REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE 28 March 1996

3. REPORT TYPE AND DATES COVERED
Final Report.

4. TITLE AND SUBTITLE
Acoustic Characterization of Soil

5. FUNDING NUMBERS
Contract No. DACA88-94-D-0008

6. AUTHOR(S)
W.D. O'Brien, Jr., R.G. Darmondy, & D.C. Munson, Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Dept. of Electrical & Computer Enginnering
Beckman Institute for Advanced Science & Technology
University of Illinois
405 North Mathews Avenue
Urbana, IL 61801

Dept Natural Resources & Envir. Sci.
University of Illinois
1102 S. Goodwin Avenue
Urbana, IL 61801

Coordinated Science Laboratory
Dept.. of Electrical & Computer Engineering
University of Illinois
1308 West Main Street
Urbana, IL 61801

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
SERDP
901 North Stuart St. Suite 303
Arlington, VA 22203

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
N/A

11. SUPPLEMENTARY NOTES
Report submitted to the US Army Construction Engineering Research Laboratory, 28 March 1996. This work was supported in part by Contract No. DACA88-94-D-0008.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release: distribution is unlimited

12b. DISTRIBUTION CODE
A

13. ABSTRACT ***(Maximum 200 Words)***

Detection and classification of buried cultural artifacts in ground soil are principal goals for the production, detection and processing of acoustic signals. Time domain, frequency domain, and combined time-frequency domain approaches to transmit and process acoustic signals all depend critically on the acoustic transduction device to transmit high-amplitude acoustic pressure waves and to receive low-amplitude acoustic pressure waves over a large band of frequencies.

14. SUBJECT TERMS
Acoustic Propagation, Cultural artifacts, SERDP

15. NUMBER OF PAGES
90

16. PRICE CODE N/A

17. SECURITY CLASSIFICATION OF REPORT
unclass.

18. SECURITY CLASSIFICATION OF THIS PAGE
unclass.

19. SECURITY CLASSIFICATION OF ABSTRACT
unclass.

20. LIMITATION OF ABSTRACT
UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

19980709 131

81 - 1996

FINAL REPORT

Acoustical Characterization of Soil

Contract Number DACA88-94-D-0008
US Army Construction Engineering Research Laboratory
P. O. Box 9005
Champaign, IL 61826

Prepared by

William D. O'Brien, Jr.
Department of Electrical and Computer Engineering
Beckman Institute for Advanced Science and Technology
University of Illinois
405 North Mathews Avenue
Urbana, IL 61801
217/333-2407
217/244-0105 fax
wdo@uiuc.edu

Robert G. Darmody
Department of Natural Resources and Environmental Sciences
University of Illinois
1102 South Goodwin Avenue
Urbana, IL 61801
217/333-9489
217/244-3219 fax
rdarmody@uiuc.edu

David C. Munson, Jr.
Coordinated Science Laboratory
Department of Electrical and Computer Engineering
University of Illinois
1308 West Main Street
Urbana, IL 61801
217/333-4789
217/244-1642 fax
d-munson@uiuc.edu

March 28, 1996

A. INTRODUCTION

The Final Report reports on the activities and results of the University of Illinois for the US Army Construction Engineering Research Laboratory Contract Number DACA88-94-D-0008, "Acoustical Characterization of Soil."

Detection and classification of buried cultural artifacts in ground soil are principal goals for the production, detection and processing of acoustic signals. Time domain, frequency domain and combined time-frequency domain approaches to transmit and process acoustic signals all depend critically on the acoustic transduction device to transmit high-amplitude acoustic pressure waves and to receive low-amplitude acoustic pressure waves over a large band of frequencies.

The transduction device is the means to an end and represents one of many critical components in the overall system, all of which affect the system's full utility. Other system components include the transmit/receive electronics, the broad-band electrical matching between the system's electronics and transduction device, the low-power and high-power switching capability between transmit and receive when the same transduction device is used for transmit and receive, and the broad-band mechanical matching between the transduction device and the acoustic propagation medium.

Once a cultural or archaeological resource site is identified, it must then be assessed in order to determine its significance and eligibility for National Registry of Historic Places (Executive Order 11593). A Phase II eligibility assessment currently costs about $10,000 to $30,000 per site. Given that there are approximately 120,000 archaeological sites in the Army alone, the cost of complete Phase II assessments is prohibitive. Therefore, there is an urgent need to significantly reduce the cost of data recovery at sites with an unknown probability of containing significant cultural or archaeological resources. It is hypothesized that a rapid subsurface acoustic imaging technique can serve this need.

The investigators wish to acknowledge the invaluable assistance of Nadine B. Smith and Richard N. Czerwinski for developing the data acquisition hardware and software, Richard N. Czerwinski for developing the data analysis software, Catherine Frazier for developing much of the synthetic aperture image approach. Scott Weisbrook for analyzing the soil samples, Dudley Swiney for analyzing the acoustic data, and Scott Weisbrook, David Tungate, Dan O'Brien, Brett Boege and Kay Raum for acoustic data acquisition.

B. MATERIALS AND METHODS

1. Acquisition and Evaluation of the Soils

Soils samples were collected from Illinois to represent a wide range of properties expected to influence acoustic response. The samples were chosen to include low and high contents of organic matter, sand, silt and clay, and a range of clay mineralogy. **Table 1** gives the locations, depths, horizons, and classifications of the soils.

Pedologists recognize layers in soils that are called horizons. These can be of various depths and thicknesses, but the properties of the soil material within individual horizons are different from the other horizons in a soil but internally similar. Horizons are labeled "A", "B", and "C" down from the surface. The horizon labels are given subordinate distinctions to indicate a particular subtype of horizon. The term "Ap" refers to the surface horizon of a soil that has been plowed (the "p" designation) or otherwise disturbed sometime in the past. Most surfaces of soils have been plowed or disturbed by forestry or agriculture to the point that soil scientists recognize the surface horizon (the "A") as an Ap and not simply as an "A" horizon that would indicate that it was "natural" or uncultivated or undisturbed by human activities.

Table 1. Locations and classifications of soils used for acoustic characterization.

Soil Sample	Sample Code	Soil Series	Soil Classification	Soil Horizon	Collection Depth (cm)	Location (IL County)	Latitude (deg min sec)	Longitude (deg min sec)
1	CAB	Catlin	fine-silty, mixed, mesic. Typic Argiudoll	Bt	73-85	Champaign	40 05 14	88 13 55
2	PLA	Plain-field	mixed, mesic. Typic Udipsamment	Ap	0-20	Mason	40 09 28	90 05 14
3	SAC	Sable	fine-silty, mixed, mesic. Typic Haplaquoll	Cg	300-320	Menard	40 00 00	88 42 30
4	DRA	Drum-mer	fine-silty, mixed, mesic. Typic Haplaquoll	Ap	0-20	Champaign	40 05 01	88 13 55
5	ADA	Adrian	sandy, mixed, mesic, euic. Terric Medisaprist	A	0-20	Cass	40 01 00	90 24 00
6	MEA	Med-way	fine-loamy, mixed, mesic. Fluvaquentic Hapludoll	Ap	0-20	Cass	40 00 00	90 28 00

Prior to analysis, the soil samples were air dried and sieved to exclude the >2 mm fraction. The influence of coarse fragment content, structure, and consistency was not addressed by this phase of the work. The soil samples were characterized as to particle-size distribution, organic matter content, clay mineralogy, coefficient of linear extensibility, plastic and liquid limits, plasticity index, and electrical conductivity. **Table 2** gives the soil physical characterization data.

Table 2. Physical properties of acoustically characterized soils.

Soil Code	Sand %	Silt %	Clay %	Organic Matter %	Soil Texture Class	Liquid limit	Plastic limit	Plasticity index	Conductivity dS/m	COLE %
CAB	11	53	36	0.5	Silty Clay Loam	43	24	19	143	14
PLA	97	1	2	0.4	Sand	NP	NP	NP	120	NP
SAC	2	82	16	0.1	Silt Loam	29	24	5	379	4
DRA	12	50	38	9.8	Silty Clay Loam	43	28	15	363	13
ADA	72	18	10	11.7	Fine Sandy Loam	NP	NP	NP	688	NP
MEA	38	38	24	2.3	Loam	28	19	9	258	7

Soil characterization was by standard methodology. Particle-size analysis was determined by the hydrometer method (Gee and Bauder, 1986). The silt and clay were determined by hydrometer and the sand fraction was separated by sieving. Electrical conductivity of the soils was determined on a 1:1 extract (Dahnke, 1988).

Organic carbon content was determined by wet combustion with sodium dichromate (Nelson and Sommers, 1986). Organic matter was calculated from the organic carbon content by multiplying by a conversion factor of 1.724. Liquid and plastic limits and plasticity index was determined with a glass plate and an electric liquid limit machine (AASHO, 1969). Coefficient of linear extensibility (COLE) was determined by air drying wet samples created with an extrusion device (Schafer and Singer, 1976).

Mineralogy is being determined on the silt and clay-sized material by X-ray diffraction (Whittig and Allardice, 1986). The size fractions were separated by centrifugation and analyzed on the Department of Natural Resources and Environmental Sciences' X-ray diffractometer.

Water content of the soils was determined by weight loss on drying, then conversion to volumetric water content basis by multiplying by the bulk density (Gardner, 1986).

The original experimental design detailed six soils, three compaction levels, and four moisture levels for a total of 72 experimental units. However, the Plainfield soil was evaluated only loosely compacted at the low moisture level, and experimental difficulties prevented three levels of compaction. Therefore, only the two extreme compaction levels were evaluated, *viz.*, loose and dense. Also, for the 100% moisture condition, it was necessary to measure only one compaction level because of the incompressibility of the water that saturated the soil pores. Therefore, the final experimental design included five soils at two compaction levels and three moisture levels, plus dry, loose Plainfield, plus Plainfield at two moistures and two compactions, plus all six saturated soils. This gave a total of 41 experimental units. The soils were acoustically evaluated at a minimum of three thicknesses ranging from 5-27 cm. Soil thickness was varied both by addition of new soil to the container and by removal of material. The total number of acoustic evaluations was 231 (**Appendix A1**).

The soil was placed in a calibrated plastic tub (approx. 30 cm high by 30 cm diam.) and weighed prior to acoustic evaluation. A small sample was removed to determine gravimetric moisture content at the time the acoustic evaluation was made. Soil particle density was estimated by adjusting the density of quartz, 2.65 g/cc, by the soil organic matter content at a density of 1.3 g/cc. From these data the soil sample bulk density, moisture content, and thickness were determined.

Bulk density of the soils was determined by weight per unit volume (Blake and Hartge, 1986). Porosity of the soils was determined by calculating the ratio of sample weight to volume (Danielson and Sutherland, 1986).

Initial water content of the soils for acoustic evaluation was air-dry. The two next higher water contents were nominally increased to 10 and 25% water by volume. This was achieved by carefully adding the appropriate quantity of water and thoroughly mixing and screening the moist soil through a 4 mm sieve. The saturated soil was made by adding screened moist soil to standing water. All moist soils were allowed to equilibrate for at least 24 hours before acoustic evaluation.

The loose compaction of the soils was achieved by pouring the sieved soil into the calibrated tub and striking off a level, smooth surface. The soils when air-dry were quite powdery and dusty. Compaction was achieved by adding a succession of 2 cm lifts to the bottom of the tub and applying a load by means of vigorous stomping onto a plywood disk on the soil surface. This was repeated until the desired soil thickness was achieved.

The acoustic data was recorded in files that were named to identify the experimental soil conditions. The first three letters in the file name were the soil code (see **Appendix A1**). The next two numbers were the sample thickness in cm. The next digit was the moisture code, 1, 2, 3, or 5 for air-dry, 10%, 25%, and saturated, respectively. The next digit was the compaction code, 1 for loose, 4 for compact. The last number in the file name represented the replication of those soil conditions. An additional code is added after the file name to indicate sequence number. Table A1 lists the sample files and the soil conditions at the time the acoustic data was collected.

2. Acoustic Measurement System

Data Acquisition Hardware

A block diagram of the soil characterization system is shown in **Figure 1**. The system consists of a host computer (ZEOS 133 MHz Pentium, Minneapolis, MN) which is used to control all data acquisition procedures. This computer has a 16 MB of core memory (expandable to 32 megabytes), 700 MB hard drive storage, Mitsumi 4X CD-ROM Drive, SuperVGA graphics, and is connected to the Beckman Institute internet network. Communications with the internet network is through NCSA Telnet 2.6.1. A Hewett Packard HP8116A (with option 001) programmable signal generator receives instructions *via* the IEEE-488 communication bus which produces a low-level signal. The low-level signal from the HP8116A is amplified by a 3000 W power amplifier (Industrial Test Equipment Powertron 3000S amplifier, Port Washington, NY) and impedance matched to the NRL F56 Serial 58 acoustic source (Transducer Branch, U.S. Naval Research Laboratory's Underwater Sound Reference Detachment, Orlando, FL) by a transformer (Industrial Test Equipment ET-900V transformer).

The acoustic signal is propagated through the soil sample contained in a tub which is coupled to face of the F56 acoustic source *via* a water interface. The transmitted acoustic signal is received by the NRL F42C Serial 28 hydrophone (Transducer Branch, U.S. Naval Research Laboratory's Underwater Sound Reference Detachment, Orlando, FL) which is acoustically coupled to the top of the soil sample with DOW 710 pheynlated silicon oil. The NRL F42C hydrophone is stable and calibrated with a bandwidth between 100 Hz and 200 kHz. The calibration is traceable to the NRL's Underwater Sound Reference Detachment, a national standards laboratory.

The backscattered acoustic signal is acquired by two SE1025-H acoustic emission hydrophones (Dunegan Engineering, San Juan Capistrano, CA).

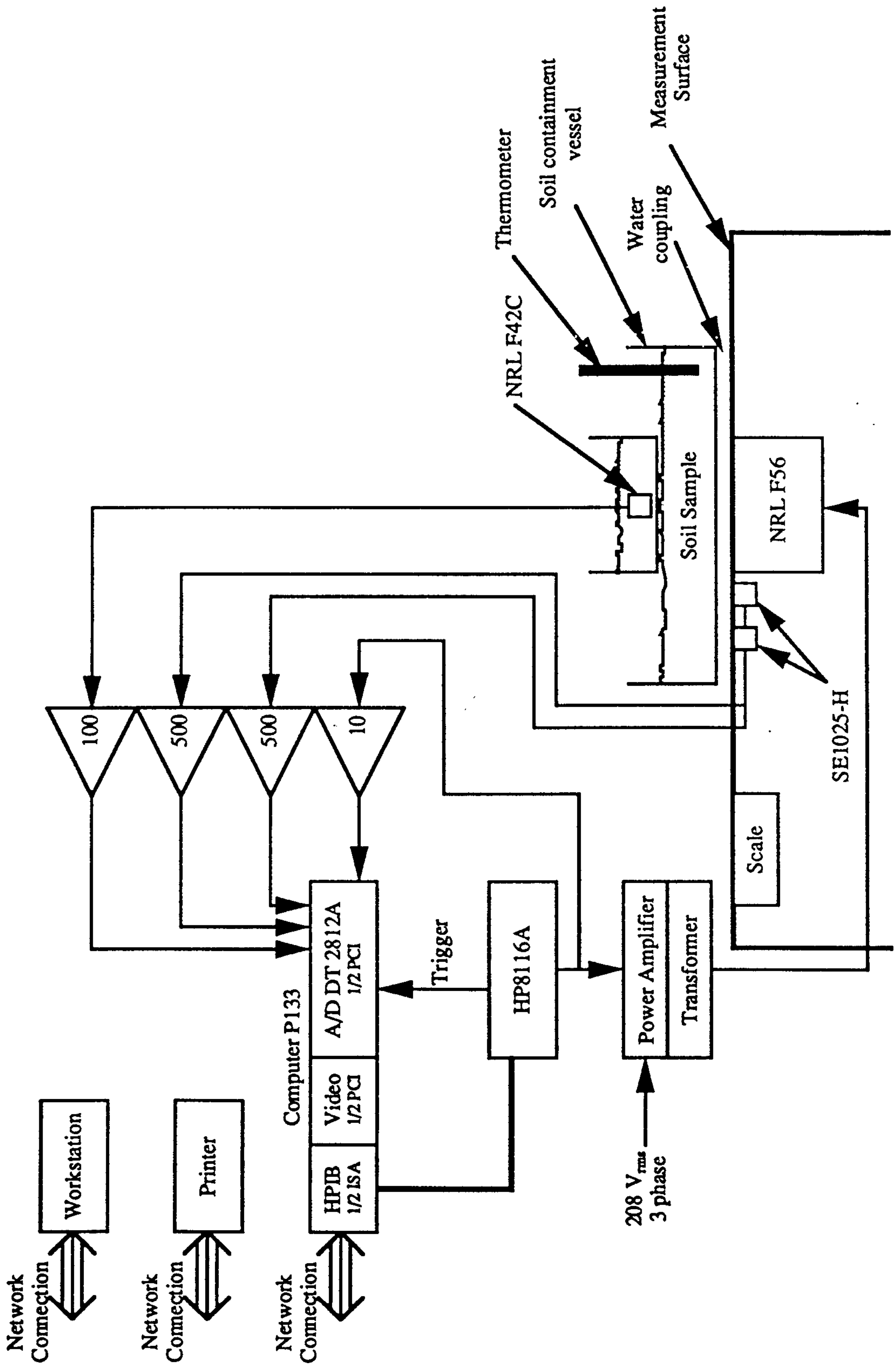
The four low level signals from the HP 8116A signal generator (Hewlett-Packard), the NRL F42C hydrophone and two SE1025-H acoustics emission hydrophones are amplified with gains of 10, 100, 500 and 500, respectively. The design of the high input impedance amplifiers is shown in **Figure 2**. The outputs from the amplifiers are digitized by a DT2812A A/D board (Data Translation, Marlboro, MA).

Data Acquisition Software

A set of programs for the data acquisition system instructs the signal generator to scan a range of amplitudes and frequencies and to receive, digitize and store in individual ASCII files the four low level signals.

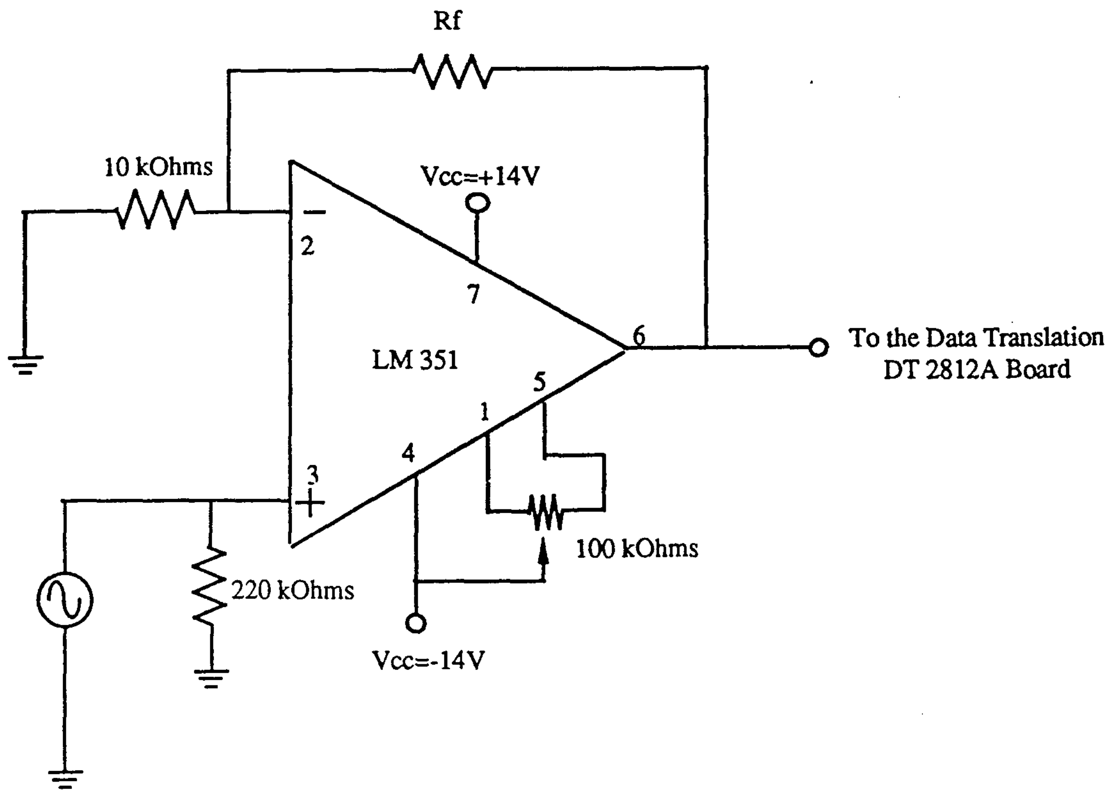
Data acquisition and command instructions to the IEEE-488 compatible devices from the ZEOS Panthera occurs thorough the General Purpose Interface Bus (GPIB) (National Instruments-GPIB/PC, 1989). The interface bus is comprised of a interface board (National Instruments PCII) installed in the motherboard of the ZEOS computer and physically connected through a GPIB cable to the electronic devices which allows the electronic devices to communicate. A program GPIB.COM (supplied by National Instruments) is a software loadable device driver which is loaded at the system start-up by DOS.

All of the communication programs and the ***autoexec.bat*** file (**Appendix A2**) are specifically written for use of the Microsoft® C6.0. Within the ***autoexec.bat*** file are instructions to the operating system as to the location of libraries and include files. The ***config.sys*** file (**Appendix A3**) is also specially written for the computer environment.



Measurement Surface
Soil containment vessel
Thermometer
Water coupling
NRL F42C
Soil Sample
NRL F56
SE1025-H
Scale
100
500
500
10
A/D DT 2812A 1/2 PCI
Video 1/2 PCI
HPIB 1/2 ISA
Computer P133
Trigger
HP8116A
Power Amplifier
Transformer
208 V rms 3 phase
Workstation
Printer
Network Connection
Network Connection
Network Connection

Figure 1



Amplified Device	Amplifier No#	Gain	Rf Value	DT2818A Channel No#
HP8116A	0	10	100 kΩ	0
NRL 42C	1	100	1.0 MΩ	1
SE1025H-1	2	500	4.7 MΩ	2
SE1025H-2	3	500	4.7 MΩ	3

Figure 2

A schematic of the high input impedance amplifier circuit for the HP8116A signal generator source and the three hydrophones (NRL 42C, SE1025H-1 and SE1025H-2). The table lists the desired gain for the individual device and the require feedback resistance value for Rf. The table also indicates the input channel for the for the Data Translation DT2812A A/D board.

The ***soil4.mak*** file (**Appendix A4**) is the make file for the ***soil4.exe*** program. The purpose of the make file is to compile all the source code for the program, define the type of memory model to use, create the stack of 32000 bytes and then link all of the object files.

For use of the software which runs the soil characterization system are 3 non Microsoft® C6.0 include files. The ***DTST_THS.H*** program, written by Data Translation (**Appendix A5**), contains a list of define (#define in C language) for the Data Translation software tools called from a Microsoft® C6.0 program. The ***DTST_XMM.H*** (**Appendix A6**) program is the definition file for the extended memory management functions using ***HIMEM.SYS*** from Microsoft® C6.0. Finally, the ***DTST_ERR.H*** (**Appendix A7**) program contains the DTI Software Tools error codes which provides 4096 different error codes.

Data translation supplies two libraries which contain error information and codes but which are too extensive to list in the appendix. The C program which calls these libraries is called *MSC_SUB.C* (**Appendix A8**). This program contains the source which basically creates, calls, clears, utilizes and deletes the buffers which store the acquired data from the channels. Detailed information about using the Data Translation libraries, include and source code can be found elsewhere (Data Translation SPO 126 Software Tool Kit, 1992).

The *soil4.c* (**Appendix A9**) program is the source code for the executable main program which controls the soil data acquisition.

The procedure to transfer raw data from soil acquisition system (bmode.brl.uiuc.edu) to workstation (ecstasy) is listed in **Appendix A10**.

Data Processing Software

The processing of the data is done in Matlab (The MathWorks, Natick, MA) on a SUNSparc2 workstation. The analysis code is modular in design to facilitate experimental analysis. Processing is performed by a succession of two primary routines, procsoil.m and process.m.

The ***procsoil.m*** (**Appendix A11**) program reads in (*via* ***loadsoil.m***, **Appendix A12**) the raw data soil measurements from a single thickness, and produces a list of estimates of the time delay between pulse transmission and reception, and the amplitude of the received pulse. The estimates are tabulated in 1 kHz increments of frequency, and each value is obtained from four different frequency measurements, *i.e.*, the 1 kHz values are obtained from the scans at 1000 Hz, 1250 Hz, 1500 Hz and 1750 Hz, from 1 kHz to 10 kHz.

Time of flight (TOF) is estimated by computing the correlation function between the transmitted and received pulses for each of the frequencies being combined, and forming a histogram of the peak locations which occur. Two estimates represent high and low values intended to define a range of possible TOF values, a third estimate is the median of the first two, and the fourth estimate is the most frequently occurring histogram peak location. In practice, this should be the most accurate estimate.

The amplitude values returned by ***procsoil.m*** are correlation function values corresponding to the TOF estimates. These values are directly proportional to the desired quantity, with a scaling factor equal to the square of the transmitted pulse amplitude; as long as data sets of different transmitting signal amplitude are not mixed, there is no need to normalize measurements by this quantity.

The ***peak.m*** (**Appendix A13**) program is a function which does the work on the signals within *procsoil.m* (except for the portions dealing with user interface). It allows for greater ease of software development by containing the commands frequently invoked in ***procsoil.m*** into a single file.

The ***process.m*** **(Appendix A14)** program computes frequency dependent speed of sound estimates by performing linear regressions with respect to thickness. Speed of sound is calculated as the slope of line which best fits (in a least squares sense) the plot of TOF *vs.* thickness. Attenuation is the slope of the line which best fits the plot of 20*log(received amplitude) *vs.* thickness. The 20 log() operation is performed so that attenuation can be reported in dB/cm-kHz. The ***process.m*** program also reports the coefficient of the regression to indicate how well the points fit the regression line.

The ***process.m*** program also reports the frequency dependent results from the log decrement analysis. The ***log_dec.m*** **(Appendix A15)** program is a function used to analyze the data received by the acoustic emissions transducers, positioned at the bottom of the tub of soil. It computes the center frequency and 3 dB bandwidths of the exponentially decaying signal which appears after the excitation signal is turned off. By the Biot theory of wave propagation in porous media, the 3 dB bandwidth is related to the decay rate of the decaying component; this decay rate is calculated directly and from the 3 dB bandwidth.

The ***coef.m*** **(Appendix A16)** program is a function which evaluates the correlation function between two data sequences at a certain point. The ***maxes.m*** **(Appendix A17)** program is a function which return the indices of local maxima in a data sequence. It is used here to determine the positions of correlation peaks. The ***regression.m*** **(Appendix A18)** program is a function used to compute the best fit line, in a least squares sense, to a set of points.

3. Acoustic Measurement Technique

Samples were hand-packed in the soil containment vessel for acoustic evaluation. This hand-packed sample was at a relatively low density (loose compaction). After this sample was evaluated acoustically, the sample density (compaction) was increased by placing a piston-type arrangement on top of the soil in the soil containment vessel and applying an impulsive-type force to the piston.

C. SOIL RESULTS AND DISCUSSION

Tables 3 and **4** provide the mean values of attenuation coefficient (dB/cm-kHz) and propagation speed (m/s) for the six soil types (ADA, CAB, DRA, MEA, PLA, SAC) as a function of four soil moistures and two soil compactions. The first digit of the code in the left hand column designated soil moisture (1, 2, 3, 5) where 1 designated dry soil and 5 designates fully

Table 3. Mean value of the attenuation coefficient (dB/cm-kHz)

	ADA	CAB	DRA	MEA	PLA	SAC
11	0.31	0.14	0.21	0.12	0.23	0.39
14	0.20	0.59	0.53	0.63	–	0.36
21	0.68	0.36	0.38	0.39	0.35	0.46
24	0.58	0.91	0.56	0.62	0.50	0.69
31	0.39	0.45	0.51	0.28	0.81	0.50
34	0.71	0.90	0.71	0.90	0.16	0.96
51	0.49	0.58	0.58	0.31	0.75	0.38

Table 4. Mean value of the propagation speed (m/s)

	ADA	CAB	DRA	MEA	PLA	SAC
11	153	190	177	154	260	140
14	121	150	159	126	-	139
21	89	114	118	151	138	117
24	147	102	136	93	103	122
31	87	150	107	161	122	121
34	86	105	245	105	253	176
51	102	102	118	139	189	207

saturated soil. The second digit of the code in the left hand column designates soil compaction (1,4) where 1 designates loose compaction and 4 designates dense compaction. **Appendix A19** provides a more complete listing of the acoustic properties as a function of the detailed soil characteristics.

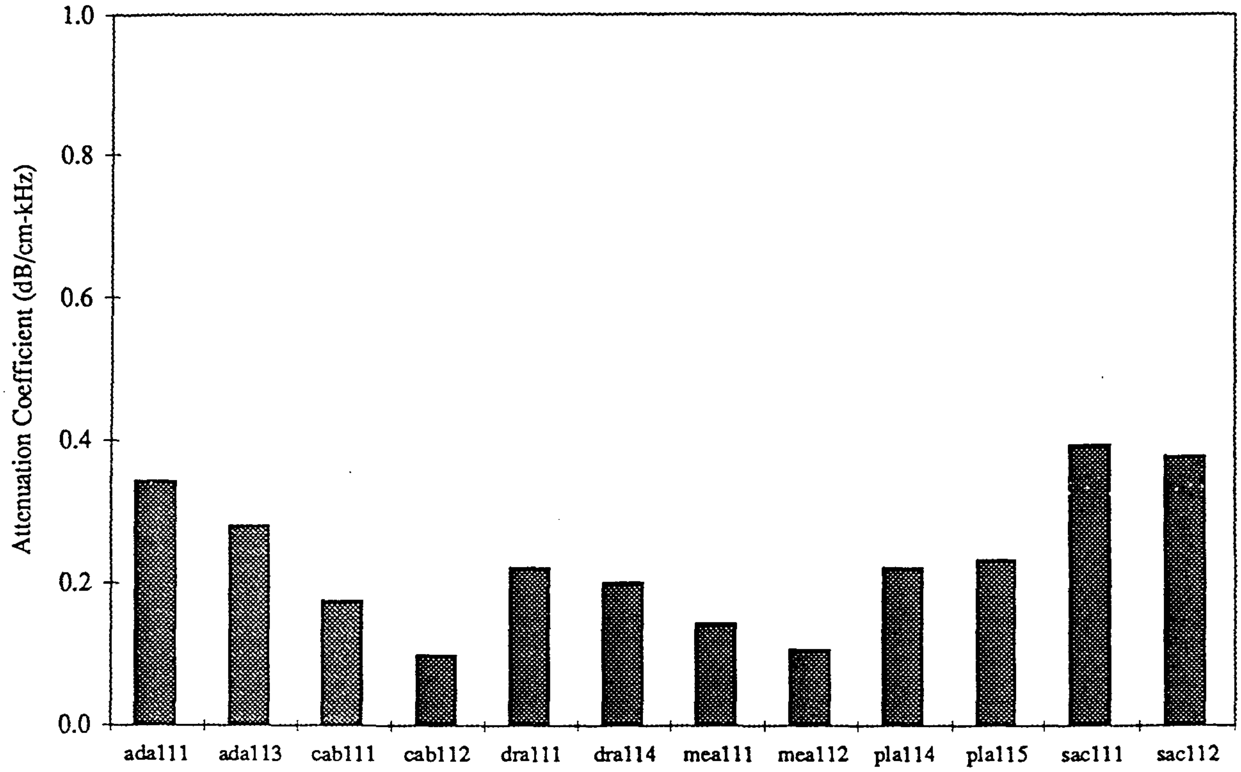
Figures 3-9 graphically represent the attenuation coefficient (dB/cm-kHz) and propagation speed (m/s) for each of the seven soil conditions (11, 14, 21, 24, 31, 34, 51). The last digit in the code in the figures is a sequence number which was used for repeat experimental units.

The experimental protocol called for measuring the acoustical properties in relatively homogeneous soil samples. It is thus suggest that the experimental approach for homogeneous soil preparation is the principal factor which contributed to the relatively narrow range of measured acoustic propagation property values. Additionally, because the soil samples were relatively homogeneous, this precluded an evaluation of the acoustic scattering properties because scattering results from acoustic heterogeneities.

