by each instrument. The identifier may or may not persist through power cycles, and is at least 32-bits long. This identifier may be assigned by a control program to enable it distinguish between multiple instruments that may share the same data channel.
Command Byte: ASCII ‘u’; Hexadecimal 0x75.
Command Modifiers: None.
Command Data: 4 or more bytes, depending on the size of the identifier space within the reader. The reader will either truncate the data to fit the available space, or zero pad the data if the space is more than the available space. The value sent is literal data (i.e. not an ASCII representation, but the actual numeric value. No translation is done).
Command Terminator: Present.

n; Get Identifier String: this retrieves the current value stored in the identifier space of the instrument.
Command Byte: ASCII ‘n’; Hexadecimal 0x6E.
Command Modifiers: None.
Command Data: None.
Command Terminator: Not present.

p; Initiate Internal Calibration Routines: this runs any internal calibration routines the instrument has. It may or may not result in an interactive prompt.
Command Byte: ASCII ‘p’; Hexadecimal 0x70.
Command Modifiers: Optional (instrument dependent).
Command Data: None.
Command Terminator: Present.

q; Enable Internal Post-Processing Routines: this toggles the internal post processing property of the instrument. On some instruments, there will be the capability to run some internal processing routines on the raw data obtained from the antenna measurements before output.
This may or may not modify the format of the output data.
Command Byte: ASCII ‘q’; Hexadecimal 0x71.
Command Modifiers: Optional (instrument dependent).
Command Data: Optional (instrument dependent).
Command Terminator: Present.

r; Get Internal Post-Processing Data: this streams out the current state associated with the instrument's internal post-processing routines through the main communications/control channel in a instrument dependent format.
Command Byte: ASCII 'r'; Hexadecimal 0x72.
Command Modifiers: None.
Command Data: None.
Command Terminator: Not present.
S; Adjust Internal Post-Processing Data: this initiates the internal routines to adjust the data associated with the internal post processing routines. This is typically a series of interactive prompts for information.
Command Byte: ASCII 'S'; Hexadecimal 0x53.
Command Modifiers: None.
Command Data: None.
Command Terminator: Not present.

s; Get Instrument State : this streams out the current state of the instrument's properties through the main communications/control channel in a instrument dependent format.
Command Byte: ASCII 's'; Hexadecimal 0x73.
Command Modifiers: None.
Command Data: None.
Command Terminator: Not present.

t; Adjust Internal Thresholds : this adjusts the internal threshold variables on the instrument. The effect is instrument dependent.
Command Byte: ASCII 't'; Hexadecimal 0x74.
Command Modifiers: None.
Command Data: At least one argument, each argument an ASCII representation of a base 10 number, with each argument separated by ASCII ';' (hexadecimal 0x3B). The order and significance of the arguments are instrument dependent.
Command Terminator: Present.

m; Set Target Tag Mode: this command selects the target tag to search for (for example, a instrument may be capable of searching for both silicon RFID tags and magnetic EAS tags).
Command Byte: ASCII 'm'; Hexadecimal 0x6D.
Command Modifiers: An ASCII representation of a number greater than or equal to zero.
Currently assigned values are:
0 – Resonant mode tags (such as magnetic EAS tags).
1 – Harmonic mode tags (such as magnetic domain wall tags).
Other values may be assigned in a instrument dependent fashion.
Command Data: None.
Command Terminator: Not present.

y; Reset All Constants to Default: this command resets all internal constants to their value immediately after a power-on reset. It may or may not implement a remote hardware reset.
Command Byte: ASCII 'y'; Hexadecimal 0x79.
Command Modifiers: None.
Command Data: None.
Command Terminator: Not present.

D; Discover Current Output Format: this command returns a byte sequence that details the default output format for the system. The sequence is at least 4 bytes long, and increases in multiples of 2. The first byte in the sequence is the size of the entire sequence (including the size byte). The second byte is the number of data values returned by the instrument for each frequency point. For each of the values that may be returned there is a 2-byte number that indicates the data value's size, MSB first. The sizes are listed in the order that the data values will be returned.
Command Byte: ASCII 'D'; Hexadecimal 0x44.
Command Modifiers: None.

