AN AUTOMATED SYSTEM FOR ULTRASONIC ABSORPTION MEASUREMENTS

BY

DAVID WILLIAM DUBACK
B.S., University of Illinois, 1975

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1981

Urbana, Illinois

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

THE GRADUATE COLLEGE

March 1981

WE HEREBY RECOMMEND THAT THE THESIS BY

DAVID WILLIAM DUBACK

ENTITLED AN AUTOMATED SYSTEM FOR ULTRASONIC ABSORPTION MEASUREMENTS

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

Leon A. Frizzell
Director of Thesis Research

G.W. Swenson, Jr.
Head of Department

Committee on Final Examination†

Chairman

† Required for doctor's degree but not for master's.

O-517

UNIVERSITY OF ILLINOIS
at Urbana-Champaign
The Graduate College
330 Administration Building

FORMAT APPROVAL

To the Graduate College:

The format of the thesis submitted by DAVID WILLIAM DUBACK
for the degree of M.S. is acceptable to the
department of ELEC ENG.

3/6/87
Date

(Signed) [signature]
Departmental Representative

Acknowledgements

The author wishes to thank his advisor, Professor Leon Frizzell, for his help and encouragement during this project. and Professor W. D. O'Brien for suggesting and funding the work, and providing helpful input during the development of the project. Special thanks is due Mr. Joseph Cobb for many hours spent consulting and troubleshooting, and assistance in the construction of, the hardware. Appreciation is extended to Mr. Earl Payne, Mr Michael Haney, Mr. Kenneth Chan, Dr. Ronald Johnston and Mr. Steven Foster for their assistance in solving hardware and software problems, and to Mr. Steven Vaughn for assistance during the programming of a remote interface used for the project. Thanks also to Mr. Bill McNeill for helpful discussion concerning the fabrication of microthermocouples used for the experiments, to Mr. Bob Cicone for help with the drawings appearing in this thesis, and to Mrs. Wanda Elliot, who protected him from the savage jaws of bueaurocracy during this period.

Finally, the author wishes to thank his parents and family for their interest and support during his educational career.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Over the past few years the use of diagnostic ultrasound has increased rapidly, to the point where its use is now common in many medical disciplines. Therapeutic ultrasound has been in use much longer, and new studies are being done on hyperthermic and surgical applications. Despite this common and still growing employment of ultrasound there is still much that is not understood regarding the interaction of ultrasound with biological media. Fundamental to that understanding is a knowledge of the propagation properties such as the velocity, attenuation coefficient and absorption coefficient.

The attenuation and absorption coefficients are both necessary to determine how much of the attenuated energy is scattered since attenation includes absorption and scattering (used here in the broad sense to include any redirection of energy from the sound beam). Additionally, the absorption coefficient is important to understanding ultrasonic interaction with tissue because it gives the amount of energy which is being converted to heat. Thus a knowledge of this pararmeter is important to defining the limits of ultrasound exposure that will avoid thermal damage from ultrasound, and

is also important to the use of ultrasound in hyperthermic applications.

The only means currently available to determine the absorption coefficient, as opposed to the attenuation coefficient is the transient thermoelectric technique (1,2,3). The technique involves the determination of the initial rate of temperature rise, $(dT/dt)_o$ in °C/s, in a sample due to the application of ultrasound of known intensity. The time course of temperature is measured by a small diameter thermocouple imbedded in the sample. The absorption coefficient α (Np/cm) is then determined by

$$\alpha = \frac{\rho C k}{2 I_s} \left(\frac{dT}{dt} \right)_0 \qquad (1)$$

where ρCk is the product of the surrounding medium's density and heat capacity per unit volume at constant pressure (joules/cm**3/ C) and I_s is the ultrasonic intensity at the thermocouple junction.

Most commonly used thermocouples for the application have thermoelectic powers covering the approximate range from 38 - 60 μV/ °C. The temperature rise resulting from the ultrasonic exposure typically one second in duration, must generally be limited to one degree Celsius or less, to avoid

errors resulting from a change in the medium's propagation properties with temperature (ref 3). The combination of these two facts means that systems which can measure accurately signals of a few microvolts are necessary.

Until recently these measurements were recorded on photosensitive paper by deflection of a light beam via a mirror attached to a sensitive galvonometer. The rate of temperature rise was then determined by time-consuming hand analysis of the recordings. This has likely been one reason for the limited reporting of ultrasonic absorption coefficient data for biological tissues, as evidenced in two recent comprehensive compilations of ultrasonic propagation property data (4,5).

This thesis reports the development of an automated system, employing digitization of the thermocouple response, and computer processing of the data for the determination of the absorption coefficient within seconds of the ultrasonic irradiation. This allows immediate analysis of the data to determine whether results are consistent with those from preceeding irradiations or earlier measurements. Additionaly, it provides the potential for new procedures of analysis. One example, discussed in this thesis, is the summation and averaging of data from repeated responses in order to obtain acceptable signal-to-noise ratios with siganls which are at lower levels than normal. With this

feature, lower ultrasonic intensities can be employed, which will facilitate measurements at higher frequencies than have been used previously and will ensure that infinitesimal wave theory applies.

The organization of the remainder of this thesis is as follows. Chapter two describes the equipment used, and chapter three the software developed to allow operator-controlled measurements via a mini-computer. Chapter four presents results obtained using this system, and chapter five discusses suggestions for the improvement of the existing system and related programming. Included in the appendices are program listings with detailed documentation for each program unit.

CHAPTER 2

EQUIPMENT

A block diagram of the experimental system is shown in Figure 1. The output of a thermocouple, which has been imbedded in the tissue under study, is connected to the input of a precision modular instrumentation amplifier (Analog Devices model 606M). This amplifier serves as a preamplifier and has a fixed gain of 80 dB. The output of the preamplifier is connected to the positive input of an integrated circuit instrumentation amplifier (Analog Devices model 522B). The negative input to this section of the amplifier is either a ground level or a variable DC level used to compensate for the effect of bias when measurements are made in physiological saline. The voltage supplied can be either manually selected or computer-controlled. The gain of the second stage is variable and has four settings: 9.5, 15.1, 20.8 and 26.4 dB; this allows for measurements under a variety of conditions.

The output of the second stage is passed through an active filter having a four-pole Butterworth response with a 3-dB corner frequency of one hundred hertz. It is then presented to the input of an analog-to-digital converter interfaced to a Perkin-Elmer 7/32 minicomputer which

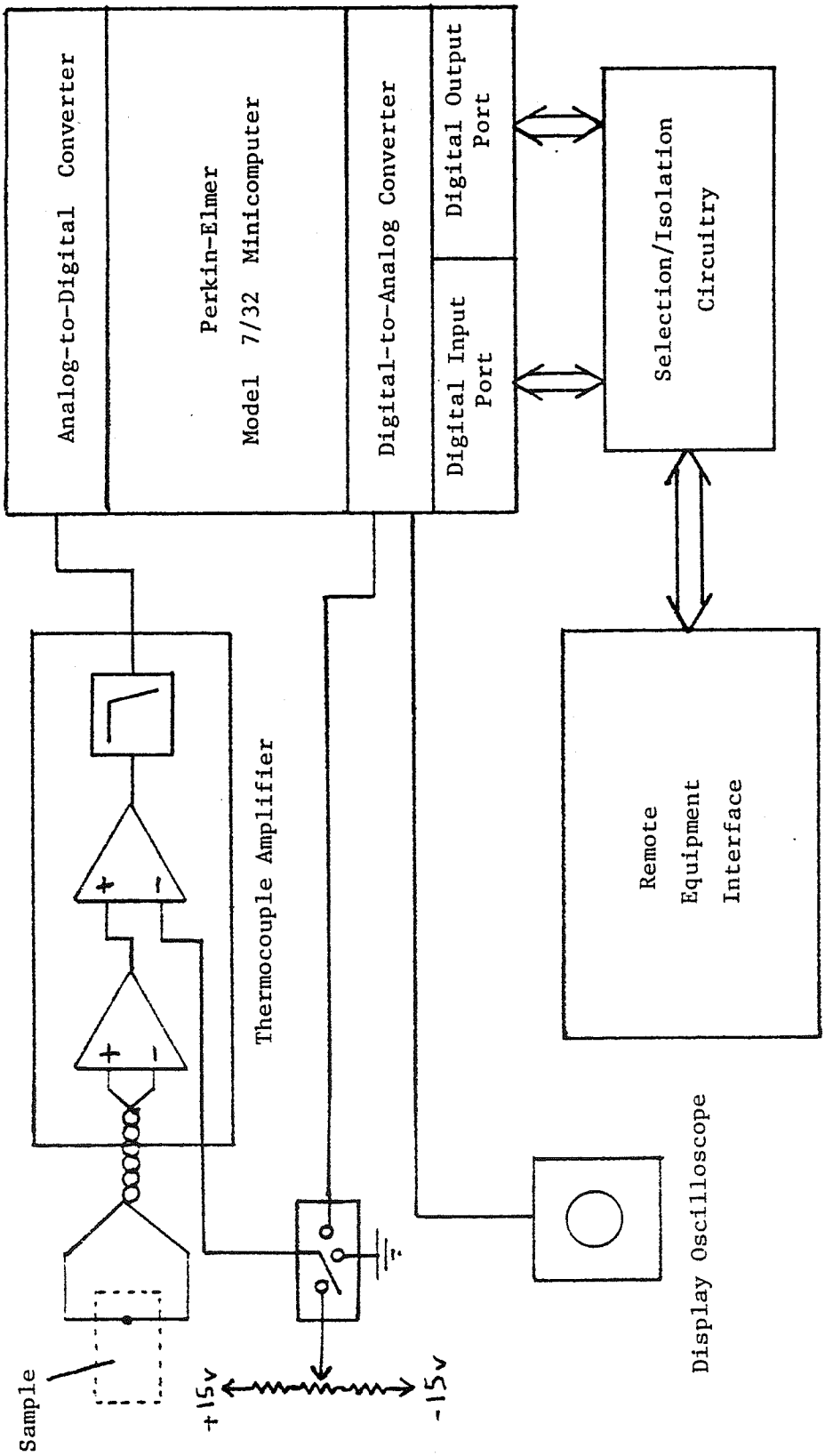


Figure 1 Block diagram of the experimental system.

controls the experiment and processes the digitized data to compute the absorption coefficient. Ten bits of precision are achieved at a sampling rate of 16,384 samples/second; since the frequencies of the signals under study are much lower than this rate, not all data points are stored.

The Perkin-Elmer 7/32 minicomputer, which serves as the mainframe in the course of an experiment, is a thirty-two bit microprogrammed machine. It is capable of time-sharing operation, and one hundred thirty kilobytes of memory are available for program and data storage. Peripherals include two magnetic tape drives, a high speed line printer and cathode ray terminals for operator interaction. In addition, a digital input-output port which is connected directly to the multiplexor bus of the processor interfaces external user circuits, placing them under program control. A digital-to-analog converter allows display of data (stored and/or calculated) on an oscilloscope.

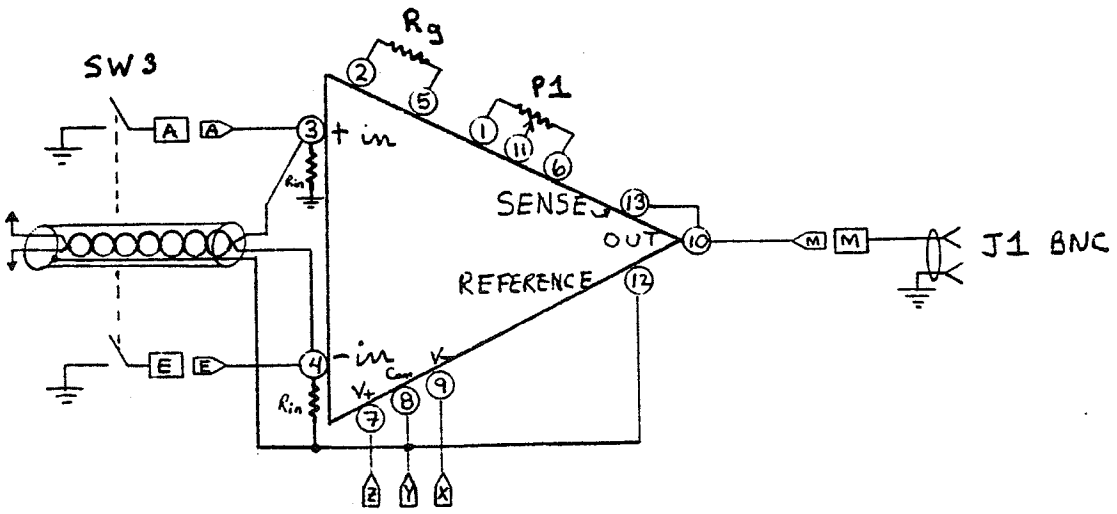
The selection/isolation circuitry gives program control of the equipment used in the experiment by providing an electronic path to and from the equipment with electrical isolation between the computer and the remote equipment. A selection feature allows independent use of the digital input/output port by other programs.

The equipment interface itself controls the position of the transducer relative to the tissue and thermocouple (in three dimensions), independent movement of the thermocouple through the tissue, the intensity and the duration of the ultrasound exposure. The following sections describe these systems in detail.

Thermocouple amplifier

The schematic of the first stage of amplification, the preamplifier, is shown in Figure 2. The power supply used to provide the $\pm$ 15 volts needed for the amplifiers is a Power One model AA15-.8. It outputs $\pm$ 15 volts at 800 ma., and is also used for the filter section, as well as for duplicate amplifiers in multiple thermocouple applications. Tantalum filter capacitors (10 μf, 35 volt) are used at the power pins of all devices, to minimize power supply ripple.