The attenuation coefficient over the 1-10 kHz frequency range ranged between a low of about 0.1 dB/cm-kHz which was most prevalent for the loose dry (11) samples and a high close to 1 dB/cm-kHz which was most prevalent for the more moist compact (34) samples.

The attenuation coefficient has a direct influence over the imaging depth for a given dynamic range. **Figure 10** demonstrated the influence of the roundtrip propagation loss as a function of attenuation coefficient (top panel) and frequency (bottom panel). For a dynamic range of 140 dB (typical of diagnostic ultrasound imaging systems but unknown yet for a sub-surface acoustic imaging system), the top panel in **Figure 10** suggests an imaging depth of 4.6 feet is achievable for an attenuation coefficient of 0.5 dB/cm-kHz at a frequency of 1 kHz and an imaging depth of 6 feet is easily achievable for lesser attenuation coefficient values. For the same dynamic range, the bottom panel in **Figure 10** suggests an imaging depth of 2.3 feet is achievable for a frequency of 5 kHz at an attenuation coefficient of 0.2 dB/cm-kHz, increases to 3.8 feet for a frequency of 3 kHz and to more than 6 feet for a frequency of 1 kHz.

It is also necessary to consider the acoustic reflection coefficient from the object to be imaged in soil when considering the overall roundtrip propagation loss, but, in general, it appears that the acoustic reflection coefficient will be around 0 dB. This is based on the experimental results obtained herein. The characteristic acoustic impedance of soil is in the range of (1-3) x 10^5 Pa-s/m (density $\approx$1000 kg/m^3; propagation speed $\approx$100-300 m/s). A plastic object might have a


1.0
0.8
0.6
0.4
0.2
0.0
Attenuation Coefficient (dB/cm-kHz)
ada111
ada113
cab111
cab112
dra111
dra114
mea111
mea112
pla114
pla115
sac111
sac112

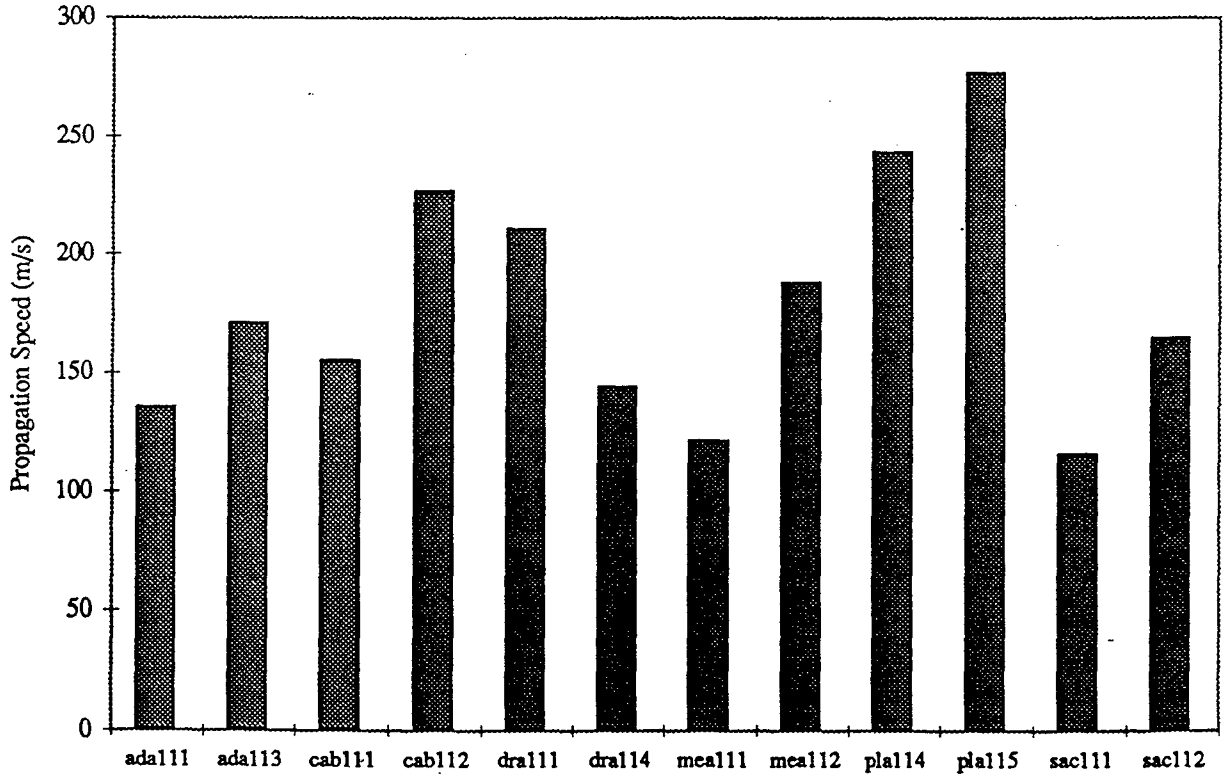

300
250
200
150
100
50
0
Propagation Speed (m/s)
ada111
ada113
cab111
cab112
dra111
dra114
mea111
mea112
pla114
pla115
sac111
sac112

Figure 3

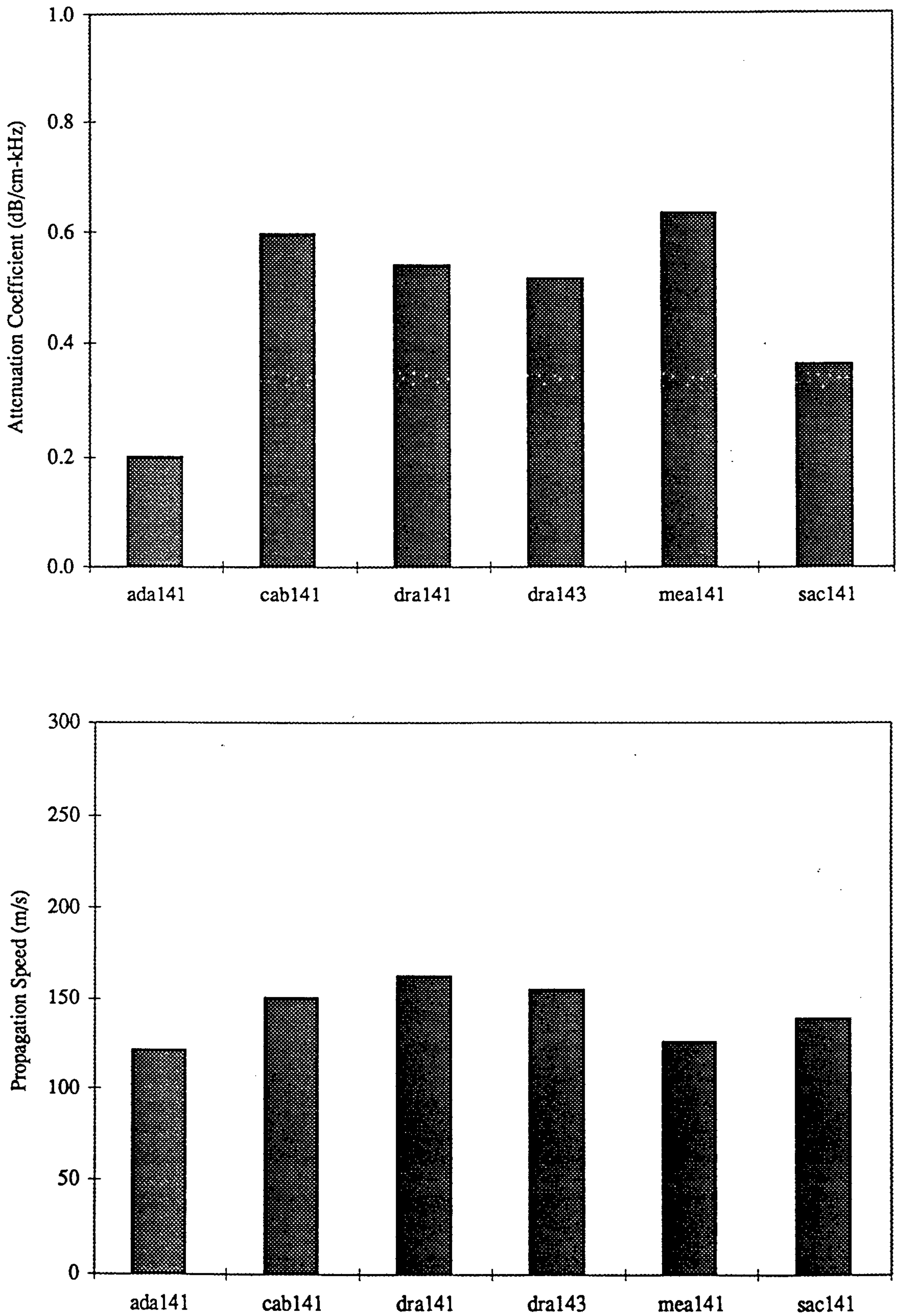
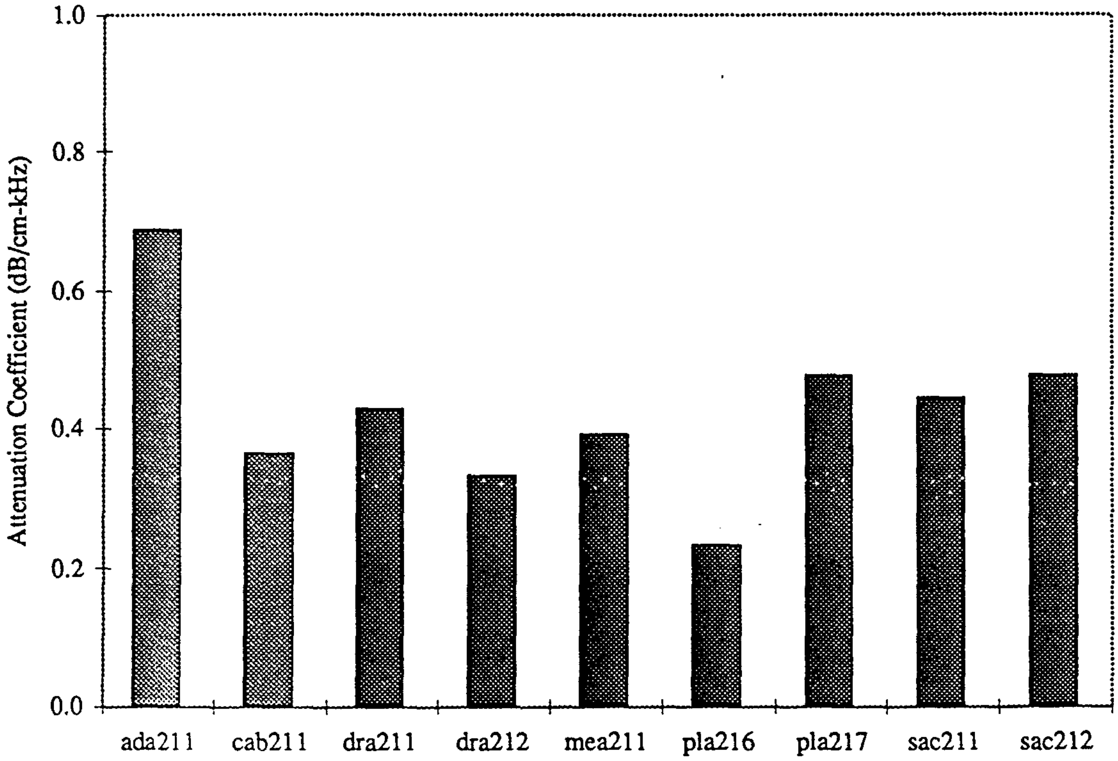
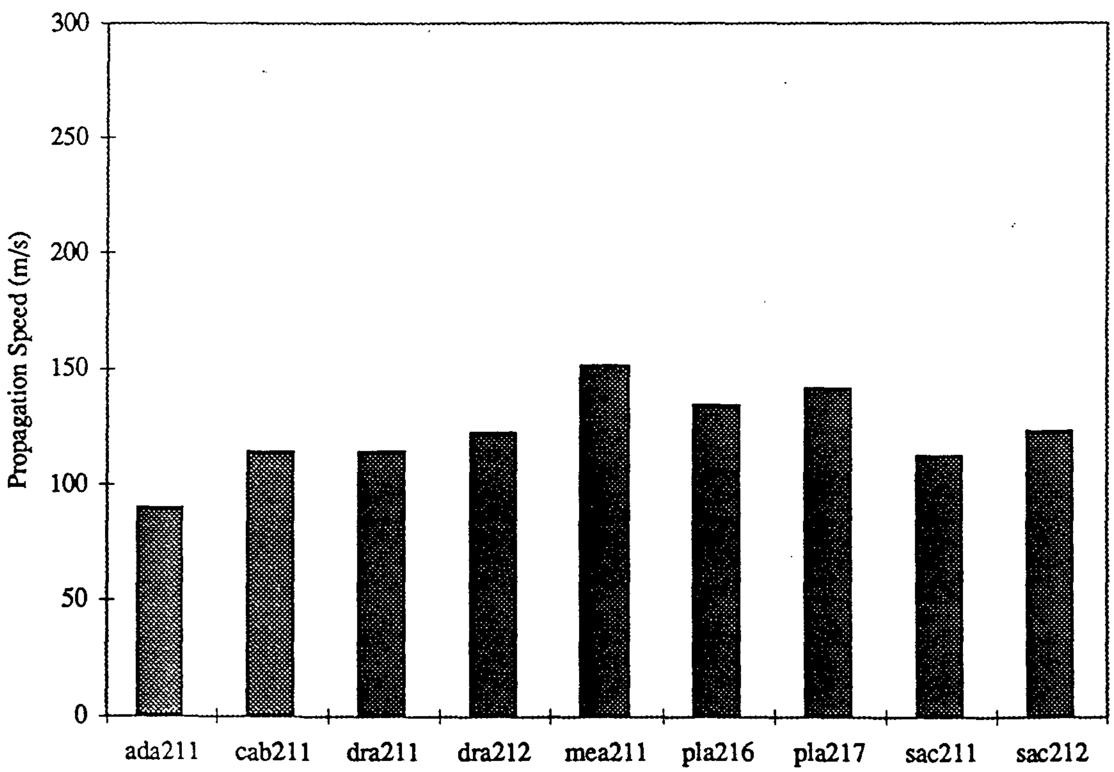

1.0
0.8
0.6
0.4
0.2
0.0
Attenuation Coefficient (dB/cm-kHz)
ada141
cab141
dra141
dra143
mea141
sac141
300
250
200
150
100
50
0
Propagation Speed (m/s)
ada141
cab141
dra141
dra143
mea141
sac141

Figure 4



1.0
0.8
0.6
0.4
0.2
0.0
Attenuation Coefficient (dB/cm-kHz)
ada211
cab211
dra211
dra212
mea211
pla216
pla217
sac211
sac212



300
250
200
150
100
50
0
Propagation Speed (m/s)
ada211
cab211
dra211
dra212
mea211
pla216
pla217
sac211
sac212

Figure 5

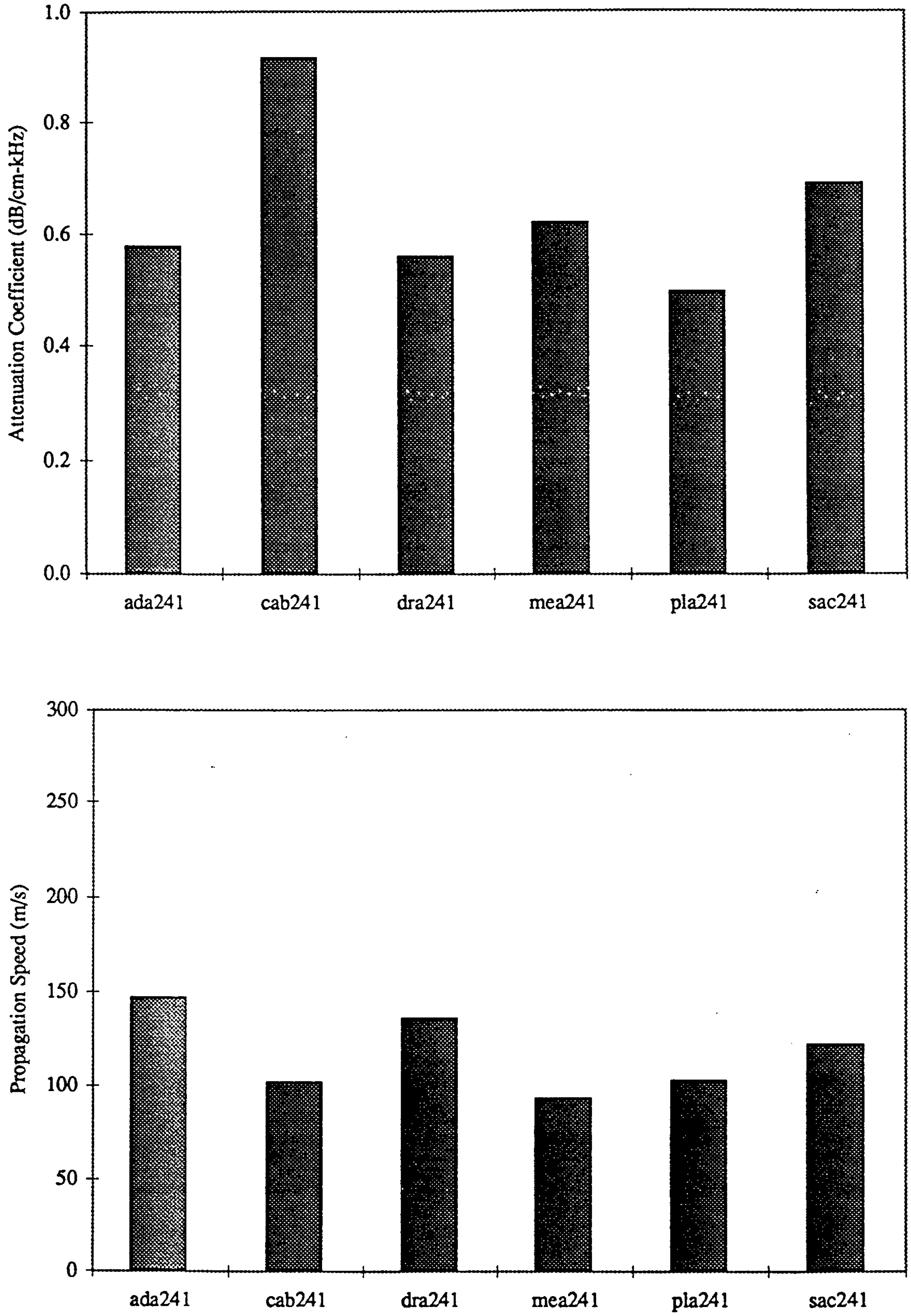
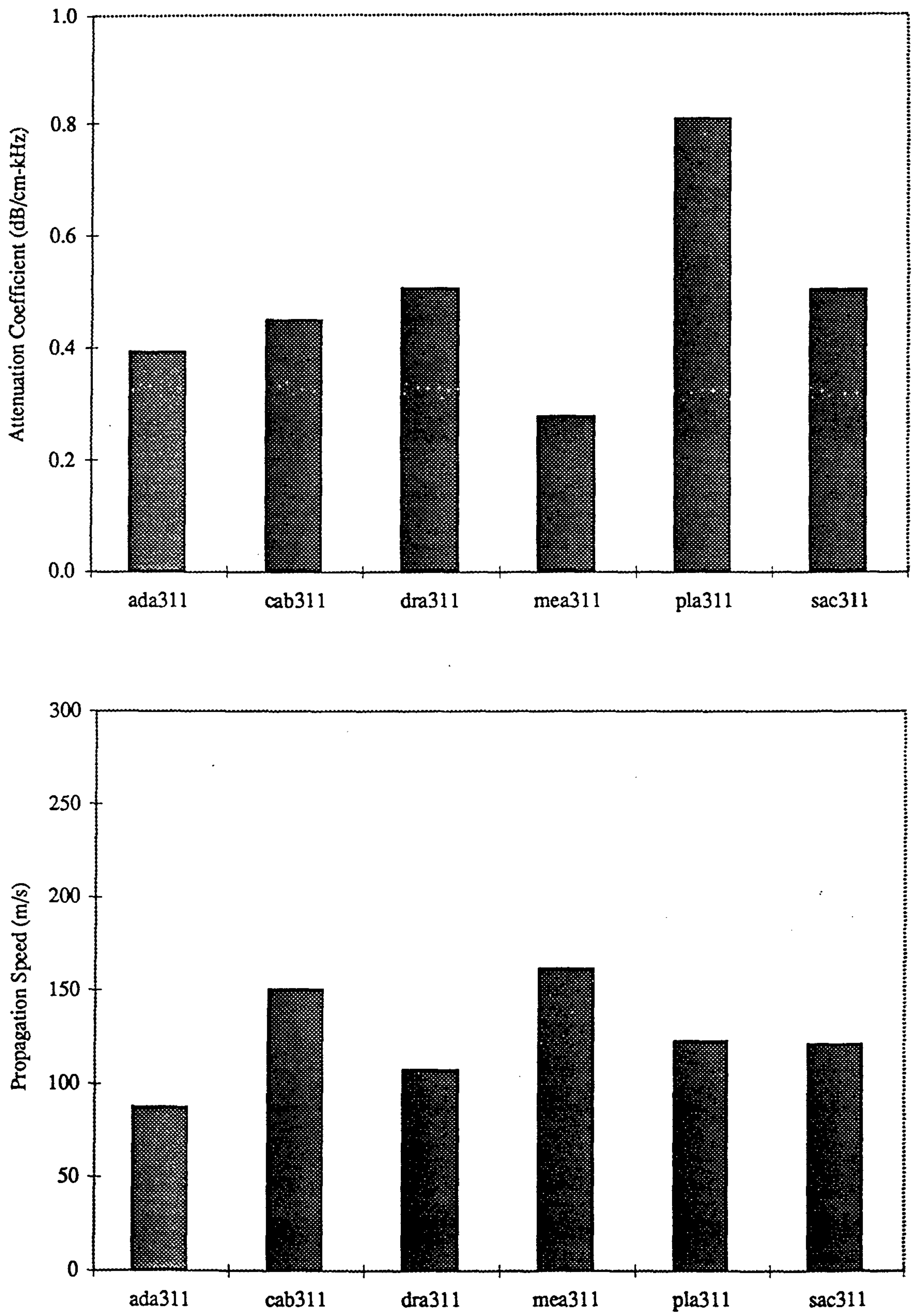

1.0
0.8
0.6
0.4
0.2
0.0
Attenuation Coefficient (dB/cm-kHz)
ada241
cab241
dra241
mea241
pla241
sac241
300
250
200
150
100
50
0
Propagation Speed (m/s)
ada241
cab241
dra241
mea241
pla241
sac241

Figure 6



1.0
0.8
0.6
0.4
0.2
0.0
Attenuation Coefficient (dB/cm-kHz)
ada311
cab311
dra311
mea311
pla311
sac311
300
250
200
150
100
50
0
Propagation Speed (m/s)
ada311
cab311
dra311
mea311
pla311
sac311

Figure 7

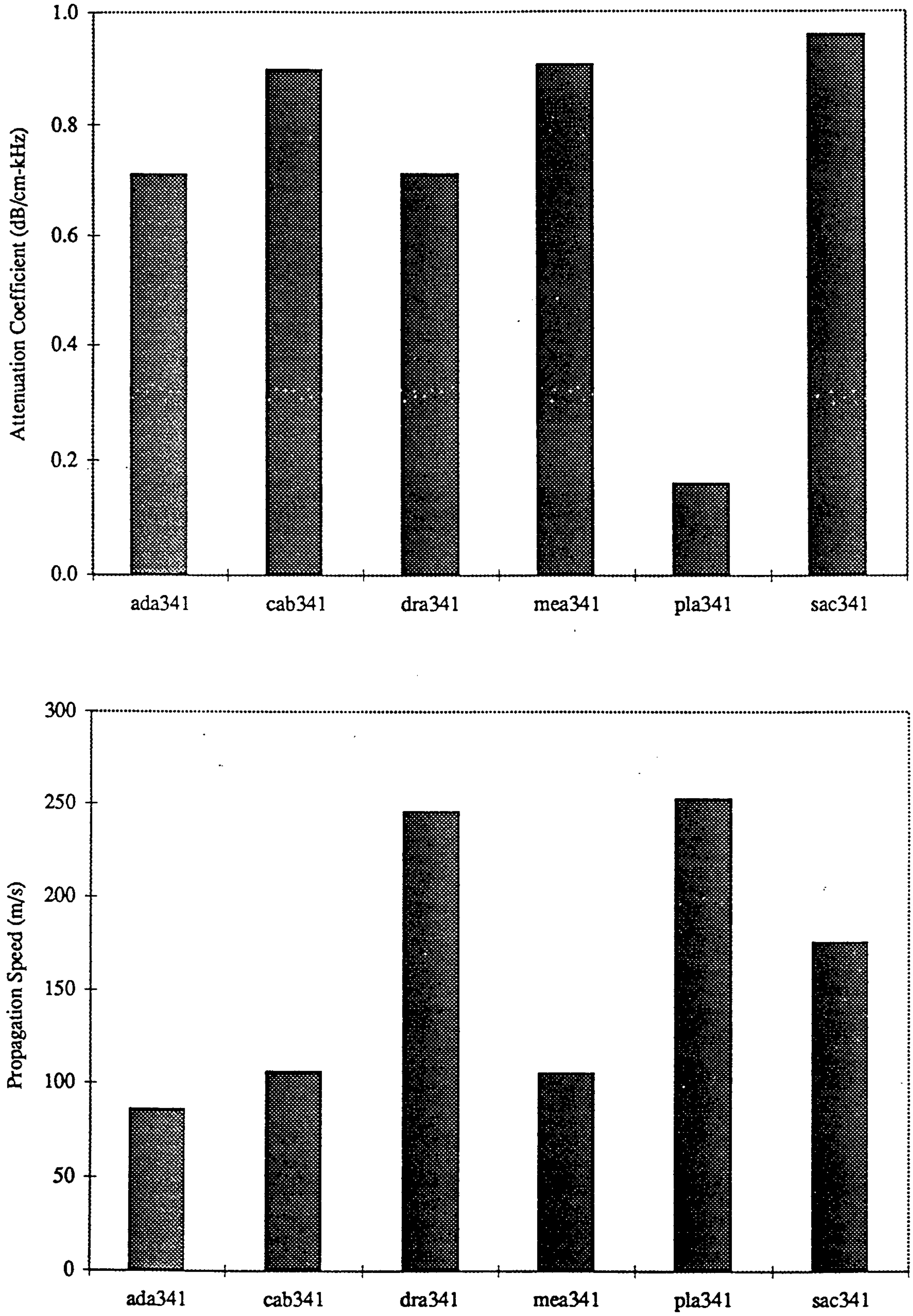

1.0
0.8
0.6
0.4
0.2
0.0
Attenuation Coefficient (dB/cm-kHz)
ada341
cab341
dra341
mea341
pla341
sac341
300
250
200
150
100
50
0
Propagation Speed (m/s)
ada341
cab341
dra341
mea341
pla341
sac341

Figure 8

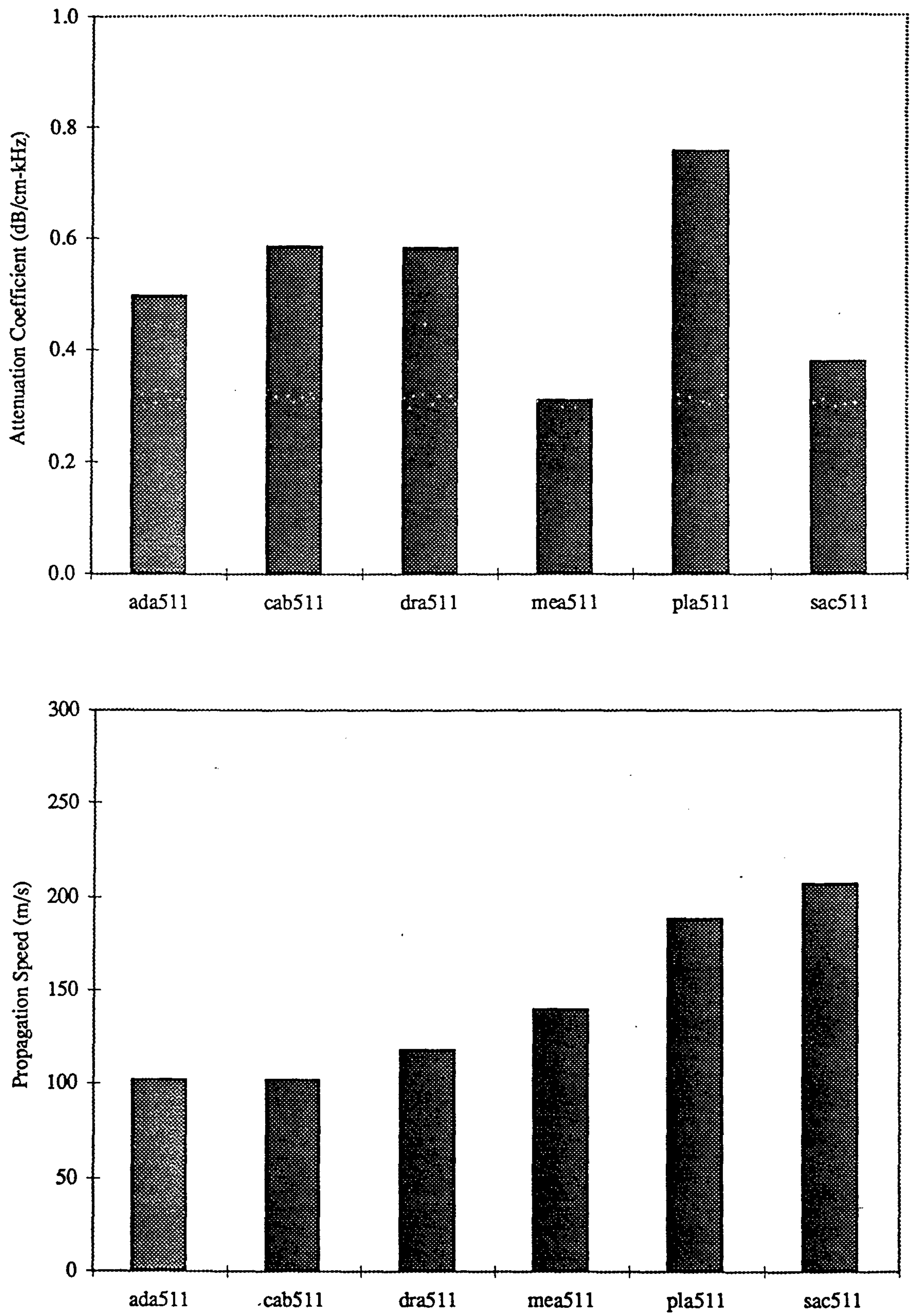
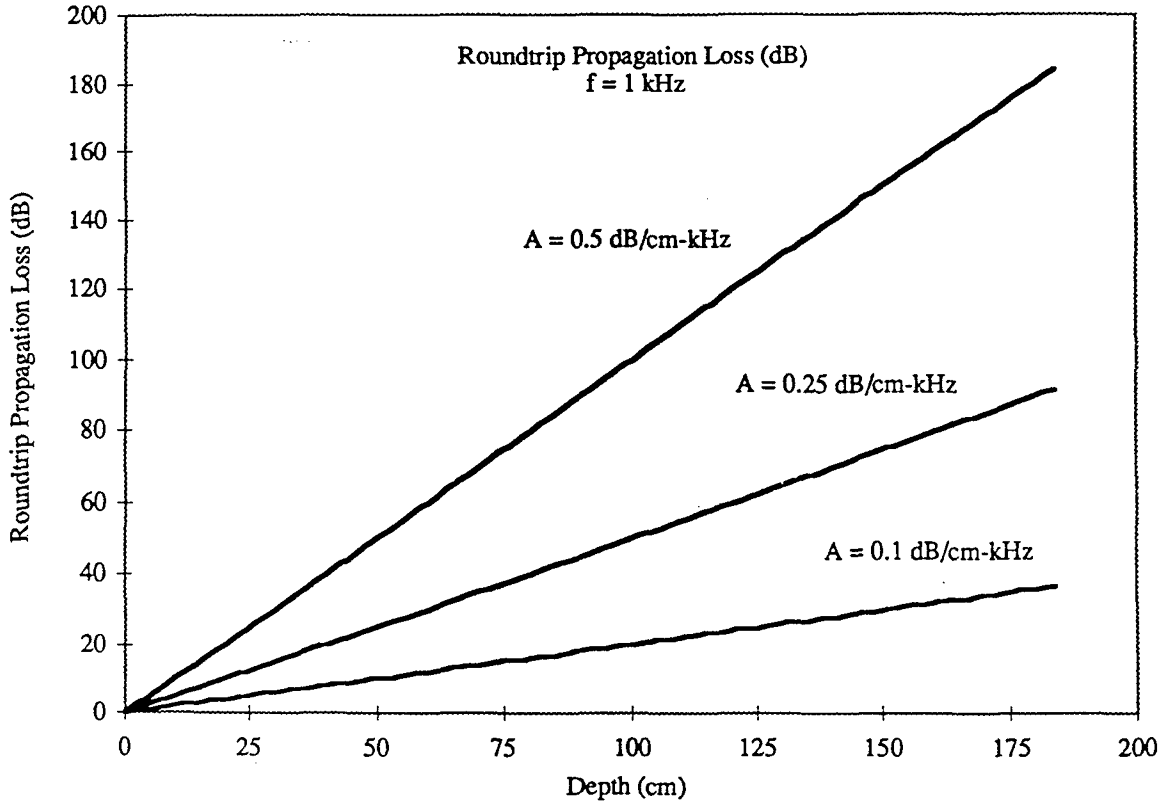