Command Data: None.
Command Terminator: Not present.

Scanning Commands
These commands are those that set whether or not the instrument does a frequency scan and set the related frequency parameters.

w; Set Scanning Mode: this toggles the scanning property of the instrument. When in scanning mode, the instrument performs a search for tags between the current scan start and stop frequencies, in frequency intervals set by the scan step frequency. Otherwise the instrument searches for a tag at a single frequency.
Command Byte: ASCII ‘w’; Hexadecimal 0x77.
Command Modifiers: None.
Command Data: None.
Command Terminator: Not present.

g; Restart Frequency Scan: if a scan is in progress, this command restarts the scan from the current scan start frequency. Otherwise, it is ignored.
Command Byte: ASCII ‘g’; Hexadecimal 0x67.
Command Modifiers: None.
Command Data: None.
Command Terminator: Not present.

a; Set Start Frequency: this sets the frequency at which the scan starts.
Command Byte: ASCII ‘a’; Hexadecimal 0x61.
Command Modifiers: None.
Command Data: An ASCII representation of the frequency value to start the scan at. The number must be greater than 0. The instrument determines the range of allowable values.
Command Terminator: Present.

b; Set Stop Frequency: this sets the frequency at which the scan stops.
Command Byte: ASCII ‘b’; Hexadecimal 0x62
Command Modifiers: None.
Command Data: An ASCII representation of the frequency value to start the scan at. The number must be greater than 0. The instrument determines the range of allowable values.
Command Terminator: Present.

c; Set Step Frequency: this sets the frequency difference between each point in the scan.
Command Byte: ASCII ‘c’; Hexadecimal 0x63
Command Modifiers: None.
Command Data: An ASCII representation of the value of the difference between each frequency point in the scan. The number must be greater than 0. The instrument determines the range of allowable values.
Command Terminator: Present.

f; Set Single Frequency: this sets the frequency at which the instrument searches for tags when not in scanning mode.
Command Byte: ASCII ‘f’; Hexadecimal 0x66.
Command Modifiers: None.
Command Data: An ASCII representation of the frequency value at which the instrument performs the search for a tag. The number must be greater than 0. The instrument determines the range of allowable values.
Command Terminator: Present.

B; Set Harmonic Base Frequency: this sets the base frequency at which the instrument transmits (for instruments with harmonic detection capabilities, where the instrument transmits at a certain base frequency and searches for harmonics of that frequency in the response of the tag).
Command Byte: ASCII ‘B’; Hexadecimal 0x42.

Command Modifiers: None.
Command Data: An ASCII representation of the frequency value at which the instrument transmits when in harmonic mode. The number must be greater than 0. The instrument determines the range of allowable values.
Command Terminator: Present.

Legacy Commands
These commands are included to deal with current implementations of the interface. They will be eliminated as soon as an appropriate format can be implemented to support an arbitrary number of constants for special purposes.