The input to this amplifier consists of two leads from the thermocouple and a third lead connected to the shield for the input lines at its furthest point from the amplifier. This latter line is grounded within the amplifier, and reduces noise pickup in the input lines. The two input leads themselves are of bare copper, beginning with the connections to the thermocouple wires and ending at the amplifier inputs.



R_{in} = 40K ¼w, 1% metal film resistor (2 places)

R_g = 40 " " " " "

P1 = 100K 5% ten turn cermet potentiometer

Values in ohms; K = 1000 ohms

Input leads of bare copper; connections to pins 3 and 4 made with low temperature solder.

Figure 2 Schematic of first amplifier stage.

Connections for the inputs (pins 3 and 4; see figure 2) are made with a special "low-temperature" solder to eliminate drifts due to temperature variations at the input.

The nullification of offset at the input terminals is provided by the ten-turn cermet potentiometer (P1) connected to pins 1, 11 and 6. Once adjusted for zero offset, the amplifier is specified to have less than one-quarter of a microvolt per degree Celsius drift vs. temperature, and less than three microvolts drift per volt variation in power supply. The observed drift after initial zeroing has repeatedly been verified to be negligible.

At the output, this amplifier provides means for sensing the voltage at remote loads (SENSE), and the offsetting of the output with respect to an arbitrary voltage level (REFERENCE). Neither of these features is used (the sense line is usually used when some amount of current is being passed over long lines, which might result in a loss of voltage at the load; this is not the case here.), and the connections shown are per manufacturers' directions for applications not requiring them, i. e., SENSE connected to the output and REFERENCE connected to ground.

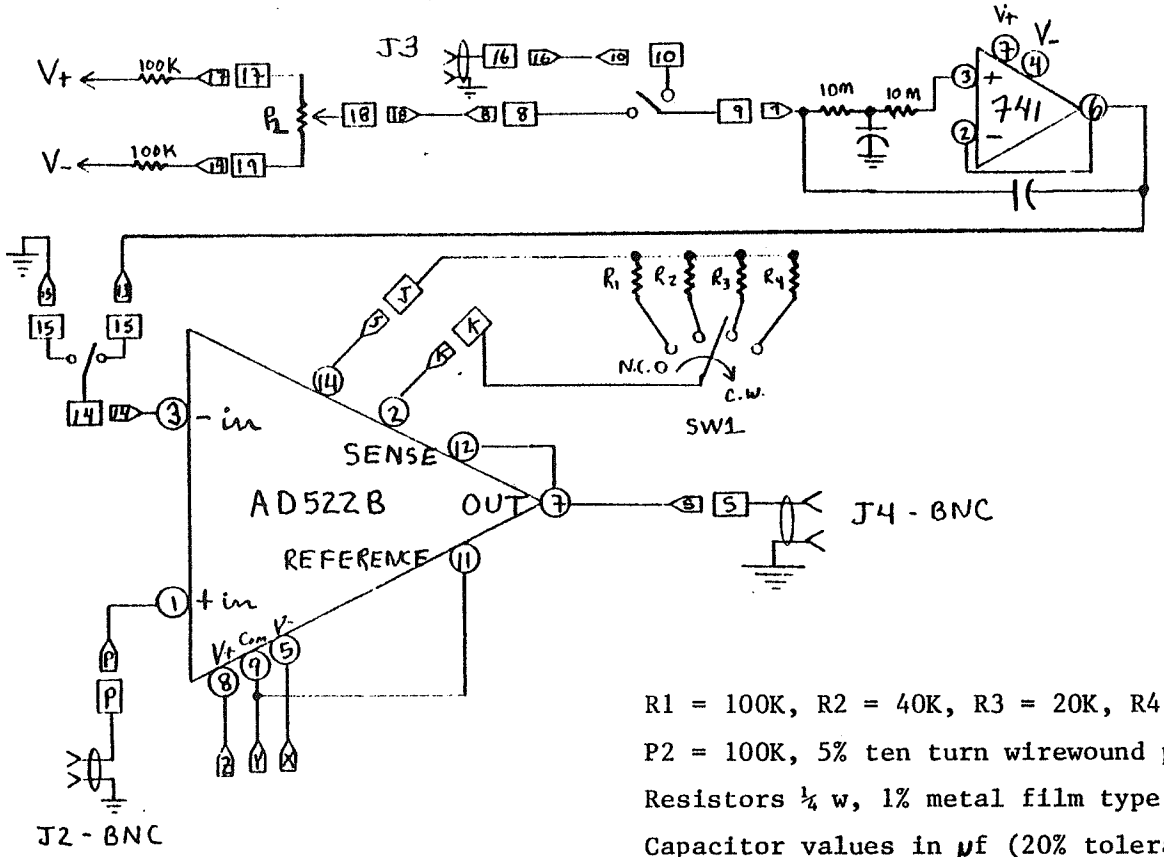
The gain of the amplifier is fixed by the value of the gain resistor (Rg) connected between terminals 2 and 5. The value of this resistor determines the gain according to the equation

$$\text{Gain} = 1 + \frac{400{,}000}{Rg}$$

where Rg is given in ohms. is metal film (1/4W, 1%) with a nominal resistance of 40 ohms; hence, the gain of this stage is fixed at (nominally) 10,001, or 80 db.

The second stage of the amplifier consists of an Analog Devices Model 522B Integrated Circuit Instrumentation Amplifier (see Figure 3). It, too requires only the addition of an external gain resistor to be employed as an amplifier. Of the differential input of the second stage, the positive input is connected to the output of the first stage and the negative input is switchable between a steady ground level and a variable voltage which can be supplied either by the digital-to-analog converter on the 7/32, or by manually adjusting the potentiometer in the resistor-potentiometer dividing circuit. This variable voltage is necessary to compensate for the effects of large shifts in the steady-state output of the preamplifier.

The voltage level shifts mentioned above are induced when measurements are made in physiological saline.



R1 = 100K, R2 = 40K, R3 = 20K, R4 = 10K

P2 = 100K, 5% ten turn wirewound potentiometer.

Resistors ¼ w, 1% metal film type; K = 1000Ω

Capacitor values in μf (20% tolerance, 25WVDC)

Figure 3 Schematic of second amplifier stage.

The different metals in the thermocouple wires used (Chromel is an alloy of Nickel and Chromium;Constantan is an alloy of Copper and Nickel) together with the presence of salt ions in solution gives rise to half-cell potentials which cause a current to flow in a loop between the thermocouple wire and the tissue. This bias current has to given rise to a voltage of approximately 3.5v at the output of the first stage, when using a 13 μm diameter lap-soldered thermocouple junction. This level would ordinarily saturate the second stage, which nominally has a gain of 10. A positive level equal to the "bias" level is applied to the negative input of the 522B to maintain output levels within the 10v linear range of the amplifier.

The nullifying voltage may be supplied by the digital-to-analog converter of the 7/32 as dictated by the software. In this case, the output of the first stage is observed using the 7/32's analog-to-digital converter with an (approximately) equal voltage applied to the input of an active filter. The filter is necessary to eliminate the effects of high frequency noise observed at the output of the digital-to-analog converter. The noise originates in the processor ground of the 7/32, and is due to the high switching currents in the processor. The filter has a two-pole Butterworth response with a 3db corner frequency of 0.01 Hz. The output of the filter is input to the negative input of the 522B.

When manually adjusting the voltage, P1 of figure 3 is turned until the voltage at the negative input is slightly below (in absolute value) that of the output of the first stage. This ensures that all inputs to the A/D are positive (the A/D only converts positive voltages). The numercial routines used in calculations allow for some offset in the steady state, and thus no attempt is made to completely nullify the bias voltage.

The gain of the second stage is determined with a single resistor, as in the first stage; however, since this stage is configured to have variable gain, several different resistors are used as "gain resistors". The gain equation for the 522B is

$$\text{Gain} = 1 + \frac{200{,}000}{Rg}$$

where Rg is the value of the gain resistor in ohms. Metal film resistors (1/4w, 1%) are used, having values of 100K, 40K, 20K, and 10K, yielding selectable gains of 3, 6, 11, and 21, respectively (9.5, 15.6, 20.8 and 26.4 dB.).

The terminals provided for nullification of offsets in the 522B are not used, since experience has shown these offsets to be negligible. Sense and reference terminals are also provided for the 522B; they are not used and are connected as per manufacturers directions for applications not requiring them, i.e., SENSE connected to the output and

REFERENCE connected to ground. The DATA GUARD terminal (ordinarily driven by the shield of the input lines) is not used, since the first stage to second stage connection is short and is contained within a shielded enclosure. It is left unconnected, as per manufacturer's directions for applications not requiring its use.

The output of the second stage is input to a low-pass filter which provides a four-pole Butterworth response with a corner frequency of 100 Hz. The filter consists of two series-connected Datel UFT-2 active filter chips, each of which implements a two-pole lowpass Butterworth function. When cascading two-pole sections to obtain a four-pole response, the 3 db corner frequency of the individual sections is the same, but the damping factor is different for the two stages. The mathematics to show this is straightforward, and a discussion of it can be found in any handbook of active filter design (see for example ref. 8)

The construction of an active filter from Datel UFT-2 chips is described in Datel Application Note Number FLUAJ05901. For corner frequencies above 50 Hz., the implementation of a lowpass filter in the inverting configuration involves the connection of four external resistors, as per Table 1.

TABLE 1 - Values of external resistors for an inverting lowpass filter.

R1	R2	R3	R4=R5
100K	OPEN	$\frac{100K}{3.80Q-1}$	5.03 x 10

The values of the damping factors for the individual sections of this implementation were obtained from the text ACTIVE FILTER COOKBOOK , by Donald Lancaster. (ref. 8, p. 75) They are 1.848 and 0.765, for the first and second stages of filtering, respectively. These Q's must be multiplied by a factor of 0.5 to make them compatible with the Q's used by Datel, as the definition of Q is different for the two sources.

The application of the formulas in Table 1 to the two desired damping factors and assuming a 3db corner frequency of 100 Hz. yields the values for the resistors shown in Table 2.

TABLE 2 - Values of external resistors used for this system

	R1	R2	R3	R4=R5
1st Section	100K	OPEN	90.9K	500K
2nd Section	100K	OPEN	23.7K	500K

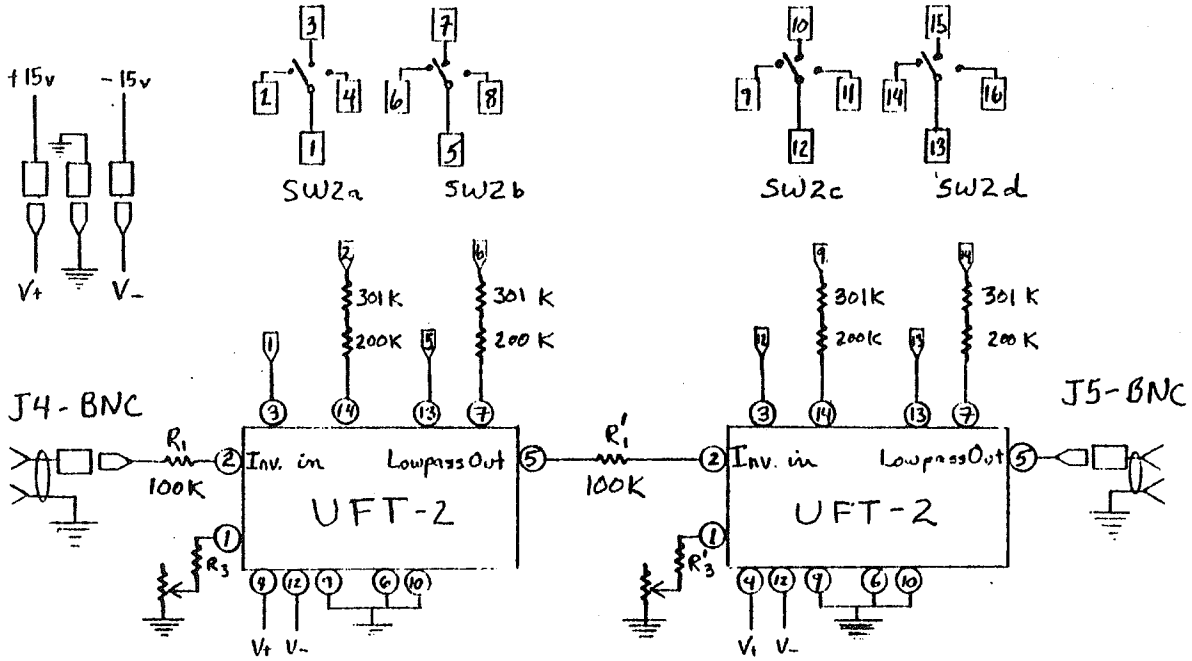
The use of two inverting filters in series gives a non-inverted output.

The schematic for the assembled filter is shown in Figure 4. The output of the filter is input to one channel of the analog-to-digital converter on the 7/32.

Analog-to-Digital Conversion

The analog-to-digital converter (A/D) is a peripheral device of the 7/32 and is described in Perkin-Elmer publication number 29-443R03, "Mini-I/O System Instruction Manual". Ten bits of precision are achieved for an input range of 0 to +10.24v . The A/D is normally free-running and converts at a rate of approximately 25 kHz. Software latency times account for the fact that the actual rate achievable is somewhat less than this; rates of about 22 kHz have been achieved using specially written assembly-language programs (see Appendix II description of subroutine ATD for an example of one).

Under normal circumstances, the free-running conversion rate of the A/D is not constant, leading to some uncertainty in asssociating time values with samples. A uniform time base for the A/D converter was established so that sampling rates would be constant and controlled.



R3 = 90.9K, R3' = 23.7K

P3 = 10K P3' = 5K

Resistors ¼ w, 1% metal film type

Capacitor values in f (20%, 25WVDC tantalum)

Potentiometers 5% ten turn cermet type

Figure 4 Schematic of filter stage.