1.0
0.8
0.6
0.4
0.2
0.0
Attenuation Coefficient (dB/cm-kHz)
ada511
cab511
dra511
mea511
pla511
sac511
300
250
200
150
100
50
0
Propagation Speed (m/s)
ada511
cab511
dra511
mea511
pla511
sac511

Figure 9


Roundtrip Propagation Loss (dB)
f = 1 kHz
A = 0.5 dB/cm-kHz
A = 0.25 dB/cm-kHz
A = 0.1 dB/cm-kHz
Roundtrip Propagation Loss (dB)
Depth (cm)
0
20
40
60
80
100
120
140
160
180
200
0
25
50
75
100
125
150
175
200

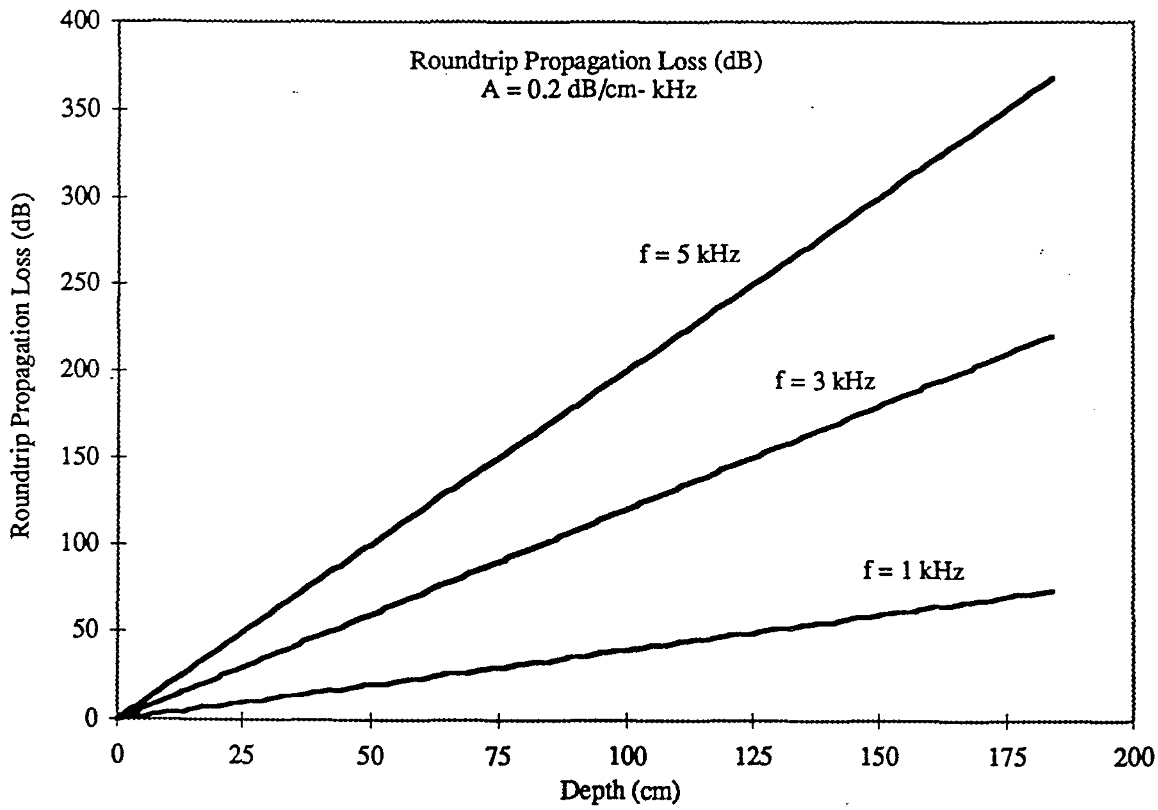

Roundtrip Propagation Loss (dB)
A = 0.2 dB/cm- kHz
f = 5 kHz
f = 3 kHz
f = 1 kHz
Roundtrip Propagation Loss (dB)
Depth (cm)
0
50
100
150
200
250
300
350
400
0
25
50
75
100
125
150
175
200

Figure 10

characteristic acoustic impedance in the range of 3.2 x 10^6 Pa-s/m (for Lucite: density = 1200 kg/m^3; propagation speed = 2650 m/s) whereas a metallic object would have a greater value of characteristic acoustic impedance. The pressure reflection coefficient, at normal incidence, is

$$\mathbf{R} = \frac{Z_{object} - Z_{soil}}{Z_{object} + Z_{soil}} \tag{1}$$

where, for Z_{object} = 3.2 x 10^6 Pa-s/m and Z_{soil} = 3 x 10^5 Pa-s/m, **R** = 0.826 (or -1.6 dB).

The imaging depth is, thus, inversely proportional to frequency which results in the important engineering tradeoff between depth of penetration into soil and image resolution. As a crude rule of thumb (dealt with more precisely in Section D), the image resolution can be approximated to that of the acoustic wavelength ($\lambda = c/f$) where c is the propagation speed and f is the acoustic frequency. Thus, as frequency increases, the wavelength decreases (resolution improves) and depth of penetration decreases.

A detailed analysis to the mechanisms responsible for the interaction of the propagated acoustic wave with soil is beyond the scope of the project. However, the initial hypothesis suggested that the acoustic propagation properties of in soil might be a function of soil moisture and compaction. A first order examination of the acoustic propagation properties as a function of moisture and compaction (see **Figures 11-16**) do not reveal any obvious trends.

D. SUB-SURFACE IMAGING

1. Analytical component

The subsurface image formation problem has multiple strategies. Whether performing conventional B-scan or using synthetic aperture techniques to create an image, the RF data are collected by scanning the transducer or the transducer's beam along a line. For B-scan imaging, the positions where the RF data are collected must be far enough apart that the signals can be considered independent of each other while still satisfying the spatial sampling criterion. Each returned (backscattered) acoustic echo is detected and the image is created by stacking the detected scans side by side so that one of the scan dimensions represents lateral position of the scan, and the other represents the axial depth into the medium, and the pixel brightness represents the amplitude of the detected acoustic echo. In this case, the ideal beam would be very narrow to yield good lateral resolution, and the ideal transmitted pulse would have narrow time duration to yield good axial resolution. Within certain limits of approximation, lateral resolution is that of the beamwidth and described mathematically at the focus as

$$\text{Lateral Resolution} = \left(\frac{\text{FL}}{\text{D}}\right)\lambda = \text{f/\#}\,\lambda \tag{2}$$

where FL is the focal length, D is the source diameter, λ is the acoustic wavelength and f/# is the f-number (ratio of focal length to source diameter). In practical situations, the beam is narrow only over a small depth near its focus. This means that only part of the image is actually in focus. To compensate for this, the received echoes can be dynamically focused on receive, thus creating multiple focal regions, each described by a different lateral resolution. **Figure 17** schematically shows three focal regions for the same source. The final image results from a composite of multiple focal regions, thus optimizing (minimizing) the lateral resolution along a single B-scan image line.

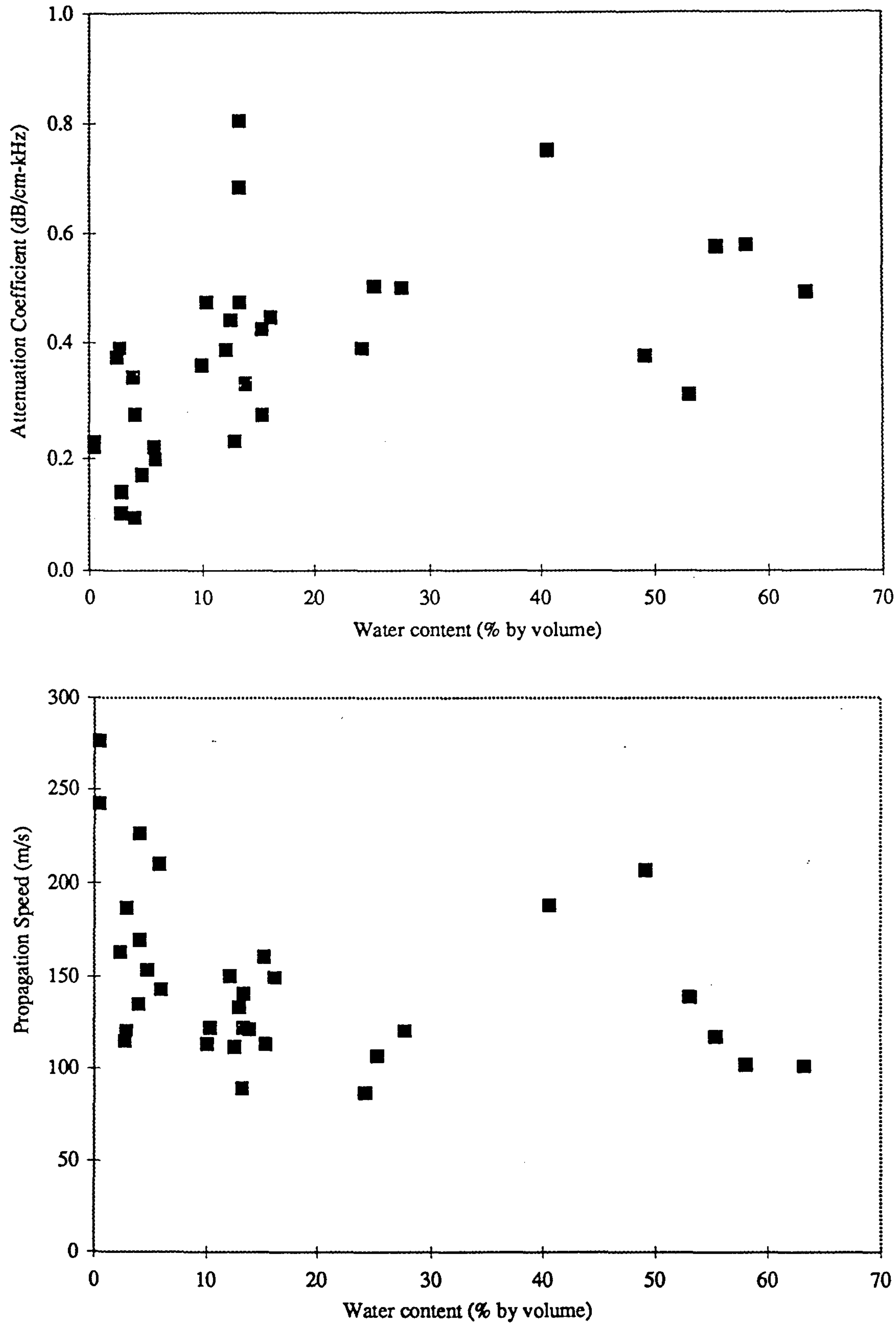
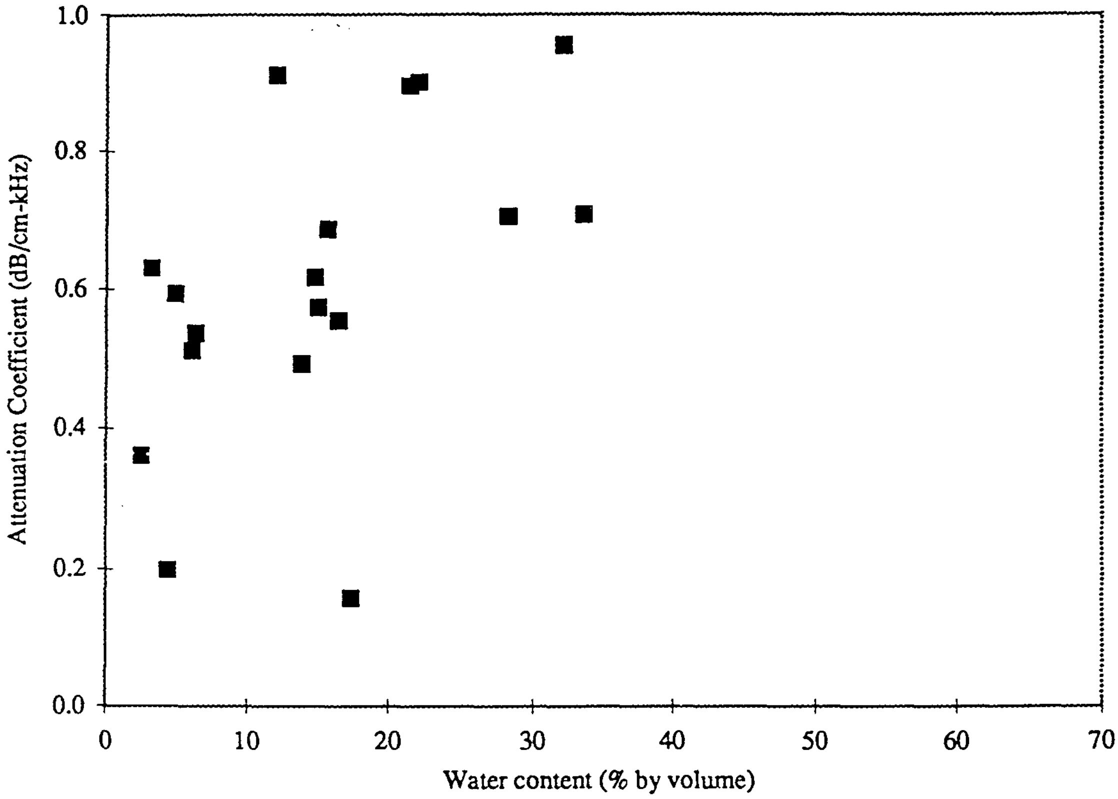

1.0
0.8
0.6
0.4
0.2
0.0
Attenuation Coefficient (dB/cm-kHz)
0
10
20
30
40
50
60
70
Water content (% by volume)
300
250
200
150
100
50
0
Propagation Speed (m/s)
0
10
20
30
40
50
60
70
Water content (% by volume)

Figure 11


1.0
0.8
0.6
0.4
0.2
0.0
Attenuation Coefficient (dB/cm-kHz)
0
10
20
30
40
50
60
70
Water content (% by volume)

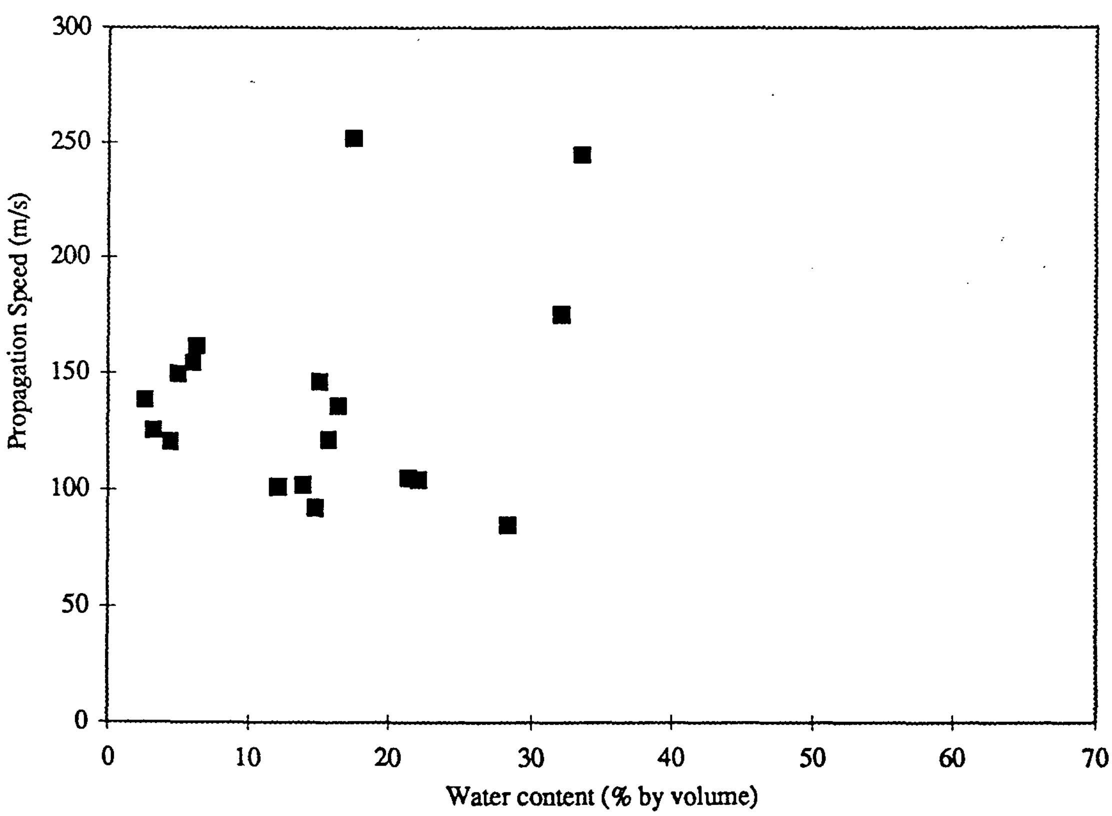
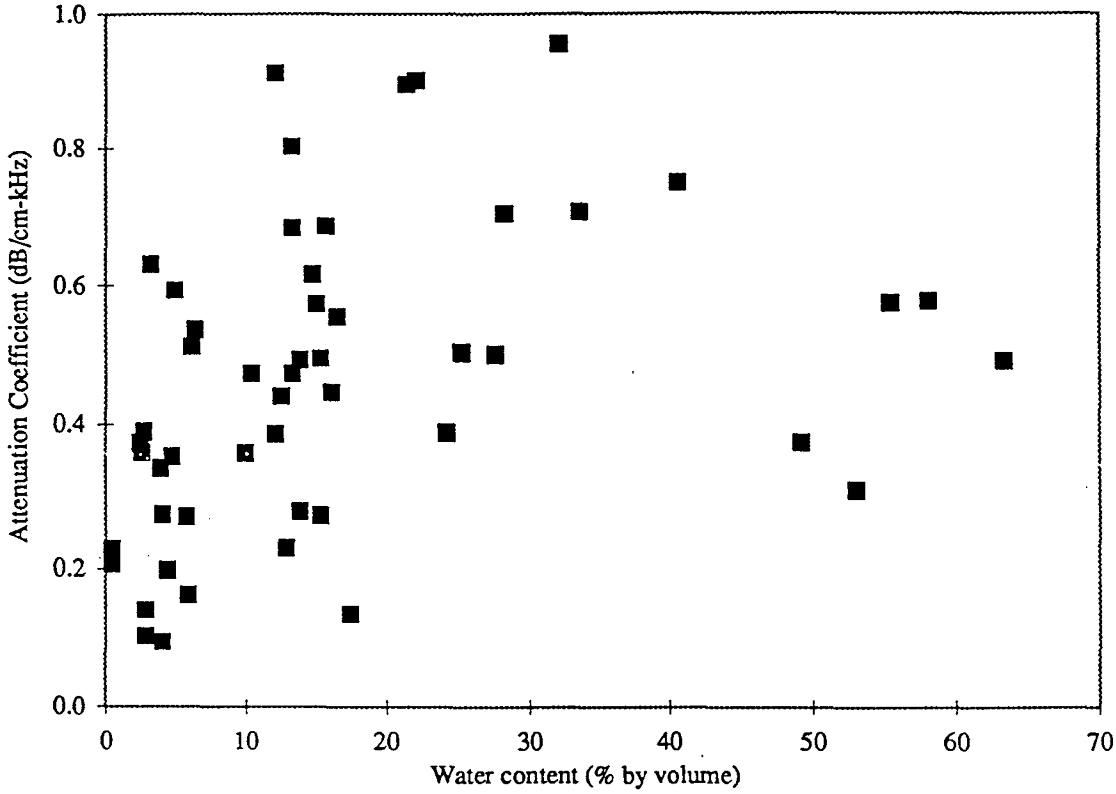

300
250
200
150
100
50
0
Propagation Speed (m/s)
0
10
20
30
40
50
60
70
Water content (% by volume)

Figure 12


1.0
0.8
0.6
0.4
0.2
0.0
Attenuation Coefficient (dB/cm-kHz)
0
10
20
30
40
50
60
70
Water content (% by volume)

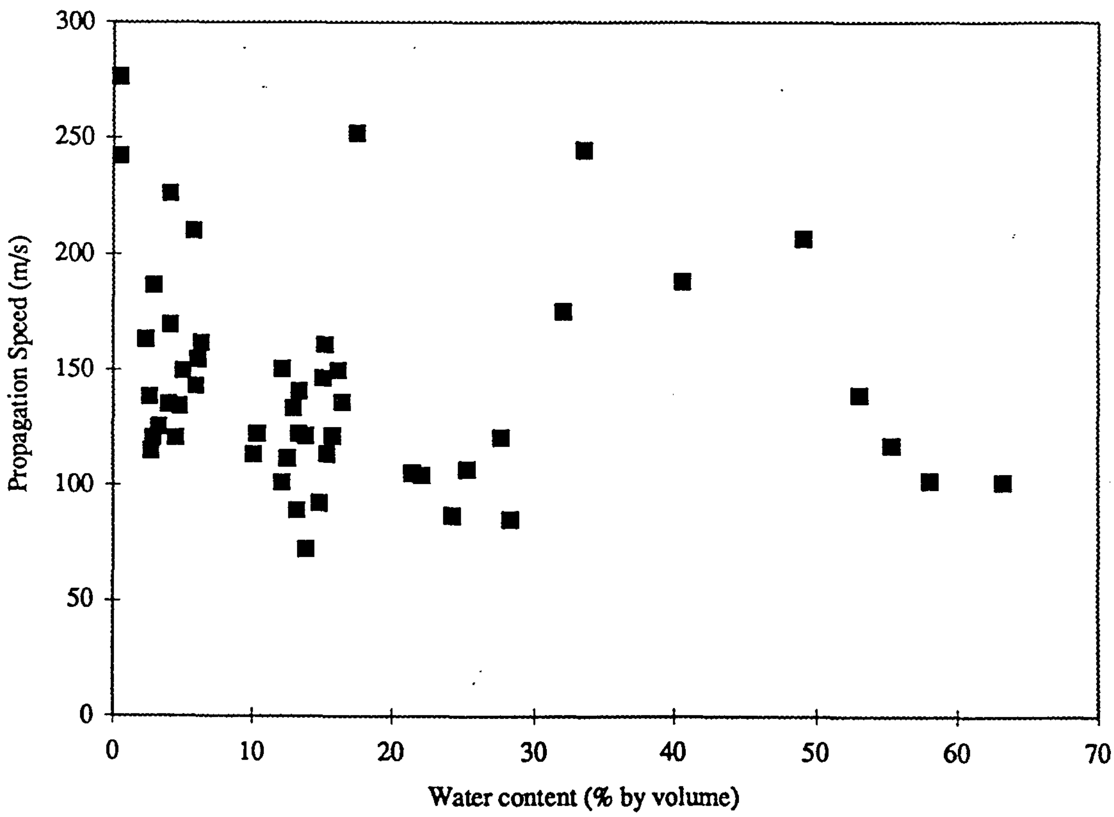
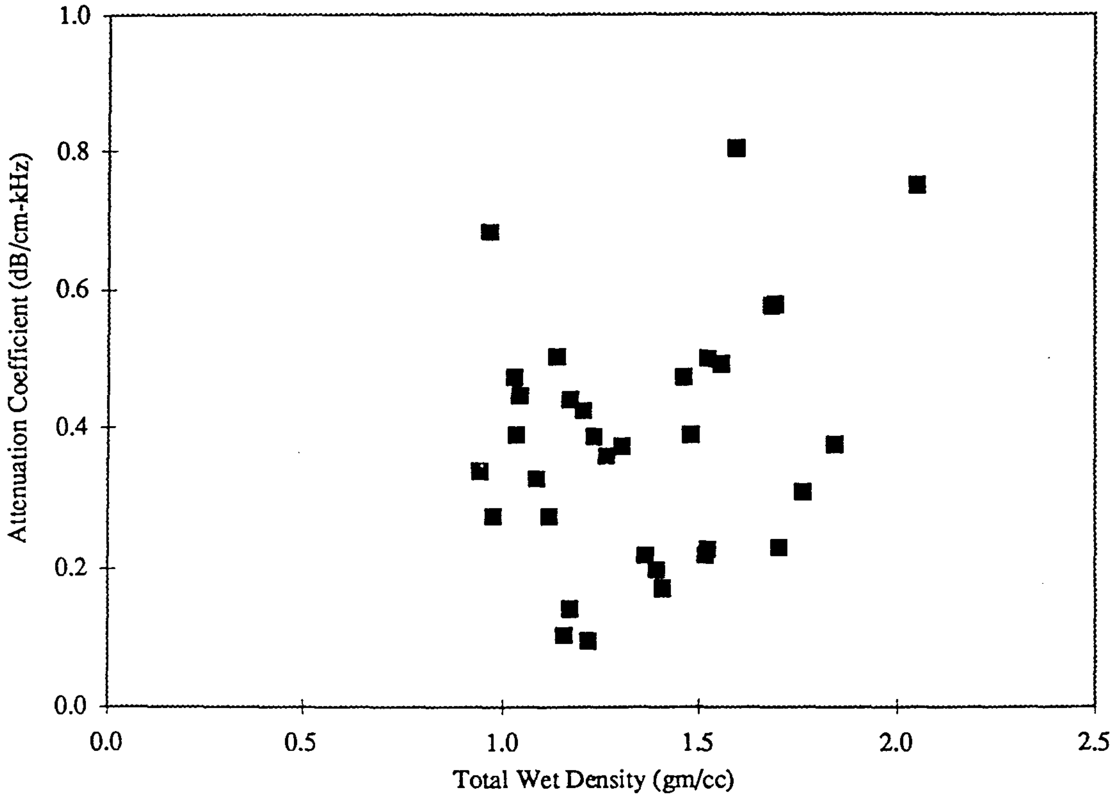

300
250
200
150
100
50
0
Propagation Speed (m/s)
0
10
20
30
40
50
60
70
Water content (% by volume)

Figure 13


1.0
0.8
0.6
0.4
0.2
0.0
Attenuation Coefficient (dB/cm-kHz)
0.0
0.5
1.0
1.5
2.0
2.5
Total Wet Density (gm/cc)

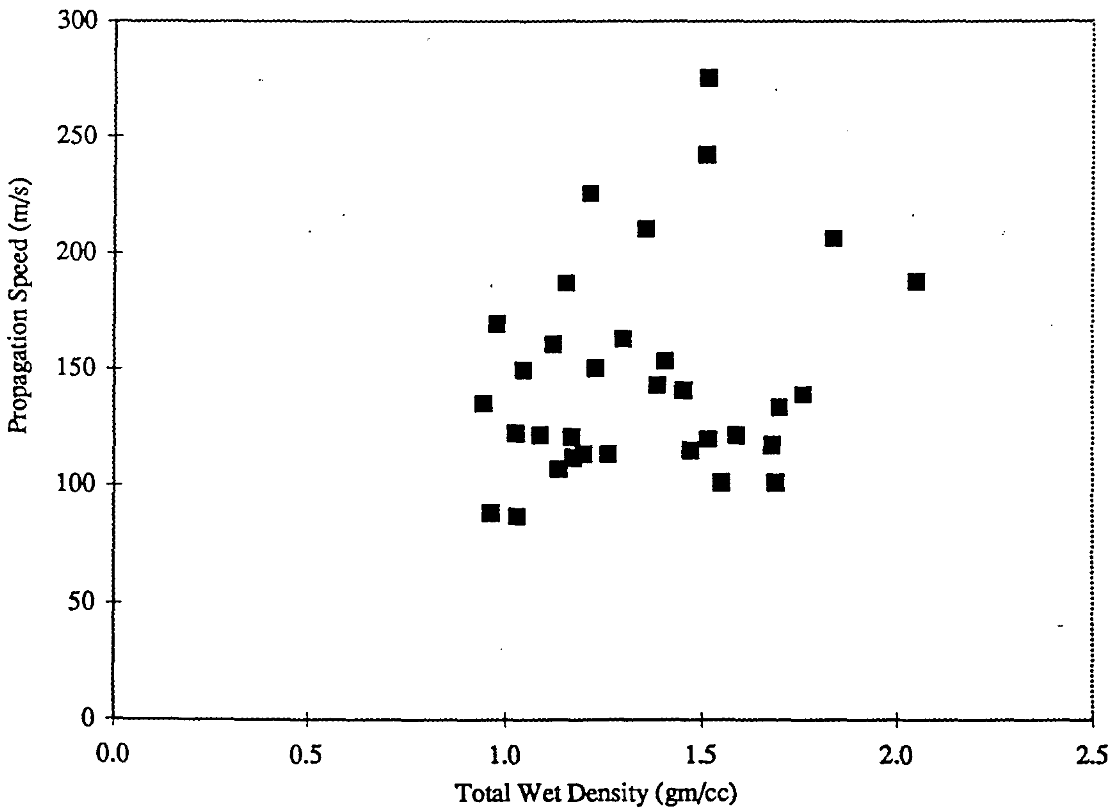
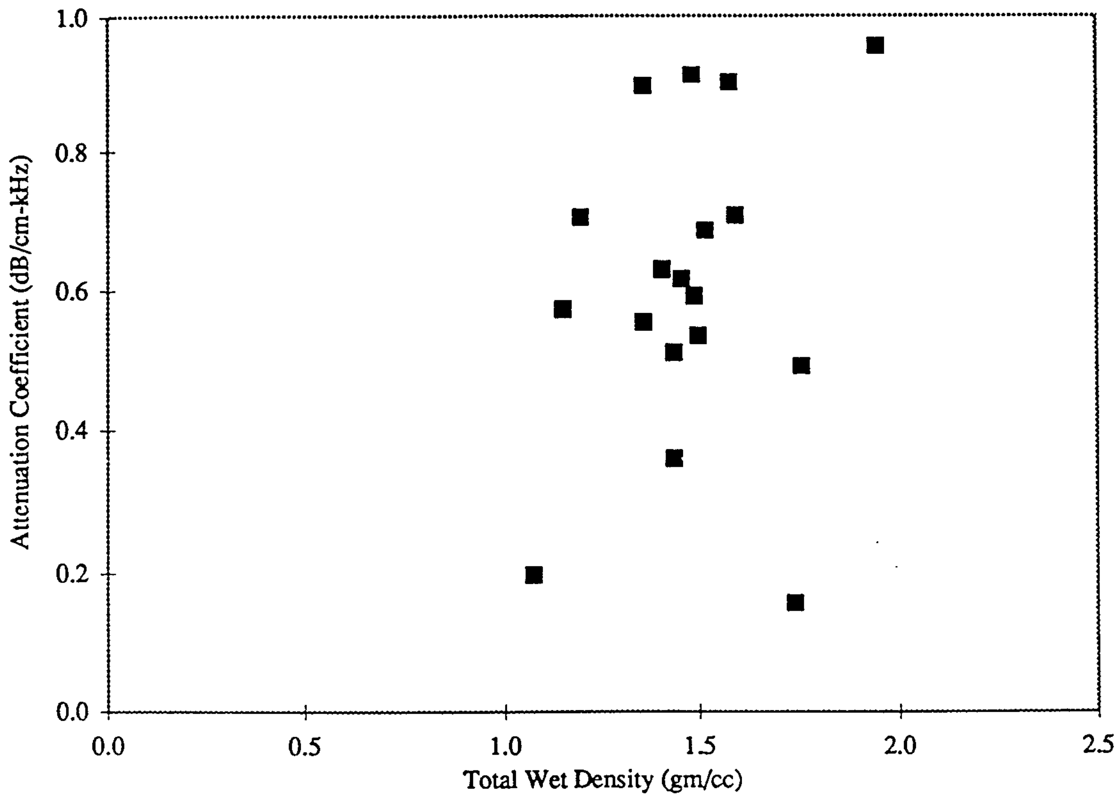