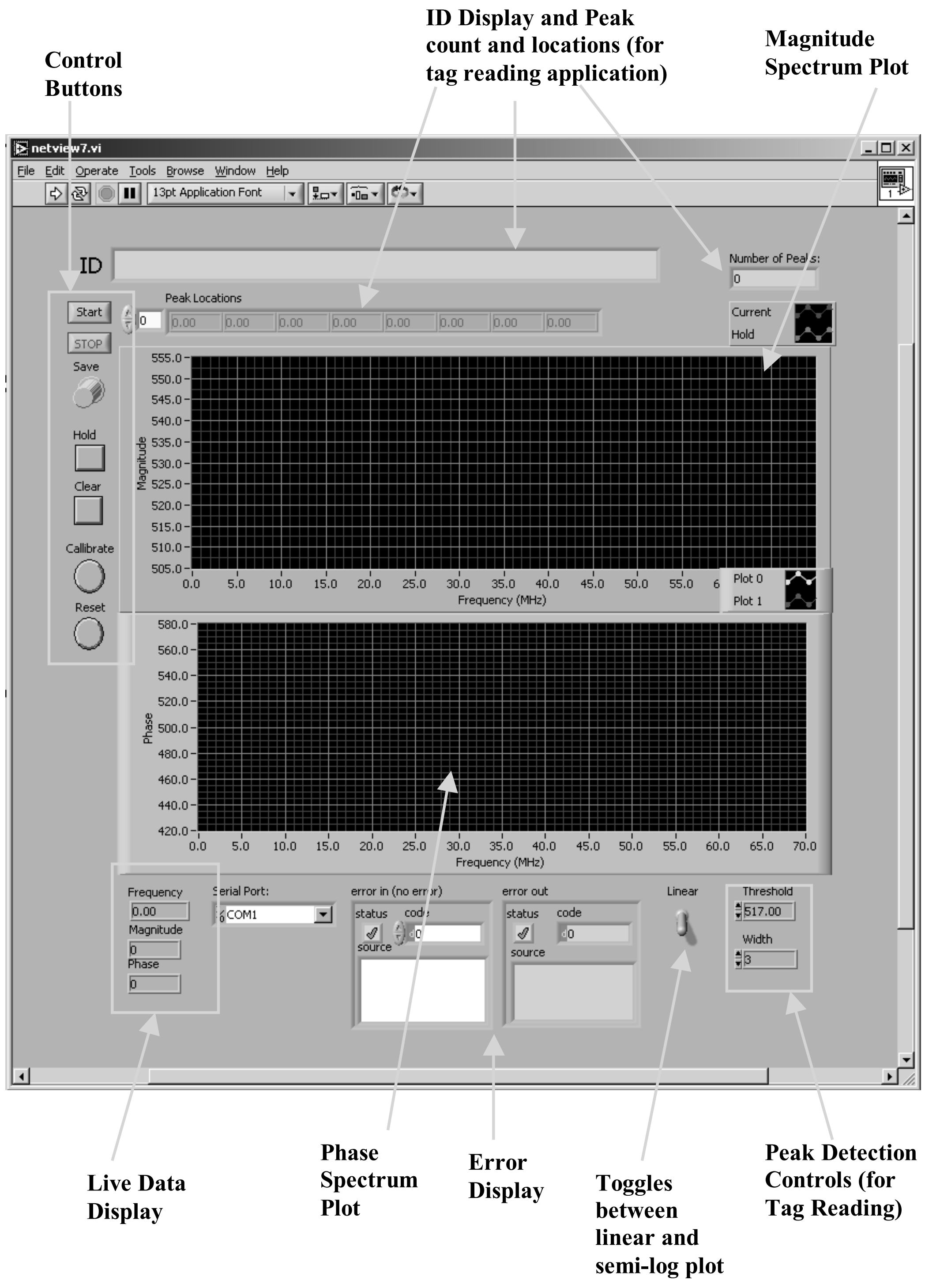
j; Set Transmit Pulse Period: when implemented, this allows the modification of the length of time of the radio frequency pulse the instrument transmits while performing a search. The length is a multiple of 10µs.
Command Byte: ASCII 'j'; Hexadecimal 0x6A.
Command Modifiers: None.
Command Data: An ASCII representation of the length of time (in multiples of 10µs) that the instrument should transmit for. The number must be greater than 0. The instrument determines the range of allowable values.
Command Terminator: Present.

k; Set Transient Delay Period: when implemented, this allows the modification of the length of time after the end of the transmit pulse that the instrument waits for either transient energy in the transmit coil to subside, or the frequency generator to stabilize. This delay is of use when the same antenna is used for the RF transmission and tag response detection.
Command Byte: ASCII 'k'; Hexadecimal 0x6B.
Command Modifiers: None.
Command Data: An ASCII representation of the length of time (in multiples of 10µs) that the instrument should wait after the end of the transmit pulse before measuring the response from the tag. The number must be greater than 0. The instrument determines the range of allowable values.
Command Terminator: Present.

l; Set Measurement Period: when implemented, this allows the modification of the length of time that the measurement of the tag response is taken. This delay is of use when a time dependent measurement technique is in use (such as an integrator).
Command Byte: ASCII 'l'; Hexadecimal 0x6C.
Command Modifiers: None.
Command Data: An ASCII representation of the length of time (in multiples of 10µs) that the instrument should take the measurement. The number must be greater than 0. The instrument determines the range of allowable values.
Command Terminator: Present.