This was accomplished by constructing a crystal-controlled clock and connecting it to the A/D. It is found on the same circiut card as the selection/isolation circuitry, as shown in Figure 5a. A schematic of the clock is shown in Figure 6.

The operation of the clock is as follows: The two inverters IC10a and IC10b are biased into their active regions by 470 ohm resistors and coupled by a 47,000 pf capacitor. In parallell with these two inverters is a 6.5536 MHz crystal in series with a trimming capacitor. Together, these components form a 6.5536 MHz oscillator which can be trimmed slightly to provide a 6.5384 MHz signal. This frequency, when divided by four hundred, yields a frequency (16.384 kHz) which is a power of two, a feature which eases digital processing of the signals. The output of the oscillator is buffered by inverters IC10c and IC10d, and appears at the input to IC9. IC9 is a 74LS490 dual decade counter, the two sections of which have been wired to provide a divide-by-one-hundred function. The output of the second counter (pin 9) is thus a 65.384 kHz square wave. This is divided by four by the two J-K flip-flops of IC8 to produce the desired 16.384 kHz square wave, which appears at pin 11 of IC8.

In order to be clocked externally, the A/D system on the 7/32 requires that pin 100 of connector 5 on the 7/32 A/D board be grounded.

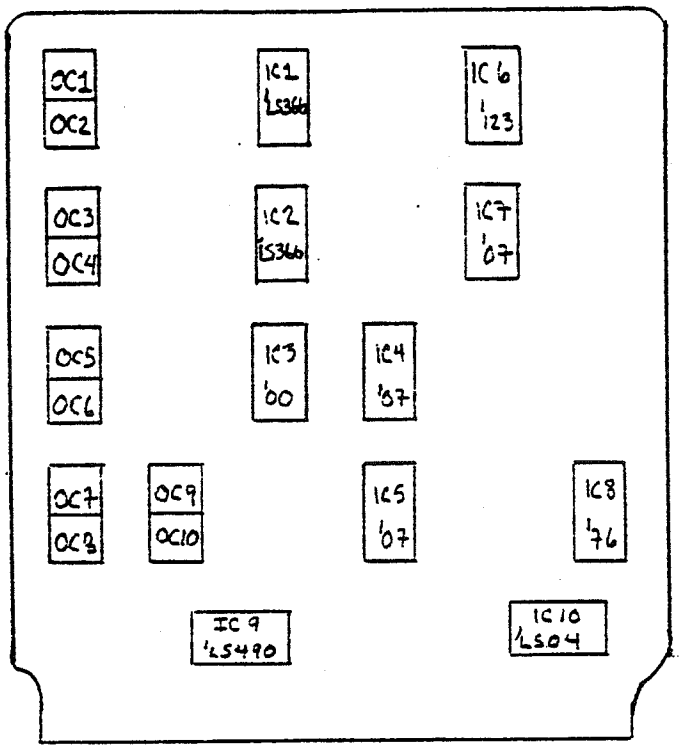


Figure 5a Card layout - selection/isolation circuitry and external clock for A/D

Figure 5b Front Panel for selection/isolation and ext. A/D clock.

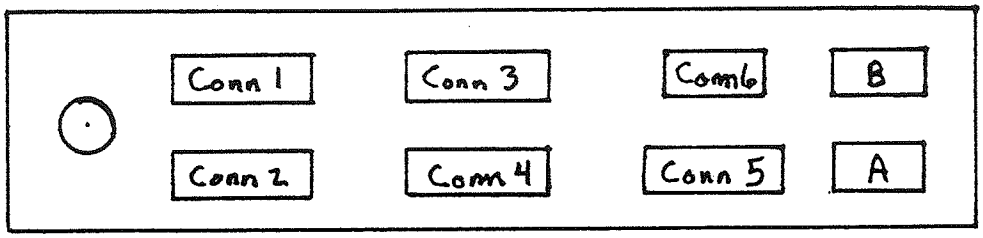


Figure 5c Rear Panel Connections for selection/isolation and ext. A/D clock.

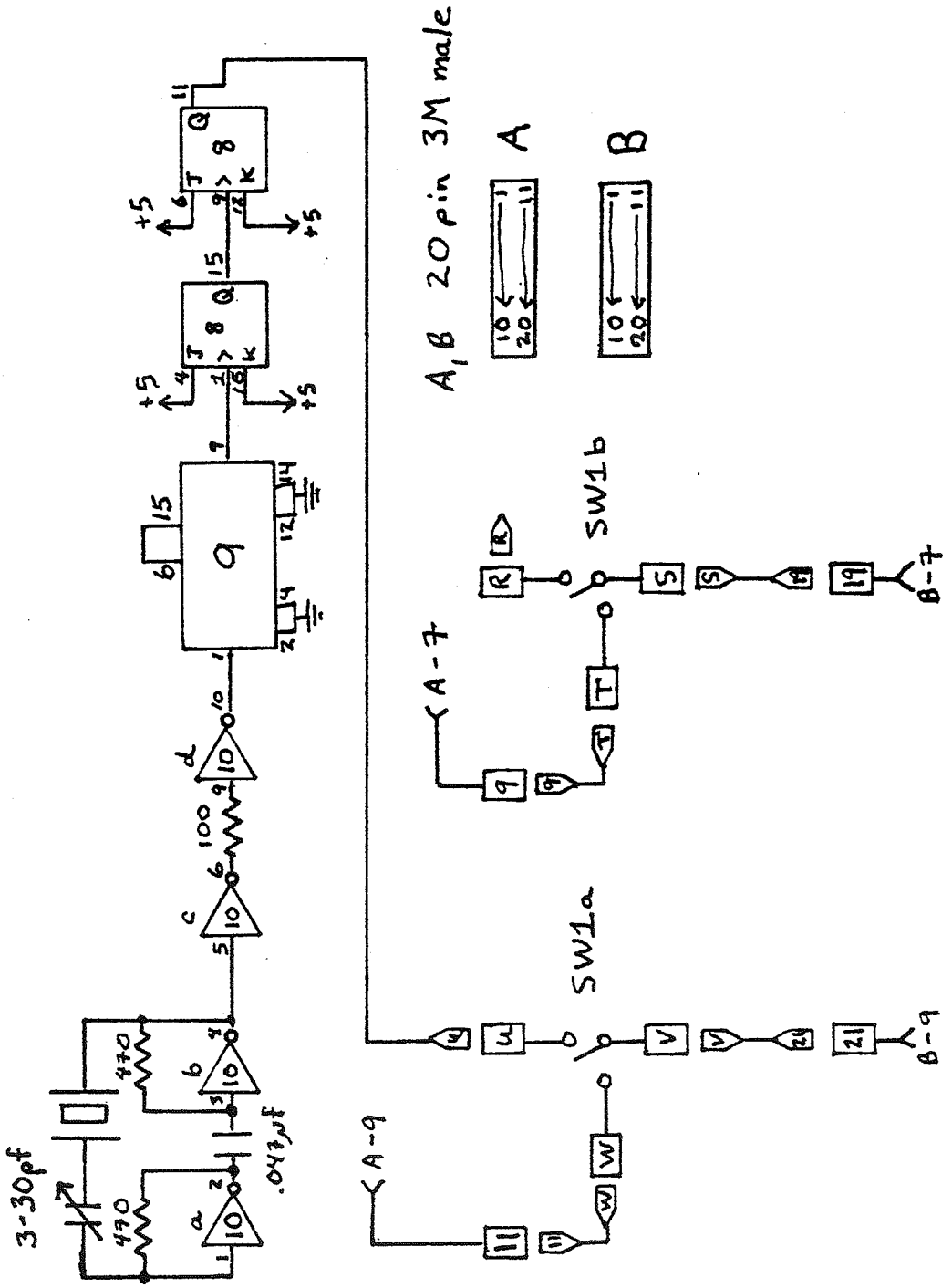


Figure 6 Schematic of external clock for A/D.

This signal is supplied at pin 7 of connector B, and a ribbon cable routes the signal to the 7/32. Figure 5b shows the front panel of the selection/isolation and external A/D clock unit, for reference. Whenever SW2 is in the "external clock 1" mode, a ground level is supplied to this pin. Whenever SW2 is in the "external clock 2" mode (down), the signal appearing at pin 7 of connector A is routed to the 7/32. This allows the switching of another user's external clock onto the 7/32 A/D board. The grounding of pin 100 of the A/D board enables a TTL-level clock applied at pin 101 of the A/D board to determine the start of the next analog-to-digital conversion. This line is connected to pin 9 of connector B. The signal supplied to the 7/32 can be either the above-mentioned clock ("external clock 1"; 16.384 kHz.) or a clock supplied by another user of the 7/32, "external clock 2", which is found at pin 9 of connector A. Figure 5c shows the rear panel connections for the selection/isolation and external A/D clock unit.

It is very important to note that unless the data presented by the A/D is read by the processor before the next rising edge of the clock, no new conversion will be started. A software routine (ATD, described in Appendix I) assures that each clock edge starts a new conversion.

Selection/Isolation circuitry

The selection/isolation circuitry provides an electronic path from the digital input-output (DIO) ports of the 7/32 to the experimental equipment associated with the project. Electrical isolation is provided between the two systems, and a selection feature assures that only those programs specifically requesting interaction with this equipment affect its use. This selection is necessary because there are many devices connected simultaneously to the DIO ports.

A schematic of the outputs from the 7/32 and their controls appears in Figure 7 and input lines together with their controls appear in Figure 8. The sixteen bits of the digital output port appear in groups of eight, each bit with a ground, at pins 1-8 of connector 3 and pins 1-8 of connector 4. The handshake lines which control the exchange of information to and from the output port appear at pins 9 and 10 of connector 3.

Referring to Figure 7, bits 0-3 of the output port are decoded by IC3 to respond to the device code "2". The low level resulting from the selection of device 2 (DEV 2) enables the one-shots in IC6 (which control handshakes from the DIO ports) and the tri-state inverters (IC1 and IC2 of Figure 8) which control inputs of handshakes to both the input and output ports of the DIO, as well as the