300
250
200
150
100
50
0
Propagation Speed (m/s)
0.0
0.5
1.0
1.5
2.0
2.5
Total Wet Density (gm/cc)

Figure 14


1.0
0.8
0.6
0.4
0.2
0.0
Attenuation Coefficient (dB/cm-kHz)
0.0
0.5
1.0
1.5
2.0
2.5
Total Wet Density (gm/cc)

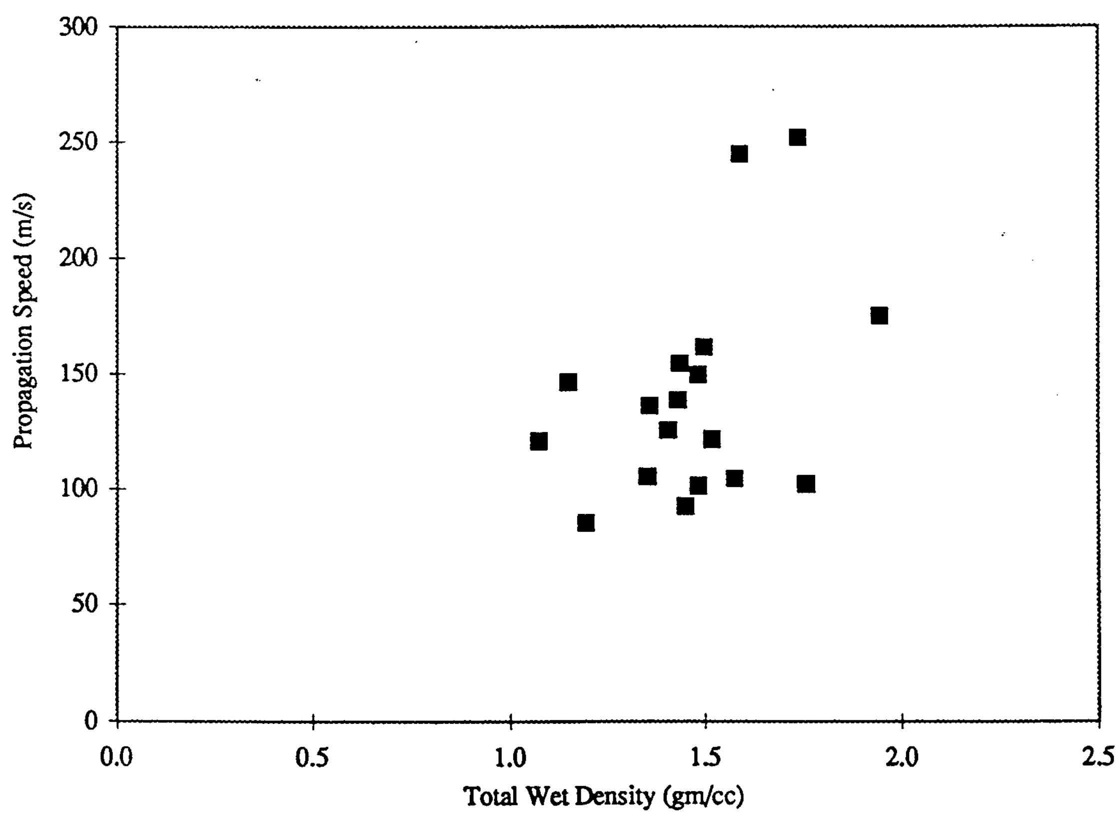
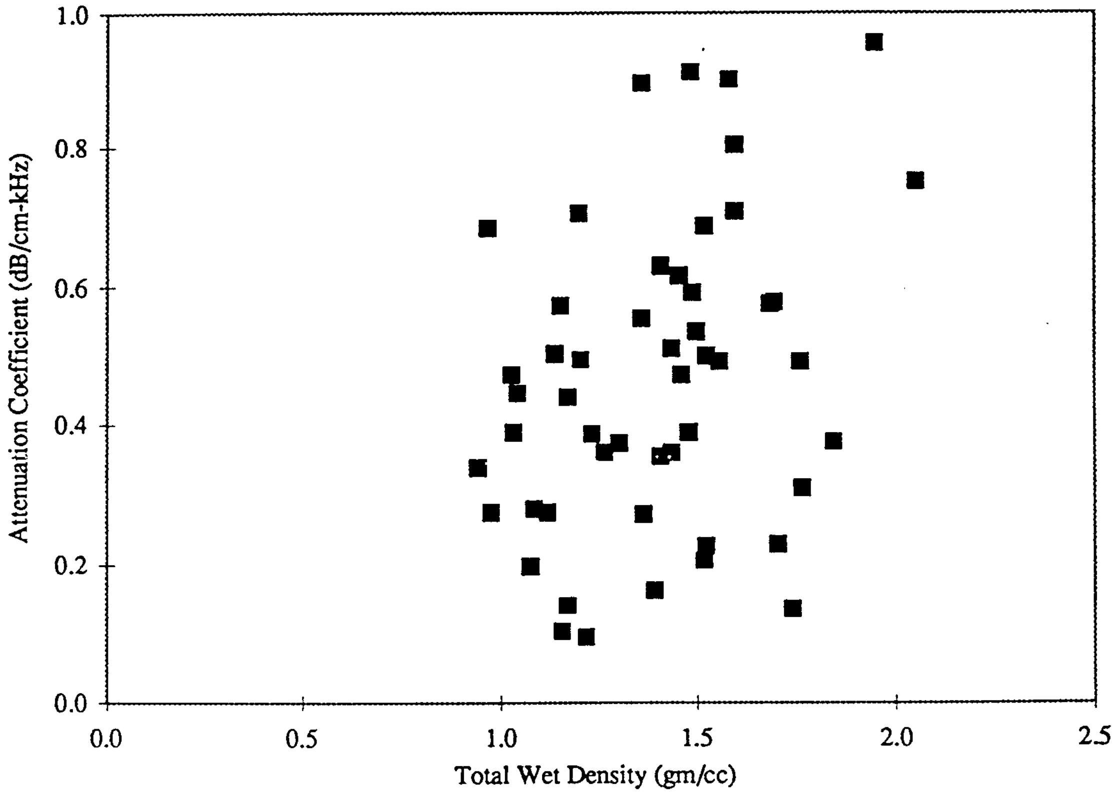

300
250
200
150
100
50
0
Propagation Speed (m/s)
0.0
0.5
1.0
1.5
2.0
2.5
Total Wet Density (gm/cc)

Figure 15


1.0
0.8
0.6
0.4
0.2
0.0
Attenuation Coefficient (dB/cm-kHz)
0.0
0.5
1.0
1.5
2.0
2.5
Total Wet Density (gm/cc)

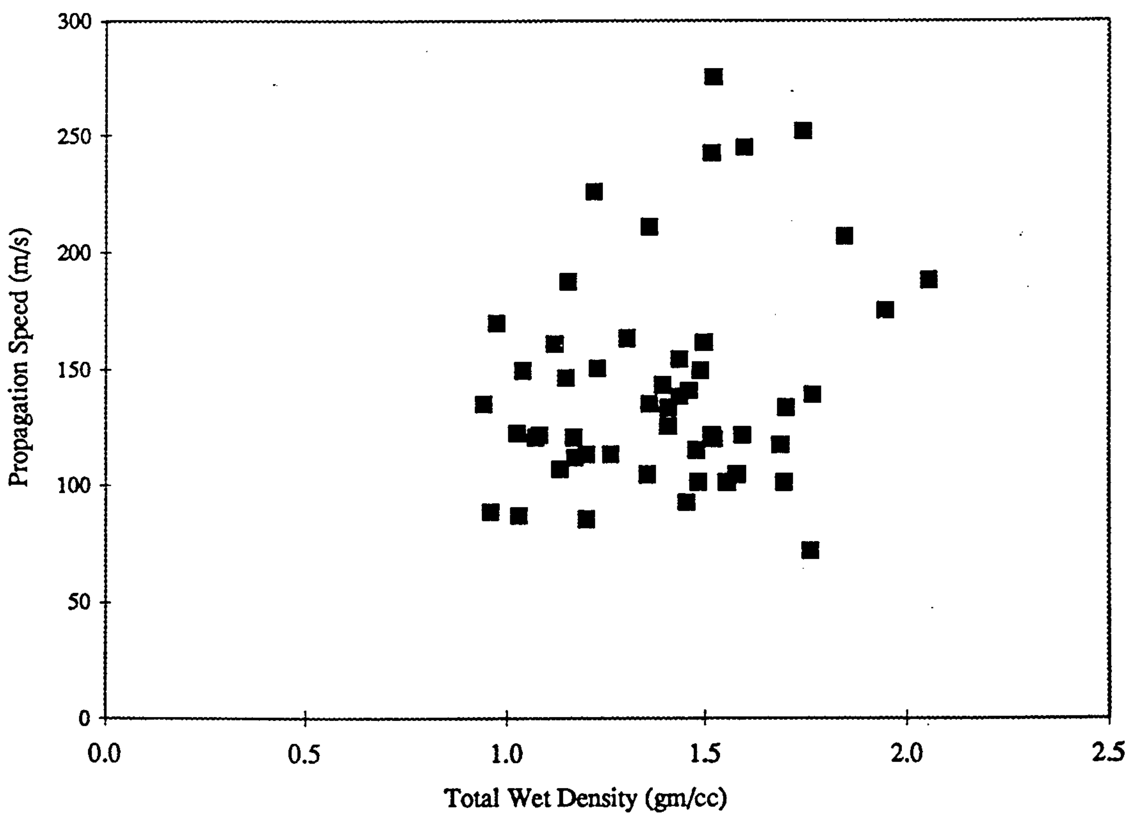

300
250
200
150
100
50
0
Propagation Speed (m/s)
0.0
0.5
1.0
1.5
2.0
2.5
Total Wet Density (gm/cc)

Figure 16

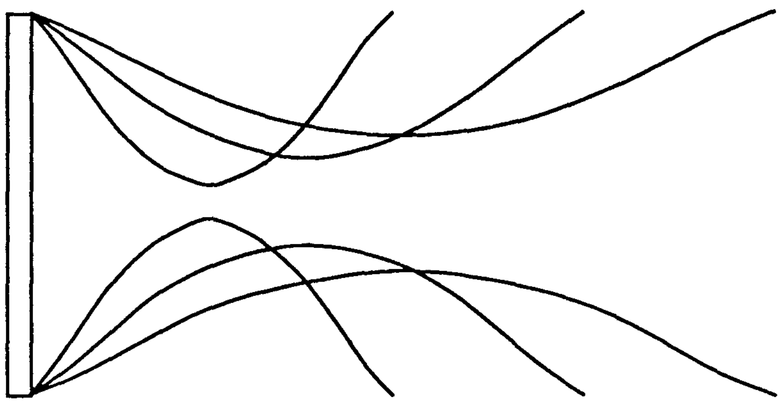


Figure 17

Also within certain limits of approximation, axial resolution is that of the space occupied by an acoustic pulse and described mathematically as

$$\text{Axial Resolution} = Q\lambda = \left(\frac{f}{\Delta f}\right)\lambda = \frac{c}{\Delta f} \tag{3}$$

where the system Q is the ratio of center frequency f to system bandwidth Δf and c is the acoustic propagation speed.

In the synthetic aperture case, the RF data are collected using a transducer with a *wide* beam pattern and a correspondingly *small* aperture. Again, the ideal pulse has a narrow time duration (or large bandwidth). RF data are collected at lateral positions close enough together that the area of interest is illuminated by many transmit beams. In order to synthesize the wide aperture of a focused transducer, several neighboring beams are coherently summed. Obtaining a focused beam at a given position is a matter of making sure the beams add constructively at that point or, in other words, have the same phase at that point. This can be accomplished by delaying the beams near the center of the sampled aperture. It is proposed to adapt signal processing techniques from synthetic aperture radar (SAR).

SAR is a microwave imaging technique that produces imagery having better lateral resolution than that dictated by the source's beamwidth. In the most widely known form of SAR, called strip-mapping, the source's beam sweeps out a strip on the ground, a sequence of closely spaced pulses is transmitted, and the returned waveforms are recorded (Munson and Visenten, 1989). The source moves only a small fraction of its beamwidth between pulse transmissions so that each point in the scene is illuminated by many pulses. The returned RF signals are correlated in the range direction with a replica of the transmitted signal, and in the cross-range direction with a linear FM waveform to produce a high-resolution radar image. Within certain limits of approximation, utilizing this form of processing with a source diameter D yields lateral resolution of D/2. Thus, a smaller source actually provides *BETTER* lateral resolution! It is our hypothesis that a similar form of processing can be successful in the acoustic scenario. For example, using a linear array of acoustic sources, and moving the array across the soil surface, it should be possible to obtain high lateral resolution in the direction of travel of the array. The traditional SAR imaging algorithm must be modified, however, to account for signal attenuation in the soil, and to counter the effects of noise. For near-field imaging, we plan to investigate a modified SAR imaging algorithm.

In the acoustic subsurface imaging scenario, the "object" to be imaged (*i.e.*, cultural artifacts in soil) will literally abut the transducer, probably through a water interface. Thus, imaging at small depths will take place in the near field of the transducer. In the case of near-field imaging, the Fourier approximation breaks down, so that the typical correlation-based imaging algorithm for SAR will fail. For near-field SAR-based imaging we suggest a modified version of the so-called wavenumber approach (Cafforio *et al.*, 1991; Choi and Munson, submitted). The wavenumber algorithm is equivalent to the wave migration method in seismic data processing. This technique has the advantage that wavefront curvature is incorporated into the imaging model so that superb image reconstructions can be produced at close ranges. The computational demands of this algorithm are comparable to those for the correlation-based approach.

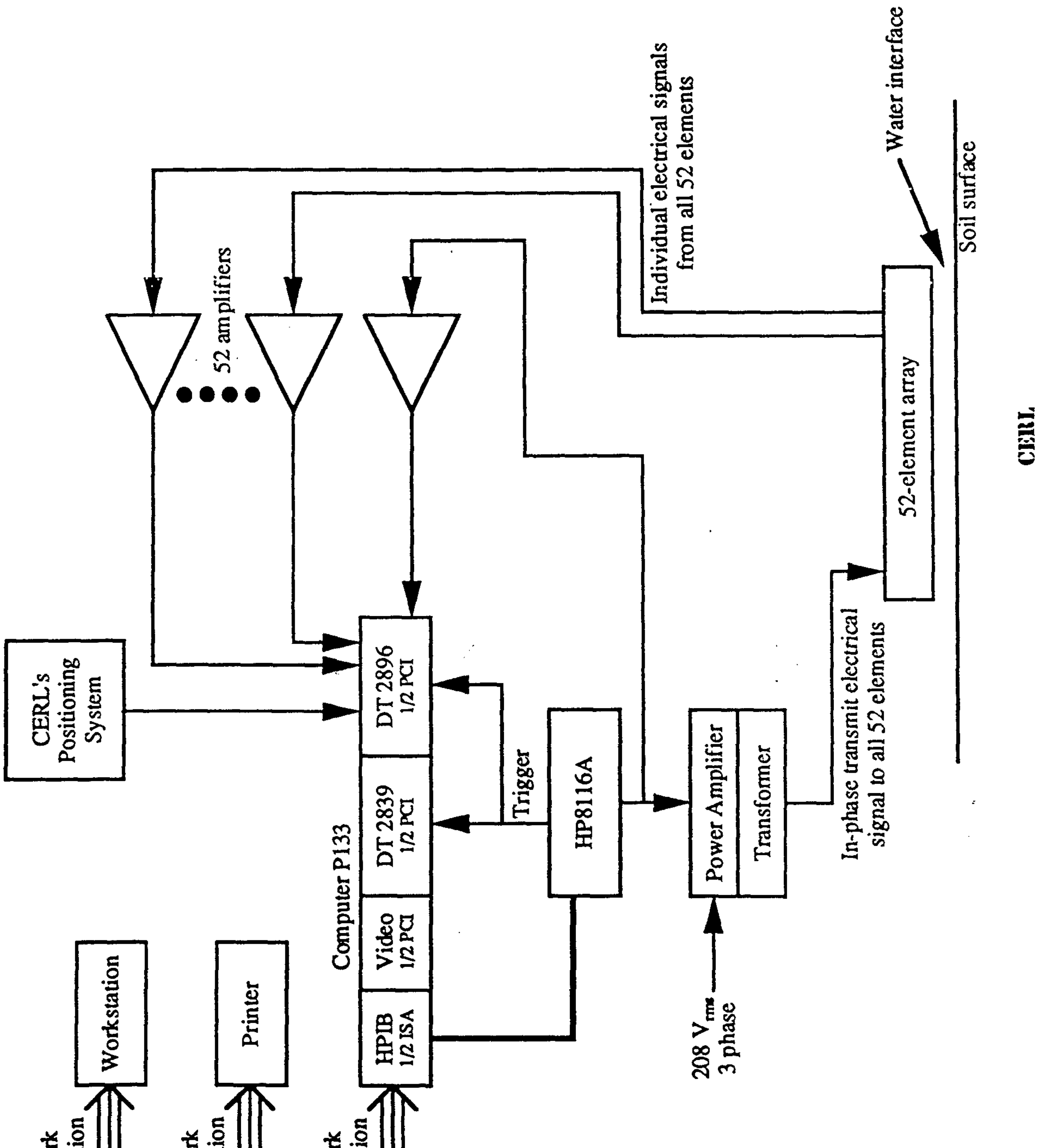
The SAR technique may be confounded by noise introduced in the subsurface imaging scenario. To combat this potential difficulty we can utilize a new SAR imaging algorithm (Lee *et al.*, 1996) derived from a geophysics approach used to estimate the conductivity distribution from wellbore induction measurements. This approach to SAR imaging implements a full nonseparable two-dimensional inversion of the SAR measurement equation. Using the spectral representation of the SAR measurement kernel, this approach decomposes the SAR imaging problem into a set of integral equations, described by a nonseparable convolution in one dimension and a projection in the other dimension. The inversion is performed in two stages: multichannel deconvolution, followed by back-projection. By dropping small singular values in the spectral representation and also using a regularized deconvolution filter, this approach to SAR imaging is inherently robust to the presence of noise.

A third image processing strategy is the same conventional B-scan imaging. The synthetic aperture processing offers the advantage that the summed beams can be focused at any depth; therefore, the entire image can be in focus. However, the disadvantage of synthetic aperture imaging is that the signal to noise ratio is low. These two techniques will be combined and evaluated. At lower acoustic frequencies, with a wider beam pattern and a correspondingly smaller aperture, a synthetic approach is more likely whereas at higher acoustic frequencies, with a more collimated beam pattern, a B-Scan approach is more logical.

2. Experimental component

Professor O'Brien attended the Ultrasound Transducer Workshop at Penn State, August 16-18, 1995, as an invited speaker. There he made valuable contacts with engineers and scientists at Penn State's Applied Research Laboratory. This laboratory has a major activity in the design and development of high-power sonar transducers in our acoustic frequency range, mostly arrays. The cooperation with Penn State's Applied Research Laboratory provides valuable expertise with the two-dimensional acoustic transduction device and some of the support electronics including RF signal access to the electrical signals received at each of the 52 elements. In addition, significant expertise exists in the quantitative and qualitative evaluation of the imaging system's overall development and performance (see **Figure 18**).

A 52-element sonar array (8x8 3.56 cm^2 close-packed elements with 3 elements in each corner missing) can be driven in phase to produce the transmit acoustic source. This procedure will produce essentially a transmit plane wave. This is a cost-effective means to evaluate the feasibility of subsurface imaging. A more complete (and costly) system could include individual control of each transmit element, thus allowing for transmit focusing and beam steering.



Network Connection
Network Connection
Network Connection
Workstation
Printer
CERL's Positioning System
Computer P133
HPIB 1/2 ISA
Video 1/2 PCI
DT 2839 1/2 PCI
DT 2896 1/2 PCI
52 amplifiers
Trigger
HP8116A
Power Amplifier
Transformer
208 V_{rms} 3 phase
In-phase transmit electrical signal to all 52 elements
Individual electrical signals from all 52 elements
52-element array
Water interface
Soil surface
CERL

Figure 18

On receive, the backscattered echo signals from each of the 52 elements will be amplified and digitized. Transmit electronics are available from a previously funded CERL Project (DACA88-94-D-0008). The sonar array with transmit/receive switches and receive preamps on each element can be modified by Penn State's Applied Research Laboratory. The receive digitizing capability will be purchased as part of this project and consists of one Data Translation DT2839 A/D boards capable of 100 MHz throughput with the existing 133 MHz Pentium PC in combination with a DT 2896 Channel Expander capable of switching between and digitizing the received echoes from the 52 elements. This system operates at a 12-bit digitizing rate of 1 MHz to provide the operational A/D rate per channel up to 1 MHz. Therefore, this cost-effective A/D arrangement will collect backscattered echo signals from all 52 elements. This proposed approach will require multiple (52) transmit-receive pulses since the digitizing capability will acquire one echo for each transmit pulse. Again, this is a cost-effective means to evaluate the imaging feasibility. A more complete system would include the capability to acquire the electrical signals from each transducer element for a single transmit pulse, thus decreasing the imaging data acquisition time and providing the capability to average received signals to improve the signal-to-noise ratio.

Digitizing the receive echo signals permits off-line processing for dynamic focusing on receive. **Table 5** provides one-way receive focusing properties of the 52-element sonar array assuming an active aperture diameter of 28 cm in a medium with a propagation speed of 250 m/s. The regions denoted by dashes in **Table 5** do not permit focusing under the indicated conditions (focusing must occur in the Fresnel region of the field) and thus, in these regions, synthetic aperture processing is more logical to optimize lateral resolution.

Table 5. Beam properties for a 28 cm active aperture diameter in a medium with a propagation speed of 250 m/s. A 20% bandwidth is assumed for axial resolution.

				Approximate Lateral Resolution at Focus (cm)				Axial
f (kHz)	ka	λ (cm)	a^2/λ (cm)	f/1 FL=28 cm	f/1.5 FL=42 cm	f/2 FL=56 cm	f/2.5 FL=70 cm	Resolution (cm)
1	3.5	25	7.8	—	—	—	—	1.3
2	7.0	13	16	—	—	—	—	0.63
3	11	8.3	24	—	—	—	—	0.42
4	14	6.3	31	6.3	—	—	—	0.31
5	18	5.0	39	5.0	—	—	—	0.25
6	21	4.2	47	4.2	6.3	—	—	0.21
7	25	3.6	55	3.6	5.4	—	—	0.18
8	28	3.1	63	3.1	4.7	6.3	—	0.16
9	32	2.8	71	2.8	4.2	5.6	—	0.14
10	35	2.5	78	2.5	3.8	5.0	6.3	0.13
11	39	2.3	86	2.3	3.4	4.5	5.7	0.11
12	42	2.1	94	2.1	3.1	4.2	5.2	0.10
13	46	1.9	102	1.9	2.9	3.8	4.8	0.096
14	49	1.8	110	1.8	2.7	3.6	4.5	0.089
15	53	1.7	118	1.7	2.5	3.3	4.2	0.083

At a digitizing rate of 200 kHz (at least 10 times greater that the highest imaging frequency), and assuming a soil propagation speed of 250 m/s, 400 bytes of data are collected per meter depth (denoted a single A-line). Assuming 1 kB/A-line (2.5 m depth; approximately 8.2 feet), data acquisition from one array position yields 52 digitized A-lines and 52 kB/position.

The array can be moved with two geometries, one in a straight line to yield a 2-D image (denoted B-mode image) and the other in a 2-D pattern to yield a 3-D image. The spacing between array positions should be about 2 cm to take full advantage of independent A-lines (note: beam diameter at focus is about 2 cm at 15 kHz for f/1 receive focusing, see **Table 5**) for B-mode image processing and possibly a higher density (<2mm spacing) for synthetic array image processing. About 10.4 MB of data (200x52 kB) are collected for 2-D, B-mode imaging with 200 array positions (potential 2-D field of view: 4 m horizontally x 2.5 m in depth), whereas about 520 MB of data (100x100x52 kB) are collected for 3-D imaging with a 100 x 100 square 2-D pattern (potential 3-D field of view: 2 x 2 m horizontally x 2.5 m in depth). The advantage of the 3-D imaging data acquisition procedure is that not only multiple 2-D images perpendicular to the soil surface can be generated but also multiple 2-D horizontal images parallel to the soil surface (denoted C-mode images) at any soil depth can be generated.

Test bed recommendations It is recommended that CERL's test imaging field include a series of 2 cm and 5 cm diameter metal pipes separated at various distances horizontally and vertically and at various depths to provide high contrast targets so that axial and lateral resolutions can be estimated experimentally. Further, it is recommended that CERL's test imaging field include a similar series of thin-walled plastic containers filled with materials of various acoustic scattering properties to estimate experimentally contrast resolution. A 3-D positioning system will be required to support the 300-pound array and be capable of moving the array in about 2 cm increments with a precision of about 2 mm (about a factor of 10 better than the 2 cm incremental movement). The top surface of the test bed should be water (depth about 6-8 inches) to provide acoustic coupling between the array source and soil. Finally, the array positioning system should be interfaced with a computer to precisely move and record the position. If possible, it would be desirable to have the positioning system interfaced to the data acquisition system. Also, the drive electronics for the 52-element array require a 208 volt, three-phase line.

E. FUTURE PLANS

The long-term objective of the proposed research program is to *classify* precisely subsurface cultural artifacts through acoustic imaging means. Six generalized research phases are envisioned to accomplish the long-term objective. **Phase 1** is the currently funded activity program (CERL Contract Number DACA88-94-D-0008, "Acoustical Characterization of Soil") and is providing base-line evaluations of acoustic properties of a limited set of soils. **Phase 2** evaluates the feasibility for subsurface detection of cultural artifacts using various acoustic imaging approaches under laboratory, test-bed conditions and to initiate studies of *in situ* soil acoustic properties. **Phase 3** optimizes both the imaging system and the image formation algorithms for detection of cultural artifacts under laboratory, test-bed conditions. **Phase 4** develops classification strategies to classify precisely subsurface cultural artifacts under laboratory, test-bed conditions, and to test detection strategies under *in situ*, field conditions. **Phase 5** evaluates and optimizes detection and classification strategies under selected *in situ*, field conditions This phase consists of developing the acoustic imaging system on an appropriate mobile platform. **Phase 6** evaluates detection and classification capabilities under a wide variety of *in situ*, field conditions, and modifies these capabilities as appropriate.

All of these phases require continued evaluation of the acoustic propagation and scattering properties of soils. Thus, it is proposed to continue these fundamental acoustic characterizations of well-characterized soil samples to develop a better understanding of the acoustical properties under more typical soil conditions. The acoustic imaging feasibility evaluation in CERL's test bed will be conducted in one type of soil whereas there are numerous and widely varying soil conditions around the world. The increased understanding of soil's acoustic properties will permit a fundamental basis for extrapolating imaging capabilities in soil types that have not been evaluated in a soil test bed.

It is proposed to add carbonate content, plastic limit, liquid limit, maximum density and coefficient of linear extensibility (COLE) testing to the analyses currently being done on the soil samples currently under evaluation. Further, it is proposed to drop the soil characteristics that do not correlate with acoustic properties and re-evaluate using those that do correlate along with the additional soil characteristics. These analyses are critical for assessing whether the soil properties can provide a fundamental basis for estimating the soil's acoustic properties and hence the acoustic imaging capabilities. The additional analyses will be conducted on the current 6 soil types.

A second component to be investigated is the effect of soil layering which can be performed with the soil samples that have already been collected and evaluated, and with the existing measurement system. Here, the layering would be accomplished with a thin plastic film separating the layers. The soil layers in a single study would include different soils (or the same soil with different water contents or compactions) where the selection of the specific soils would be based on their individual acoustic properties. Mathematical modeling would be employed to determine whether the acoustic properties of individual soil samples can estimate the acoustic properties of the layered media.

A third component to be investigated is undisturbed core samples. A coring truck exists to collect about 6" diameter core samples about 24" deep. This is an ideal size for the existing measurement system. The intent would be to evaluate undisturbed core samples of soil types that have already been evaluated and thus to compare the differences between the same soil types that have been prepared under laboratory conditions and those that are more typical of undisturbed *in situ* conditions. Ten paired core samples will be obtained, one for acoustic characterization and the other for soil characterization.

F. REFERENCES

AASHO 1969. Standard method for determining liquid limit of soils (T 89-60). Soils Manual, Asphalt Institute. College Park, MD.

Blake, G.R. and K.H. Hartge. 1986. Bulk density.p.363-376 In Methods of soil analysis. Second edition. (A. Klute. ed.). Agronomy Monograph no.9. American Society of Agronomy. Madison, Wisconsin.

Broadband Power Amplifier Instruction Manual Model A150. Electronic Navigation Industries, Inc., Rochester, NY, 1987.

Cafforio, C., C. Prati, and F. Rocca. 1991 SAR data focusing using seismic migration techniques, IEEE Trans. on Aerospace and Electronic Systems, 27:194-207.

Choi, H. and D. C. Munson, Jr. On the wavenumber model in synthetic aperture radar, submitted to IEEE Trans. on Image Processing.

Dahnke W.C. 1988. Recommended chemical soil test procedures. Bull. 499. North Dakota Agricultural Experimental Experiment Station. North Dakota State University.

Danielson, R.E. and P.L. Sutherland. 1986 p.443-462 In Methods of soil analysis. Second edition. (A. Klute. ed.). Agronomy Monograph no.9. American Society of Agronomy. Madison, Wisconsin.

Data Translation SPO 126 Software Tool Kit, 1992.

Dunn, F., Edmonds, P.D., and Fry, W.J., Absorption and Dispersion of Ultrasound in Biological Media. In Biological Engineering, H.P. Schwan, ed. McGraw Hill, New York, pp. 205-322, 1969.

Gardner, W.H. 1986. Water content.p.493-544 In Methods of soil analysis. Second edition. (A. Klute. ed.). Agronomy Monograph no.9. American Society of Agronomy. Madison, Wisconsin.

Gee, G.W. and J.W. Bauder. 1986. Particle-size analysis.p.383-412 In Methods of soil analysis. Second edition. (A. Klute. ed.). Agronomy Monograph no.9. American Society of Agronomy. Madison, Wisconsin.

HP8116A Programmable Pulse / Function Generator 50 MHz Operating and Service Manual. Hewlett-Packard GPIB, Federal Republic of Germany, 1984.

Jackson, M.L., C. H. Lim, and L. W. Zelazny. 1986. Oxides, hydroxides, and aluminosilicates. p.101-150 In Methods of soil analysis. Second edition. (A. Klute. ed.). Agronomy Monograph no.9. American Society of Agronomy. Madison, Wisconsin.

Lee, J. A. C., O. Arikan, and D. C. Munson, Jr. 1996 Formulation of a generalized imaging algorithm for high-resolution synthetic aperture radar, Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, Atlanta, Georgia, May 7-10.

Microsoft® C6.0, Microsoft Corporation, Redmond WA, 1989.

Munson, D. C., Jr. and R. L. Visentin. 1989 A signal processing view of strip-mapping synthetic aperture radar, IEEE Trans. on Acoustics, Speech, and Signal Processing, 37:2131-2147.

Nelson, D.W. and L.E. Sommers. 1982. p.539-580 In Methods of soil analysis. Second edition. (A.L. Page. ed.). Agronomy Monograph no.9. American Society of Agronomy. Madison, Wisconsin.

Schafer, W.M. and M.J. Singer. 1976. A new method of measuring shrink-swell potential using soil paste. Soil Sci. Soc. Am. J. 40:805-806.