8.5 Labview Interface

User Panel



Number of Peaks: 0

Peak Locations: 0 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00

Frequency: 0.00

Magnitude: 0

Phase: 0

Serial Port: COM1

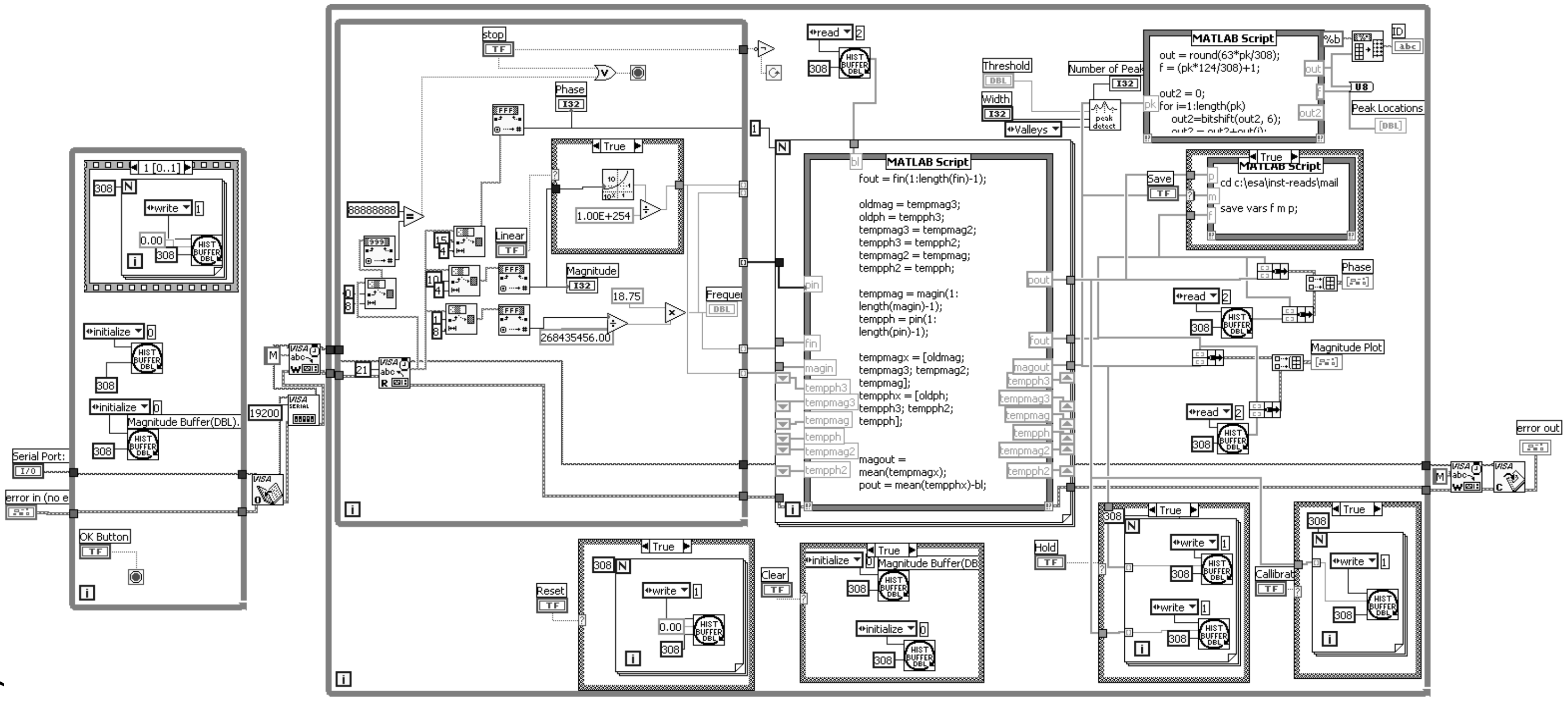
error in (no error): status ✓ code 0

error out: status ✓ code 0

Threshold: 517.00

Width: 3

Diagram (Code)



```
MATLAB Script
fout = fin(1:length(fin)-1);

oldmag = tempmag3;
oldph = tempph3;
tempmag3 = tempmag2;
tempph3 = tempph2;
tempmag2 = tempmag;
tempph2 = tempph;

tempmag = magin(1:
length(magin)-1);
tempph = pin(1:
length(pin)-1);

tempmagx = [oldmag;
tempmag3; tempmag2;
tempmag];
tempphx = [oldph;
tempph3; tempph2;
tempph];


magout =
mean(tempmagx);
pout = mean(tempphx)-bl;
```

```
MATLAB Script
out = round(63*pk/308);
f = (pk*124/308)+1;

out2 = 0;
for i=1:length(pk)
    out2=bitshift(out2, 6);
    out2 = out2+out(i);
```

```
MATLAB Script
cd c:\esa\inst-reads\mail

save vars f m p;
```

8.6 Labview Interface Features and Operating Instructions

The Labview interface was developed to serve as a user-friendly interface between the PC and the network analyzer board via the microcontroller. While the microcontroller board accepts character (ASCII) inputs from the keyboard, this can be quite inconvenient for a user who is unfamiliar with the various commands that the microcontroller accepts. In addition, displaying the data collected by the network analyzer board graphically is desirable, as it allows the user to have a visual sense of the data being collected.

The Labview interface that was developed had the following features:

1) Graphical displays of the Phase and magnitude spectrum
2) Trace holding allowing comparison between different test objects/samples
3) Autoscaling on/off function for graphs
4) Error message display
5) Data saving function, allowing acquired spectra to be stored.
6) Baseline correction, to correct for frequency dependence of the probe/instrument

The operating requirements for the Labview interface are as follows:

1) Windows operating system
2) Labview 6.0 and Matlab (6.0 or 6.1)
3) Sufficiently fast PC/laptop (> 600 MHz)

The sequence for operating the interface program is:

1) Start up and execute the Labview interface program virtual instrument
2) Power up network analyzer board
3) Click on the 'Start' button in the user panel

At this stage, the board will begin sweeping across the frequency range (1 MHz to 125 MHz) and display the acquired data.

To save current data in a Matlab vars.mat file,

4) Click on the 'Save' button on the user panel

To calibrate the baseline,

5) Click on the 'Calibrate' button
6) Clicking on 'Reset' undoes ceases the baseline subtraction function, and the raw data will be plotted

To hold the current plot,

7) Click on the 'Hold' button'. The current magnitude and phase plots will be held as blue traces.
8) Clicking on 'Clear' removes the held traces from the plots

For chipless RFID tag reading,