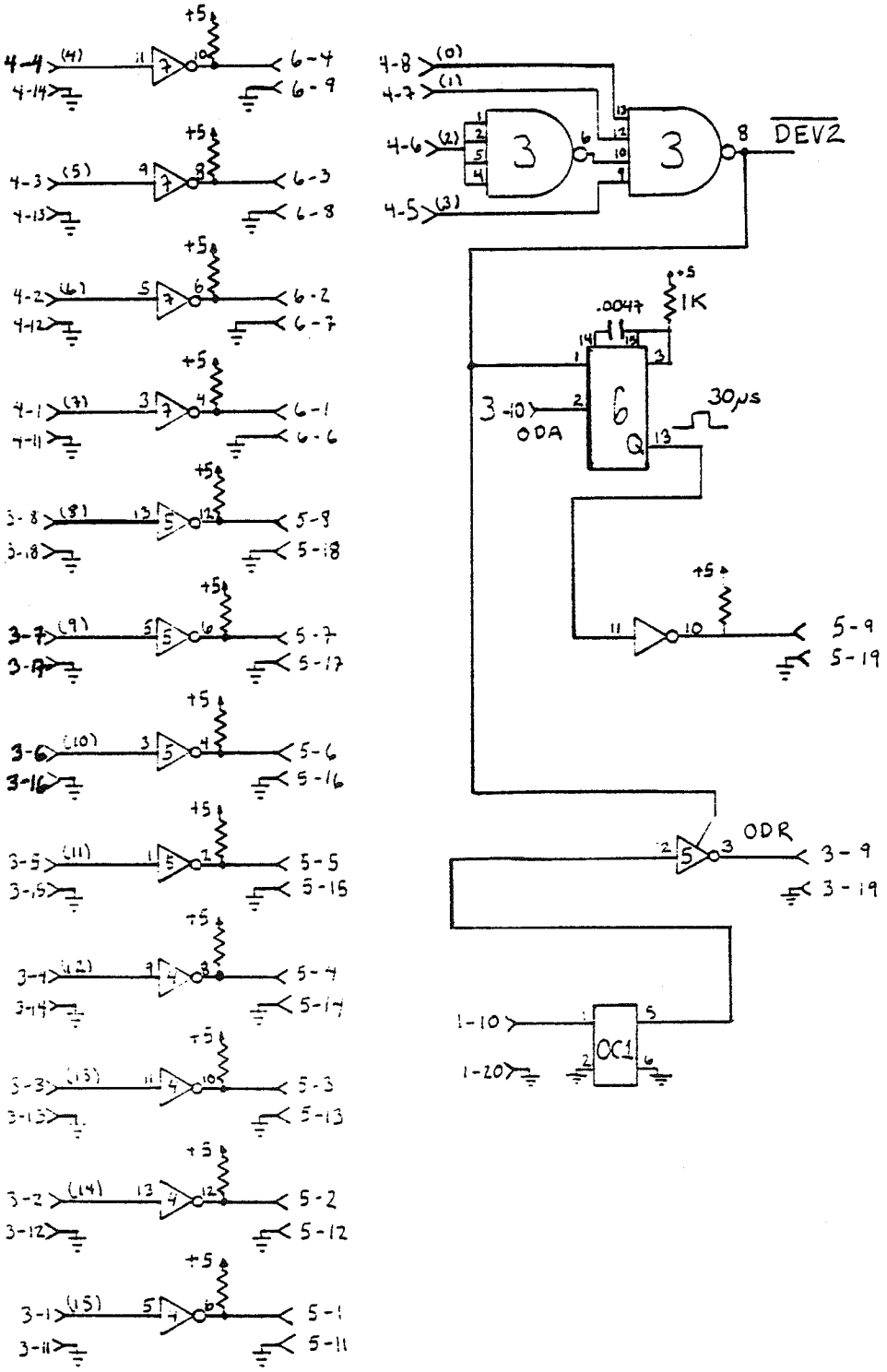


Figure 7 Schematic of the outputs from the 7/32

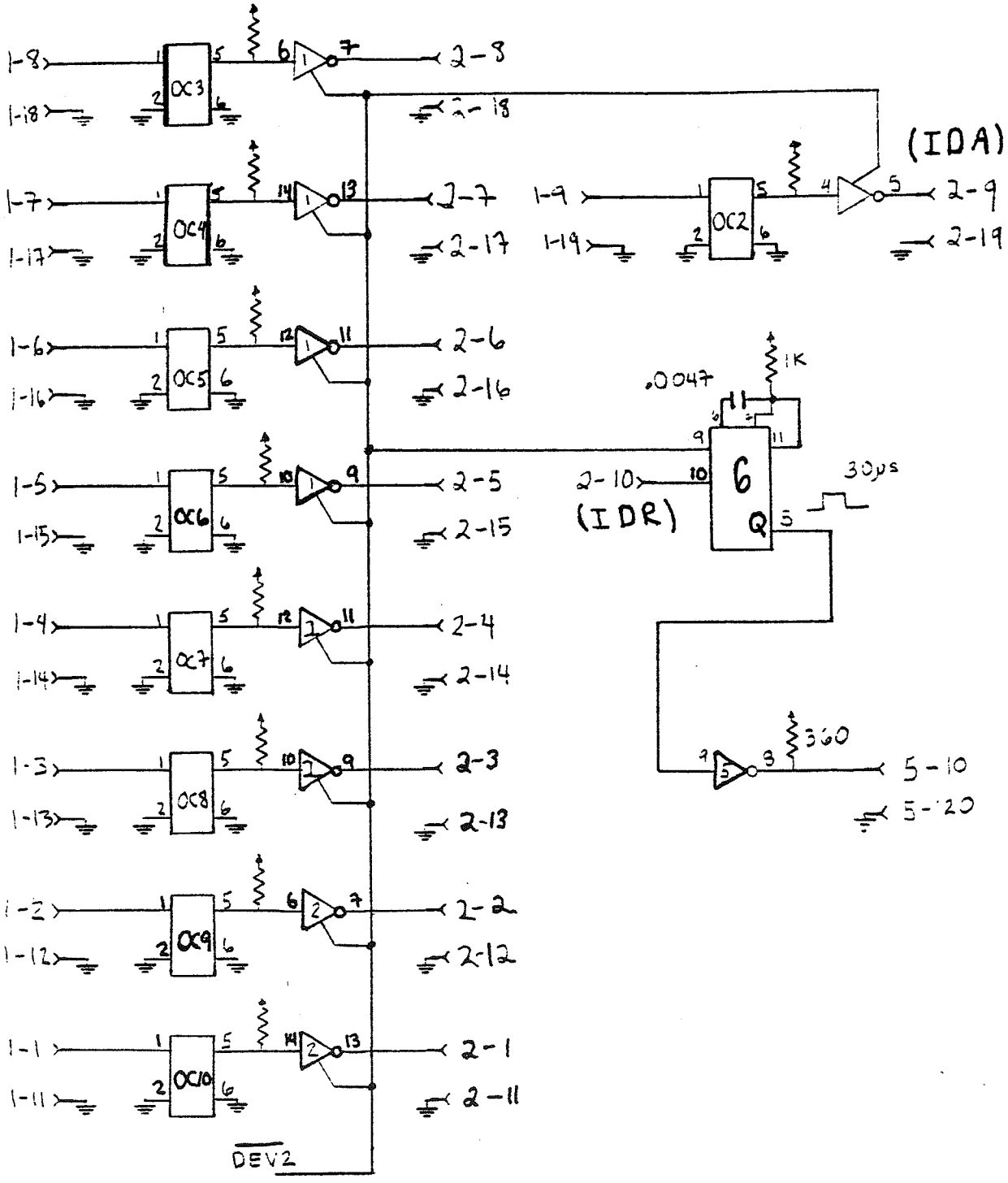


Figure 8 Schematic of the inputs to the 7/32.

actual transfer of data to the input port. In addition, LED2 is driven by IC7 to indicate to users of the 7/32 that this circuitry is active. Bits 4-15 of the digital output port are buffered by the open-collector inverters of IC4, IC5 and IC7 and appear at pins 1-4 of connector 6 (bits 4-7) and pins 1-8 of connector 5 (bits 8-15).

The one microsecond pulse ODA (OUTPUT DATA AVAILABLE) (connector 3, pin 10) which appears whenever a sixteen bit halfword is written by the processor to the output port, is stretched by the one-shot IC6a. This is necessary because the transmission of data over the lines terminated at connectors 5-6 occurs at approximately 16 KHz. A one microsecond pulse cannot be reliably transmitted over a thirty meter path at this rate; a widened pulse works quite well.

The streched pulse is buffered by IC5 and appears at pin 10 of connector 5. Note that the stretched pulse occurs only if Device 2 is selected in the above-mentioned manner. At the other end of the lines terminated by connector 5, opto-isolators (Monsanto MCT-2) assure electrical isolation between the DIO and the remote interface (see ref. 9).

The input lines to the selection/isolation circuitry are also terminated by opto-isolators; hence there is complete isolation between the 7/32 and the remote interface (see

Figure 8). The signals appearing at connector 1 are buffered by opto-isolators OC3-OC10 and converted to TTL levels. These levels appear at the inputs to IC1 and IC2. IC1 and IC2 are in the active state (as opposed to the high-impedance "Z" state) only when a logical zero is present at both pins 1 and 15 of the respective IC. As mentioned, this low level (DEV2) is provided when Device 2 is selected by bits 0-3. When enabled by DEV2, the signals from pins 1-8 of connector 1 appear at pins 1-8 of connector 2, which is connected to the DIO input. When not enabled, these tri-state inverters do not affect the use of the DIO by other users.

The actual input of data to the DIO is controlled by the handshake IDA (INPUT DATA AVAILABLE), which is found at pin 9 of connector 2. When this line is strobed, the status of the DIO input port is changed to "not busy" (BUSY = 0), indicating to the processor that data is available. Whenever a 16-bit halfword is read from the DIO by the processor, a one microsecond pulse appears at the IDR (INPUT DATA RECIEVED) line of the input (pin 10 of connector 2). This pulse is stretched by IC6b to thirty microseconds, for the same reasons discussed above for ODA.

The input line appearing at pin 10 of connector 1 is isolated by OC1 , buffered by IC1 and appears at pin 10 of connector 3. This pin is connected to the ODR (OUTPUT DATA RECIEVED) line of the output port, and indicates to the

processor that data previously written to the output port has been recieved. Again, the tri-state inverter IC1, when not enabled, does not affect the use of the DIO by other 7/32 users.

Remote Equipment Interface

The equipment interface at the remote location is discussed in detail elsewhere (ref. 9). It is capable, under program control, of transducer placement in three dimensions, and determining the placement of the thermocouple with respect to the thermocouple. In addition, the setting of a variable capacitor used to control the voltage across the transducer (and thus the intensity produced by the transducer) can be accomplished, as well as the remote switching of the r. f. to the transducer, for programmable amounts of time. The output of position encoders for the placement controllers and variable capacitor can be read remotely via this interface, as can the status of a down counter used to control all of the above-mentioned features of the remote interface. Thus, the programs used for data collection have complete control of experimental equipment. The programs necessary to interact with this equipment (ATD, READMS) are found in Appendix I.

Chapter 3. Software

This chapter deals with the programming of the Perkin-Elmer 7/32 minicomputer which is used to control the experiment. This software can be divided into units covering three basic areas: 1) real-time interaction with the experimental system; 2) calculation and outputting of results and 3) interaction with the operator. These areas will be treated independently, though it should be noted that there exists at all times links between various program units.

The heart of the data collection is a routine which digitizes the output of the thermocouple amplifier during the application of ultrasound to the specimen being examined. Voltage data in the range 0 to +10.24v are converted to binary data in the range 0 to 1023. Digitization occurs at a crystal-controlled frequency of 16.384 kHz; since the frequencies involved are much lower than this, not all data points are stored. At present, one of every two samples is stored, yielding 8192 samples for a one-second application of ultrasound, the standard duration for this application. Of the 8192 samples stored, every fifteenth sample is used for calculations, the samples being taken beginning at t=0.25s. This yields approximately 567 samples which are fit with a polynomial from which the rate of temperature rise can be computed.

The initial temperature rise recorded by the thermocouple is dominated by the contribution of viscous heating at the wire surface due to relative motion between the thermocouple and the sample. Goss et al (5) have shown that for junction sizes smaller than 13 μm, the slope of the temperature vs time curve at 0.5 seconds can be used for $(dT/dt)_0$. This time represents a compromise between negating the effects of the viscous interaction with the surrounding medium and minimizing the effects of diffusion, which increase with time.

Thus, (dT/dt) as described in Chapter one is determined at t=0.5 seconds. Experience has shown that if data points are included from the portion of data corresponding to the initial (viscous) temperature rise, the least square fit to the data in the region centered at t=0.5 seconds is affected adversely. This portion generally can be seen to approach equilibrium beginning at approximately 0.1 - 0.2 seconds, and 0.25 seconds was chosen as the beginning of the fit to minimize the effect of the viscous heating. Samples are read by the A/D for a short time past the end of the application of r. f. to the transducer; however, only those corresponding to the time when the r. f. is on are included (i. e., normally to t = 1 second).

The data points, together with their associated time values are stored in arrays which are operated on by a

least-squares fitting program. This program, LEAST (see Appendix I) allows for the selection of the degree of the polynomial used in the fit; experience has shown a fit of degree three to work well. After the fit has been computed, a companion routine, EVAL (see Appendix I) is called to evaluate the fit at arbitrary points. Presently, the fit is output at eleven points (0 to 1.0 seconds at 0.1 second intervals). In addition, a modification to the evaluation routine (EVAL) yields the first derivative of the fit for these points. The first derivative corresponds to dT/dt, and when evaluated at 0.5 seconds is used as the $(dT/dt)_o$ of equation 1 to compute the absorption coefficient. The following sections describe how the operator interacts with the program, and what "utilities" are available to the operator.

The program is loaded by typing ABSORB at the system console of the 7/32. The cathode-ray terminal used by the operator is cleared and written with program information. A typical screen pattern is shown in Figure 9. The date and time are obtained via system software, and automatically updated as changes occur. At the bottom of the screen, a prompt will be issued for a command to be input. To execute a command, the key corresponding to the one-letter abbreviation of the command is pressed. After execution of a given command, the prompt is again issued, and the process continues until the program is cancelled from the system

console. A list of commands is given in Figure 10. An upper-case character enclosed in brackets (e.g., {A}) indicates that to execute the sequence described, that character (in this case, "A") would be pressed. For clarity, the commands are grouped as to their function in the discussion below.

```
21:20:17                DATA FOR SHOT   1 FOLLOWS

ABSORPTION PROGRAM **       TODAY IS    10/29/80          TIME 21:20:17

2) TISSUE TYPE:  CAL # 41, D. C. 710    3) PREPARATION:  TAP WATER, 37  DEG C
                                        4) TRANSDUCER:   1 MHZ FOCUSSED
1) OPERATOR:  DAVE DUBACK               5) TCOUPLE: CHROMEL-CONSTANTAN

17) RCPK   1.724          12)TSENS     60.000      14) TDIAM    3.000

11) DEGREE:     8         22) TDEPTH    0.000      21) INTENSITY: 2.166
9) # OF PTS: 9000         16) ATTEN:    0.000          ISUBS:     2.166
10) # USED:   567
                          6) VCAL:    452.700      23) CS SET:  200.000
                          7) ICAL:      5.000          V:       284.000
                          RANGE:        2.000
SHOT #      1                                      18) GAIN:     11.000
                #SHOTS IN BLOCK:      5            19)FILTER:   100.000
ALPHA: 0.4964368E-01  TAKEN:          0            CSCS         994.000

29)INPUT EPS: -1.0000     20) LENGTH:   1.000      24) X:        -0.031
RMS ERROR:     0.0519                              25) Y:         0.059
                                                   26) Z:        -0.026
                                                   28) ICODE:      22
```

Figure 9. Example of terminal screen output.

CONTROL OF EXPERIMENTAL EQUIPMENT

{T} "Take"

The ultrasound is turned on for the length of time given by the program variable LENGTH. Voltage data are recorded by the A/D and stored in memory.

{D} "Display"

Same as {T}, but the information recorded is considered "scratch". This allows the operator to setup equipment without affecting the state of statistical recording.

{A} "Average"

Same as {T}, but the data is added, point for point, to that previously stored. This allows averaging of low-level signals.

{B} "Baseline"

Data is recorded for one second and a least squares fit of degree one calculated. The slope of the fit indicates the amount of temperature drift in the experimental setup. For future, totally automated experiments, this drift will be compared to an arbitrary criterion to determine whether ensuing data is valid (large drifts preclude accurate measurement) .

{U} "Update"

A voltage value is entered by the operator, and the output of the digital-to-analog converter channel connected to the thermocouple amplifier is adjusted to this voltage.

Figure 10 - Summary of operator commands

CALCULATIONS

{L} "Least"

The A/D data stored in memory is converted to REAL format and a least squares fit is performed for the degree indicated by DEGREE.

{K} "Krunch"

The A/D data stored using {A} ("Average") is operated upon, as for {L}, but the results are divided by the number of thermocouple responses averaged.

OUTPUT OF CONTROL INFORMATION AND DATA

{I} "Information"

The screen is cleared and written with the current status of the variables which are under operator control.

{O} "Output data"

The data corresponding to the last least-squares fit is output to the screen.

{S} "Summarize"

A statistical summary of experimental activities is output (not yet fully programmed).

{W} "Write"

The data currently displayed on the screen (via {I}, {O} or {S}) is output on the line printer with a heading indicating date, time and number of responses processed thus far.

Figure 10 -continued.