USRD Series F42 Transducers, Naval Research Laboratory Underwater Sound Reference Detachment, Orlando FL, 32856-8337. January 1978.

Whittig, L.D. and W.R. Allardice. 1986. X-ray diffraction techniques.p.331-362 In Methods of soil analysis. Second edition. (A. Klute. ed.). Agronomy Monograph no.9. American Society of Agronomy. Madison, Wisconsin.

APPENDICES

Appendix A1. Soil conditions at time of acoustic evaluation.

Sample File Number	Net Total kg	Dry Soil kg	Net H_2O kg	Tub Depth cm	Total Vol L	Dry Soil Den gm/cc	Total Wet Den gm/cc	Soil Part Den gm/cc	Total Solid %	Total Pores %	H_2O Filled Pores %	% H_2O (vol)	% H_2O (dry)	% H_2O (wet)
ada16111-A4	15.6	14.9	0.6	16.0	17.16	0.87	0.91	2.49	34.9	65.1	5.7	3.7	4.3	4.1
ada19111-A3	19.3	18.5	0.8	19.0	20.49	0.90	0.94	2.49	36.2	63.8	6.1	3.9	4.3	4.1
ada22111-A2	22.3	21.4	0.9	22.0	23.82	0.90	0.94	2.49	36.0	64.0	6.0	3.9	4.3	4.1
ada25111-A1	26.7	25.6	1.1	25.0	27.15	0.94	0.98	2.49	37.8	62.2	6.5	4.0	4.3	4.1
ada16113-B1	16.2	15.5	0.7	16.0	17.10	0.91	0.94	2.49	36.3	63.7	6.1	3.9	4.3	4.1
ada19113-B2	20.5	19.7	0.8	19.0	20.43	0.96	1.00	2.49	38.6	61.4	6.7	4.1	4.3	4.1
ada22113-B3	23.2	22.2	1.0	22.0	23.76	0.93	0.97	2.49	37.5	62.5	6.4	4.0	4.3	4.1
ada25113-B4	26.3	25.2	1.1	25.0	27.09	0.93	0.97	2.49	37.4	62.6	6.4	4.0	4.3	4.1
ada08141-C1	8.2	7.9	0.3	8.0	8.24	0.95	1.00	2.49	38.3	61.7	6.6	4.1	4.3	4.1
ada11141-C2	12.8	12.2	0.5	11.0	11.57	1.06	1.10	2.49	42.4	57.6	7.9	4.5	4.3	4.1
ada14141-C3	16.6	15.9	0.7	14.0	14.90	1.07	1.11	2.49	42.9	57.1	8.0	4.6	4.3	4.1
ada13211-D4	13.0	11.4	1.6	13.0	13.77	0.83	0.94	2.49	33.1	66.9	17.7	11.8	14.3	12.5
ada16211-D3	15.7	13.7	2.0	16.0	17.10	0.80	0.92	2.49	32.2	67.8	17.0	11.5	14.3	12.5
ada19211-D2	20.2	17.2	3.0	19.0	20.43	0.84	0.99	2.49	33.8	66.2	22.1	14.7	17.4	14.8
ada22211-D1	23.6	20.1	3.5	22.0	23.76	0.85	0.99	2.49	33.9	66.1	22.3	14.7	17.4	14.8
ada10241-E1	11.7	9.9	1.7	10.0	10.44	0.95	1.12	2.49	38.1	61.9	26.8	16.5	17.4	14.8
ada13241-E2	16.1	14.1	2.0	13.0	13.77	1.02	1.17	2.49	41.0	59.0	24.8	14.6	14.3	12.5
ada16241-E3	19.5	17.1	2.4	16.0	17.10	1.00	1.14	2.49	40.0	60.0	23.8	14.3	14.3	12.5
ada19241-E4	23.7	20.7	3.0	19.0	20.43	1.01	1.16	2.49	40.6	59.4	24.4	14.5	14.3	12.5
ada16311-F4	17.2	13.2	4.0	16.0	17.10	0.77	1.01	2.49	30.9	69.1	34.2	23.7	30.8	23.5
ada19311-F3	21.1	16.1	5.0	19.0	20.43	0.79	1.03	2.49	31.7	68.3	35.6	24.3	30.8	23.5
ada22311-F2	24.8	19.0	5.8	22.0	23.76	0.80	1.04	2.49	32.0	68.0	36.1	24.6	30.8	23.5
ada25311-F1	28.0	21.6	6.4	25.0	27.09	0.80	1.03	2.49	32.0	68.0	34.9	23.7	29.8	23.0
ada10341-G4	12.3	9.4	2.9	10.0	10.48	0.90	1.17	2.49	36.0	64.0	43.2	27.7	30.8	23.5
ada13341-G3	16.3	12.5	3.8	13.0	13.81	0.90	1.18	2.49	36.2	63.8	43.5	27.8	30.8	23.5
ada16341-G2	21.1	16.1	5.0	16.0	17.14	0.94	1.23	2.49	37.8	62.2	46.6	29.0	30.8	23.5
ada19341-G1	24.6	18.8	5.8	19.0	20.47	0.92	1.20	2.49	36.9	63.1	44.8	28.3	30.8	23.5
ada16511-H3	24.8	14.7	10.1	15.0	15.99	0.92	1.55	2.49	36.8	63.2	100.1	63.2	68.9	40.8
ada19511-H4	26.5	15.7	10.8	16.0	17.10	0.92	1.55	2.49	36.8	63.2	100.1	63.2	68.9	40.8
ada20511-H6	33.4	19.8	13.6	20.0	21.54	0.92	1.55	2.49	36.8	63.2	100.1	63.2	68.9	40.8
ada22511-H2	36.8	21.8	15.0	22.0	23.76	0.92	1.55	2.49	36.8	63.2	100.1	63.2	68.9	40.8
ada23511-H5	38.5	22.8	15.7	23.0	24.87	0.92	1.55	2.49	36.8	63.2	100.1	63.2	68.9	40.8
ada25511-H1	42.0	24.9	17.1	25.0	27.09	0.92	1.55	2.49	36.8	63.2	100.1	63.2	68.9	40.8

Appendix A1. Soil conditions at time of acoustic evaluation (continued).

Sample File Number	Net Total kg	Dry Soil kg	Net H_2O kg	Tub Depth cm	Total Vol L	Dry Soil Den gm/cc	Total Wet Den gm/cc	Soil Part Den gm/cc	Total Solid %	Total Pores %	H_2O Filled Pores %	% H_2O (vol)	% H_2O (dry)	% H_2O (wet)
cab16111-A4	23.6	22.8	0.8	16.0	16.97	1.34	1.39	2.64	50.9	49.1	9.4	4.6	3.4	3.3
cab19111-A3	28.7	27.7	1.0	19.0	20.30	1.37	1.41	2.64	51.7	48.3	9.7	4.7	3.4	3.3
cab22111-A2	32.7	31.6	1.1	22.0	23.63	1.34	1.38	2.64	50.5	49.5	9.3	4.6	3.4	3.3
cab25111-A1	38.6	37.3	1.3	25.0	26.96	1.38	1.43	2.64	52.3	47.7	10.0	4.7	3.4	3.3
cab16112-B1	19.4	18.7	0.6	16.0	17.17	1.09	1.13	2.64	41.2	58.8	6.4	3.7	3.4	3.3
cab19112-B2	24.9	24.1	0.8	19.0	20.50	1.17	1.21	2.64	44.4	55.6	7.3	4.0	3.4	3.3
cab22112-B3	29.7	28.7	1.0	22.0	23.83	1.20	1.25	2.64	45.6	54.4	7.6	4.1	3.4	3.3
cab25112-B4	34.7	33.5	1.2	25.0	27.16	1.24	1.28	2.64	46.7	53.3	8.0	4.2	3.4	3.3
cab10141-C1	15.0	14.5	0.5	10.0	10.31	1.41	1.45	2.64	53.2	46.8	10.3	4.8	3.4	3.3
cab13141-C2	20.0	19.3	0.7	13.0	13.64	1.41	1.46	2.64	53.5	46.5	10.4	4.9	3.4	3.3
cab16141-C3	25.7	24.8	0.9	16.0	16.97	1.46	1.51	2.64	55.4	44.6	11.3	5.0	3.4	3.3
cab19141-C4	30.6	29.6	1.0	19.0	20.30	1.46	1.51	2.64	55.1	44.9	11.2	5.0	3.4	3.3
cab16211-D4	21.6	19.8	1.8	16.0	16.97	1.17	1.27	2.64	44.2	55.8	18.5	10.3	8.8	8.1
cab19211-D3	25.3	23.2	2.1	19.0	20.30	1.15	1.25	2.64	43.3	56.7	17.9	10.1	8.8	8.1
cab22211-D2	30.0	27.7	2.3	22.0	23.63	1.17	1.27	2.64	44.3	55.7	17.5	9.8	8.3	7.7
cab25211-D1	33.9	31.3	2.6	25.0	26.96	1.16	1.26	2.64	43.9	56.1	17.2	9.7	8.3	7.7
cab10241-E1	14.4	13.2	1.2	10.0	10.27	1.29	1.40	2.64	48.7	51.3	22.2	11.4	8.8	8.1
cab13241-E2	19.2	17.6	1.6	13.0	13.57	1.30	1.41	2.64	49.2	50.8	22.6	11.5	8.8	8.1
cab16241-E3	25.7	23.6	2.1	16.0	16.87	1.40	1.52	2.64	53.0	47.0	26.3	12.4	8.8	8.1
cab19241-E4	31.9	29.3	2.6	19.0	20.17	1.45	1.58	2.64	55.0	45.0	28.5	12.8	8.8	8.1
cab16311-F4	17.2	14.5	2.7	16.0	17.01	0.85	1.01	2.64	32.2	67.8	23.4	15.9	18.6	15.7
cab19311-F3	21.2	17.9	3.3	19.0	20.34	0.88	1.04	2.64	33.4	66.6	24.0	16.0	18.1	15.4
cab22311-F2	24.3	20.6	3.7	22.0	23.67	0.87	1.03	2.64	32.9	67.1	23.5	15.8	18.1	15.4
cab25311-F1	29.1	24.6	4.5	25.0	27.00	0.91	1.08	2.64	34.5	65.5	25.3	16.5	18.1	15.4
cab10341-G4	12.7	10.7	2.0	10.0	10.35	1.03	1.23	2.64	39.1	60.9	31.7	19.3	18.6	15.7
cab13341-G3	19.0	16.0	3.0	13.0	13.68	1.17	1.39	2.64	44.3	55.7	39.2	21.8	18.6	15.7
cab16341-G2	22.9	19.3	3.6	16.0	17.01	1.13	1.35	2.64	42.9	57.1	37.1	21.2	18.6	15.7
cab19341-G1	29.6	24.9	4.7	19.0	20.34	1.23	1.46	2.64	46.4	53.6	42.7	22.9	18.6	15.7
cab10511-H6	17.7	11.6	6.1	10.0	10.46	1.11	1.69	2.64	42.0	58.0	100.0	58.0	52.2	34.3
cab13511-H5	23.3	15.3	8.0	13.0	13.79	1.11	1.69	2.64	42.0	58.0	100.0	58.0	52.2	34.3
cab16511-H4	28.9	19.0	9.9	16.0	17.12	1.11	1.69	2.64	42.0	58.0	100.0	58.0	52.2	34.3
cab17511-H8	30.8	20.2	10.6	17.0	18.23	1.11	1.69	2.64	42.0	58.0	100.0	58.0	52.2	34.3
cab19511-H3	34.6	22.7	11.9	19.0	20.45	1.11	1.69	2.64	42.0	58.0	100.0	58.0	52.2	34.3
cab21511-H7	38.3	25.2	13.1	21.0	22.67	1.11	1.69	2.64	42.0	58.0	100.0	58.0	52.2	34.3
cab22511-H2	40.2	26.4	13.8	22.0	23.78	1.11	1.69	2.64	42.0	58.0	100.0	58.0	52.2	34.3
cab25511-H1	45.8	30.1	15.7	25.0	27.11	1.11	1.69	2.64	42.0	58.0	100.0	58.0	52.2	34.3

Appendix A1. Soil conditions at time of acoustic evaluation (continued).

Sample File Number	Net Total kg	Dry Soil kg	Net H_2O kg	Tub Depth cm	Total Vol L	Dry Soil Den gm/cc	Total Wet Den gm/cc	Soil Part Den gm/cc	Total Solid %	Total Pores %	H_2O Filled Pores %	% H_2O (vol)	% H_2O (dry)	% H_2O (wet)
dra12111-A4	17.0	16.2	0.7	12.0	12.46	1.30	1.36	2.52	51.8	48.2	11.9	5.7	4.4	4.2
dra17112-A3	23.9	22.8	1.0	17.0	17.91	1.28	1.33	2.52	50.7	49.3	11.4	5.6	4.4	4.2
dra22113-A2	31.6	30.3	1.3	22.0	23.36	1.30	1.35	2.52	51.5	48.5	11.8	5.7	4.4	4.2
dra27112-A1	40.2	38.5	1.7	27.0	28.81	1.33	1.39	2.52	53.0	47.0	12.5	5.9	4.4	4.2
dra12114-B4	18.8	18.0	0.8	12.0	12.75	1.41	1.47	2.52	56.1	43.9	14.2	6.2	4.4	4.2
dra17114-B3	25.2	24.1	1.1	17.0	18.30	1.32	1.37	2.52	52.3	47.7	12.2	5.8	4.4	4.2
dra22114-B2	31.7	30.3	1.3	22.0	23.85	1.27	1.33	2.52	50.5	49.5	11.3	5.6	4.4	4.2
dra27114-B1	40.6	38.8	1.7	27.0	29.40	1.32	1.38	2.52	52.5	47.5	12.3	5.8	4.4	4.2
dra12141-C1	17.0	16.2	0.7	12.0	12.46	1.30	1.36	2.52	51.8	48.2	11.9	5.7	4.4	4.2
dra17141-C2a	27.8	26.7	1.2	17.0	17.91	1.49	1.55	2.52	59.1	40.9	16.1	6.6	4.4	4.2
dra17142-C2b	27.8	26.7	1.2	17.0	17.91	1.49	1.55	2.52	59.1	40.9	16.1	6.6	4.4	4.2
dra23141-C3	37.0	35.4	1.6	23.0	24.45	1.45	1.51	2.52	57.5	42.5	15.0	6.4	4.4	4.2
dra05143-D1	6.3	6.0	0.3	5.0	4.83	1.24	1.29	2.52	49.2	50.8	10.8	5.5	4.4	4.2
dra08143-D2	12.0	11.4	0.5	8.0	8.10	1.41	1.48	2.52	56.1	43.9	14.2	6.2	4.4	4.2
dra10143-D3	15.8	15.1	0.7	10.0	10.28	1.47	1.53	2.52	58.3	41.7	15.5	6.5	4.4	4.2
dra16211-E4	20.3	17.7	2.6	16.0	16.88	1.05	1.20	2.52	41.6	58.4	26.1	15.3	14.6	12.7
dra19211-E3	24.1	21.0	3.1	19.0	20.15	1.04	1.19	2.52	41.4	58.6	25.9	15.2	14.6	12.7
dra22211-E2	28.4	24.8	3.6	22.0	23.42	1.06	1.21	2.52	42.0	58.0	26.6	15.4	14.6	12.7
dra25211-E1	31.7	27.7	4.0	25.0	26.69	1.04	1.19	2.52	41.2	58.8	25.7	15.1	14.6	12.7
dra16212-F1	17.9	15.6	2.3	16.0	16.91	0.92	1.06	2.52	36.7	63.3	21.3	13.5	14.6	12.7
dra19212-F2	21.3	18.6	2.7	19.0	20.21	0.92	1.05	2.52	36.5	63.5	21.1	13.4	14.6	12.7
dra22212-F3	25.7	22.4	3.3	22.0	23.51	0.95	1.09	2.52	37.9	62.1	22.4	13.9	14.6	12.7
dra25212-F4	30.2	26.4	3.8	25.0	26.81	0.98	1.13	2.52	39.1	60.9	23.5	14.3	14.6	12.7
dra10241-G1	13.8	12.1	1.7	10.0	10.35	1.17	1.33	2.52	46.6	53.4	30.1	16.1	13.7	12.1
dra13241-G2	18.5	16.3	2.2	13.0	13.68	1.19	1.35	2.52	47.2	52.8	30.9	16.3	13.7	12.1
dra16241-G3	22.8	20.1	2.7	16.0	17.01	1.18	1.34	2.52	46.8	53.2	30.4	16.2	13.7	12.1
dra19241-G4	28.5	25.1	3.4	19.0	20.34	1.23	1.40	2.52	49.0	51.0	33.1	16.9	13.7	12.1
dra16311-H4	18.2	14.4	3.8	16.0	16.91	0.85	1.07	2.52	33.8	66.2	33.8	22.4	26.3	20.8
dra19311-H3	23.3	18.0	5.3	19.0	20.21	0.89	1.15	2.52	35.4	64.6	40.4	26.1	29.3	22.6
dra22311-H2	27.0	20.9	6.1	22.0	23.51	0.89	1.15	2.52	35.2	64.8	40.1	25.9	29.3	22.6
dra25311-H1	31.1	24.1	7.0	25.0	26.81	0.90	1.16	2.52	35.7	64.3	40.8	26.3	29.3	22.6
dra10341-I1	15.1	12.0	3.1	10.0	10.31	1.16	1.46	2.52	46.1	53.9	56.6	30.5	26.3	20.8
dra13341-I2	22.5	17.8	4.7	13.0	13.61	1.31	1.65	2.52	51.9	48.1	71.4	34.4	26.3	20.8
dra16341-I3	26.9	21.2	5.7	16.0	16.91	1.25	1.59	2.52	49.7	50.3	67.5	33.9	27.1	21.3
dra18341-I4	30.7	24.2	6.5	17.5	18.56	1.30	1.65	2.52	51.7	48.3	73.0	35.3	27.1	21.3
dra15511-J8	26.9	18.0	8.8	15.0	15.99	1.13	1.68	2.52	44.8	55.2	100.1	55.3	49.0	32.9
dra16511-J4	28.7	19.3	9.5	16.0	17.10	1.13	1.68	2.52	44.8	55.2	100.1	55.3	49.0	32.9
dra18511-J7	32.5	21.8	10.7	18.0	19.32	1.13	1.68	2.52	44.8	55.2	100.1	55.3	49.0	32.9
dra19511-J3	34.3	23.0	11.3	19.0	20.43	1.13	1.68	2.52	44.8	55.2	100.1	55.3	49.0	32.9
dra21511-J6	38.1	25.5	12.5	21.0	22.65	1.13	1.68	2.52	44.8	55.2	100.1	55.3	49.0	32.9
dra22511-J2	39.9	26.8	13.1	22.0	23.76	1.13	1.68	2.52	44.8	55.2	100.1	55.3	49.0	32.9
dra24511-J5	43.6	29.3	14.4	24.0	25.98	1.13	1.68	2.52	44.8	55.2	100.1	55.3	49.0	32.9
dra25511-J1	45.5	30.5	15.0	25.0	27.09	1.13	1.68	2.52	44.8	55.2	100.1	55.3	49.0	32.9

Appendix A1. Soil conditions at time of acoustic evaluation (continued).

Sample File Number	Net Total kg	Dry Soil kg	Net H_2O kg	Tub Depth cm	Total Vol L	Dry Soil Den gm/cc	Total Wet Den gm/cc	Soil Part Den gm/cc	Total Solid %	Total Pores %	H_2O Filled Pores %	% H_2O (vol)	% H_2O (dry)	% H_2O (wet)
mea16111-A4	19.6	19.1	0.5	16.0	17.12	1.11	1.14	2.62	42.6	57.4	4.8	2.7	2.5	2.4
mea19111-A3	24.0	23.4	0.6	19.0	20.45	1.14	1.17	2.62	43.6	56.4	5.0	2.8	2.5	2.4
mea22111-A2	27.9	27.2	0.7	22.0	23.78	1.14	1.17	2.62	43.6	56.4	5.0	2.8	2.5	2.4
mea25111-A1	32.2	31.4	0.8	25.0	27.11	1.16	1.19	2.62	44.2	55.8	5.1	2.8	2.5	2.4
mea16112-B1	19.3	18.8	0.5	16.0	17.14	1.10	1.13	2.62	42.0	58.0	4.7	2.7	2.5	2.4
mea19112-B2	23.6	23.0	0.6	19.0	20.47	1.12	1.15	2.62	42.9	57.1	4.8	2.8	2.5	2.4
mea22112-B3	27.4	26.7	0.7	22.0	23.80	1.12	1.15	2.62	42.9	57.1	4.8	2.8	2.5	2.4
mea25112-B4	32.0	31.2	0.8	25.0	27.13	1.15	1.18	2.62	44.0	56.0	5.1	2.8	2.5	2.4
mea10141-C1	14.5	14.1	0.3	10.0	10.46	1.35	1.38	2.62	51.5	48.5	6.5	3.1	2.3	2.3
mea13141-C2	19.4	19.0	0.4	13.0	13.79	1.37	1.41	2.62	52.5	47.5	6.7	3.2	2.3	2.3
mea16141-C3	23.3	22.8	0.5	16.0	17.12	1.33	1.36	2.62	50.8	49.2	6.3	3.1	2.3	2.3
mea19141-C4	30.0	29.3	0.7	19.0	20.45	1.43	1.47	2.62	54.7	45.3	7.4	3.3	2.3	2.3
mea16211-D4	20.9	18.7	2.1	16.0	17.12	1.09	1.22	2.62	41.8	58.2	21.2	12.3	11.3	10.1
mea19211-D3	25.5	22.9	2.6	19.0	20.45	1.12	1.24	2.62	42.7	57.3	22.0	12.6	11.3	10.1
mea22211-D2	28.9	26.1	2.8	22.0	23.78	1.10	1.21	2.62	41.9	58.1	20.0	11.6	10.6	9.6
mea25211-D1	33.5	30.2	3.2	25.0	27.11	1.12	1.23	2.62	42.6	57.4	20.6	11.8	10.6	9.6
mea10241-E1	15.2	13.7	1.5	10.0	10.31	1.33	1.47	2.62	50.6	49.4	30.2	14.9	11.3	10.1
mea13241-E2	19.6	17.6	2.0	13.0	13.61	1.29	1.44	2.62	49.4	50.6	28.8	14.6	11.3	10.1
mea16241-E3	24.5	22.0	2.5	16.0	16.91	1.30	1.45	2.62	49.7	50.3	29.2	14.7	11.3	10.1
mea19241-E4	29.0	26.1	2.9	19.0	20.21	1.29	1.43	2.62	49.2	50.8	28.6	14.5	11.3	10.1
mea16311-F4	19.0	16.3	2.7	16.0	16.87	0.97	1.13	2.62	37.0	63.0	25.0	15.7	16.2	14.0
mea19311-F3	23.2	20.1	3.1	19.0	20.17	1.00	1.15	2.62	38.0	62.0	25.0	15.5	15.6	13.5
mea22311-F2	26.2	22.7	3.5	22.0	23.47	0.97	1.12	2.62	36.9	63.1	23.8	15.0	15.6	13.5
mea25311-F1	28.8	24.9	3.9	25.0	26.77	0.93	1.08	2.62	35.5	64.5	22.5	14.5	15.6	13.5
mea10341-G1	15.1	13.0	2.1	10.0	10.36	1.25	1.46	2.62	47.9	52.1	39.0	20.3	16.2	14.0
mea13341-G2	22.2	19.1	3.1	13.0	13.69	1.40	1.62	2.62	53.3	46.7	48.4	22.6	16.2	14.0
mea16341-G3	27.4	23.6	3.8	16.0	17.02	1.39	1.61	2.62	52.9	47.1	47.7	22.5	16.2	14.0
mea19341-G4	32.6	28.0	4.6	19.0	20.35	1.38	1.60	2.62	52.6	47.4	47.2	22.4	16.2	14.0
mea10511-H6	18.1	12.7	5.5	10.0	10.31	1.23	1.76	2.62	47.0	53.0	99.9	53.0	43.1	30.1
mea12511-H5	22.0	15.4	6.6	12.0	12.51	1.23	1.76	2.62	47.0	53.0	99.9	53.0	43.1	30.1
mea13511-H10	24.0	16.7	7.2	13.0	13.61	1.23	1.76	2.62	47.0	53.0	99.9	53.0	43.1	30.1
mea15511-H4	27.8	19.5	8.4	15.0	15.81	1.23	1.76	2.62	47.0	53.0	99.9	53.0	43.1	30.1
mea16511-H9	29.8	20.8	9.0	16.0	16.91	1.23	1.76	2.62	47.0	53.0	99.9	53.0	43.1	30.1
mea18511-H3	33.6	23.5	10.1	18.0	19.11	1.23	1.76	2.62	47.0	53.0	99.9	53.0	43.1	30.1
mea19511-H8	35.6	24.9	10.7	19.0	20.21	1.23	1.76	2.62	47.0	53.0	99.9	53.0	43.1	30.1
mea21511-H2	39.4	27.6	11.9	21.0	22.41	1.23	1.76	2.62	47.0	53.0	99.9	53.0	43.1	30.1
mea22511-H7	41.4	28.9	12.5	22.0	23.51	1.23	1.76	2.62	47.0	53.0	99.9	53.0	43.1	30.1
mea24511-H1	45.2	31.6	13.6	24.0	25.71	1.23	1.76	2.62	47.0	53.0	99.9	53.0	43.1	30.1

Appendix A1. Soil conditions at time of acoustic evaluation (continued).

Sample File Number	Net Total kg	Dry Soil kg	Net H_2O kg	Tub Depth cm	Total Vol L	Dry Soil Den gm/cc	Total Wet Den gm/cc	Soil Part Den gm/cc	Total Solid %	Total Pores %	H_2O Filled Pores %	% H_2O (vol)	% H_2O (dry)	% H_2O (wet)
pla11111-A6	17.7	17.6	0.1	11.0	11.37	1.55	1.56	2.65	58.7	41.3	1.1	0.5	0.3	0.3
pla14111-A5	21.8	21.7	0.1	14.0	14.67	1.48	1.49	2.65	56.0	44.0	1.0	0.4	0.3	0.3
pla17111-A4	26.3	26.2	0.1	17.0	17.97	1.46	1.46	2.65	55.2	44.8	0.9	0.4	0.3	0.3
pla20111-A3	32.6	32.5	0.1	20.0	21.27	1.53	1.53	2.65	57.8	42.2	1.1	0.4	0.3	0.3
pla23111-A2	37.1	37.0	0.1	23.0	24.57	1.51	1.51	2.65	56.9	43.1	1.0	0.4	0.3	0.3
pla26111-A1	41.2	41.1	0.1	25.5	27.32	1.50	1.51	2.65	56.8	43.2	1.0	0.4	0.3	0.3
pla12112-B1	17.8	17.7	0.1	12.0	12.67	1.40	1.40	2.65	53.0	47.0	0.9	0.4	0.3	0.3
pla17113-B2	29.2	29.1	0.1	17.0	18.17	1.60	1.61	2.65	60.6	39.4	1.2	0.5	0.3	0.3
pla22112-B3	37.1	37.0	0.1	22.0	23.67	1.56	1.57	2.65	59.1	40.9	1.1	0.5	0.3	0.3
pla27112-B4	48.1	47.9	0.1	27.0	29.17	1.64	1.65	2.65	62.1	37.9	1.3	0.5	0.3	0.3
pla12114-C4	19.1	19.0	0.1	12.0	12.47	1.53	1.53	2.65	57.7	42.3	1.1	0.4	0.3	0.3
pla17114-C3	27.1	27.0	0.1	17.0	18.28	1.48	1.48	2.65	55.9	44.1	1.0	0.4	0.3	0.3
pla22114-C2	36.3	36.1	0.1	22.0	23.85	1.52	1.52	2.65	57.3	42.7	1.0	0.4	0.3	0.3
pla27114-C1	44.5	44.3	0.1	27.0	29.38	1.51	1.51	2.65	57.0	43.0	1.0	0.4	0.3	0.3
pla12115-D1	17.5	17.4	0.1	12.0	12.73	1.37	1.37	2.65	51.7	48.3	0.8	0.4	0.3	0.3
pla17115-D2	28.0	27.9	0.1	17.0	17.97	1.55	1.56	2.65	58.7	41.3	1.1	0.5	0.3	0.3
pla22115-D3	35.3	35.2	0.1	22.0	23.47	1.50	1.50	2.65	56.7	43.3	1.0	0.4	0.3	0.3
pla26115-D4	43.9	43.7	0.1	26.0	27.87	1.57	1.57	2.65	59.3	40.7	1.1	0.5	0.3	0.3
pla27115-D5	45.6	45.4	0.1	27.0	28.97	1.57	1.57	2.65	59.3	40.7	1.1	0.5	0.3	0.3
pla18216-E4	31.6	29.2	2.4	18.0	19.07	1.53	1.66	2.65	57.9	42.1	29.8	12.5	8.2	7.6
pla21216-E3	37.9	35.0	2.9	21.0	22.37	1.57	1.69	2.65	59.2	40.8	31.4	12.8	8.2	7.6
pla24216-E2	43.8	40.5	3.3	24.0	25.67	1.58	1.71	2.65	59.6	40.4	32.0	12.9	8.2	7.6
pla27216-E1	50.1	46.3	3.8	27.0	28.97	1.60	1.73	2.65	60.4	39.6	33.0	13.1	8.2	7.6
pla19217-F1	28.0	25.4	2.5	19.0	20.09	1.26	1.39	2.65	47.8	52.2	24.3	12.7	10.0	9.1
pla22217-F2	33.3	30.2	3.0	22.0	23.36	1.29	1.42	2.65	48.9	51.1	25.4	13.0	10.0	9.1
pla25217-F3	41.3	37.5	3.8	25.0	26.63	1.41	1.55	2.65	53.2	46.8	30.1	14.1	10.0	9.1
pla10241-G1	17.8	16.4	1.4	10.0	10.27	1.60	1.73	2.65	60.4	39.6	34.4	13.6	8.5	7.9
pla13241-G2	23.2	21.4	1.8	13.0	13.57	1.58	1.71	2.65	59.6	40.4	33.2	13.4	8.5	7.9
pla16241-G3	29.9	27.5	2.4	16.0	16.87	1.63	1.77	2.65	61.7	38.3	36.4	13.9	8.5	7.9
pla19241-G4	36.5	33.6	2.9	19.0	20.17	1.67	1.81	2.65	63.0	37.0	38.5	14.2	8.5	7.9
pla16311-H4	26.7	23.9	2.7	16.0	16.82	1.42	1.58	2.65	53.8	46.2	34.9	16.2	11.4	10.2
pla19311-H3	32.1	29.6	2.5	19.0	20.09	1.47	1.60	2.65	55.8	44.2	27.8	12.3	8.4	7.7
pla22311-H2	37.3	34.4	2.9	22.0	23.36	1.47	1.59	2.65	55.6	44.4	27.7	12.3	8.4	7.7
pla25311-H1	41.9	38.6	3.2	25.0	26.63	1.45	1.57	2.65	54.8	45.2	26.8	12.1	8.4	7.7
pla10341-I1	17.6	16.2	1.4	10.0	10.51	1.54	1.67	2.65	58.4	41.6	31.4	13.2	8.5	7.8
pla13341-I2	25.3	22.6	2.7	13.0	13.84	1.63	1.83	2.65	61.7	38.3	51.0	19.5	11.9	10.7
pla16341-I3	29.1	26.0	3.1	16.0	17.17	1.51	1.69	2.65	57.2	42.8	42.3	18.1	11.9	10.7
pla19341-I4	36.0	32.2	3.8	19.0	20.50	1.57	1.76	2.65	59.3	40.7	46.0	18.7	11.9	10.7
pla15511-J4	30.0	22.9	7.1	15.0	15.77	1.45	2.05	2.65	54.9	45.1	99.9	45.1	31.1	23.7
pla18511-J3	37.2	29.1	8.1	18.0	19.07	1.53	2.05	2.65	57.7	42.3	99.9	42.3	27.7	21.7
pla21511-J2	44.7	35.8	8.8	21.0	22.37	1.60	2.05	2.65	60.5	39.5	100.1	39.5	24.7	19.8
pla24511-J1	53.1	44.0	9.0	24.0	25.67	1.72	2.05	2.65	64.9	35.1	100.0	35.1	20.5	17.0

Appendix A1. Soil conditions at time of acoustic evaluation (continued).