9) Set up peak detection function by specifying the threshold and peak width in the appropriate control boxes. A default value for these has already been specified, but this can be adjusted.

Miscellaneous:

10) Clicking on the 'Linear' button toggles the plot scale from a semi-log scale (default) to a linear sacle.

To end,

11) Click on 'Stop'.
12) Turn off board power.

8.7 Cost Analysis of Board

The following table gives the various components that make up the instrument, as well as their prices in various quantities. These have been totaled up to obtain the estimated parts cost for each board, which is given at the bottom of the table.

Part Type	Desig-nator	Foot-print	Descrip-tion	Comments	Price-10	Price-25	Price-35	Price-100	Price-1000
.1u	C10	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
.1u	C11	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
.1u	C12	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
.1u	C13	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
.1u	C14	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
.1u	C15	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
.1u	C16	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
.1u	C17	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
.1u	C18	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
.1u	C19	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
.1u	C20	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
.1u	C21	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
.1u	C22	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
.1u	C23	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
.1u	C24	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
.1u	C25	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
.1u	C6	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
.1u	C7	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
.1u	C8	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
.1u	C9	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
0.1u	C1	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
0.1u	C2	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
0.1u	C3	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
0.1u	C4	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
0.1u	C5	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
0.1uf	C27	805	Capacitor		0.16	0.16	0.16	0.1128	0.0656
0.47uf	C30	1206	Capacitor		0.255	0.255	0.255	0.182	0.1092
10000p	C126	805	Capacitor		0.097	0.097	0.097	0.0656	0.03384
10000p	C127	805	Capacitor		0.097	0.097	0.097	0.0656	0.03384
10000p	C26	805	Capacitor		0.097	0.097	0.097	0.0656	0.03384
1000p	C122	805	Capacitor		0.069	0.069	0.069	0.0406	0.01625
1000p	C123	805	Capacitor		0.069	0.069	0.069	0.0406	0.01625
1000p	C124	805	Capacitor		0.069	0.069	0.069	0.0406	0.01625
1000p	C125	805	Capacitor		0.069	0.069	0.069	0.0406	0.01625
100p	C130	805	Capacitor		0.069	0.069	0.069	0.0406	0.01625
100uF	C57	100UTA	Capacitor		1.644	1.644	1.644	1.1436	0.69