{P} "Plot"

The data currently displayed on the oscilloscope is plotted on the line printer. This data may be any or all of the following: Actual A/D data, fitted data, the instantaneous slope of the fitted data, or the slope of the fitted data at 0.5 seconds (displayed as a tangent).

{V} "Volume"

The A/D data stored in memory is saved on magnetic tape, with a header giving the current state of control information.

{H} "Help"

A summary of the commands available and their effect is written to the screen.

{C} "Comment"

Lines entered by the operator are written to the printer. This serves as a log of observations made by the operator.

{R} "Read"

A prompt is issued for a parameter number (1-29). After the operator enters the number, the paramter corresponding to that number is read from the terminal. A prompt is issued prior to the read, explaining what, if any, units are to be assumed.

{G} "Get"

All variables under operator control are read in, without the operator entering individual paramter numbers as for {R}.

Figure 10 - continued

Figure 11 gives a list of variables under operator control. To the left of each variable is a parameter number which appears on the terminal screen (see also Figure 9) Access to a given variable via command {R} is gained when this parameter number is entered as a response to the prompt which follows {R}. A brief explanation of each variables' purpose is given; for more detailed information, Appendix I should be consulted.

(1)NAME — The name of the operator...

(2)SPECIMEN — Specimen under study...

(3)PREPARATION — Any special preparation (e. g., degassed saline solution.

(4)TRANSDUCER — Transducer used to deliver the ultrasound

(5)TCOUPLE — Thermocouple type used (e. g., Chromel-Constantan, etc.)

(6)VCAL [volts] — A calibrating voltage associated with a given transducer which is used together with ICAL (below) to determine the ultrasonic intensity applied for a given voltage across the transducer . This voltage is known for a given setting of the variable capacitor (CS).

(7)ICAL [W/cm**2] — The intensity at which VCAL was calculated. The intensity produced by the transducer for a given voltage across it is given by the formula

$$I = \left(\frac{V}{VCAL}\right)^2 \times ICAL$$

where I is the intensity produced, V is the voltage applied to the transducer and VCAL and ICAL are the calibrating voltage and intensity, respectively.

(8)RANGE — Range of the variable capacitor CS. Range 2 is used for low intensity measurements and range 3 for high intensity measurements.

(9),(10) — - no longer used...

(11)DEGREE — The degree of the polynomial used to fit stored data. Usually 3 but variable from 0 to 13.

(12)TSENS — The sensitivity of the thermocouple used, in microvolts per degree Celsius.

Figure 11 - Variables under operator control.

(13) - no longer used...

(14)TDIAM [cm] Diameter of the thermocouple junction. Not used for calculations but important for the record (see the above discussion on viscous effects).

(15) - no longer used...

(16)ATTEN Attenuation of the specimen in Nepers/cm. It is used to determine the site intensity according to the equation

$$I_s = I_o e^{-2\alpha x}$$

where I is the site intensity, I_o is the intensity calculated from VCAL and ICAL above, α is the attenuation coefficient and x is the depth of the thermocouple junction in the surrounding medium (see (22)TDEPTH).

(17)RCPK The product of the C in equation 1 and a constant used to ensure proper units in calculations. [J/cm**3- ^{o}C]

(18)GAIN The gain of the second stage of amplification. Since the first stage has fixed gain, this is taken into account in the calculations.

(19)FILTER The cutoff frequency of hte filter used to limit noise from the thermocouple amplifier.

(20)LENGTH The amount of time, in seconds, for which the ultrasound is to be applied and data taken via the thermocouple amplifier.

(21)INTENSITY The intensity, in Watts/cm**2, desired from the transducer. In future programming, CS will be adjusted to produce an appropriate voltage across the transducer. At present, CS is calculated and displayed so that the operator can manually set the intensity. (22)TDEPTH The depth of the thermocouple junction (cm) in the surrounding medium (see above discussion for (16)ATTEN).

Figure 11 - continued.

(23)CS SET — The setting of the variable capacitor (CS) which determines the voltage across the transducer. CS SET is calculated whenever the operator changes the desired intensity; it may also be entered as a variable, in which case the intensity is calculated using the voltage produced for the chosen CS setting. The voltage across the transducer is a linear function of the CS setting; it is periodically calibrated and thus no constants will be given here.

(24)X
(25)Y
(26)Z — These variables will control the position of the transducer relative to the thermocouple junction in future programming. At present, the current position is recorded via the remote interface, but the operator must manually move the transducer.

(27)NSINBK — The number of signals to be averaged in low intensity measurements. The averaging program will require large buffers in memory, and is not yet fully programmed...

(28)ICODE — This variable sets the displays which will appear on the oscilloscope. It is a two-digit number which is the sum of te codes noted below for each desired display:

1) Slope vs. time of the fit
2) Fit vs. time
4) A/D data vs. time
8) Test pattern
16) Tangent at t=0.5

For example, an ICODE of 22 represents the simultaneous outputting of a digitized response, the fit to the digital data, and the tangent at 0.5 seocnds, which would be used in determining the absorption coefficient.

(29)EPS — This is the desired rms error for the fit, and should always be set negative to force the code for LEAST to use the degree chosen by the operator.

Figure 11 - continued.

CHAPTER 4

RESULTS AND SUMMARY

The system as described has been constructed and tested. Preliminary absorption data however, were obtained using an earlier design which differed in that it relied on a filter which consisted of discrete operational amplifiers, rather than the filter described in chapter 2. These results show good agreement with the previous system employing a galvanometer to record temperature data.

Measurements of the ultrasonic absorption coefficient were made for bovine liver and Dow Corning 710 Silicone Fluid using the system as described, except for the filter section, which at the time was implemented using discrete operational amplifiers in a classic low-pass filter configuration. At the same time that the new system was used, the previous system employing a galvanometer to record the thermocouple response was also used, for purposes of comparison. The results of these measurements are given in Table 3.

TABLE 3 - Comparison of absorption coefficient (Np/cm) obtained using two systems.

Material	Previous System			New System		
D.C. 710	0.0525	0.0087	(6)	0.0499	0.0042	(19)
Beef Liver	0.0234	0.0020	(8)	0.0230	0.0033	(19)

The reuslts are given as mean standard deviation. The number in parentheses following each result indicates the the number of measurements made. Measurements were made at 1 MHz and 37 C using degassed distilled water as a coupling medium. at 1 MHz and 37 ^{o}C. For the Dow Corning 710 Silicone Fluid, the means of a series of measurements with each system agree to within 6%. The results for both systems are higher than expected, because viscous heating between the wire and the surrounding medium is very large for this viscous oil, and because streaming can occur in the liquid (see ref. 5). The means of the results for beef liver agree to within 1.7%; the value of the absorption coefficient obtained for the galvanometer-based measurements was in perfect agreement with that reported by Goss et al (ref. 10), with the value obtained by the new system slightly lower. Thus the new system gives results in excellent agreement with those reported using measurement techniques employed previously.

The results of the automated measurements appear within seconds of the application of ultrasound (see Figure 12 for an example of numerical output and Figure 13 for an example of plotted data) and therefore serve to point to possible anomalies during the course of an experiment. Among typical anamolies observed are isolated measurements which fall several standard deviations away from the mean of all measurements for a given session, with the remainder of the

data points showing little variance. Unfortunately, further measurements were not possible due to a breakdown in the r. f. equipment used to drive the transducer. It is expected that new tests will continue to confirm the good performance of the automated system.

The results of calibration of the system components are given in Table 4 (DC gains of all stages) and Figure 14 (frequency response of filter). It should be noted that equivalent calibrations were made for the system as used for the above-mentioned measurements, but since the system with the discrete op-amp filter has been dismantled, the calibration results are not included. Indeed, the superposition of the two frequency response curves upon one another showed little discernible difference.

Table 4. DC Gains of system components

Amplifier	Second Stage	Filter
10013.2	3.001	1.03
	6.000	
	10.997	
	20.998	

Remember that only the second stage has variable gain; the theoretical gains are 10001 for the amplifier, 3, 6, 11 and 21 for the second stage, and 1.00 for the filter.

In summary, this system should prove very useful in future determinations of the ultrasonic absorption coefficient. It provides a fast, accurate means of temperature data aquisition and processing, and the software included gives the experimentor a number of features which help to follow the progress of the experiment in a "real-time" sense.

21:20:33 DATA FOR SHOT 1 FOLLOWS

VOLTAGE	TEMP	DEG/SEC	ALPHA	TIME
0.371977	0.563602E-01	17.0198	1.02620	0.00
1.19920	0.181697	3.28561	0.198105	0.10
1.40713	0.213202	1.58076	0.953113E-01	0.20
1.55277	0.235874	1.38840	0.837132E-01	0.30
1.67682	0.254063	1.0085	0.68101E-01	0.40
1.76573	0.267534	0.823352	0.496437E-01	0.50
1.84840	0.280060	0.823352	0.501205E-01	0.60
1.92650	0.291893	0.694281	0.418614E-01	0.70
1.98609	0.300923	0.542563	0.32137E-01	0.80
2.05038	0.310663	0.780585	0.47065E-01	0.90
2.08999	0.316665	-0.840821	-0.519029E-01	1.00

Figure 12 Example of Numerical Output

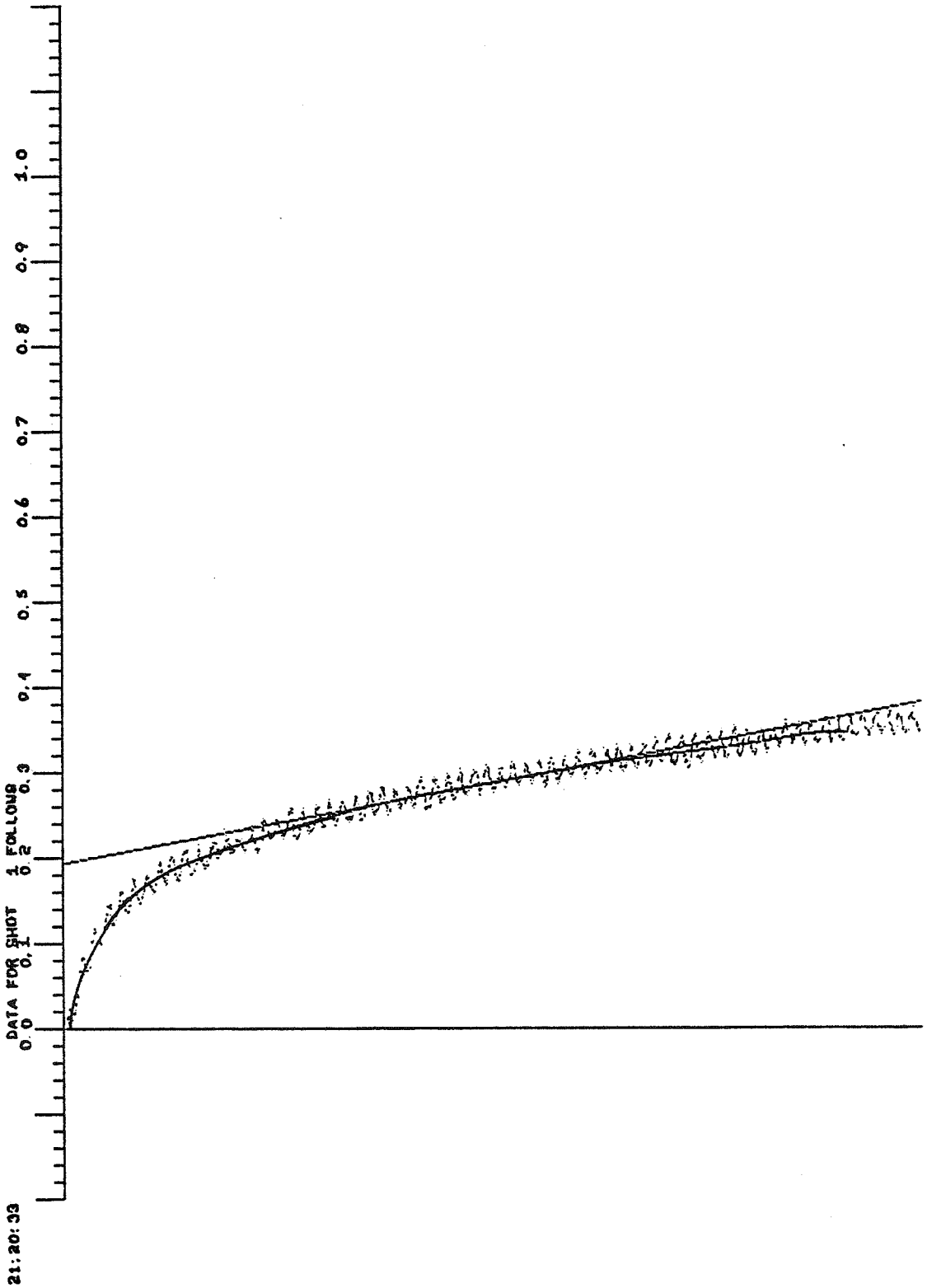


Figure 13 Example of plotted output.