Sample File Number	Net Total kg	Dry Soil kg	Net H_2O kg	Tub Depth cm	Total Vol L	Dry Soil Den gm/cc	Total Wet Den gm/cc	Soil Part Den gm/cc	Total Solid %	Total Pores %	H_2O Filled Pores %	% H_2O (vol)	% H_2O (dry)	% H_2O (wet)
sac16111-A4	25.0	24.5	0.4	16.0	17.02	1.44	1.47	2.65	54.3	45.7	5.8	2.6	1.8	1.8
sac19111-A3	28.4	27.8	0.5	19.0	20.35	1.37	1.39	2.65	51.6	48.4	5.2	2.5	1.8	1.8
sac22111-A2	35.5	34.8	0.6	22.0	23.68	1.47	1.50	2.65	55.5	44.5	6.1	2.7	1.8	1.8
sac25111-A1	41.7	40.9	0.8	25.0	27.01	1.52	1.54	2.65	57.2	42.8	6.5	2.8	1.8	1.8
sac16112-B1	21.2	20.8	0.4	16.0	16.91	1.23	1.25	2.65	46.5	53.5	4.2	2.3	1.8	1.8
sac19112-B2	26.3	25.8	0.5	19.0	20.21	1.28	1.30	2.65	48.2	51.8	4.5	2.3	1.8	1.8
sac22112-B3	30.6	30.0	0.6	22.0	23.51	1.28	1.30	2.65	48.3	51.7	4.5	2.3	1.8	1.8
sac25112-B4	36.2	35.5	0.7	25.0	26.81	1.33	1.35	2.65	50.1	49.9	4.9	2.4	1.8	1.8
sac08141-C1	10.5	10.3	0.2	7.5	7.71	1.34	1.36	2.65	50.5	49.5	5.0	2.4	1.8	1.8
sac11141-C2	16.6	16.3	0.3	11.0	11.59	1.40	1.43	2.65	52.9	47.1	5.5	2.6	1.8	1.8
sac14141-C3	21.7	21.3	0.4	14.0	14.92	1.43	1.45	2.65	53.9	46.1	5.7	2.6	1.8	1.8
sac17141-C4	27.0	26.5	0.5	17.0	18.25	1.45	1.48	2.65	54.8	45.2	5.9	2.7	1.8	1.8
sac16211-D4	19.3	17.4	1.9	16.0	17.01	1.02	1.13	2.65	38.5	61.5	18.6	11.4	11.2	10.1
sac19211-D3	23.9	21.3	2.6	19.0	20.34	1.05	1.18	2.65	39.6	60.4	21.0	12.7	12.1	10.8
sac22211-D2	28.3	25.2	3.1	22.0	23.67	1.07	1.20	2.65	40.3	59.7	21.6	12.9	12.1	10.8
sac25211-D1	31.6	28.1	3.4	25.0	27.00	1.04	1.17	2.65	39.4	60.6	20.8	12.6	12.1	10.8
sac16212-E1	17.1	15.3	1.7	16.0	17.10	0.90	1.00	2.65	33.9	66.1	15.2	10.0	11.2	10.1
sac19212-E2	20.6	18.5	2.1	19.0	20.43	0.90	1.01	2.65	34.2	65.8	15.4	10.1	11.2	10.1
sac22212-E3	24.6	22.1	2.5	22.0	23.76	0.93	1.03	2.65	35.1	64.9	16.0	10.4	11.2	10.1
sac25212-E4	28.8	25.9	2.9	25.0	27.09	0.95	1.06	2.65	36.0	64.0	16.7	10.7	11.2	10.1
sac10241-F1	14.7	13.1	1.5	10.0	10.36	1.27	1.41	2.65	47.9	52.1	28.1	14.6	11.6	10.4
sac13241-F2	20.9	18.7	2.2	13.0	13.69	1.37	1.52	2.65	51.5	48.5	32.6	15.8	11.6	10.4
sac16241-F3	25.8	23.1	2.7	16.0	17.02	1.36	1.52	2.65	51.3	48.7	32.2	15.7	11.6	10.4
sac19241-F4	32.7	29.3	3.4	19.0	20.35	1.44	1.60	2.65	54.3	45.7	36.4	16.6	11.6	10.4
sac13311-G4	20.7	16.9	3.7	13.0	13.77	1.23	1.50	2.65	46.4	53.6	50.5	27.1	22.0	18.1
sac16311-G3	26.2	21.4	4.7	16.0	17.10	1.25	1.53	2.65	47.3	52.7	52.4	27.6	22.0	18.1
sac19311-G2	30.6	25.0	5.6	19.0	20.43	1.22	1.50	2.65	46.2	53.8	50.6	27.3	22.3	18.2
sac22311-G1	36.7	30.0	6.7	22.0	23.76	1.26	1.54	2.65	47.6	52.4	53.7	28.1	22.3	18.2
sac10341-H1	21.0	17.5	3.5	10.0	10.51	1.67	2.00	2.65	63.0	37.0	88.9	32.9	19.7	16.5
sac13341-H2	26.2	21.9	4.3	13.0	13.84	1.58	1.89	2.65	59.7	40.3	77.3	31.2	19.7	16.5
sac16341-H3	32.8	27.4	5.4	16.0	17.17	1.59	1.91	2.65	60.2	39.8	78.8	31.4	19.7	16.5
sac17341-H4	36.2	30.2	6.0	17.0	18.28	1.65	1.98	2.65	62.4	37.6	86.5	32.6	19.7	16.5
sac14511-J7	27.0	19.8	7.2	14.0	14.67	1.35	1.84	2.65	50.9	49.1	100.1	49.1	36.4	26.7
sac16511-J3	31.0	22.8	8.3	16.0	16.87	1.35	1.84	2.65	50.9	49.1	100.1	49.1	36.4	26.7
sac17511-J6	33.1	24.2	8.8	17.0	17.97	1.35	1.84	2.65	50.9	49.1	100.1	49.1	36.4	26.7
sac19511-J2	37.1	27.2	9.9	19.0	20.17	1.35	1.84	2.65	50.9	49.1	100.1	49.1	36.4	26.7
sac20511-J5	39.1	28.7	10.4	20.0	21.27	1.35	1.84	2.65	50.9	49.1	100.1	49.1	36.4	26.7
sac23511-J4	45.2	33.1	12.1	23.0	24.57	1.35	1.84	2.65	50.9	49.1	100.1	49.1	36.4	26.7
sac25511-J1	46.2	33.9	12.3	23.5	25.12	1.35	1.84	2.65	50.9	49.1	100.1	49.1	36.4	26.7

Appendix A2. Soil data acquisition system's ***autoexec.bat*** file.

```
@ECHO OFF
PROMPT $_$t$_$d$_$p$g
PATH
C:\WINDOWS;C:\WINDOWS\COMMAND;C:\SyQuest;\WINDOWS;\DOS;\xtgold;\ncsa2;\c600\b
in;\c600\binb\;rich\;\pkzip;
PATH C:\NU;%PATH%
SET SYMANTEC=C:\SYMANTEC
SET NU=C:\NU
SET TEMP=C:\DOS
SET MOUSE=C:\MOUSE
SET INCLUDE=C:\C600\INCLUDE;
SET LIB=C:\C600\LIB;
SET HELPFILES=C:\C600\HELP\*.HLP
SET INIT=C:\C600\INIT
set qhpib=c:\hpib
set gpc=c:\graphic\fonts
C:\WINDOWS\COMMAND\MODE CON:RATE=32 DELAY=1
rem - By Windows Setup - LH C:\WINDOWS\COMMAND\MSCDEX.EXE /D:MTMIDE01
/M:10
rem - By Windows Setup - C:\MOUSE\MOUSE.EXE /Q
LH C:\GO95\DMS MONITOR
doskey
c:\mouse\mouse.exe
C:\WINDOWS\COMMAND\doskey
ECHO.
ECHO TYPE "WIN" TO START WINDOWS FOR WORKGROUPS 3.11
rem cd soil
rem C:\NU\NDD C:/Q
rem C:\NU\IMAGE C:
c:\ncsa2\pkt8000 0x60 10 0x220 0xe000
c:\winsock\winpkt 0x60
```

Appendix A3. Soil data acquisition system's ***config.sys*** file.

```
rem DEVICE=C:\BUSLOGIC\BTDOSM.SYS /D
DEVICE=C:\DOS\HIMEM.SYS
DEVICE=C:\LIB\SP0126.SYS
DEVICE=C:\PLUGPLAY\DRIVERS\DOS\DWCFGMG.SYS
DEVICE=C:\DOS\EMM386.EXE NOEMS /X=C800-CBFF   /X=DC00-DFFF
BUFFERS=30
FILES=45
DOS=UMB
DEVICEHIGH=C:\DOS\SETVER.EXE
DOS=HIGH
DEVICEHIGH=C:\MTM\MTMCDAI.SYS /D:MTMIDE01 /P:170,15
SHELL=C:\DOS\COMMAND.COM C:\DOS\ /E:2048 /P
DEVICE=C:\WINDOWS\IFSHLP.SYS
STACKS=9,256
rem to fix error#12 to the EMM386 30.12.95
rem STACKS=18,512
rem DEVICE=C:\SyQuest\SYQASPI\ASPI2DOS.SYS
rem DEVICE=C:\SyQuest\SCSI\SQDRIVER.SYS
LASTDRIVE=Z
```

Appendix A4. Soil data acquisition system's ***soil4.mak*** file.

```
########################################################################
#       SOIL4.MAK
#       NMake File for DT software tools C Example programs
#       Compiler/Linker switches used:
#                       c               compile only
#                       Zi              Symbolic debugging information
#                       Gs              disable stack checking
#                       AL              large memory model
#                       st:0xXXXX       Stack size
#                       co              CodeView
#                                       #
########################################################################

source = soil4
cflags = /c /AL /Gs
cdebug = /Zi /Od
lflags =
ldebug = /co /st:0x2000

all:  $(source).exe $(source).obj msc_sub.obj

$(source).obj: $(source).c dtst_xmm.h dtst_err.h dtst_tls.h
        cl $(cflags) $(source).c

msc_sub.obj:  msc_sub.c dtst_xmm.h dtst_err.h dtst_tls.h
        cl $(cflags) msc_sub.c

$(source).exe: $(source).obj
        link $(lflags) $(source),,NUL,c:\c600\lib\llibc7.lib+clhpib dtswtool
```

Appendix A5. Soil data acquisition system's *DTST_THS.H* file.

```
/*-----------------------------------------------------------------*
 *-----------------------------------------------------------------*
 *-        Module:  DTST_TLS.H                                   -*
 *-        Function: This file is the Data Translation, Inc.      -*
 *-                definition file for the software tools        -*
 *-                called from Microsoft C V5.1                  -*
 *-----------------------------------------------------------------*
 *-----------------------------------------------------------------*/

#define AD_SECTION    0x0000        /* Select the ad section */
#define DA_SECTION    0x0001        /* Select the da section */
#define AD_DA_SECTION 0x0002        /* Select both sections - SIMUL. START */

typedef struct BOARDS_STRUCT {
    int  num_units;              /* number of boards installed */
    int  board_id;               /* device identifier (see manual) */
    int  base_address;           /* base address of this board */
    char  driver_version[10];    /* driver name asciiz format */
    int  dma_channel_1;          /* 1st dma channel being used */
    int  dma_channel_2;          /* 2nd dma channel being used */
    int  interrupt_level;        /* interrupt being used */
    int  board_timeout;          /* device timeout in seconds */
    int  mux_id;                 /* multiplexer id (reserved) */
    int  ad_setup_bits;          /* a/d acquisition setup bits */
    int  da_setup_bits;          /* d/a acquisition setup bits */
    int  ad_sect_range;          /* current ad input range in millivolts */
    int  da_sect_range0;         /* current da 0 input range in millivolts */
    int  da_sect_range1;         /* current da 1 input range in millivolts */
    int  chan_onbrd;             /* on board input channels */
    int  chan_offbrd;            /* off board input channels */
    int  thrsh;                  /* dac 0 sample for theshold triggering */
}BOARDS;

/** definitions for the ad_setup_bits field of the BOARDS structure **/

#define AD_DATA      0x4000     /* 1 = ad binary data,  0 = ad 2s comp data */
#define AD_POLARITY  0x2000     /* 1 = ad unipolar,    0 = ad bipolar */
#define AD_INPUTS    0x1000     /* 1 = se inputs,      0 = diff inputs */
#define AD_CLOCK     0x0800     /* 1 = ext clock,      0 = int clock */
#define AD_TSEL      0x0400     /* 1 = analog threshold 0 = digital */
#define AD_TPOL      0x0200     /* 1 = rising,         0 = falling */
#define AD_TRIGGER   0x0100     /* 1 = ext clock,      0 = int clock */

/** definitions for the da_setup_bits field of the BOARDS structure **/

#define DA_POLAR1    0x0800     /* 1 = dac 1 unipolar,   0 = da bipolar */
#define DA_POLAR0    0x0400     /* 1 = dac 0 unipolar,   0 = da bipolar */
#define DA_DATA1     0x0200     /* 1 = dac 1 binary data, 0 = da 2s comp data */
#define DA_DATA0     0x0100     /* 1 = dac 0 binary data, 0 = da 2s comp data */
#define DA_TSEL      0x0080     /* 1 = analog threshold 0 = digital */
#define DA_TPOL      0x0040     /* 1 = rising,          0 = falling */
#define DA_TRIGGER   0x0020     /* 1 = ext trigger,     0 = int trigger */
```

```
#define DA_CLOCK     0x0010        /* 1 = ext clock,      0 = int clock */

#define DAC_1       0x0002        /* enable dac 1 */
#define DAC_0       0x0001        /* enable dac 0 */
#define NO_DACS     0x0000        /* disable both dacs */

typedef struct DIGITALIO_STRUCT {
   int    port;               /* digital io port to operate on */
   int    command;            /* digital io command */
   int    direction;          /* digital io direction (input/output) */
   long   dio_value;          /* digital value for input or output */
} DIGITALIO;

/** definitions for the port field of the DIGITALIO structure **/

#define PORT_1      0x0002        /* select dio port 1 */
#define PORT_0      0x0001        /* select dio port 0 */

/** definitions for the command field of the DIGITALIO structure **/

#define SYNCH_MODE  0x0003        /* select synchronized dio mode */
#define SET_XFER    0x0002        /* set direction and then set / get value */
#define XFER_VAL    0x0001        /* set / get a port value */
#define SET_DIR     0x0000        /* set port direction command */

/** definitions for the direction field of the DIGITALIO structure **/

#define DIO_OUTPUT  0x0001        /* set port as outputs */
#define DIO_INPUT   0x0000        /* set port as inputs */

typedef struct FEATURES_STRUCT {
   int    supported_features;     /* major subsystem support */
   char   device_name[20];        /* asciiz encoded device name */
   char   module_name[20];        /* asciiz encoded module name */
   int    max_se_channels;        /* max number of ad se inputs supported */
   int    max_di_channels;        /* max number of ad di inputs supported */
   int    cg_list;                /* depth of channel gain list */
   int    ad_support_bits;        /* ad support flags */
   int    ad_sect_resolution;     /* resolution (in bits) of ad section */
   float  ad_sect_thruput;        /* max thruput of ad section */
   float  ad_sect_clock;          /* ad section base clock frequency */
   int    ad_sect_maxpos;         /* max positive input voltage (in millivolts) */
   int    ad_sect_maxneg;         /* max negative input voltage (in millivolts) */
   int    da_support_bits;        /* da support flags */
   int    da_sect_number;         /* number of dacs on board */
   int    da_sect_resolution;     /* resolution (in bits) of da section */
   float  da_sect_thruput;        /* max thruput of da section */
   float  da_sect_clock;          /* da section base clock frequency */
   int    da_sect_maxpos;         /* max positive input voltage (in millivolts) */
   int    da_sect_maxneg;         /* max negative input voltage (in millivolts) */
   int    digital_io_ports;       /* number of digital io ports */
   int    digital_io_size;        /* size of each digital io port */
   int    num_user_ctrs;          /* number of counter/timers */
   int    ex_support;             /* extended major subsystem support flags */
```

```
} BOARD_FEATURES;

/** definitions for the supported_features field of the BOARD_FEATURES structure **/

#define XSUP      0x8000   /* device has extended support */
#define SSH       0x4000   /* device has SSH module */
#define TSCAN     0x2000   /* device supports triggered channel scans */
#define AAF       0x1000   /* device has anti-aliasing filters on board */
#define ADFIFO    0x0800   /* device has FIFOs on input data stream */
#define SCNFR     0x0400   /* device supports software configurability */
#define MUX       0x0200   /* device supports external multiplexer */
#define DTC       0x0100   /* device supports DT Connect */
#define THTRIG    0x0080   /* device supports on board microprocessor (intelligent) */
#define DSP       0x0040   /* device supports on board DSP */
#define CAL       0x0020   /* device supports on board auto calibration */
#define CLK       0x0010   /* device supports programmable clock functions */
#define TMR       0x0008   /* device supports programmable timer functions */
#define DIO       0x0004   /* device supports digital input / output */
#define DA        0x0002   /* device supports da */
#define AD        0x0001   /* device supports ad */

/** definitions for the ad_support_bits and da_support_bits
        fields of the BOARD_FEATURES structure      **/

#define PGL       0x0200   /* ad subsystem has PGL module */
#define PGH       0x0100   /* ad subsystem has PGH module */
#define SE        0x0080   /* a/d supports single ended inputs */
#define DI        0x0040   /* a/d supports differential inputs */
#define BIN       0x0020   /* adc & dac support binary data */
#define TWOS      0x0010   /* adc & dac support twos complement data */
#define BIP       0x0008   /* adc & dac support bipolar inputs */
#define UNI       0x0004   /* adc & dac support unipolar inputs */
#define XCLK      0x0002   /* adc & dac support external clock */
#define XTRIG     0x0001   /* adc & dac support external trigger */

/** definitions for the ex_supp field of the BOARD_FEATURES structure      **/

#define TRIGLOUT  0x0004   /* device support trigger lockout */
#define SYNCHDO   0x0002   /* device supports synch dio */
#define DT2896    0x0001   /* device supports CGL extension thru DT2896 */

typedef struct SET_ACQ_PARAMS_STRUC {
   int    section;         /* a/d or d/a section of the board */
   int    transfer_type;   /* type of acquisition transfer */
   float  clock_rate;      /* acquisition clock rate */
   float  trig_rate;       /* re-trigger rate */
   float  cut_off;         /* cut off frequency */
} SET_ACQ_PARAMS;

/** definitions for the transfer_type field of the SET_ACQ_PARAMS structure **/

#define BC_INTERRUPT      0x0010  /* buffer cycle - interrupt driven */
#define BC_AUTOINIT       0x0011  /* buffer cycle - auto init dma */
```

```
#define B_INTERRUPT          0x0020  /* burst - interrupt driven */
#define B_INTRPT_TRIG_SCAN          0x0024  /* burst - interrupt triggered scan */
#define B_INTRPT_XTRIG_BUFF                0x0028  /* burst - interrupt xtrigger/buffer */
#define B_INTRPT_XTRIG_BUFF_TRIG_SCAN 0x002C  /* burst - interrupt xtriggered scan
*/

#define B_SINGLE             0x0021  /* burst - single dma channel */
#define B_SINGLE_TRIG_SCAN          0x0025  /* burst - single dma triggered scan */
#define B_SINGLE_XTRIG_BUFF         0x0029  /* burst - single dma xtrigger/buffer */
#define B_SGL_XTRIG_BUFF_TRIG_SCAN   0x002D  /* burst - single dma xtriggered scan
*/

#define B_DUAL                      0x0022  /* burst - dual dma channels (gap free) */
#define B_DUAL_TRIG_SCAN            0x0026  /* burst - dual dma triggered scan */
#define B_DUAL_XTRIG_BUFF           0x002A  /* burst - dual dma xtrigger/buffer */
#define B_DUAL_XTRIG_BUFF_TRIG_SCAN  0x002E  /* burst - dual dma xtriggered scan */

#define C_INTERRUPT                 0x0040  /* continuous - interrupt driven */
#define C_INTRPT_TRIG_SCAN          0x0044  /* continuous - interrupt triggered scan */
#define C_INTRPT_XTRIG_BUFF         0x0048  /* continuous - interrupt xtrigger/buffer */
#define C_INTRPT_XTRIG_BUFF_TRIG_SCAN 0x004C  /* continuous - interrupt xtriggered
scan */

#define C_SINGLE             0x0041  /* continous - single dma channel */
#define C_SINGLE_TRIG_SCAN          0x0045  /* continuous - single dma triggered scan */
#define C_SINGLE_XTRIG_BUFF         0x0049  /* continuous - single dma xtrigger/buffer */
#define C_SINGLE_XTRIG_BUFF_TRIG_SCAN 0x004D  /* continuous - single dma
xtriggered scan */

#define C_DUAL                      0x0042  /* continous - dual dma channels (gap free) */
#define C_DUAL_TRIG_SCAN            0x0046  /* continuous - dual dma triggered scan */
#define C_DUAL_XTRIG_BUFF           0x004A  /* continuous - dual dma xtrigger/buffer */
#define C_DUAL_XTRIG_BUFF_TRIG_SCAN  0x004E  /* continuous - dual dma xtriggered
scan */

typedef struct SINGLE_STRUC {
   int   sing_chan;          /* channel for single value ad operation */
   int   sing_gain;          /* gain for single value ad operation */
   int   sing_dacs;          /* dac mode select */
} SINGLE;

typedef struct TIME_DATE_STRUC {
   int   year;             /* year of time/date stamp */
   int   month;              /* month of time/date stamp */
   int   day;              /* day of time/date stamp */
   int   hour;              /* hour of time/date stamp */
   int   min;              /* minate of time/date stamp */
   int   sec;             /* seconds of time/date stamp */
} TIME_DATES;

typedef struct COUNTER_TIMER_STRUC {
   int   ct_command;            /* counter/timer command */
   int   counter_select;        /* counter to operate on */
   int   clock_select;         /* 0 = internal; 1 = external */
```

```
    long   clock_count;         /* countdown/divider count */
    long   event_flag;          /* users' event counter/flag */
    long   frequency;           /* measured frequency */
} COUNTER_TIMERS;

/*** ct_command definitions ***/

#define RESET_COUNTER  0          /* reset the selected counter */
#define SQUARE_WAVE    1          /* generate a square wave */
#define GENERATE_RATE  2          /* generate pulses at specified rate */
#define SW_ONE_SHOT    3          /* software triggered one shot */
#define HW_ONE_SHOT    4          /* hardware triggered one shot */
#define MEASURE_FREQ   5          /* measure frequency */
#define COUNT_EVENT    6          /* count events */
#define GET_COUNT      7          /* get current clock count */

/*** clock_select definitions ***/

#define CT_INTERNAL    0          /* use internal clock (600KHz) */
#define CT_EXTERNAL    1          /* use external clock */

/*** counter_select definitions ***/

#define DT2812_TMR_CTR 0          /* use timer/counter 0 */

/*** channel scan enable - used for auto-channel sequence ***/

#define CHANNEL_SCAN   0x80       /* enable channel scan */

/*-------------------------------------------------------------------
                    function prototypes
 -------------------------------------------------------------------*/

int far cdecl dt_counter_timer (int unit, int dev_handle,
                    COUNTER_TIMERS far *cnttmrs);
int far cdecl dt_check_buffer  (int unit, int section, int far *buf_handle,
                    int far *buf_status);
int far cdecl dt_create_buffer (int unit, int section, int far *buf_size,
                    int far *buf_ptr, int far *buf_handle);
int far cdecl dt_create_cgl    (int num_ch_ga, int far *chan_array,
                    int far *gain_array);
int far cdecl dt_delete_buffer (int unit, int section, int buf_handle);
int far cdecl dt_get_acq       (int unit, int dev_handle, int section,
                    SET_ACQ_PARAMS far *set_acq_params);
int far cdecl dt_get_board     (int unit, int dev_handle,
                    BOARDS far *brd_struct);
int far cdecl dt_get_buf_tstamp (int buffer_handle, TIME_DATES far *tstamp);
int far cdecl dt_get_features  (int unit, int dev_handle,
                    BOARD_FEATURES far *fea_struct);
int far cdecl dt_initialize    (char far *dev_name, int far *dev_handle);
int far cdecl dt_link_xm_buffer (int unit, int section, int far *xm_buffer_size,
                    long xm_buffer_ptr, int far *buffer_handle);
int far cdecl dt_read_727_channel (int unit, int dev_handle, int channel,
                    int gain, int far *data_ptr);
```

```
int far cdecl dt_reset         (int unit, int dev_handle);
int far cdecl dt_reset_buffer  (int buffer_handle);
int far cdecl dt_set_acq       (int unit, int dev_handle, int section,
                               SET_ACQ_PARAMS far *set_acq_params);
int far cdecl dt_set_board     (int unit, int dev_handle,
                               BOARDS far *brd_struct);
int far cdecl dt_set_dio       (int unit, int dev_handle,
                               DIGITALIO far *dio_struct);
int far cdecl dt_single_acq    (int unit, int dev_handle, int section,
                               long far *sing_samples, SINGLE far *sngl_operation);
int far cdecl dt_start_acq     (int unit, int dev_handle, int section);
int far cdecl dt_stop_acq      (int unit, int dev_handle, int section);
int far cdecl dt_terminate     (int dev_handle);
int far cdecl dt_wait_buffer   (int unit, int section, int far *buffer_handle);
```

Appendix A6. Soil data acquisition system's ***DTST_XMM.H*** file.

```
/*-----------------------------------------------------------------*
 *-----------------------------------------------------------------*
 *-        Module:  DTST_XMM.H                                    -*
 *-        Function: This file is the Data Translation, Inc.      -*
 *-                  definition file for the extended memory      -*
 *-                  management functions using Microsoft         -*
 *-                  HIMEM.SYS called from Microsoft C V5.1.      -*
 *-----------------------------------------------------------------*
 *-----------------------------------------------------------------*/

int far cdecl dt_xm_allocate_block   (int, int far *);
int far cdecl dt_xm_free_block       (int);
int far cdecl dt_xm_get_emb_info     (int, int far *, int far *, int far *);
int far cdecl dt_xm_get_free_mem     (int far *, int far *);
int far cdecl dt_xm_get_version      (int far *);
int far cdecl dt_xm_initialize       ();
int far cdecl dt_xm_lock_block       (int, long far *);
int far cdecl dt_xm_move_dos_to_dos  (int far *, int far *, long);
int far cdecl dt_xm_move_dos_to_xm   (int far *, int, long, long);
int far cdecl dt_xm_move_xm_to_dos   (int, long, int far *, long);
int far cdecl dt_xm_move_xm_to_xm    (int, long, int, long, long);
int far cdecl dt_xm_reallocate_block (int, int);
int far cdecl dt_xm_unlock_block     (int);
```

Appendix A7. Soil data acquisition system's ***DTST_ERR.H*** file.

```
/*********************************************************************
 *                     FILE -- DTST_ERR.H                          *
 *                DTI Software Tools Error Codes               *
 *********************************************************************

 This file contains the DTI Software Tools error codes.  The error
 codes are a word in length.  The upper 4 bits encode the error value
 into one of 16 different range.  This provides for a count of 4096
 different error codes for each range.

 Range                    Type
 ----------------------------------------------------------
 0x0000 - 0x0FFF    INFORMATION (not necessarily an error)
 0x1000 - 0x1FFF    AD SECTION
 0x2000 - 0x2FFF    DA SECTION
 0x3000 - 0x3FFF    BUFFERING
 0x4000 - 0x4FFF    SYSTEM
 0x5000 - 0x5FFF    DIGITAL IO SECTION
 0x6000 - 0x6FFF    EXTENDED MEMORY
 0x7000 - 0x7FFF    COUNTER/TIMER SECTION
 0x8000 - 0xFFFF    RESERVED
 -----------------------------------------------------------------*/


/* -----------------------------------------------------------------
  |                     INFORMATION CODES                        |
  -----------------------------------------------------------------*/

#define errComplete          0x0000   /* command complete */
#define errInProcess         0x0001  /* command in process */
#define errInvalidSection     0x0002  /* invalid board section specified */
#define errNoIOInProcess      0x0003        /* there is no analog i/o in process */
#define errStoppedByUser      0x0004        /* user stopped function in process */
#define errIOInProcess        0x0005  /* analog i/o in process */
#define errDriverNotInit      0x0006  /* driver has not been initialized */
#define errToolkitAlreadyInit 0x0007  /* toolkit has previously been initialized */

/* -----------------------------------------------------------------
  |                     AD SECTION CODES                         |
  -----------------------------------------------------------------*/

#define errAdTrigger         0x1000   /* ad trigger error */
#define errAdOverrun                0x1001    /* ad overrun error */
#define errAdNotReady               0x1002    /* ad system not ready */
#define errAdXferUnsupported  0x1003      /* unsupported ad xfer type */
#define errInvalidChannel     0x1004 /* invalid channel in cgl */
#define errInvalidGain       0x1005   /* invalid gain in cgl */
#define errInvalidNumEntries  0x1006        /* illegal number of cgl entries */
#define errInvalidAdSetup     0x1007  /* illegal ad setup bits in SET_BOARD */
#define errInvalid727Channel  0x1008        /* invalid dt727 channel selected */
#define errChannelSequence    0x1009        /* illegal channel sequence */
```

```
/* ---------------------------------------------------------------------
  |                      DA SECTION CODES                              |
  ---------------------------------------------------------------------*/

#define errDaTrigger          0x2000   /* da trigger error */
#define errDaUnderrun         0x2001   /* da underrun error */
#define errDaNotReady         0x2002   /* da system not ready */
#define errDaXferUnsupported  0x2003   /* unsupported da xfer type */
#define errIllegalDac         0x2004   /* illegal value for dac selected */
#define errInvalidDaSetup     0x2005   /* illegal da setup bits in SET_BOARD */

/* ---------------------------------------------------------------------
  |                      BUFFERING CODES                               |
  ---------------------------------------------------------------------*/

#define errBufferNotFound     0x3000   /* Buffer is not in the BTL */
#define errBufferSize         0x3001   /* illegal buffer size requested */
#define errNoMoreBCBs         0x3002   /* there are no more BCB's avaliable */
#define errBufferOverrun      0x3003   /* buffer not returned in time */
#define errBufferNotReady     0x3004   /* buffer not ready for io */
#define errBuffOnOddAddress   0x3005   /* buffers must reside on a word boundry */
#define errBuffXsDmaBoundry   0x3006   /* Buffer croses dma page boundry */
#define errNoBuffers          0x3007   /* There are not any BCB's in the BTL */
#define errIllegalBtl         0x3008   /* Illegal BTL for requested operation */

/* ---------------------------------------------------------------------
  |                      SYSTEM CODES                                  |
  ---------------------------------------------------------------------*/

#define errUnknownCommand     0x4000   /* unknown command error */
#define errInvalidLun         0x4001   /* invalid logical unit error */
#define errTimeout            0x4002   /* timeout error */
#define errDmaConflict        0x4003   /* dma channel conflict error */
#define errInterruptConflict  0x4004   /* interrupt channel conflict error */
#define errIllegalDma         0x4005   /* illegal dma channel selection      */
#define errIllegalInterrupt   0x4006   /* illegal interrupt channel selection */
#define errUnsupportedFunction 0x4007  /* unsupported function request */
#define errIllegalDriverCall  0x4008   /* illegal driver level call detected */
#define errIllegalBufferType  0x4009   /* illegal buffer for requested op. */
#define errNoInterruptChannel 0x400A   /* function requires int chan selection */
#define errNoDmaChannel       0x400B   /* function requires dma chan selection */
#define errIllegalRate        0x400C   /* invalid clock rate entered */
#define errIllegalScanRate    0x400D   /* illegal channel scan rate */
#define errInvalidScanRate    0x400E   /* invalid scan rate for selected conversion rate */
#define errReducedRate        0x400F   /* gain selection has reduced conversion rate */
#define errDrvNotFound        0x4010   /* driver was not found on open */
#define errTooManyFiles       0x4011   /* cannot open driver, too many files open */
#define errInvalidAccess      0x4012   /* cannot open driver, invalid access specified      */
#define errOpenOther          0x4013   /* unknown error on open driver call */
#define errInvalidHandle      0x4014   /* invalid device handle */
#define errCloseOther         0x4015   /* unknown error closing driver */
#define errInvalidFunction    0x4016   /* invalid driver function requested */
#define errSendOther          0x4017   /* unknown error sending driver packet      */
#define err727notAttached     0x4018   /* no dt727 connected to selected unit */
```

```
#define errIllegalCutOff      0x4019  /* illegal input cut-off frequency */
#define errMinimumRate        0x401A  /* minimum conversion rate not met */
#define errInvalidTimeout     0x401B  /* invalid timeout value */
#define errInvalidBtl         0x401C  /* invalid BTL */
#define errInvalidCgl         0x401D  /* invalid CGL */

/* -----------------------------------------------------------------
  |                 DIGITAL IO SECTION CODES                        |
  -----------------------------------------------------------------*/

#define errIllegalDioPort     0x5000  /* illegal digital port selection */
#define errIllegalDioCommand  0x5001  /* illegal digital port command */
#define errDioDirection       0x5002  /* invalid digital i/o direction */

/* -----------------------------------------------------------------
  |                 EXTENDED MEMORY                                 |
  -----------------------------------------------------------------*/

#define errXMSDrvrNotInstalled 0x6000  /* XMS driver is not installed */
#define errFuncNotImp          0x6001  /* XMS function is not implemented */
#define errVDiskFound          0x6002  /* VDisk device installed */
#define errAllXMAllocated      0x6003  /* all extended memory is allocated */
#define errAllXMHandlesAlloc   0x6004  /* all XMS handles have been allocated */
#define errInvalidXMSHandle    0x6005  /* invalid XMS handle used */
#define errXMSBlockLocked      0x6006  /* block must be unlocked for selected operation */
#define errOverflowLockCount   0x6007  /* the block lock count has overflowed */
#define errLockFailed          0x6008  /* block could not be locked */
#define errBlockUnlocked       0x6009  /* block is already unlocked */
#define errA20LineError        0x600A  /* an error on the A20 address line has occured */
#define errInvalidSrcHdl       0x600B  /* invalid source handle */
#define errInvalidSrcOff       0x600C  /* invalid source offset
#define errInvalidDestHdl      0x600D  /* invalid destination handle */
#define errInvalidDestOff      0x600E  /* invalid destination handle */
#define errInvalidLength       0x600F  /* invalid transfer count in move block */
#define errInvalidOverlap      0x6010  /* cannot perform move block, invalid block overlap */
#define errParity              0x6011  /* cannot perform move block, parity error */
#define errIllegalBufSize      0x6012  /* requested buffer size is invalid */
#define errUnknownXMSError     0x6013  /* an unknown XMS error has occured */

/* -----------------------------------------------------------------
  |                 COUNTER/TIMER SECTION                           |
  -----------------------------------------------------------------*/

#define errInvalidCTCommand   0x7000  /* Invalid counter/timer command */
#define errInvalidCTSelect    0x7001  /* illegal counter/timer selected */
#define errInvalidClkSelect   0x7002  /* illegal clock selected */
#define errInvalidClkCount    0x7003  /* illegal clock count selected */
#define errInvalidSource      0x7004  /* invalid count source specified */
#define errInvalidGate        0x7005  /* invalid gate specified */
#define errCountOverFlow      0x7006  /* counter has overflowed */
#define errInvalidPeriod      0x7007  /* invalid period specified */
#define errInvalidRatio       0x7008  /* invalid ratio specified */
```