		NT							
100uF	C58	100UTA NT	Capacitor		1.644	1.644	1.644	1.1436	0.69
10u	C128	1210	Capacitor		0.518	0.518	0.518	0.4522	0.2975
10u	C129	1210	Capacitor		0.518	0.518	0.518	0.4522	0.2975
10uF	C32	1210	Capacitor		0.518	0.518	0.518	0.4522	0.2975
10uf	C33	1210	Capacitor		0.518	0.518	0.518	0.4522	0.2975
12p	C74	805	Capacitor		0.078	0.078	0.078	0.0527	0.0272
12p	C75	805	Capacitor		0.078	0.078	0.078	0.0527	0.0272
16p	C60	402	Capacitor		0.341	0.341	0.341	0.2015	0.0806
16p	C62	402	Capacitor		0.341	0.341	0.341	0.2015	0.0806
1uF	C55	1206	Capacitor		0.748	0.748	0.748	0.442	0.1768
22p	C120	805	Capacitor		0.078	0.078	0.078	0.0527	0.0272
22p	C121	805	Capacitor		0.078	0.078	0.078	0.0527	0.0272
22uF	C41	1210	Capacitor		0.622	0.622	0.622	0.5434	0.3575
27p	C64	805	Capacitor		0.078	0.078	0.078	0.0527	0.0272
27p	C66	805	Capacitor		0.078	0.078	0.078	0.0527	0.0272
2p	C68	805	Capacitor		0.078	0.078	0.078	0.0527	0.0272
2p	C70	805	Capacitor		0.078	0.078	0.078	0.0527	0.0272
330u	C44	HA-CAP	Capacitor		0.443	0.411	0.411	0.38	0.19
330u	C53	HA-CAP	Capacitor		0.443	0.411	0.411	0.38	0.19
39p	C76	805	Capacitor		0.078	0.078	0.078	0.0527	0.0272
39p	C77	805	Capacitor		0.078	0.078	0.078	0.0527	0.0272
47p	C72	805	Capacitor		0.078	0.078	0.078	0.0527	0.0272
47p	C73	805	Capacitor		0.078	0.078	0.078	0.0527	0.0272
47uF	C56	2220	Capacitor		1.644	1.644	1.644	1.436	0.69
82pF	C54	805	Capacitor		0.046	0.046	0.046	0.0312	0.01613
8p	C78	805	Capacitor		0.078	0.078	0.078	0.0527	0.0272
8p	C94	805	Capacitor		0.078	0.078	0.078	0.0527	0.0272
CON2	J16	SMA	Connector	Johnson Components Inc	4.446	4.446	4.446	3.276	2.34
CON2	J7	SMA	Connector	Johnson Components Inc	4.446	4.446	4.446	3.276	2.34
CON2	J9	2-PIN TERM BLOCK	Connector	-					
CON3	J2	3-PIN TERM BLOCK	Connector	-					
CON3	J4	POWER CON	Connector	CUI Inc	0.33	0.33	0.33	0.254	0.191
DB9	J1	DB9/F	Connector	Amp/Tyco	1.94	1.617	1.617	1.294	1.074
SG-636	X3	SG636	Crystal	Epson Electronics	6.15	5.125	5.125	5.125	2.95
XM-1	X4	XM-1	Crystal		6.525	6.525	6.525	5.22	3.915
AD9854	U8	SQ-80	DDS	Analog	33	30.5	30.5	28	27.29