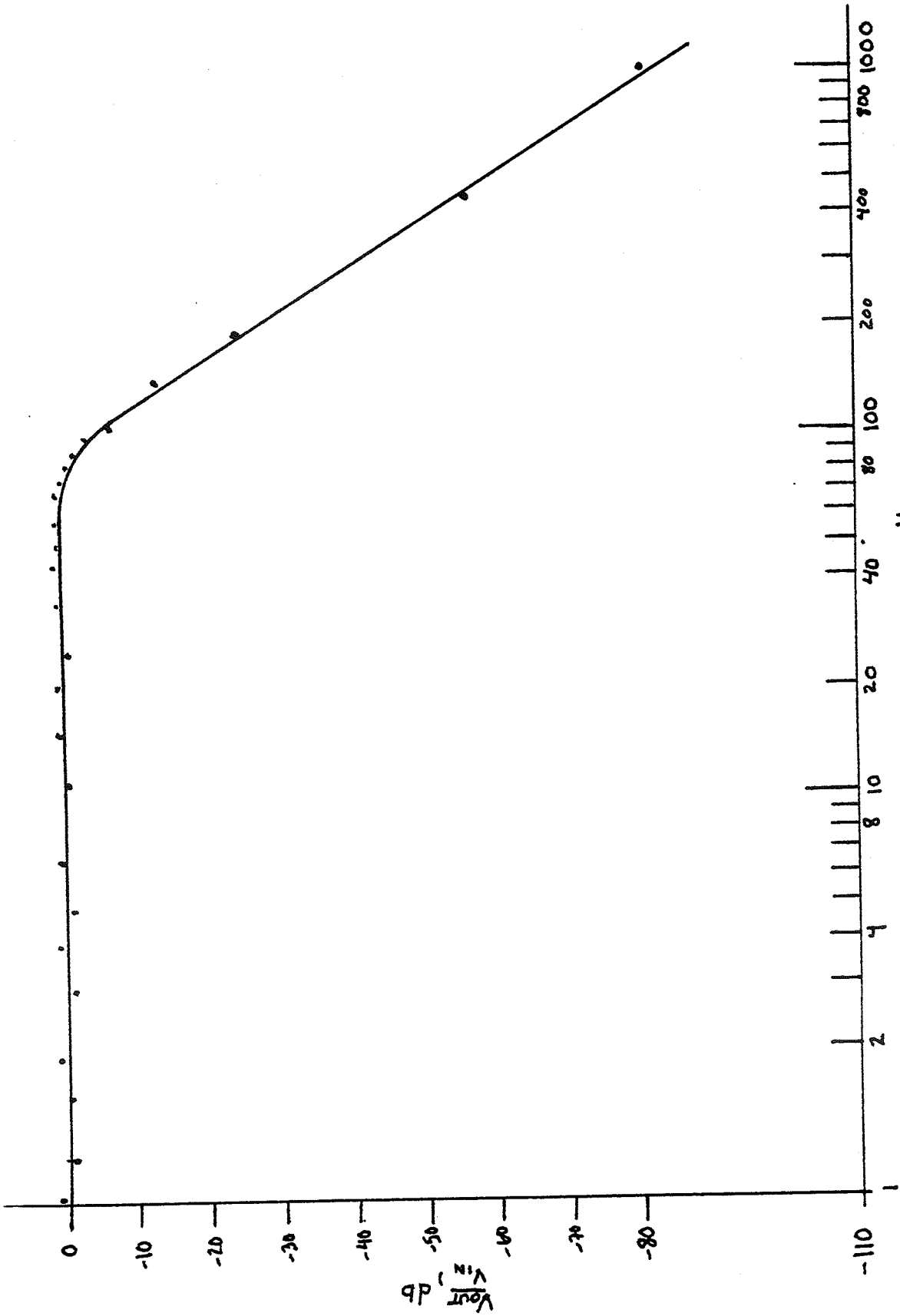


Figure 14 Frequency response of filter.

CHAPTER 5

SUGGESTIONS FOR FUTURE IMPROVEMENT

The system as desribed needs little added to it in terms of hardware. The only addition necessary to the thermocouple amplifier would be programmability via the remote interface, to allow measurements under a variety of conditions without operator intervention. There is some question in the author's mind as to the desiribilty of this, as the operator must be present anyway, and the addition of circuitry to effect remote gain switching must be placed in a highly sensitive feedback loop in the second stage; it is perhaps wisest to leave gain switching a manual operation and avoid the introduction of noise and/or gain errors.

Reference 8 gives a description of a stabilizing circuit used in connection with an r. f. amplifier to provide the transducer with a driving voltage. In practice, this stabilizing circuit must be "fine-tuned" by manually setting an attenuator in the amplifier input circuit. The attenuator is part of a sophisticated signal generator which has the capability of being remotely programmed. A modification to the remote interface to allow program control of this attenuator is necessary before completely automated experiments can be carried out, since the attenuator must be

re-tuned for each change in intensity desired.

The software described is relatively complete in terms of carrying out the functions necessary for data aquisition and processing. A completely automated system, however, requires additional programming. This programming would take the form of a routine which would accept a description of an experiment, and decode this description into a series of subroutine calls. This is exactly what the main program does at present, the only difference being that the present program accepts only "one-step" experiments!

The 7/32 assembly-language instruction set includes a series of list processing features which are quite useful in the maintenence of "stacks"; a FIFO (first-in-first-out) list of subroutines generated by the decoding program could be accessed sequentially by a program almost exactly like that of the present "main" program, the difference being that the new program would accept its commands from the FIFO list instead of directly from the operator (via the terminal).

In addition, the recording on magnetic tape of experimental results needs to be written in such a fashion as to allow easy access via searching and sorting routines, to facilitate analysis based on models different from the current one (see ref. 5)

REFERENCES

1. W. D. O'Brien, Jr., "Ultrasonic Dosimetry," in Ultrasound: Its Application in Medicine and Biology, ed. F. J. Fry, pp 343-391, Elsevier Scientific Pub. Co., NY, (1978).

2. R. L. Johnston, S. A. Goss, V. Maynard, J. K. Brady, L. A. Frizzell, W. D. O'Brien, Jr. and F. Dunn, "Elements of Tissue Characterization. Part I. Ultrasonic Propagation Properties," in Ultrasonic Tissue Characterization II, NBS Special Publication 525, pp 19-27, U. S. Government Printing Office, Washington, D. C., (1979).

3. W. J. Fry and R. B. Fry, "Determination of Absolute Sound Levels and Acoustic Absorption Coefficients by Thermocouple Probes - Theory," J. Acoust. Soc. Am. 26, 294-310 (1954)

4. W. J. Fry and R. B. Fry, "Determination of Absolute Sound Levels and Acoustic Absorption Coefficients by Thermocouple Probes--Experiment," J. Acoust. Soc. Am. 26, 311-317 (1954).

5. S. A. Goss, J. W. Cobb, and L. A. Frizzell, "Effect of Beam Width and Thermocouple Size on the Measurement of Ultrasonic Absorption Using the Transient Thermoelectric Technique," 1977 Ultasonics Sysmposium Proceedings, IEEE Cat. No. 77CH 1264-1SU, 206-211. IEEE, New York, N. Y. (1977).

6. S. A. Goss, R. L. Johnston and F. Dunn, "Comprehensive Compilation of Empirical Ultrasonic Properties of Mammalian Tissues," J. Acoust. Soc. Am. 64, 423-457 (1978).

7. S. A. Goss, R. L. Johnston and F. Dunn, "Compilation of Empirical Ultasonic Properties of Mammalian Tissues. II," J. Acoust. Soc. Am. 68, 93-108 (1980).

10. D. Lancaster, Active Filter Cookbook, Howard W. Sams & Co., Indianapolis, pp. 43-76 (1978)

9. Vaughn, S. "Automation of Ultrasound Dosimetry Experiment" M. S. Thesis in Electrical Engineering, University of Illinois (1980).

10. S. A. Goss, L. A. Frizzell and F. Dunn, "Ultrasonic Absorption and Attenuation in Mammalian Tissues," Ultrasound Med. Biol. (in press).

APPENDIX A

PROGRAM LISTINGS AND DESCRIPTIONS

This Appendix contains a description of the programs used to obtain and process temperature data, together with those which provide the operator control of an experiment and those which are used to report results to the operator. The programs are divided into the following categories (program names in parentheses): 1) Main program and entry of system control variables (MAIN and GETPAR) 2) Interaction with the remote interface, analog-to-digital converter and digital-to-analog converter (READMS, ATD, DTASWP and DTA) 3) Processing of temperature data. (LEAST, EVAL, GETDAT, CALC, BASELN and AVG) 4) Display of control information and results (REFR, BLANK, CLEAR, IBELL, PRNTSC, PLTDOT, TOF, HASHMK AND PLTSHT).

A listing of the main program is found in Figure 15. Its purpose is to provide the operator access to all of the utilities described above. All are accessible as seperate functions; several have been lumped together to form sequences of functions which can be executed through the use of a single command.

A list of the commands and their associated effects was given in Figure 9. The operation of the main program is described in this section.

When the program is started, the tables used by subroutines REFR and GETPAR, namely, CMDT, HEAD, WCUR and APTR are read from disc files. The logical unit assignments for these files, together with their format and content description, is included in the listing for the main program. The integer equivalents of the real-valued array INFO are set equal to their counterparts, and the screen is written with the current data and information. From this point forward, the following sequence is repeated, until a request to end execution is issued:

Subroutine WONKB is called. This initiates a read from the operator terminal. Program execution, however, continues. The oscilloscope is driven by DTASWP to display the information which the operator has chosen through the use of variable 29 (ICODE). Every few seconds, the screen is

```
C
C
      INTEGER CLABEL(26),LABEL(26)
      INTEGER SHOOT,NOSHOT,CHAN2
      INTEGER NN(15)
      INTEGER DCHAN,POSTWT
      INTEGER STACK(100),MCMD(8,8),LINE(20)
      INTEGER CRT,PR,OUT,IHELP
      INTEGER RGSTG(16)
C
      INTEGER*2 CMMD
C
C
      REAL SVCS(100),SVINT(100),SVALPH(100)
      REAL CSV(15),MALPHA(15),SIGMA(15)
C
C
C
C/COMST
C
      REAL LENGTH,INTENS,ICAL,INFO(40),ISUBS,NPTS,NUSED
C
      INTEGER HEAD(40,6),AINFO(7,6),APTR(40),STATUS
C
      INTEGER*2 WCUR(40,4),CMDT(40,2)
C
      COMMON/REFRSH/HEAD(40,6),WCUR(40,4),AINFO(7,6),APTR(40),INFO(40),
     1  STATUS,CMDT(40,2)
C
C
      INTEGER*2 ADBUF(9000)
C
      COMMON/ATD/ ADBUF(9000)
C
C
      INTEGER PBLK(5),PBLK1(5),LU,CRTYPE
C
      INTEGER*2 CMD(40)
C
      COMMON /KEYBD/ PBLK(5),PBLK1(5),CMD(40),LU,CRTYPE
C
      COMMON /READMS/ IX,IY,IZ,ICS,IDAT,ISTAT,IDC
C
C
      LOGICAL TSTOUT,CROUT,SLPOUT,TANOUT,CALCD
      LOGICAL SHTOUT,INFOUT,DATOUT,SUMOUT,MAKCHK,AVDONE,CRNCHD
C
      COMMON /CNTRL/ TSTOUT,CROUT,SHTOUT,SLPOUT,TANOUT,CALCD,
     1   INFOUT,DATOUT,SUMOUT,MAKCHK,AVDONE,CRNCHD
C
C
C
C
      REAL TX(600),ADF(600),ARRAY(1245)
      REAL TF(11),TT(11),TS(11),ALPH(11)
      REAL DTF(11),DTS(11),DTR(11)
      REAL MEAN,MSLOPE
      DOUBLE PRECISION R(600)
C
      COMMON/CRUNCH/ TX(600),ADF(600),ARRAY(1245),
     1               TF(11),TT(11),TS(11),ALPH(11),
     2               DTF(11),DTS(11),DTR(11),R(600),
     3               MEAN,MSLOPE
C
C
C
      INTEGER*2 ISBUF(600),ICRBUF(600),ISLBUF(600)
      INTEGER*2 TANBUF(600)
      COMMON /SCOP/ ISBUF(600),ICRBUF(600),ISLBUF(600),TANBUF(600)
C
      INTEGER SHEND,ASHEND
C
      COMMON /NINFO/ NRNGE,NNPTS,NNUSED,IDEG,MAXDEG,NDESRD,NSINBK,
     1    SHEND,ASHEND,ICODE
C
C
      EQUIVALENCE (INFO(6),VCAL),(INFO(7),ICAL),(INFO(8),TSENS),
     1 (INFO(9),TDIAM),(INFO(10),ATTEN),(INFO(11),RCPK),(INFO(12),GAIN),
     2 (INFO(13),FILTER),(INFO(14),LENGTH),(INFO(38),INTENS),
     3 (INFO(16),X),(INFO(17),Y),(INFO(18),Z),(INFO(20),SHOT),
     4 (INFO(25),ALPHA),(INFO(26),NPTS),(INFO(27),NUSED),
     5 (INFO(28),DEG),(INFO(29),EPS),(INFO(34),TDEPTH),(INFO(35),V),
     6 (INFO(36),CS),(INFO(37),ISUBS),(INFO(39),RNGE)
     7 ,(INFO(40),DESRD),(INFO(22),RMSERR)
     8  ,(INFO(15),SINBK),(INFO(19),CODE),(INFO(21),CSCS)
C/COMEND
      DATA SHOOT/1/,NOSHOT/0/
      DATA CRT/3/,PR/10/,OUT/3/,IHELP/12/
      DATA CHAN2/2/
      DATA NINTNS/15/
      DATA MAG1/7/
```