Appendix A8. Soil data acquisition system's *MSC_SUB.C* file.

```
/************************************************************************
Copyright (C) 1990. Data Translation, Inc., 100 Locke Drive,
Marlboro Massachusetts 01752-1192.

All rights reserved. This software is furnished to purchaser under a
license for use on a single computer system and can be copied (with
the inclusion of DTI's copyright notice) only for use in such system,
except as may be otherwise provided in writing by Data Translation,
Inc., 100 Locke Drive, Marlboro, Massachusetts 01752-1192.

The information in this document is subject to change without notice
and should not be construed as a commitment by Data Translation, Inc.
Data Translation, Inc. assumes no responsibility for any errors that
may appear in this document.

Data Translation cannot assume any responsibility for the use of any
portion of this software on any equipment not supplied by Data
Translation, Inc.
************************************************************************/

#include <stdio.h>
#include <conio.h>
#include "dtst_tls.h"
#include "dtst_err.h"
#include "dtst_xmm.h"

void wait_for_user ()
{
        printf("\n\n--> Keystroke to continue...");
        flushall();
    getchar();
}

void featur ( int unit, int dev_handle)
{
BOARD_FEATURES features;
int             status;

        printf ("\nGetting Device Features: STATUS = 0x%x\n",
                status = dt_get_features (unit, dev_handle, &features));

        if (status == errComplete)
        {
                printf ("\nFeatures Word...........0x%x",features.supported_features);
                printf ("\nDevice Name..............%s"      ,features.device_name);
                printf ("\nModule Name..............%s"      ,features.module_name);
                printf ("\nInput SE Channels........%d"      ,features.max_se_channels);
                printf ("\nInput DI Channels........%d"      ,features.max_di_channels);
                printf ("\nCGL Size.................%d"      ,features.cg_list);
                printf ("\nAD Section Support.......0x%x",features.ad_support_bits);
                printf ("\nAD Resolution............%d"      ,features.ad_sect_resolution);
                printf ("\nAD Thruput..............%.2f",features.ad_sect_thruput);
```

```
            printf ("\nAD Base Clock............%.2f",features.ad_sect_clock);
            printf ("\nAD Max + Input...........%d"       ,features.ad_sect_maxpos);
            printf ("\nAD Max - Input...........%d"       ,features.ad_sect_maxneg);
            printf ("\nDA Section Support.......0x%x",features.da_support_bits);
            printf ("\nNumber DACS.............%d"        ,features.da_sect_number);
            printf ("\nDA Resolution............%d"       ,features.da_sect_resolution);
            printf ("\nDA Thruput...............%.2f",features.da_sect_thruput);
            printf ("\nDA Base Clock............%.2f",features.da_sect_clock);
            printf ("\nDA Max + Output..........%d"       ,features.da_sect_maxpos);
            printf ("\nDA Max - Output..........%d"       ,features.da_sect_maxneg);
            printf ("\nNumber DIO Ports.........%d"       ,features.digital_io_ports);
            printf ("\nSize DIO Ports (bits)....%d"       ,features.digital_io_size);
        printf ("\nNum user counter/timers..%d"  ,features.num_user_ctrs);
          printf ("\nExtended support.........%d"  ,features.ex_support);
    }

    wait_for_user();

} /* end featur routine */


/*****************************************************************

                    function put_boar
                    =================
    This routine tests the dt_set_board toolkit routine.

*****************************************************************/

void put_boar (int unit, int dev_handle, int dma1, int dma2,
        int intrpt, int ad_setup, int da_setup, int tmout)

{
BOARDS      board;

    board.dma_channel_1  = dma1;
    board.dma_channel_2  = dma2;
    board.interrupt_level = intrpt;
    board.board_timeout  = tmout;
    board.ad_setup_bits  = ad_setup;
    board.da_setup_bits  = da_setup;
    board.chan_onbrd     = 0;
    board.chan_offbrd    = 0;
    board.thrsh          = 0;

    printf ("\nSetting device's BOARD structure:  STATUS = 0x%x\n",
            dt_set_board (unit, dev_handle, &board));

} /* end routine put_boar */

/*****************************************************************

                    function get_boar
                    =================
```

```
        This routine tests the dt_get_board toolkit routine.

 **********************************************************************/

void get_boar ( int unit, int dev_handle )

{
BOARDS      board;
int         status;

        printf ("\nGetting device's BOARD structure  STATUS: = 0x%x\n",
                status = dt_get_board (unit, dev_handle, &board));

        if (status == errComplete)
        {
                printf ("\nNumber Installed Units...%d"         ,board.num_units);
                printf ("\nDevice ID................0x%x"  ,board.board_id);
                printf ("\nDevice Base Address......0x%x"  ,board.base_address);
                printf ("\nDevice Driver Version....%s"         ,board.driver_version);
                printf ("\nDMA Channel 1............%d"         ,board.dma_channel_1);
                printf ("\nDMA Channel 2............%d"         ,board.dma_channel_2);
                printf ("\nInterrupt Channel........%d"         ,board.interrupt_level);
                printf ("\nTimeout Value (secs).....%d"         ,board.board_timeout);
                printf ("\nA/D Setup Value..........0x%04x",board.ad_setup_bits);
                printf ("\nD/A Setup Value..........0x%04x",board.da_setup_bits);
                printf ("\nAD Range.................%d"         ,board.ad_sect_range);
                printf ("\nDA0 Range................%d"         ,board.da_sect_range0);
                printf ("\nDA1 Range................%d"         ,board.da_sect_range1);
                printf ("\nMuxID....................%2X"    ,board.mux_id);
              printf ("\nChannels onboard.........%d"    ,board.chan_onbrd);
        printf ("\nChannels offboard........%d"    ,board.chan_offbrd);
             printf ("\nThreshold trigger value..%d"    ,board.thrsh);
        }

        wait_for_user();

} /* end routine get_boar */

/**********************************************************************

                        function get_acq
                        ================
        This routine tests the dt_get_acq toolkit routine.

 **********************************************************************/

void get_acq (int unit, int dev_handle, int section)

{
SET_ACQ_PARAMS          sap;
int                     status;

        printf ("\nGetting acquisition information  STATUS: 0x%x\n",
                status = dt_get_acq (unit, dev_handle, section, &sap));
```

```
        if (status == errComplete)
        {
                printf ("\nSection.................%d"   ,sap.section);
                printf ("\nTransfer Type...........0x%x",sap.transfer_type);
                printf ("\nConversion Rate.........%.2f",sap.clock_rate);
                printf ("\nTrigger Rate............%.2f",sap.trig_rate);
                printf ("\nCut off frequency.......%.2f",sap.cut_off);
        }

        wait_for_user();

} /* end get_acq routine */

/**********************************************************************

                        function set_acq
                        ================
        This routine tests the dt_set_acq toolkit routine.

 **********************************************************************/

void set_acq (int unit, int dev_handle, int section,
             int tran_type, float rate, float trig_rate, float cut_off)

{
SET_ACQ_PARAMS          sap;

        sap.section       = section;
        sap.transfer_type = tran_type;
        sap.clock_rate    = rate;
        sap.trig_rate     = trig_rate;
     sap.cut_off       = cut_off;

        printf ("\nSetting Acquisition Information  STATUS = 0x%x\n",
                dt_set_acq (unit, dev_handle, section, &sap));

} /* end set_acq routine */
```

Appendix A9. Soil data acquisition system's *soil4.c* file.

```
/*==================================================================||
        soil4.c
|               Written for U.S. Army CERL Soil Project
|               Version 1.0  16.11.95
|                               1.1  12.12.95 Write lots o' data.
|                               1.2  14.12.95 Checks for bad data and waits for |
                               good.
|                               1.3  18.12.95 Includes writes to comment files.
|                                                    Close to user version.
|                               1.4  27.12.95 Save one or multiple files.
|                                                    Ready for use.
|
|               Written by Rich Czerwinski & Nadine Barrie Smith
|                       & Copyright (C)  1990. Data  Translation,  Inc. &
|
|               Functions:
|                               change_settings(char *codes)
|                               collect(numchan, numbuf, numtrials, bsize, clk_rate)
|                               control_panel(int *rpt, int *bur, int *frq, int *amp)
|                               enter_data_file()
|                               enter_one_file()
|                               error_handler (error, routine)
|                               refresh_traces(int rpt)
|                               sample(int numchan, int numbuf, int bsize, int clk_rate)
|                               timer()
|
/*==================================================================
#include <cfunc.h>
#include <chpib.h>
#include <sicl.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <graph.h>
#include "dtst_tls.h"
#include "dtst_err.h"
#include "dtst_xmm.h"

#include <sys\types.h>
#include <sys\timeb.h>

#define NUMCHAN  4
#define NUMBUF          2
#define SAMPLES         2048
#define CLK_RATE  100000.
#define SANITYNUMTRIALS         1
#define NUMTRIALS 1
#define MAXAMP          5.
#define F1      59
#define F2      60
#define F3      61
#define F4      62
```

```
#define F5     63
#define F6     64
#define F7     65
#define F8     66
#define ESC    27

/*-------------------------------
| Global File definations      |
--------------------------------*/
FILE  *out;

int
        j,
        real_trig,
        sample(),
        change_settings(),
        buffer[NUMBUF][SAMPLES*NUMCHAN],
        isc = 7,
        wait_time = 2,                    /* wait 2s before  next action */
        source = 716;                     /* HPIB parameters
*/

void
        collect(),
        control_panel(),
        refresh_traces();

float
        localbuf[SAMPLES*NUMCHAN],
        data[NUMCHAN][SAMPLES];

struct timeb xtime;                   /* for the timer function          */

long
   time1,
        time2;

double
        limits[4],
        minuslimits[4];

char
        str_buffer[100],
        file_name[20],
        name[50];

FILE
        *fp,
        *fopen();

main()
{

        char
```

```
        *codes;

    int
        error,
        rpt = 70,
        bur = 5,
        frq = 2000,
        amp = 100,
        i,
        window_length = SAMPLES;

        limits[0] = 5.;
        limits[1] = 5.;
        limits[2] = 3.5;
        limits[3] = 1.25;
        minuslimits[0] = -5;
        minuslimits[1] = -5;
        minuslimits[2] = -1.5;
        minuslimits[3] = 0.25;

        real_trig = (int)(0.5 +0.5*rpt*CLK_RATE/(1000*NUMCHAN));

/*--------------------------------------
| initialize HPIB
--------------------------------------*/
        error = IORESET (isc);
        error_handler (error, "IORESET");
        error = IOTIMEOUT (isc, 5.0);
        error_handler (error, "IOTIMEOUT");
        error = IOCLEAR (isc);
        error_handler (error, "IOCLEAR");
        codes = "M7,CT0,T0,W1,H0,A0,C0,DTY 50 %,OFS 0.0,L0,FRQ 2 KHZ, AMP
100MV,BUR 5 #, RPT 70 MS, D0";
        error = change_settings(codes);
        error_handler (error, "IOOUTPUTS");

        /*--------------------------------------
        | set up graphics for wave display
        --------------------------------------*/
        _setvideomode( _MAXRESMODE );
        _clearscreen( _GCLEARSCREEN );

        for (j = 0; j < NUMCHAN; j++)
                {
                _setcolor(9);
                _setviewport(10,10+j*70,600,75+j*70 );
                _setwindow(0, real_trig, limits[j], window_length -real_trig,minuslimits[j]);
                _rectangle_w(_GBORDER,real_trig,limits[j],window_length-
real_trig,minuslimits[j]);
                }

        control_panel(&rpt, &bur, &frq, &amp);

        /* if ESC is selected */
```

```
        strcpy(codes, "FRQ 2000 HZ, AMP 10 MV");
        error = change_settings(codes);
        error_handler (error, "IOOUTPUTS");
        _setvideomode( _DEFAULTMODE );

} /* END MAIN */



/*-----------------------------------------------------------
|
|       Function:  refresh_traces(int rpt)
|
-----------------------------------------------------------*/
void refresh_traces(int rpt)
{
        int i, j, window_length = SAMPLES, real_trig;

        real_trig = (int)(0.5 + 0.5*rpt*CLK_RATE/(1000*NUMCHAN));

        /*--------------------------------------
        | Clear out buffers
        --------------------------------------*/
        for(i = 0; i < NUMBUF; i++)
                for (j = 0; j < NUMCHAN*SAMPLES; j ++)
                        buffer[i][j] = 0;
        for (j = 0; j < NUMCHAN*SAMPLES; j ++) localbuf[j] = 0.;

        /*--------------------------------------
        | set up graphics for wave display
        --------------------------------------*/

        _setvideomode( _MAXRESMODE );
        _clearscreen( _GCLEARSCREEN );

        /*--------------------------------------
        | Get data from board
        --------------------------------------*/
        collect(NUMCHAN, NUMBUF, SANITYNUMTRIALS, NUMCHAN*SAMPLES,
CLK_RATE);
        for (j = 0; j < NUMCHAN; j++)
                {
                _setcolor(9);
                _setviewport(10,10+j*70,600,75+j*70 );
                _setwindow(0, real_trig, limits[j], window_length -real_trig,minuslimits[j]);
                _rectangle_w(_GBORDER,real_trig,limits[j],window_length-
real_trig,minuslimits[j]);
                _setcolor(0);
                _moveto_w( 0, -MAXAMP*data[j][real_trig]);
                for (i =real_trig; i<SAMPLES; i++)
                        {
                        data[j][i]=MAXAMP*localbuf[i*NUMCHAN +j];

                        _setcolor(10+j);
```

```
                        _lineto_w( i, -data[j][i] );
                        }
                }
} /*  end refresh_traces  */


/*--------------------------------------------------------------------
| Function:change_settings(char *codes)
|
--------------------------------------------------------------------*/
int change_settings(char *codes)
{
        return (IOOUTPUTS (source, codes, strlen(codes) ));
}  /* end change_settings        */



/*------------------------------------------------------------
|
|       Function: control_panel
|
------------------------------------------------------------*/
void control_panel(int *rpt, int *bur, int *frq, int *amp)
{
        static char codes[80]= "", *rptstr, *burstr, *frqstr, *ampstr, key;
        static int error, i, enable = 1;
        int stop_freq;
        int freq=0;
        int amplitude=10;
        int x,y;

        char freq_ext[5];
        char amp_ext[5];

        while (key != ESC )
         {
         _settextcolor(9);
         _settextposition( 20, 1 );
         sprintf(str_buffer,"[F1]  FRQ %d Hz   ", *frq);
         _outtext( str_buffer );

         _settextposition( 21, 1 );
         sprintf(str_buffer,"[F2]  BUR %d cycles per burst   ", *bur);
         _outtext( str_buffer );

         _settextposition( 22, 1 );
         sprintf(str_buffer,"[F3]  RPT %d ms  ", *rpt);
         _outtext( str_buffer );

         _settextposition( 23, 1 );
         sprintf(str_buffer,"[F4]  AMP %d mV  ", *amp);
         _outtext( str_buffer );

         _settextposition( 24, 1 );
```

```
if (enable)
        sprintf(str_buffer,"[F5]  Turn that !&*$@ thing off!");
else
        sprintf(str_buffer,"[F5]  OK, turn it back on again.");
_outtext( str_buffer );

_settextposition( 25, 1 );
sprintf(str_buffer,"[F6]  Refresh trace display");
_outtext( str_buffer );

_settextposition( 26, 1 );
sprintf(str_buffer,"[F7]  Save to Multiple files");
_outtext( str_buffer );

_settextposition( 20, 40 );
sprintf(str_buffer,"[F8]  Save current data to file");
_outtext( str_buffer );

_settextposition( 27, 1 );
sprintf(str_buffer,"[ESC] Quit");
_outtext( str_buffer );


key = getch();
switch (key)
                {
                case F1:
                        _settextposition( 27, 1 );
                        sprintf(str_buffer,"New Frequency (in Hz):");
                        _outtext( str_buffer );
                        scanf("%s", frqstr);
                        *frq = atoi(frqstr);
                        strcat(codes, "FRQ ");
                        strcat(codes, frqstr);
                        strcat(codes, "HZ");
                        error = change_settings(codes);
                        error_handler (error, "IOOUTPUTS");
                        _settextposition( 27, 1 );
                        sprintf(str_buffer,"                              ");
                        _outtext( str_buffer );
                        control_panel(rpt, bur, frq, amp);
                        break;          /* break case F1  */

                case F2:
                        _settextposition( 27, 1 );
                        sprintf(str_buffer,"New burst length (in cycles):");
                        _outtext( str_buffer );
                        scanf("%s", burstr);
                        *bur = atoi(burstr);
                        strcat(codes, "BUR ");
                        strcat(codes, burstr);
                        strcat(codes, " #");
                        error = change_settings(codes);
                        error_handler (error, "IOOUTPUTS");
```

```
            _settextposition( 27, 1 );
            sprintf(str_buffer,"                              ");
            _outtext( str_buffer );
            control_panel(rpt, bur, frq, amp);
            break;/* break case F2  */

case F3:
            _settextposition( 27, 1 );
            sprintf(str_buffer,"New pulse repitition interval (in ms):");
            _outtext( str_buffer );
            scanf("%s", rptstr);
            *rpt = atoi(rptstr);
            strcat(codes, "RPT ");
            strcat(codes, rptstr);
            strcat(codes, "MS");
            error = change_settings(codes);
            error_handler (error, "IOOUTPUTS");
            _settextposition( 27, 1 );
            sprintf(str_buffer,"                                      ");
            _outtext( str_buffer );
            control_panel(rpt, bur, frq, amp);
            break; /* break case F3  */

case F4:
            _settextposition( 27, 1 );
            sprintf(str_buffer,"New pulse amplitude (in mV):");
            _outtext( str_buffer );
            scanf("%s", ampstr);
            *amp = atoi(ampstr);
            strcpy(codes, "AMP ");
            strcat(codes, ampstr);
            strcat(codes, "MV");
            error = change_settings(codes);
            error_handler (error, "IOOUTPUTS");
            _settextposition( 27, 1 );
            sprintf(str_buffer,"                              ");
            _outtext( str_buffer );
            control_panel(rpt, bur, frq, amp);
            break; /* break case F4  */

case F5:
            if (enable)
                  {
                  strcpy(codes, "AMP 10 MV");
                  enable = 0;
                  }
            else
                  {
                  strcpy(codes, "AMP ");
                  itoa(*amp, ampstr, 10);
                  strcat(codes, ampstr);
                  strcat(codes, " MV");
                  enable = 1;
                  }
```

```
                error = change_settings(codes);
                error_handler (error, "IOOUTPUTS");
                control_panel(rpt, bur, frq, amp);
                break;   /* break case F5  */

case F6:
                refresh_traces(*rpt);
                control_panel(rpt, bur, frq, amp);
                break;

case F7:
                enter_data_file();

                amplitude = 5;
                stop_freq=37;                          /* stop at 10 kHz */
                for ( y = 0; y < 5; y++)
                        {
                        amplitude = 2 * amplitude;
                        itoa(y, amp_ext, 10);
                        freq=750;
                        if ( y == 3       || y == 4 ) /* stop_freq=6kHz */
                                {
                                stop_freq = 21;
                                }
                        for (x = 0; x < stop_freq; x++)
                                {
                                itoa(amplitude, ampstr, 10);
                                strcpy(codes, "AMP ");
                                strcat(codes, ampstr);
                                strcat(codes, " MV, ");

                                itoa(x, freq_ext, 10);

                                freq=freq+250;
                                itoa(freq, frqstr, 10);
                                strcat(codes, "FRQ ");
                                strcat(codes, frqstr);
                                strcat(codes, " HZ");

                                _settextposition( 22, 5);
                                sprintf(str_buffer,"  %s ", codes );
                                _outtext( str_buffer );

                                error = change_settings(codes);
                                error_handler (error, "IOOUTPUTS");

                                strcpy(name, "C:\\soil\\data\\");
                                strcat(name, file_name);
                                strcat(name, ".");
                                strcat(name, amp_ext);
                                if (x < 10) strcat (name, "0");
                                strcat(name, freq_ext);
                                _settextposition( 23, 5);
                                sprintf(str_buffer,"  %s ", name );
```

```
                                        _outtext( str_buffer );


                                        timer();

                                        fp = fopen(name, "w");
                                        collect(NUMCHAN, NUMBUF,
NUMTRIALS, NUMCHAN*SAMPLES, CLK_RATE);
                                        for (i = real_trig; i<SAMPLES; i++)
                                             {
                                   /*        fprintf(fp, "%d   ", i);   */
                                             for (j = 0; j < NUMCHAN; j++)
                                                  {
                                                  fprintf(fp, "%f   ",
MAXAMP*localbuf[i*NUMCHAN +j]);

                                                  }
                                             fprintf(fp, "\n");
                                             }
                                        fclose(fp);
                                        refresh_traces(*rpt);
                                   }    /* end for (x = 0; x < 20; x++)        */
                              } /* end for (y = 0; x < 6; x++)        */

                              strcpy(codes, "FRQ 2000 HZ, AMP 10 MV");
                              error = change_settings(codes);
                              error_handler (error, "IOOUTPUTS");
                              _setvideomode( _DEFAULTMODE );
                              exit(0);
                              break;   /* break case F7  */

                    case F8:
                              enter_one_file();
                              strcpy(codes, "FRQ 2000 HZ, AMP 10 MV");
                              error = change_settings(codes);
                              error_handler (error, "IOOUTPUTS");
                              _setvideomode( _DEFAULTMODE );
                              exit(0);
                              break;  /* break case F8  */

                    default:
                              control_panel(rpt, bur, frq, amp);

               }      /* end switch  */
          } /* end       while (key != ESC )  */
} /* end control_panel()  */


/*---------------------------------------------------------------------
| Function:collect(numchan, numbuf, numtrials, bsize, clk_rate)              |
---------------------------------------------------------------------*/
void collect(int numchan, int numbuf, int numtrials, int bsize, int clk_rate)
{
```

```
        int i, j, buf_num;

        for (j = 0; j < bsize; j ++)
                localbuf[j] = 0.;

        for (i = 0; i < numtrials; i++)
                {
                _settextposition( 29, 1 );
                sprintf(str_buffer,"Trial number: %d.", i);
                _outtext( str_buffer );

                while ((buf_num = sample(numchan, numbuf, bsize, clk_rate)) == -1);
                for (j = 0; j < bsize; j ++)
                        localbuf[j]+=((float)buffer[buf_num][j]-2048.)/((float)numtrials*2048.);
                }
} /* end collect        */



/*------------------------------------------------------------------------
| Function: sample(int numchan, int numbuf, int bsize, int clk_rate)          |
------------------------------------------------------------------------*/
int sample(int numchan, int numbuf, int bsize, int clk_rate)
        {
        int istat, unit = 0, handle, bhandle, flag;
        int i, j, k, hdl[NUMBUF], channel, gain;
        int buf_num;
        char *driver_name = "DT2812$0";
        BOARDS board;
        SET_ACQ_PARAMS sap;

        flag = 0;

        /***** initialize and reset the DT2812 board. *****/
        istat = dt_initialize (driver_name, &handle);
        if (istat != 0)
                {
                flag = -1;
                _settextposition( 28, 1 );
                sprintf(str_buffer,"Error x%x in dt_initialize.", istat);
                _outtext( str_buffer );
                }


        /***    create a couple of buffers        ***/
        for (i = 0; i < numbuf; i ++)
        istat = dt_create_buffer(unit,AD_SECTION,&bsize,&buffer[i][0],&hdl[i]);
        if (istat != 0)
                {
                flag = -1;
                _settextposition( 28, 1 );
                sprintf(str_buffer,"Error x%x in dt_create_buffer.              ", istat);
                _outtext( str_buffer );
```

```
                /* getch(); */
        }

    /***** create a channel/gain list          *****/
    channel = (numchan -1)|CHANNEL_SCAN;
    gain = 0;
    dt_create_cgl (1, &channel, &gain );
    if (istat != 0)
        {
        flag = -1;
        _settextposition( 28, 1 );
        sprintf(str_buffer,"Error x%x in dt_create_cgl.            ", istat);
        _outtext( str_buffer );
        }

    /***** set board functions. *****/
    board.dma_channel_1    = 0;
    board.interrupt_level = 0;
    board.board_timeout    = 15;
    board.ad_setup_bits    = 0x5100;
    board.da_setup_bits    = 0;

    dt_set_board (unit, handle, &board);
    if (istat != 0)
        {
        flag = -1;
        _settextposition( 28, 1 );
        sprintf(str_buffer,"Error x%x in dt_set_board.             ", istat);
        _outtext( str_buffer );
        }

    /* setup the acquisition parameters for a burst adc.     */
    sap.section        = AD_SECTION;
    sap.transfer_type = 0x49; /* continuous, triggered single channel DMA */
    sap.clock_rate     = CLK_RATE;
    dt_set_acq (unit, handle, AD_SECTION, &sap);
    if (istat != 0)
        {
        flag = -1;
        _settextposition( 28, 1 );
        sprintf(str_buffer,"Error x%x in dt_set_acq.               ", istat);
        _outtext( str_buffer );
        }

    /***** start collection operation. *****/
    istat = dt_start_acq ( unit, handle, AD_SECTION);
    if (istat != 0)
        {
        flag = -1;
        _settextposition( 28, 1 );
        sprintf(str_buffer,"Error x%x in dt_start_acq.             ", istat);
        _outtext( str_buffer );
        }
```

```
            istat = dt_wait_buffer(unit, AD_SECTION, &bhandle);
            if (istat != 0)
                    {
                    flag = -1;
                    _settextposition( 28, 1 );
                    sprintf(str_buffer,"Error x%x in dt_wait_buffer.            ", istat);
                    _outtext( str_buffer );
                    }

            buf_num = 0;
            while ( bhandle != hdl[buf_num]) buf_num ++;

            dt_reset_buffer(bhandle);
            if (istat != 0)
                    {
                    flag = -1;
                    _settextposition( 28, 1 );
                    sprintf(str_buffer,"Error x%x in dt_reset_buffer.           ", istat);
                    _outtext( str_buffer );
                    /* getch(); */
                    }

    /***** now stop the a/d conversions *****/
    dt_stop_acq ( unit, handle, AD_SECTION );
    for (i = 0; i < numbuf; i ++)
            {
            dt_delete_buffer(unit, AD_SECTION, hdl[i]);
            if (istat != 0)
                    {
                    flag = -1;
                    _settextposition( 28, 1 );
                    sprintf(str_buffer,"Error x%x in dt_delete_buffer, buffer number %d.",
istat, i);
                    _outtext( str_buffer );
                    }
            }

    /***** all done, terminate toolkit/driver communication. *****/
    istat = dt_terminate (handle);
    if (istat != 0)
            {
            flag = -1;
            _settextposition( 28, 1 );
            sprintf(str_buffer,"Error x%x in dt_terminate.            ", istat);
            _outtext( str_buffer );
            }

    if (flag == 0) return (buf_num); else return (-1);

} /* end sample() */


/*-----------------------------------------------------------------
| Function:  enter_data_file()
```

```
---------------------------------------------------------------*/
enter_data_file()
{
int
        error;

char
        *codes,
        option,
        destination[50],
        thickness_buffer[3],
        weight_buffer[10],
        moisture_buffer[3],
        bucket_buffer[3],
        compaction_buffer[3],
        sequence_buffer[3],
        time_buffer[10],
        date_buffer[10];

                _strtime( time_buffer );
                _strdate( date_buffer );

                _settextcolor(10);
                for ( j = 0; j < 12; j++)
                        {
                        _settextposition( 19+j, 1 );
                        sprintf(str_buffer,"                                              ");
                        _outtext( str_buffer );
                        }

                _settextposition( 20, 1 );
                sprintf(str_buffer,"Enter soil type.           (3 chars only) : ");
                _outtext( str_buffer );
                scanf("%s", file_name);

                _settextposition( 21, 1 );
                sprintf(str_buffer,"Enter soil thickness in cm.  (2 chars only) : ");
                _outtext( str_buffer );
                scanf("%s", thickness_buffer );

                _settextposition( 22, 1 );
                sprintf(str_buffer,"Enter soil weight in kg.                  : ");
                _outtext( str_buffer );
                scanf("%s", weight_buffer );


                _settextposition( 23, 1 );
                sprintf(str_buffer,"Enter Moisture.            (1 chars only) : ");
                _outtext( str_buffer );
                scanf("%s", moisture_buffer );

                _settextposition( 24, 1 );
                sprintf(str_buffer,"Enter Compaction.          (1 chars only) : ");
                _outtext( str_buffer );
```

```
scanf("%s", compaction_buffer );

_settextposition( 25, 1 );
sprintf(str_buffer,"Enter Sequence.          (1 chars only) : ");
_outtext( str_buffer );
scanf("%s", sequence_buffer );

_settextposition( 26, 1 );
sprintf(str_buffer,"Enter Bucket Number.                 : ");
_outtext( str_buffer );
scanf("%s", bucket_buffer );

strcat(file_name, thickness_buffer);
strcat(file_name, moisture_buffer);
strcat(file_name, compaction_buffer);
strcat(file_name, sequence_buffer);
strcpy(destination, "c:\\soil\\data\\");
strcat(destination, file_name);
strcat(destination, ".cmt");

_settextposition( 27, 1 );
sprintf(str_buffer," File Name = %s  ", destination);
_outtext( str_buffer );

while(option != 'y')
{
_settextcolor(11);
_settextposition( 28, 40 );
sprintf(str_buffer," Is the file name correct (y/n)  ");
_outtext( str_buffer );
option = getche();
switch(option)
        {
        case 'n':
                enter_data_file();
        default :
                break;
        } /* end switch */
} /* end while */


if( !access( destination, 0 ) )
    {
    _settextcolor(10);
    for ( j = 0; j < 12; j++)
        {
        _settextposition( 19+j, 1 );
        sprintf(str_buffer,"                         ");
        _outtext( str_buffer );
        _settextposition( 19+j, 35 );
        sprintf(str_buffer,"                         ");
        _outtext( str_buffer );
        }
    _settextposition( 21, 1 );
```

```
                    sprintf(str_buffer," %s  ", destination);
                    _outtext( str_buffer );
                    _settextposition( 22, 1 );
                    sprintf(str_buffer,"File Alredy Exists. Hit any key to exit. ");
                    _outtext( str_buffer );
                    getch();
                    strcpy(codes, "FRQ 2000 HZ, AMP 10 MV");
                    error = change_settings(codes);
                    error_handler (error, "IOOUTPUTS");
                    _setvideomode( _DEFAULTMODE );
                    exit(0);
                    }
            else
              {
              out = fopen(destination,"w");
                            fprintf(out,"\n %s     ",file_name);
                            fprintf(out,"\n %s     ",thickness_buffer);
                            fprintf(out,"\n %s     ",weight_buffer);
                            fprintf(out,"\n %s     ",moisture_buffer);
                            fprintf(out,"\n %s     ",compaction_buffer);
                            fprintf(out,"\n %s     ",sequence_buffer);
                            fprintf(out,"\n %s     ",bucket_buffer);
                            fprintf(out,"\n %s     ",time_buffer);
                            fprintf(out,"\n %s     ",date_buffer);
                            fclose(out);
                    }


            _settextcolor(15);
            for ( j = 0; j < 10; j++)
                    {
                    _settextposition( 19+j, 1 );
                    sprintf(str_buffer,"                                                  ");
                    _outtext( str_buffer );
                    }

            return(0);
} /* end enter_data_file() */


/*-------------------------------------------------------
|
|       Function: enter_one_file()
|
--------------------------------------------------------*/
enter_one_file()
{
int
        i,
        error;

char
        *codes,
```

```
option,
destination[50];

        _settextcolor(10);
        for ( j = 0; j < 12; j++)
                {
                _settextposition( 19+j, 1 );
                sprintf(str_buffer,"                                                    ");
                _outtext( str_buffer );
                }

        _settextposition( 20, 1 );
        sprintf(str_buffer,"Enter file name ( name & extennsion ) : ");
        _outtext( str_buffer );
        scanf("%s", file_name);

        strcpy(destination, "c:\\soil\\data\\");
        strcat(destination, file_name);

        _settextposition( 27, 1 );
        sprintf(str_buffer," File Name = %s  ", destination);
        _outtext( str_buffer );

        if( !access( destination, 0 ) )
                {
                _settextcolor(10);
                for ( j = 0; j < 12; j++)
                        {
                        _settextposition( 19+j, 1 );
                        sprintf(str_buffer,"                         ");
                        _outtext( str_buffer );
                        _settextposition( 19+j, 35 );
                        sprintf(str_buffer,"                         ");
                        _outtext( str_buffer );
                        }
                _settextposition( 21, 1 );
                sprintf(str_buffer," %s  ", destination);
                _outtext( str_buffer );
                _settextposition( 22, 1 );
                sprintf(str_buffer,"File Alredy Exists. Hit any key to exit. ");
                _outtext( str_buffer );
                getch();
                strcpy(codes, "FRQ 2000 HZ, AMP 10 MV");
                error = change_settings(codes);
                error_handler (error, "IOOUTPUTS");
                _setvideomode( _DEFAULTMODE );
                exit(0);
                }
        else
          {
                fp = fopen(destination, "w");
                collect(NUMCHAN, NUMBUF, NUMTRIALS,
NUMCHAN*SAMPLES, CLK_RATE);
                for (i = real_trig; i<SAMPLES; i++)
```

```
                                {
                                /*              fprintf(fp, "%d   ", i);   */
                                for (j = 0; j < NUMCHAN; j++)
                                        {
                                        fprintf(fp, "%f   ",
MAXAMP*localbuf[i*NUMCHAN +j]);
                                        }
                                        fprintf(fp, "\n");
                                }
                        fclose(fp);
                        strcpy(codes, "FRQ 2000 HZ, AMP 10 MV");
                        error = change_settings(codes);
                        error_handler (error, "IOOUTPUTS");
                        _setvideomode( _DEFAULTMODE );
                        exit(0);
                        }

} /* end enter_one_file() */




/*------------------------------------------------------------------
| Function:  error_handler (error, routine)
-------------------------------------------------------------------*/
error_handler (error, routine)
int          error;
char          *routine;
{
 char        *estring;
 char        ch;

 if (error != NOERR)
   {
    printf ("Error in call to %s \n", routine);
    printf ("    Error = %d : %s \n", error, errstr(error));
    printf ("Press <RETURN> to continue: ");
    scanf ("%c", &ch);
   }
}  /* end error_handler */


/*-------------------------------------------------------------
|        Function:  timer()
|        Purpose :  pause for x-seconds between taking data
|                        so the signal can settle.
--------------------------------------------------------------*/
timer()
{
        ftime(&xtime);
        time1=(long)xtime.time;
        while(1)
```

```
        {
        ftime(&xtime);
        time2=(long)xtime.time;
        if ( time2-time1 > wait_time-1)
            break;
        }
} /* end timer */
```