				Devices					
LED	D1	1206	Diode		0.165	0.165	0.165	0.1375	0.11
LED	D2	1206	Diode		0.165	0.165	0.165	0.1375	0.11
TDC-10-1	U11	TDC-10-1	Directional Coupler	Minicircuits	14.9	14.9	14.9	14.0	13.3
TDC-10-1	U14	TDC-10-1	Directional Coupler	Minicircuits	14.9	14.9	14.9	14.0	13.3
AD8302	U25	SSO-14/G4.2	Gain/Phase Detector	Analog Devices	23.58	20.5	20.5	17	16.59
10uH	L1	5CCD	Inductor	Toko America	2.2568	2.2568	2.2568	1.1944	1.168
560uH	L2	5CCD	Inductor	Toko America	2.568	2.568	2.568	1.944	1.168
68n	L14	805	Inductor	Panasonic	0.851	0.851	0.851	0.668	0.437
68n	L15	805	Inductor	Panasonic	0.851	0.851	0.851	0.668	0.437
68n	L16	805	Inductor	Panasonic	0.851	0.851	0.851	0.668	0.437
68n	L8	805	Inductor	Panasonic	0.851	0.851	0.851	0.668	0.437
82n	L3	805	Inductor	Panasonic	0.851	0.851	0.851	0.668	0.437
82n	L7	805	Inductor	Panasonic	0.851	0.851	0.851	0.668	0.437
HEADER 2	JP1	JUMPER	Jumper	Norcomp Inc	0.984	0.984	0.984	0.811	0.55
HEADER 2	JP2	JUMPER	Jumper	Norcomp Inc	0.984	0.984	0.984	0.811	0.55
HEADER 5	JP3	HEADER5	Jumper	Norcomp Inc	1.53	1.53	1.53	1.26	0.87
JUMPER	JP4	JUMPER	Jumper	Norcomp Inc	0.984	0.984	0.984	0.811	0.55
JUMPER	JP5	JUMPER	Jumper	Norcomp Inc	0.984	0.984	0.984	0.811	0.55
16F877	U10	PLCC44	Micro-controller	Microchip	10.05	6.15	6.15	5.95	5.95
AD8011	U2	SO-8	Opamp	Analog Devices	3.94	3.3	3.3	2.51	2.51
AD8011	U5	SO-8	Opamp	Analog Devices	3.94	3.3	3.3	2.51	2.51
AD8057	U6	SO-8	Opamp	Analog Devices	1.4	1.2	1.2	1	1
AD8057	U3	SO-8	Opamp	Analog Devices	1.4	1.2	1.2	1	1
IRU1030	U7	TO-220	Regulator	International Rectifier	1.1	0.979	0.979	0.846	0.633
LM2940-5	U26	SOT-223	Regulator	National Semiconductor	1.8	1.32	1.32	0.96	0.624
TPS6735	U9	SO-8	Regulator	TI	2.68	2.14	2.14	1.67	1.52
24	R51	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
24	R52	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
51	R53	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
51	R54	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
51	R12	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682

51	R13	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
51	R15	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
51	R17	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
51	R48	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
51	R49	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
52.3	R55	805	Resistor	1%	0.09	0.0494	0.0484	0.0484	0.019
52.3	R56	805	Resistor	1%	0.09	0.0494	0.0484	0.0484	0.019
75	R18	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
75	R2	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
110	R14	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
110	R16	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
130	R4	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
130	R8	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
160	R1	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
160	R6	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
510	R3	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
510	R5	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
510	R7	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
510	R9	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
1.3k	R21	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
1k	R10	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
1k	R11	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
2k	R50	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
6.2k	R19	805	Resistor	5%	0.08	0.043	0.043	0.043	0.01682
MAX233	U1	SOL-20	RS-232 Interface	Maxim	10.76	8.12	8.12	5.2	5.2
1N5817	D3	1N5817	Schottky Diode	Diodes Inc	0.54	0.54	0.54	0.405	0.216
SW DPST	S1	PUSH-BUTTON	Switch	Omron Electronics	0.23	0.202	0.202	0.16	0.1215
SW SPDT	S3	SIP3	Switch	E-Switch	0.64	0.6	0.64	0.5454	0.315
PCB*				Advanced Circuits	102.30	46.52	36.09	9.53	5.00
					285.19	211.91	201.52	148.89	125.36

*3 day turn around for 10, 25 and 35, 4 week turn around for 100 and 1000

9. Bibliography and References

A.R. Blythe, Electrical Properties of Polymers, Cambridge University Press, Cambridge 1979.