Figure 15 MAIN

```
C
 120  FORMAT(2I2)
 121  FORMAT(6A4,T32,4I2,T50,I2)
 122  FORMAT(F8.3)
 123  FORMAT(I6)
 125  FORMAT('ENTER COMMENT- RETURN TO END COMMENT')
 126  FORMAT(20A4)
 129  FORMAT(I1)
 130  FORMAT(8A2)
 132  FORMAT('ENTER TWO DIGIT PARAMETER #; RETURN FOR COMMAND')
 133  FORMAT(T10,'N',T25,'CS',T35,'INTENSITY',
     1    T55,'ALPHA')
 134  FORMAT(T9,I3,T21,F8.3,T31,F8.3,T51,F8.5)
C
 135  FORMAT('ILLEGAL COMMMAND!!!!!!')
      FIRST=LAST+1
      ASSIGN 801 TO CLABEL(1)
      ASSIGN 802 TO CLABEL(2)
      ASSIGN 803 TO CLABEL(3)
      ASSIGN 804 TO CLABEL(4)
      ASSIGN 805 TO CLABEL(5)
      ASSIGN 806 TO CLABEL(6)
      ASSIGN 807 TO CLABEL(7)
      ASSIGN 808 TO CLABEL(8)
      ASSIGN 809 TO CLABEL(9)
      ASSIGN 810 TO CLABEL(10)
      ASSIGN 811 TO CLABEL(11)
      ASSIGN 812 TO CLABEL(12)
      ASSIGN 813 TO CLABEL(13)
      ASSIGN 814 TO CLABEL(14)
      ASSIGN 815 TO CLABEL(15)
      ASSIGN 816 TO CLABEL(16)
      ASSIGN 817 TO CLABEL(17)
      ASSIGN 818 TO CLABEL(18)
      ASSIGN 819 TO CLABEL(19)
      ASSIGN 820 TO CLABEL(20)
      ASSIGN 821 TO CLABEL(21)
      ASSIGN 822 TO CLABEL(22)
      ASSIGN 823 TO CLABEL(23)
      ASSIGN 824 TO CLABEL(24)
      ASSIGN 825 TO CLABEL(25)
      ASSIGN 826 TO CLABEL(26)
      ASSIGN 20 TO NEXT
      READ(9,120) ((CMDT(I,J),J=1,2),I=1,40)
      READ(9,121) ((HEAD(I,J),J=1,6),(WCUR(I,K),K=1,4),APTR(I),I=1,40)
      READ (9,122) (INFO(I),I=1,40)
      NRNGE=RNGE
      NNPTS=NPTS
      NNUSED=NUSED
      NDESRD=DESRD
      READ(9,123)DCHAN,POSTWT,NPARS
      CODE=8.0
      ICODE=CODE
      MAXDEG=DEG
      RK100=1023.0/10.2375
      ISTATE=1
      CALL CLEAR
      CALL REFR(1,40)
      INFOUT=.TRUE.
      SUMOUT=.FALSE.
      DATOUT=.FALSE.
      CRNCHD=.FALSE.
      CALCD=.FALSE.
      AVDONE=.FALSE.
      MAKCHK=.FALSE.
      SHEND=5000
      ASHEND=5000
C
      DO 5 I=1,7
      DO 5 J=1,6
      AINFO(I,J)=Y'20202020'
 5    CONTINUE
C
      DO 6 I=1,600
      W(I)=1.0
 6    CONTINUE
C
      GOTO 20
C
```

Figure 15- continued

```
 17    CONTINUE
       CALL BLANK(22,23)
       CMMD=CMD(1)/256
       IF(CMMD.EQ.X'20')GOTO 20
       IF(CMMD.EQ.X'0A')GOTO 20
       IF(CMMD.EQ.X'0D')GOTO 20
       IF(CMMD.GT.X'40'.AND.CMMD.LE.X'5A')GOTO 2
       CALL BLANK(19,19)
       WRITE(CRT,135)
       GOTO 20
 2     ICMD=CMMD-X'40'
       GOTO CLABEL(ICMD)
C
 20    CONTINUE
       CALL CALC(20)
       CALL WONKB
 21    CONTINUE
       DO 211 I=1,300
       IF(TSTOUT.OR.SHTOUT)CALL DTASWP(DCHAN,POSTWT,ISBUF,600)
       IF(TSTOUT.OR.CROUT)CALL DTASWP(DCHAN,POSTWT,ICRBUF,600)
       IF(TSTOUT.OR.SLPOUT)CALL DTASWP(DCHAN,POSTWT,ISLBUF,600)
       IF(TANOUT)CALL DTASWP(DCHAN,POSTWT,TANBUF,600)
       IKBST=IAND(PBLK(1),Y'FFFF')
       IF(IKBST.EQ.0)GOTO 17
       IF(I.LT.300)GOTO 211
       CALL KBCA
       CALL READMS
       X=IX/1000.0
       Y=IY/1000.0
       Z=IZ/1000.0
       CSCS=ICS
       CALL REFR(33,33)
       IF(INFOUT)CALL REFR(16,18)
       IF(INFOUT)CALL REFR(21,21)
       GOTO 20
 211   CONTINUE
       GOTO 21
C
```

Figure 15- continued

```
C
 801  CONTINUE
      CALL AVG
      GOTO NEXT
C
 802  CONTINUE
      CALL BASELN(MEAN, SLOPE, CHAN2, CRT)
      GOTO NEXT
C
 803  CONTINUE
      CALL BLANK(22,23)
      WRITE(CRT,125)
 8031 CONTINUE
      CALL BLANK(22,23)
      READ(CRT,126)(LINE(I),I=1,20)
      DO 8032 I=1,20
      IF(LINE(I).NE.Y'20202020')GOTO 8033
 8032 CONTINUE
      GOTO NEXT
 8033 CONTINUE
      WRITE(PR,126)(LINE(I),I=1,20)
      GOTO 8031
C
 804  CONTINUE
      CALL ATD(ADBUF,LENGTH,NNPTS,SHOOT,CHAN2,SHEND)
      CALCD=.FALSE.
      CRNCHD=.FALSE.
      GOTO NEXT
C
 805  CONTINUE
      GOTO NEXT
C
 806  CONTINUE
      GOTO NEXT
C
 807  CONTINUE
      DO 8071 I=1,NPARS
      CALL GETPAR(I)
 8071 CONTINUE
      GOTO NEXT
C
 808  CONTINUE
      CALL CLEAR
      REWIND IHELP
      DO 8081 I=1,10
      READ(IHELP,126)(LINE(J),J=1,20)
      WRITE(CRT,126)(LINE(J),J=1,20)
 8081 CONTINUE
      GOTO NEXT
C
 809  CONTINUE
      INFOUT=.TRUE.
      DATOUT=.FALSE.
      SUMOUT=.FALSE.
      CALL CLEAR
      CALL REFR(1,40)
      GOTO NEXT
C
 810  CONTINUE
      GOTO NEXT
C
 811  CONTINUE
      CALL SETUP(ASHEND,NNUSED)
 8111 CONTINUE
      RMSERR=EPS
      CALL LEAST(NNUSED,RMSERR,MAXDEG,IDEG)
      CRNCHD=.TRUE.
      GOTO 815
C
 812  CONTINUE
      CALL SETUP(SHEND,NNUSED)
      GOTO 8111
C
 813  CONTINUE
      GOTO NEXT
C
```

Figure 15- continued

```
 814  CONTINUE
      CALL IBELL
      CALL GETPAR(27)
      CALL INITAV
      GOTO NEXT
C
 815  CONTINUE
      DATOUT=.TRUE.
      INFOUT=.FALSE.
      SUMOUT=.FALSE.
      CALL CLEAR
      CALL REFR(33,33)
      CALL BLANK(3,3)
      CALL OUTDAT(MEAN,SLOPE,CHAN2,CRT)
      GOTO NEXT
C
 816  CONTINUE
      CALL BLANK(23,24)
      WRITE(CRT,916)
 916  FORMAT('PLOTTING...')
      CALL TOF
      WRITE(PR,9161)(AINFO(7,JKL),JKL=1,6),ISHOT
 9161 FORMAT(/6A4,'DATA FOR SHOT ',I3,' FOLLOWS')
      CALL PLTSHT(15,ADBUF(1))
      GOTO NEXT
C
 817  CONTINUE
      GOTO NEXT
C
 818  CONTINUE
      CALL BLANK(22,23)
      WRITE(3,132)
      READ(3,120)K
      IF(K.EQ.0)GOTO NEXT
      CALL GETPAR(K)
      GOTO 818
C
 819  CONTINUE
      CALL CLEAR
      WRITE(OUT,133)
      WRITE(OUT,134)(NN(I),SVCS(I),SVINT(I),MALPHA(I),I=1,NINTNS)
      GOTO NEXT
C
 820  CONTINUE
      SHOT=SHOT+1
      ISHOT=SHOT
      CALL ATD(ADBUF,LENGTH,NNPTS,SHOOT,CHAN2,SHEND)
      CRNCHD=.FALSE.
      CALCD=.FALSE.
      GOTO NEXT
C
 821  CONTINUE
      CALL GETPAR(29)
      CALL DTA(1,VOLTGE)
      GOTO NEXT
C
 822  CONTINUE
      WRITE(PR,922)ISHOT
 922  FORMAT('DATA FOR SHOT ',I3,' HAS BEEN SAVED ON TAPE...')
      CALL SYSIO(PBLK,'38',MAG1,ADBUF,18000,0)
      ENDFILE MAG1
      GOTO NEXT
C
 823  CONTINUE
      IF(INFOUT)CALL TOF
      WRITE(PR,9161)(AINFO(7,JKL),JKL=1,6),ISHOT
      IF(INFOUT)CALL PRNTSC
      IF(DATOUT)CALL OUTDAT(MEAN,MSLOPE,CHAN2,PR)
      GOTO NEXT
C
 824  CONTINUE
      GOTO NEXT
C
 825  CONTINUE
      GOTO NEXT
C
 826  CONTINUE
      GOTO NEXT
      STOP
      END
```

Figure 15- continued

updated with the current time; this continues until a key has been pressed at the keyboard. When this occurs, the character corresponding to the key has been placed in the array CMD. At present, only one character is allowed.

The first halfword of CMD will then have as its most significant byte an ASCII character between "capital A" and "capital Z". If a non-blank character is found, it is checked to make certain it is a valid command. Invalid commands are flagged by a warning message, and WONKB is called to return control to the operator.

For valid commands, a hexadecimal '40' is subtracted from the byte containing the command character. The result is an integer whose value corrspond to the position of the command character in the alphabet, i.e., 1="A", 2="B", etc. A GOTO based on the statement numbers stored in the array CLABEL (via ASSIGN statements) is executed. As stated, the command processor will return control to the operator via WONKB when 1) a blank character is encountered or 2) a non-valid command is detected.

Subroutine GETPAR accepts one INTEGER*4 argument (K), and issues a prompt for the variable number corresponding to the argument. A listing of GETPAR can be found in Figure 16. At present, there are twenty-seven variables under control of the operator. The variable is read in its proper format, and the screen is refreshed with the new value of the variable. For the variables CS,V,NRNGE,VCAL,ICAL,INTENS,TDEPTH,ATTEN and ISUBS, calculations are performed to resolve the dependencies of some of these variables on others within the group. The following table shows the relationships of these variables to one another:

CS with NRNGE gives V

V with NRNGE gives CS

V with VCAL and ICAL gives INTENS

VCAL with V and ICAL gives INTENS

INTENS with VCAL and ICAL gives V (and therfore, CS)

INTENS with TDEPTH and ATTEN gives ISUBS

ISUBS with TDEPTH and ATTEN gives INTENS

In the code, the argument K is checked to make certain that it is within bounds (greater than zero and less than twenty-seven). A computed GOTO is executed with K as its argument, and the prompt/read sequence following statement number 8XX, where XX is the value of K, is executed. The above-mentioned calculations are performed, if necessary, and REFR is called to update the screen with the new information. The association of a given input parameter number with a

corresponding output variable is given in the INTEGER*2 array CMDT. For a given parameter number, the output variable associated with the parameter is given by the element of CMDT which corresponds to the parameter number. For example, in GETPAR, the tenth input variable is NUSED. CMDT(10) is XX, so REFR would be called with XX as its argument to output the heading and updated value for NUSED. After REFR is called, GETPAR returns.

```
$ARGC
      SUBROUTINE GETPAR(K)
C/COMST
C
      REAL LENGTH, INTENS, ICAL, INFO(40), ISUBS, NPTS, NUSED
C
      INTEGER HEAD(40,6), AINFO(7,6), APTR(40), STATUS
C
      INTEGER*2 WCUR(40,4), CMDT(40,2)
C
      COMMON/REFRSH/HEAD(40,6), WCUR(40,4), AINFO(7,6), APTR(40), INFO(40),
     1   STATUS, CMDT(40,2)
C
      COMMON /NINFO/ NRNGE, NNPTS, NNUSED, IDEG, MAXDEG, NDESRD, NSINBK,
     1     SHEND, ASHEND, ICODE
C
C
      EQUIVALENCE (INFO(6),VCAL),(INFO(7),ICAL),(INFO(8),TSENS),
     1 (INFO(9),TDIAM),(INFO(10),ATTEN),(INFO(11),RCPK),(INFO(12),GAIN)
     2 (INFO(13),FILTER),(INFO(14),LENGTH),(INFO(38),INTENS),
     3 (INFO(16),X),(INFO(17),Y),(INFO(18),Z),(INFO(20),SHOT),
     4 (INFO(25),ALPHA),(INFO(26),NPTS),(INFO(27),NUSED),
     5 (INFO(28),DEG),(INFO(29),EPS),(INFO(34),TDEPTH),(INFO(35),V),
     6 (INFO(36),CS),(INFO(37),ISUBS),(INFO(39),RNGE)
     7 ,(INFO(40),DESRD),(INFO(22),RMSERR)
     8  ,(INFO(15),SINBK),(INFO(19),CODE),(INFO(21),CSCS)
C/COMEND
 930  FORMAT(15A4)
 931  FORMAT(F7.3)
 981  FORMAT(I1)
      CALL BLANK(22,24)
      IF(K.GT.0.AND.K.LE.32)GOTO 210
      WRITE(3,211) K
 211  FORMAT('ILLEGAL PARAMTER NUMBER: ',I4)
      STATUS=1
      RETURN
 210  GOTO (801,802,803,804,805,806,807,808,
     1      809,810,811,812,999,814,999,816,
     2      817,818,819,820,821,822,823,824,
     3      825,826,827,828,829,830,831,832),K
```