Appendix A10. Procedure to transfer raw data from soil acquisition sytem to worksatation.

After taking soil data stored in c:\soil\data\.....
From bmode.brl.uiuc.edu (128.174.211.58)

TYPE at the PC	ACTION
c:\ **cd c:\ncsa2\ftp -i 128.174.211.3**	ftp into ecstasy in interactive mode
username : **your name**	login t
password : **xxxxxx**	
ftp> **lcd c:**	
ftp> **lcd soil\data**	change into c:\soil\data\ on PC
ftp> **cd /bugs/soil/raw_data**	put data into soil raw data directory
ftp> **mput *.***	put all files with the test... prefix
ftp> **quit**	files from PC should be over at the SPARC

The files should now be on ECSTASY in the directory bugs/soil/raw_data.

Appendix A11. Matlab program *procsoil.m*.

```
%    procsoil.m
%
%    Produce time delay and correlation amplitude values
%    from data collected for US ARMY CERL soil project.
%
%    Version 2.0 January 1996 by RNC & NBS
%
%    Saves data into one big file for processing.


clear
soilfile = input('Enter file name, enclosed in single quotes: ');
thickness = eval(soilfile(4:5));
moisture = eval(soilfile(6));
compaction = eval(soilfile(7));
sequence = eval(soilfile(8));

% check if info is true!
disp('thickness = '); disp(thickness);

disp('Do you want to change this setting?');
ch = input('Type new thickness or -1 to retain this value.');
if (ch ~= -1) thickness = ch; end;

limit1 = round(thickness);
%limit1 = round(1.25*thickness);
limit2 = round(5*thickness);

open_file = [soilfile, '.prc'];
open_file(4:5) = 'xx';
fid = fopen( open_file, 'a');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[pk_max_ind, pk_max_val, pk_hi_ind, pk_hi_val, pk_lo_ind, pk_lo_val, pk_med_ind,
 pk_med_val] = ...
          peak([soilfile, '.200',soilfile, '.201' soilfile, '.202', soilfi
le, '.203'], limit1, limit2);
fprintf(fid, '1000 %1.1f %1.0f %1.0f %1.0f %1.15f %1.15f %1.15f %1.15f %1.15f %1
.15f %1.15f %1.15f\n', thickness, moisture, compaction, sequence,...
pk_max_ind/25000, pk_max_val, pk_hi_ind/25000, pk_hi_val, pk_med_ind/25000, pk_m
ed_val, pk_lo_ind/25000, pk_lo_val);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[pk_max_ind, pk_max_val, pk_hi_ind, pk_hi_val, pk_lo_ind, pk_lo_val, pk_med_ind,
 pk_med_val] = ...
          peak([soilfile, '.204',soilfile, '.205' soilfile, '.206', soilfi
le, '.207'], limit1, limit2);
```

```
fprintf(fid, '2000 %1.1f %1.0f %1.0f %1.0f %1.15f %1.15f %1.15f %1.15f %1.15f %1
.15f %1.15f %1.15f\n', thickness, moisture, compaction, sequence,...
pk_max_ind/25000, pk_max_val, pk_hi_ind/25000, pk_hi_val, pk_med_ind/25000, pk_m
ed_val, pk_lo_ind/25000, pk_lo_val);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[pk_max_ind, pk_max_val, pk_hi_ind, pk_hi_val, pk_lo_ind, pk_lo_val, pk_med_ind,
 pk_med_val] = ...
            peak([soilfile, '.208',soilfile, '.209' soilfile, '.210', soilfi
le, '.211'], limit1, limit2);
fprintf(fid, '3000 %1.1f %1.0f %1.0f %1.0f %1.15f %1.15f %1.15f %1.15f %1.15f %1
.15f %1.15f %1.15f\n', thickness, moisture, compaction, sequence,...
pk_max_ind/25000, pk_max_val, pk_hi_ind/25000, pk_hi_val, pk_med_ind/25000, pk_m
ed_val, pk_lo_ind/25000, pk_lo_val);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[pk_max_ind, pk_max_val, pk_hi_ind, pk_hi_val, pk_lo_ind, pk_lo_val, pk_med_ind,
 pk_med_val] = ...
            peak([soilfile, '.212',soilfile, '.213' soilfile, '.214', soilfi
le, '.215'], limit1, limit2);
fprintf(fid, '4000 %1.1f %1.0f %1.0f %1.0f %1.15f %1.15f %1.15f %1.15f %1.15f %1
.15f %1.15f %1.15f\n', thickness, moisture, compaction, sequence,...
pk_max_ind/25000, pk_max_val, pk_hi_ind/25000, pk_hi_val, pk_med_ind/25000, pk_m
ed_val, pk_lo_ind/25000, pk_lo_val);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[pk_max_ind, pk_max_val, pk_hi_ind, pk_hi_val, pk_lo_ind, pk_lo_val, pk_med_ind,
 pk_med_val] = ...
            peak([soilfile, '.216',soilfile, '.217' soilfile, '.218', soilfi
le, '.219'], limit1, limit2);
fprintf(fid, '5000 %1.1f %1.0f %1.0f %1.0f %1.15f %1.15f %1.15f %1.15f %1.15f %1
.15f %1.15f %1.15f\n', thickness, moisture, compaction, sequence,...
pk_max_ind/25000, pk_max_val, pk_hi_ind/25000, pk_hi_val, pk_med_ind/25000, pk_m
ed_val, pk_lo_ind/25000, pk_lo_val);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[pk_max_ind, pk_max_val, pk_hi_ind, pk_hi_val, pk_lo_ind, pk_lo_val, pk_med_ind,
 pk_med_val] = ...
            peak([soilfile, '.220',soilfile, '.221' soilfile, '.222', soilfi
le, '.223'], limit1, limit2);
fprintf(fid, '6000 %1.1f %1.0f %1.0f %1.0f %1.15f %1.15f %1.15f %1.15f %1.15f %1
.15f %1.15f %1.15f\n', thickness, moisture, compaction, sequence,...
pk_max_ind/25000, pk_max_val, pk_hi_ind/25000, pk_hi_val, pk_med_ind/25000, pk_m
ed_val, pk_lo_ind/25000, pk_lo_val);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[pk_max_ind, pk_max_val, pk_hi_ind, pk_hi_val, pk_lo_ind, pk_lo_val, pk_med_ind,
 pk_med_val] = ...
            peak([soilfile, '.224',soilfile, '.225' soilfile, '.226', soilfi
le, '.227'], limit1, limit2);
fprintf(fid, '7000 %1.1f %1.0f %1.0f %1.0f %1.15f %1.15f %1.15f %1.15f %1.15f %1
.15f %1.15f %1.15f\n', thickness, moisture, compaction, sequence,...
pk_max_ind/25000, pk_max_val, pk_hi_ind/25000, pk_hi_val, pk_med_ind/25000, pk_m
ed_val, pk_lo_ind/25000, pk_lo_val);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[pk_max_ind, pk_max_val, pk_hi_ind, pk_hi_val, pk_lo_ind, pk_lo_val, pk_med_ind,
 pk_med_val] = ...
            peak([soilfile, '.228',soilfile, '.229' soilfile, '.230', soilfi
le, '.231'], limit1, limit2);
fprintf(fid, '8000 %1.1f %1.0f %1.0f %1.0f %1.15f %1.15f %1.15f %1.15f %1.15f %1
.15f %1.15f %1.15f\n', thickness, moisture, compaction, sequence,...
pk_max_ind/25000, pk_max_val, pk_hi_ind/25000, pk_hi_val, pk_med_ind/25000, pk_m
ed_val, pk_lo_ind/25000, pk_lo_val);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[pk_max_ind, pk_max_val, pk_hi_ind, pk_hi_val, pk_lo_ind, pk_lo_val, pk_med_ind,
 pk_med_val] = ...
            peak([soilfile, '.232',soilfile, '.233' soilfile, '.234', soilfi
le, '.235'], limit1, limit2);
fprintf(fid, '9000 %1.1f %1.0f %1.0f %1.0f %1.15f %1.15f %1.15f %1.15f %1.15f %1
.15f %1.15f %1.15f\n', thickness, moisture, compaction, sequence,...
pk_max_ind/25000, pk_max_val, pk_hi_ind/25000, pk_hi_val, pk_med_ind/25000, pk_m
ed_val, pk_lo_ind/25000, pk_lo_val);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fclose(fid);
```

Appendix A12. Matlab program ***loadsoil.m.***

```
function data = loadsoil(filename)
%	function data = loadsoil(filename)
%
%	Load a soil data file, regardless of extra characters
%	at the end of each line.
%
%	Kate Frazier and Kay Raum figured out how to do this.
%

eval(['fp = fopen(''', filename, ''','r','g');']);
data = fscanf(fp, '%f', [4, inf]);
data = data';
fclose(fp);
```

Appendix A13. Matlab program *peak.m*.

```
function [pk_max_ind, pk_max_val, pk_hi_ind, pk_hi_val, pk_lo_ind, pk_lo_val, pk
_med_ind, pk_med_val] = peak(files, limit1, limit2);
% function [pk_max_ind, pk_max_val, pk_hi_ind, pk_hi_val, pk_lo_ind, pk_lo_val,
pk_med_ind, pk_med_val] = peak(files, limit1, limit2);
%


file = loadsoil(files(1:12));
temp1 = file(1:200,1);
temp2 = file(1:200,2);
temp1 = temp1 - mean(temp1);
temp2 = temp2 - mean(temp2);
cor = xcorr(temp1, temp2)';
maxes1 = maxes(cor);

file = loadsoil(files(13:24));
temp1 = file(1:200,1);
temp2 = file(1:200,2);
temp1 = temp1 - mean(temp1);
temp2 = temp2 - mean(temp2);
cor = xcorr(temp1, temp2)';
maxes2 = maxes(cor);

file = loadsoil(files(25:36));
temp1 = file(1:200,1);
temp2 = file(1:200,2);
temp1 = temp1 - mean(temp1);
temp2 = temp2 - mean(temp2);
cor = xcorr(temp1, temp2)';
maxes3 = maxes(cor);

file = loadsoil(files(37:48));
temp1 = file(1:200,1);
temp2 = file(1:200,2);
temp1 = temp1 - mean(temp1);
temp2 = temp2 - mean(temp2);
cor = xcorr(temp1, temp2)';
maxes4 = maxes(cor);

if (maxes1(1) == 0) maxes1 = maxes1(2:length(maxes1)); end;
if (maxes2(1) == 0) maxes2 = maxes2(2:length(maxes2)); end;
if (maxes3(1) == 0) maxes3 = maxes3(2:length(maxes3)); end;
if (maxes4(1) == 0) maxes4 = maxes4(2:length(maxes4)); end;

ind = zeros(1,401);
ind(maxes1) = ones(size(maxes1));
ind(maxes2) = ind(maxes2) + ones(size(maxes2));
ind(maxes3) = ind(maxes3) + ones(size(maxes3));
ind(maxes4) = ind(maxes4) + ones(size(maxes4));

ind = ind(200+limit1:200+limit2);
[maxind, argmaxind] = max(ind);
```

```
allmaxes = find(ind == maxind);


file = loadsoil(files(1:12));

if (length(allmaxes) == 1)
    pk_max_ind = limit1 -1 +allmaxes;
    pk_max_val = coef1(file, pk_max_ind);
else
    pk_max_val = -100;
    for i = 1:length(allmaxes)
        temp = coef1(file, limit1 -1 +allmaxes(i));
        if (temp > pk_max_val),
            pk_max_val = temp;
            pk_max_ind = limit1 -1 +allmaxes(i);
        end;
    end
end;

%find locations of peaks
pk_range = limit1 -1 +find(ind > 0);
values = zeros(size(pk_range));

%evaluate corr function at those points
for i = 1:length(pk_range),
    values(i) = coef1(file, pk_range(i));
end;

%keep only those points corresponding to significant
%correlation values.
pk_range = pk_range(find(values > 0.1*max(values)));

%find upper and lower bounds on tof.
pk_lo_ind = pk_range(1);
pk_hi_ind = pk_range(length(pk_range));
pk_med_ind = median(pk_range);

pk_lo_val = coef1(file, pk_lo_ind);
pk_hi_val = coef1(file, pk_hi_ind);
pk_med_val = coef1(file, pk_med_ind);


%this is the old way of doing this.
%if (max(ind) > 1),
%%     pk_range = limit1 -1 +find(ind > 1);
%      pk_lo_ind = pk_range(1);
%      pk_hi_ind = pk_range(length(pk_range));
%      pk_med_ind = median(pk_range);
%
%      pk_lo_val = coef1(file, pk_lo_ind);
%      pk_hi_val = coef1(file, pk_hi_ind);
%      pk_med_val = coef1(file, pk_med_ind);
%else
%%     pk_range = limit1 +find(ind > 0);
```

```
%       pk_lo_ind = pk_range(1);
%       pk_hi_ind = pk_range(length(pk_range));
%       pk_med_ind = median(pk_range);
%
%       pk_lo_val = coef1(file, pk_lo_ind);
%       pk_hi_val = coef1(file, pk_hi_ind);
%       pk_med_val = coef1(file, pk_med_ind);
%end;
```

Appendix A14. Matlab program *process.m.*

```
function process(infile, outfile)

eval(['load ', infile, '.prc']);
data = eval(infile);

temp = hist(data(:,1), (1000:250:10000));
frequencies = 750+250*find(temp ~= 0);

fid = fopen(outfile, 'w');

    fprintf(fid, 'freq speed1 rho1 atten1 rho_atten1 speed2 rho2 atten2 rho_
atten2 speed3 rho3 atten3 rho_atten3 speed4 rho4 atten4 rho_atten4 freq1 freq2 s
lope1 slope2 slope3 slope4\n');
end;

for i = 1:length(frequencies),
    freq = frequencies(i);

    % Identify thicknesses present for each frequency.
    thick_ind = find(data(:,1) == freq);
    thicknesses = data(thick_ind,2);

    % Most frequent TOF, maximum corr if a tie.
    tof1 = data(thick_ind, 6);
    [speed1, rho1] = regression(thicknesses, tof1);
    speed1 = 0.01/speed1;
    amp1 = data(thick_ind, 7);
    [atten1, rho_atten1] = regression(thicknesses, 20*log10(amp1));
    atten1 = -atten1;

    % Largest possible TOF, corresponding corr.
    tof2 = data(thick_ind, 8);
    [speed2, rho2] = regression(thicknesses, tof2);
    speed2 = 0.01/speed2;
    amp2 = data(thick_ind, 9);
    [atten2, rho_atten2] = regression(thicknesses, 20*log10(amp2));
    atten2 = -atten2;

    % Median TOF, corresponding corr.
    tof3 = data(thick_ind, 10);
    [speed3, rho3] = regression(thicknesses, tof3);
    speed3 = 0.01/speed3;
    amp3 = data(thick_ind, 11);
    [atten3, rho_atten3] = regression(thicknesses, 20*log10(amp3));
    atten3 = -atten3;

    % Lowest possible TOF, corresponding corr.
    tof4 = data(thick_ind, 12);
    [speed4, rho4] = regression(thicknesses, tof4);
    speed4 = 0.01/speed4;
    amp4 = data(thick_ind, 13);
    [atten4, rho_atten4] = regression(thicknesses, 20*log10(amp4));
```

```
    atten4 = -atten4;

    % Log decrement calculation for each frequency.
    if (i < 10),
        filename = [infile(1:3), num2str(max(thicknesses)), infile(6:8),
'.20', num2str(i -1)];
    else
        filename = [infile(1:3), num2str(max(thicknesses)), infile(6:8),
'.2', num2str(i -1)]
    end;

    [freq1, freq2, slope1, slope2, slope3, slope4] = log_dec(filename);

     fprintf(fid, '%1.0f %1.5f %1.5f %1.5f %1.5f %1.5f %1.5f %1.5f %1.5f %1.5
f %1.5f %1.5f %1.5f %1.5f %1.5f %1.5f %1.5f %1.5f %1.5f %1.5f %1.5f %1.5f %1.5f
\n', ...
        freq, speed1, rho1, atten1, rho_atten1, speed2, rho2, atten2, rh
o_atten2, ...
        speed3, rho3, atten3, rho_atten3, speed4, rho4, atten4, rho_atte
n4, ...
        freq1, freq2, slope1, slope2, slope3, slope4);
end;

fclose(fid);
```

Appendix A15. Matlab program *log_dec.m*.

```
function [freq1, freq2, slope1, slope2, slope3, slope4] = log_dec(filename);

% Set system parameters
lengthfft = 512;
samplerate = 25000;

% Load data
orig_frq = 1000 + 250*eval(filename(11:12));
limit1 = round((5/orig_frq)*25000) +2;
limit2 = limit1 + 31;
trace = loadsoil(filename);

% Keep only the 32 samples immediately following the offset of the exciting puls
e.
% Subtract out offset value, and compute FFT.
% Need special processing to handle case of upsloping limit...
s3 = trace(limit1:limit2,3);
s3 = s3 - mean(s3);
S3 = fft(s3, lengthfft);

[x,mx] = max(abs(S3(1:lengthfft/2)));
freq1 = (mx/lengthfft)*samplerate;

[p, S] = polyfit((1:length(s3)), log(abs(hilbert(s3)))', 1);
slope1 = -samplerate*(p(1)/freq1);


s4 = trace(limit1:limit2,4);
s4 = s4 - mean(s4);
S4= fft(s4, lengthfft);

[x,mx] = max(abs(S4(1:lengthfft/2)));
freq2 = (mx/lengthfft)*samplerate;

[p, S] = polyfit((1:length(s4)), log(abs(hilbert(s4)))', 1);
slope2 = -samplerate*(p(1)/freq2);



% Compute log decrement using 3dB bandwidths
band3db_3 = find(abs(S3(1:lengthfft/2)) > max(abs(S3(1:lengthfft/2)))/sqrt(2));
band3db_4 = find(abs(S4(1:lengthfft/2)) > max(abs(S4(1:lengthfft/2)))/sqrt(2));

BW3 = ((max(band3db_3) - min(band3db_3))/lengthfft)*samplerate;
BW4 = ((max(band3db_4) - min(band3db_4))/lengthfft)*samplerate;

slope3 = pi*BW3/freq1;
slope4 = pi*BW4/freq2;
```

Appendix A16. Matlab program *coef.m.*

```
% coef.m
% Compute the amplitude of the correlation function of the
% first two columns of a matrix, evaluated at a given point.

function [value] = coef(temp1, delay)

N = 200;
trace1 = temp1(1:N,1);
trace2 = temp1(1:N,2);

trace1 = trace1 - mean(trace1);
trace2 = trace2 - mean(trace2);

cor = xcorr(trace1, trace2);
value = cor(N +delay)/sum(trace1.^2);
```

Appendix A17. Matlab program *maxes.m.*

```
function [ind] = maxes(x)

% return indices of local maxima in a data stream

z = conv(x, [1 3 1]);

z1 = [z 0];
z2 = [0 z];
z3 = z1 - z2;
l = length(z3);

z4 = z3(1:l-1);
z5 = z3(2:l);

z6 = z4 .* z5;
ind = find(z6 <= 0);
ind_temp = find(z4(ind) > z5(ind));
ind = ind(ind_temp);

for i = 1:length(ind),
    ind(i) = ind(i) - 1;
end;
```

Appendix A18. Matlab program *regression.m*.

```
function [slope, coef] = regression(index, vals)

[p, S] = polyfit(index, vals, 1);
slope = p(1);

newvals = polyval(p, index, 1);
coef = sum( (newvals -mean(newvals)) .* (vals - mean(vals)))/ ...
sqrt(sum((newvals -mean(newvals)).^2) * sum((vals -mean(vals)).^2));
```

Appendix A19. Mean acoustic and soil properties.

Sample File Number	Speed m/s	Attn Coef dB/cm-kHz	Dry Soil Den gm/cc	Total Wet Den gm/cc	Soil Part Den gm/cc	Total Solid %	Total Pores %	H_2O Filled Pores %	% H20 (vol)	% H2O (dry)	% H2O (wet)	Pore Radius µm	Perme-ability $10^{-8}cm^2$
ada111	135	0.341	0.90	0.94	2.49	36.2	63.8	6.1	3.9	4.3	4.1	67.49	22.94
ada113	170	0.276	0.93	0.97	2.49	37.4	62.6	6.4	4.0	4.3	4.1	67.45	19.82
ada141	121	0.200	1.03	1.07	2.49	41.2	58.8	7.5	4.4	4.3	4.1	67.37	13.14
ada211	89	0.684	0.83	0.96	2.49	33.3	66.7	19.8	13.2	15.9	13.7	67.61	32.23
ada241	147	0.576	1.00	1.15	2.49	40.0	60.0	25.0	15.0	15.1	13.1	67.39	14.84
ada311	87	0.393	0.79	1.03	2.49	31.6	68.4	35.2	24.1	30.6	23.4	67.78	38.99
ada341	86	0.707	0.92	1.20	2.49	36.7	63.3	44.5	28.2	30.8	23.5	67.47	21.55
ada511	102	0.494	0.92	1.55	2.49	36.8	63.2	100.1	63.2	68.9	40.8	67.47	21.23
cab111	154	0.172	1.36	1.40	2.64	51.4	48.6	9.6	4.7	3.4	3.3	4.20	0.03
cab112	226	0.098	1.18	1.22	2.64	44.5	55.5	7.3	4.0	3.4	3.3	4.31	0.07
cab141	150	0.594	1.44	1.48	2.64	54.3	45.7	10.8	4.9	3.4	3.3	4.16	0.02
cab211	114	0.364	1.16	1.26	2.64	44.0	56.0	17.8	10.0	8.6	7.9	4.32	0.08
cab241	102	0.914	1.36	1.48	2.64	51.5	48.5	24.9	12.0	8.8	8.1	4.20	0.03
cab311	150	0.450	0.88	1.04	2.64	33.3	66.7	24.1	16.1	18.3	15.4	4.72	0.28
cab341	105	0.897	1.14	1.35	2.64	43.2	56.8	37.6	21.3	18.6	15.7	4.34	0.09
cab511	102	0.582	1.11	1.69	2.64	42.0	58.0	100.0	58.0	52.2	34.3	4.36	0.10
dra111	211	0.274	1.30	1.36	2.52	51.7	48.3	11.9	5.7	4.4	4.2	3.23	0.04
dra114	144	0.163	1.33	1.39	2.52	52.8	47.2	12.5	5.9	4.4	4.2	3.22	0.04
dra141	162	0.538	1.43	1.50	2.52	56.9	43.1	14.8	6.3	4.4	4.2	3.17	0.02
dra143	155	0.513	1.37	1.43	2.52	54.6	45.4	13.5	6.1	4.4	4.2	3.20	0.03
dra211	114	0.499	1.05	1.20	2.52	41.6	58.4	26.1	15.3	14.6	12.7	3.42	0.13
dra212	122	0.283	0.95	1.08	2.52	37.6	62.4	22.1	13.8	14.6	12.7	3.55	0.21
dra241	136	0.555	1.19	1.36	2.52	47.4	52.6	31.1	16.4	13.7	12.1	3.30	0.07
dra311	107	0.505	0.88	1.13	2.52	35.0	65.0	38.8	25.2	28.5	22.2	3.74	0.27
dra341	245	0.710	1.25	1.59	2.52	49.8	50.2	67.1	33.5	26.7	21.1	3.26	0.05
dra511	118	0.579	1.13	1.68	2.52	44.8	55.2	100.1	55.3	49.0	32.9	3.35	0.09
mea111	121	0.142	1.14	1.17	2.62	43.5	56.5	5.0	2.8	2.5	2.4	10.41	0.65
mea112	188	0.104	1.12	1.15	2.62	42.9	57.1	4.8	2.8	2.5	2.4	10.42	0.71
mea141	126	0.631	1.37	1.40	2.62	52.4	47.6	6.7	3.2	2.3	2.3	10.24	0.17
mea211	151	0.391	1.11	1.23	2.62	42.2	57.8	21.0	12.1	10.9	9.9	10.44	0.79
mea241	93	0.619	1.30	1.45	2.62	49.7	50.3	29.2	14.7	11.3	10.1	10.28	0.25
mea311	161	0.277	0.97	1.12	2.62	36.9	63.1	24.1	15.2	15.7	13.6	10.71	1.78
mea341	105	0.903	1.35	1.57	2.62	51.7	48.3	45.6	21.9	16.2	14.0	10.25	0.20
mea511	139	0.311	1.23	1.76	2.62	47.0	53.0	99.9	53.0	43.1	30.1	10.33	0.38
pla114	243	0.207	1.51	1.51	2.65	57.0	43.0	1.0	0.4	0.3	0.3	100.37	8.65
pla115	276	0.230	1.51	1.52	2.65	57.1	42.9	1.0	0.4	0.3	0.3	100.38	12.75
pla216	134	0.232	1.57	1.70	2.65	59.3	40.7	31.5	12.8	8.2	7.6	100.32	4.45
pla217	141	0.476	1.32	1.45	2.65	50.0	50.0	26.6	13.2	10.0	9.1	100.66	82.24
pla241	72	0.495	1.62	1.76	2.65	61.2	38.8	35.6	13.8	8.5	7.9	100.29	2.64
pla311	122	0.807	1.45	1.59	2.65	55.0	45.0	29.3	13.2	9.1	8.3	100.42	15.67
pla341	253	0.136	1.56	1.74	2.65	59.2	40.8	42.7	17.4	11.1	10.0	100.33	4.96
pla511	189	0.753	1.57	2.05	2.65	59.5	40.5	100.0	40.5	26.0	20.6	100.33	6.64
sac111	115	0.392	1.45	1.47	2.65	54.7	45.3	5.9	2.7	1.8	1.8	6.18	0.04

Sample File Number	Speed m/s	Attn Coef dB/cm-kHz	Dry Soil Den gm/cc	Total Wet Den gm/cc	Soil Part Den gm/cc	Total Solid %	Total Pores %	H_2O Filled Pores %	% H20 (vol)	% H2O (dry)	% H2O (wet)	Pore Radius μm	Perme-ability $10^{-8}cm^2$
sac112	164	0.377	1.28	1.30	2.65	48.3	51.7	4.5	2.3	1.8	1.8	6.28	0.11
sac141	139	0.363	1.41	1.43	2.65	53.1	46.9	5.5	2.6	1.8	1.8	6.20	0.05
sac211	112	0.443	1.04	1.17	2.65	39.4	60.6	20.5	12.4	11.9	10.6	6.48	0.54
sac212	123	0.477	0.92	1.02	2.65	34.8	65.2	15.8	10.3	11.2	10.1	6.79	1.26
sac241	122	0.687	1.36	1.51	2.65	51.2	48.8	32.3	15.7	11.6	10.4	6.23	0.07
sac311	121	0.502	1.24	1.52	2.65	46.9	53.1	51.8	27.5	22.2	18.1	6.30	0.14
sac341	176	0.957	1.62	1.94	2.65	61.3	38.7	82.9	32.0	19.7	16.5	6.10	0.01
sac511	207	0.380	1.35	1.84	2.65	50.9	49.1	100.1	49.1	36.4	26.7	6.23	0.07