D. Ballo, “Network Analyzer Basics”, Hewlett-Packard Company, 1998.

K. Fenske and D. Misra, “Dielectric Materials at Microwave Frequencies”, Applied Microwave and Wireless, Vol. 12 No. 10, Oct. 2000, pp 92-100.

L. Solymar and D. Walsh, Electrical Properties of Materials, Oxford University Press, Oxford 1998.

Livingston J. D., Electronic Properties of Engineering Materials, John Wiley and Sons, 1999.

Y. Maguire, “Towards a Table Top Quantum Computer”, Master’s Thesis, Massachusetts Institute of Technology, 1999.

J. B. Hagen, Radio-Frequency Electronics- Circuits and Applications, Cambridge University Press, Cambridge 1996.

M. Bramanti and E. Salerno, “Experiments on Some Particular Permittivity in Sensors in Nondestructive Testing of Dielectric Materials”, Journal of Microwave Power and Electromagnetic Energy, Vol. 27 No. 4, pp. 209-216, 1992.

R. J. Weber, Introduction to Microwave Circuits – Radio Frequency and Design Applications, IEEE Press, New York 2001.

Y. Wei and S. Sridhar, “Radiation-Corrected Open-Ended Coax Line Technique for Dielectric Measurements of Liquids up to 20 GHz”, IEEE Transactions on Microwave Theory and Techniques, Vol. 39, No. 3, March 1991.

---

[1] R. Fletcher, O. Omojola, E. Boyden and Neil Gershenfeld, “Reconfigurable Agile Tag Reader Technologies for Combined EAS RFID Capability”.

[2] M. A. Stuchly, S. S. Stuchly, “Coaxial line reflection method for measuring dielectric properties of biological substances at radio and microwave frequencies – A review,” IEEE Trans. Instrum. Meas., vol. 29, no. 3, September 1980, pp 176-183.

[3] T. J. Warnagiris, “Liquid Sensing at Radio Frequencies”, Microwave Journal, September 2000, pp. 140 – 150.

[4] A. Ramirez, W. Daily, A. Brinley, D. LaBrecque, “Complex Resistivity Tomography for Environmental Applications”, 1st World Congress on Industrial Process Tomography, April 1999, pp 14-19.

[5] Cole K. S. and Cole R.H., Journal of Chemical Physics, 1941, No. 9, 341-51.

[6] A. A. Laogun, “Dielectric Properties of Mammalian Breast Milk at Radiofrequencies”, Physics in Medicine and Biology, Vol. 31 No. 5, pp. 555-561, 1986.

[7] T. Kudra, V. Raghavan, C. Akyel, R. Bosisio and F. van de Voort, “Electromagnetic Properties of Milk and Its Constituents at 2.45 GHz”, Journal of Microwave Power and Electromagnetic Energy, Vol. 27 No. 4, 1992.

[8] A, Boughriet, Z.Wu, H. McCann and L.E. Davis, “The Measurement of Dielectric Properties of Liquids at Microwave Frequencies Using Open-ended Coaxial Probes”, 1st World Congress on Industrial Process Tomography, April 1999, pp 318-322.

[9] S. Sridhar, “Microwave Technology and Materials Research”, Microwave Journal, June 1987, pp. 117 – 123.

[10] D.K. Misra, “A quasi-static analysis of open-ended coaxial lines,” IEEE Trans. Microwave Theory Tech., vol. MTT-35, Oct. 1987, pp 925 -928.

[11] K. F. Staebell and D. Misra, “An experimental technique for in vivo permittivity measurement of materials at microwave frequencies,” IEEE Trans. Microwave Theory Tech., vol. 38, pp. 337-340, Mar. 1990.