Figure 16. GETPAR

```
801   WRITE(3,901)
901   FORMAT('PLEASE ENTER YOUR NAME: ')
      READ(3,930) (AINFO(1,I),I=1,6)
      GOTO 10
802   WRITE(3,902)
902   FORMAT('SAMPLE TISSUE: ')
      READ(3,930) (AINFO(2,I),I=1,6)
      GOTO 10
803   WRITE(3,903)
903   FORMAT('PREPARATION: ')
      READ(3,930) (AINFO(3,I),I=1,6)
      GOTO 10
804   WRITE(3,904)
904   FORMAT('TRANSDUCER: ')
      READ(3,930) (AINFO(4,I),I=1,6)
      GOTO 10
805   WRITE(3,905)
905   FORMAT('THERMOCOUPLE USED: ')
      READ(3,930) (AINFO(5,I),I=1,6)
      GOTO 10
806   WRITE(3,906)
906   FORMAT('VCAL IN VOLTS: ')
      READ(3,931) VCAL
      GOTO 5
807   WRITE(3,907)
907   FORMAT('ICAL IN WATTS/CM**2: ')
      READ(3,931) ICAL
      GOTO 5
808   WRITE(3,908)
908   FORMAT('RANGE USED: ')
      READ(3,981) NRNGE
      RNGE=NRNGE
      GOTO 5
809   WRITE(3,909)
909   FORMAT('ENTER # OF PTS')
      READ(3,9074) NNPTS
9074  FORMAT(I5)
      NPTS=NNPTS
      GOTO 10
810   WRITE(3,9072)
9072  FORMAT('ENTER # TO BE USED')
      READ(3,9074) NNUSED
      NUSED=NNUSED
      GOTO 10
811   WRITE(3,9073)
9073  FORMAT('ENTER DEGREE OF FIT')
      READ(3,981) MAXDEG
      DEG=MAXDEG
      GOTO 10
812   WRITE(3,912)
912   FORMAT('SENSITIVITY IN MICROVOLTS/DEGREE CENTIGRADE: ')
      READ(3,931) TSENS
      GOTO 10
814   WRITE(3,914)
914   FORMAT('DIAMETER IN MILS: ')
      READ(3,931) TDIAM
      GOTO 10
816   WRITE(3,916)
916   FORMAT('ATTENUATION COEFFICIENT IN NEPERS/CM: ')
      READ(3,931)  ATTEN
      GOTO 10
817   WRITE(3,917)
917   FORMAT('RHO SUB P K: ')
      READ(3,931) RCPK
      GOTO 10
818   WRITE(3,918)
918   FORMAT('GAIN OF AMP: ')
      READ(3,931) GAIN
      GOTO 10
819   WRITE(3,919)
919   FORMAT('FILTER CUTOFF - ZERO IF NOT IN CIRCIUT: ')
      READ(3,931) FILTER
      GOTO 10
```

Figure 16- continued

```
820   WRITE(3,920)
920   FORMAT('SHOT LENGTH IN SECONDS: ')
      READ(3,931) LENGTH
      GOTO 10
821   WRITE(3,921)
921   FORMAT(A5,'INTENSITY IN WATTS/CM**2')
      READ(3,931) INTENS
8151  V=VCAL*(SQRT(INTENS/ICAL))
      IF (NRNGE.EQ.3)GOTO 8152
      CS=-0.68752E02+V*(0.94596)
      GOTO 8153
8152  CS=-0.57432E+02+V*(0.20080)
8153  ISUBS=INTENS*(EXP(-2*ATTEN*TDEPTH))
      GOTO 5
822   WRITE(3,922)
922   FORMAT('TDEPTH IN CM: ')
      READ(3,931) TDEPTH
      CALL REFR(34,34)
      IF(ICAL.EQ.0.0) ICAL=1.0
      GOTO 8151
823   WRITE(3,923)
923   FORMAT('ENTER NEW VALUE OF CS')
      READ(3,931) CS
      IF(NRNGE.EQ.3)GOTO 8910
      V=(72.680)+CS*1.0571
      GOTO 8919
8910  V=286.01+CS*4.980
8919  INTENS=ICAL*(V/VCAL)**2
      GOTO 8153
824   WRITE(3,924)
924   FORMAT('ENTER NEW VALUE OF X')
      READ(3,931) X
      GOTO 10
825   WRITE(3,925)
925   FORMAT('ENTER NEW VALUE OF Y')
      READ(3,931) Y
      GOTO 10
826   WRITE(3,926)
926   FORMAT('ENTER NEW VALUE OF Z')
      READ(3,931) Z
      GOTO 10
827   CONTINUE
      WRITE(3,927)
927   FORMAT('ENTER DESIRED NUMBER OF',
     1       'SHOTS IN BLOCK')
      READ(3,932)NDESRD
932   FORMAT(I2)
      DESRD=NDESRD
      GOTO 10
828   CONTINUE
      WRITE(3,928)
928   FORMAT('ENTER SUM OF DISPLAY CODES-',
     1   '16=TAN, 8=TEST, 4=SHOT, 2=FIT, 1=SLOPE')
      READ(3,9281)ICODE
9281  FORMAT(I2)
      CODE=ICODE
      GOTO 10
829   CONTINUE
      WRITE(3,929)
929   FORMAT('ENTER EPS')
      READ(3,931)EPS
      GOTO 10
830   CONTINUE
831   CONTINUE
832   CONTINUE
5     CONTINUE
      CALL REFR(6,7)
      CALL REFR(21,22)
      CALL REFR(35,39)
      GOTO 999
 10   CONTINUE
      IK=CMDT(K,1)
      CALL REFR(IK,IK)
999   RETURN
      END
```

Figure 16- continued

Subroutine ATD reads data from the analog-to-digital converter of the 7/32 and stores it in a buffer, where it can be numerically processed, or plotted on the printer. A listing of ATD can be found in Figure 17. There are five input arguments and one output argument for ATD. The input arguments are ADBUF (INTEGER*2 vector of NNPTS elements), LENGTH (REAL*4), NNPTS (INTEGER*4), ISHOT (INTEGER*4) and CHAN (INTEGER*4). The output argument, SHEND, is INTEGER*4.

On entry to ATD, ADBUF specifies the beginning of an INTEGER*2 vector of NNPTS elements. NNPTS samples will be taken from the A/D channel specified by the value of CHAN. If ISHOT is a 1, a command sequence is output via the selection/isolation circuitry to the remote equipment interface; this sequence will turn on the r.f. power to the transducer for a period of time (in seconds) equal to the value of LENGTH. If ISHOT is a 0, a similar sequence will turn on the counter used to time the r.f. application, but the r.f. itself will not be turned on. After the specified time (with or without the r.f. power on), the program records the sample number of the last sample stored before the time elapsed. This number is output as SHEND, and serves to indicate which data points in the buffer were taken during the time the counter was active (the counter is a down-counter, and halts after the prescribed delay has been counted.). The effective sampling rate of the A/D

```
C
      SUBROUTINE ATD(ADBUF,LENGTH,NNPTS,ISHOT,CHAN,SHEND)
      INTEGER NNPTS,ISHOT,CHAN,SHEND
      INTEGER RGSTG(16)
      INTEGER*2 ADBUF(1)
C
C
      INTEGER DCHI,DCMID,DCLO
      REAL LENGTH,TLNGTH
C
      DATA M1/Y'000000FF'/
C
      TLNGTH=LENGTH
      IF(TLNGTH.GT.166.77)TLNGTH=166.77
      ILNGTH=100000*TLNGTH
C
      DCHI=ISHFT(ILNGTH,-16)
      DCHI=IAND(DCHI,M1)
C
      DCMID=ISHFT(ILNGTH,-8)
      DCMID=IAND(DCMID,M1)
C
      DCLO=IAND(ILNGTH,M1)
C
C
C
      ISPACE=TLNGTH/0.5+0.5
      IF(ISPACE.EQ.0)ISPACE=1
C
      SHEND=NNPTS
C
$ASSM
*
         STM    0,RGSTG              SAVE THE REGISTERS
*
         LHI    1,X'88'              LOAD ADDRESS OF A/D
         LHI    2,X'C6'              LOAD COMMAND WORD
*                                    (NO INTERRUPTS, SINGLE CHANNEL MODE)
*
         OCR    1,2                  OUTPUT COMMAND TO A/D
*
         LHI    2,X'A9'              LOAD ADDRESS OF DIGITAL INPUT
         LHI    3,X'A8'              LOAD ADDRESS OF DIGITAL OUTPUT
*
         LHI    4,X'2909'            THIS SELECTS THE STATUS BITS
*                                    FROM THE MOUSE ROOM AS INPUTS TO THE DIO
*
         WHR    3,4                  OUTPUT THIS TO THE DIO
         BAL    15,WAIT1             PAUSE TO LET LINES SETTLE...
*
         LHI    4,X'2100'            THIS LOADS THE DOWN COUNTER WITH
*                                    THE LOWER ORDER 8 BITS;
         O      4,DCLO               THESE ARE OR'D INTO POSITION
         WHR    3,4                  OUTPIT THIS TO THE DIO
         BAL    15,WAIT1             WAIT...
*
         LHI    4,X'2200'            THIS LOADS THE "MIDDLE" 8 BITS...
         O      4,DCMID              "OR" THEM INTO POSITION;
         WHR    3,4                  OUTPUT TO DIO;
         BAL    15,WAIT1             WAIT...
*
         LHI    4,X'2300'            THIS WILL LOAD HIGH ORDER 8 BITS...
         O      4,DCHI               "OR" THEM INTO POSITION;
         WHR    3,4                  OUTPUT TO DIO;
         BAL    15,WAIT1             WAIT...
*
         LIS      5,0              LOAD STARTING ADDRESS OF BUFFER...
         LIS    6,2                  LOAD INCREMENT VALUE FOR BXLE INSTRUCTION...
         L      15,NNPTS             LOAD ADDRESS OF NNPTS; THIS VALUE WILL
         L      7,0(15)              GIVE THE NUMBER OF HALFWORDS IN BUFFER...
         SLLS   7,1                  MULTIPLY BY TWO AND
         SIS    7,2                  SUBTRACT TWO TO GIVE THE FINAL ADDRESS OF THE
*                                    BUFFER; THIS IS COMPATIBLE WITH THE
*                                    BXLE INSTRUCTION USED IN THE LOOP BELOW...
*
*
         LIS    8,1                  LOAD STARTING COUNT OF "SPACING" ROUTINE
         LIS    9,1                  LOAD INCREMENT VALUE
         L      10,ISPACE            THIS VALUE WILL REDUCE THE NUMBER OF SAMPLE
*                                    POINTS SAVED BY A FACTOR OF "ISPACE", THUS
*                                    REDUCING THE APPARENT SAMPLING RATE...
*
         L      15,CHAN              LOAD ADDRESS OF WORD CONTAINING THE CHANNEL
         L      13,0(15)             LOAD THE CHANNEL NUMBER...
*
```

Figure 17 ATD

```
         L      15,ISHOT
         L      14,0(15)        THIS VALUE WILL BE 1 IF A SHOT IS DESIRED
*                                AND ZERO OTHERWISE...
         BNZ    SHOOT
*
         LHI    4,X'2504'       THE RF WILL NOT GO ON...
         WHR    3,4             TELL THE DOWN TO START ANYWAY;
*                                IT SERVES AS A TIMER...
*        BAL    15,WAIT1        LINES MUST SETTLE BEFORE THE LOOP
*                                BELOW BEGINS CHECKING STATUS...
         B      START           START READING...
*
SHOOT    LHI    4,X'2544'       THIS TURNS ON THE RF FOR THE
*                                DURATION OF THE COUNT...
         WHR    3,4             OUTPUT COMMAND TO MOUSE ROOM...
START    L      15,SHEND
         L      11,ADBUF
         LIS    4,0             ZERO THE FLAG REGISTER...
         RHR    1,12
*
         LI     3,Y'36F0'       LOAD STATUS PORTION OF PSW
*                                WHICH WILL TURN OFF ALL OTHER
*                                INTERRUPTS DURING DATA TAKING...
         EPSR   14,3            EXCHANGE PSW'S
         WHR    1,13
*        WHR    1,13            REG 13 CONTAINS THE CHANNEL NUMBER;
*                                THIS STARTS A CONVERSION WITH THE NEXT
*                                PULSE FROM THE EXTERNAL CLOCK ON THE A/D
*
*
ADLUP    BXLE   8,LR4           REGS 8-10 CONTROL THE "SPACING" THROUGH
*                                THIS INSTRUCTION...
*
         LIS    8,1             HERE IF "ISPACE" SAMPLES HAVE BEEN TAKEN;
*                                THE LAST SAMPLE TAKEN WILL BE PRESERVED
*                                BY THE INCREMENTING OF THE INDEX FOR THE
*                                BUFFER ADDRESS; THIS OCCURS
*                                IN THE INSTRUCTION DIRECTLY BELOW...
*
         BXLE   5,LR4           INCREMENT POINTER; CONTINUE IF BUFFER
*                                IS NOT FULL; ELSE,
         B      RLD             BRANCH TO EXIT SEQUENCE...
*
LR4      CLHI   5,10
         BL     WT
         LR     4,4             SET CONDITION CODE TO CHECK "FLAG";
*                                REG 4 CONTAINS A ONE IF THE SHOT
*                                (OR EQUIVALENT TIME) IS OVER...
         BNZ    WT              IF FLAG IS SET, BRANCH AROUND THE
*                                FOLLOWING STEP...
*
         RHR    2,12            READ THE STATUS BITS FROM THE DOWN COUNTER...
         SRHLS  12,1            SHIFT RIGHTMOST BIT INTO CARRY
         BNC    WT
         L      15,SHEND
         ST     5,0(15)         STORE THE VALUE OF THE POINTER
*                                WHEN TIME RAN OUT...
         LIS    4,1             SET THE FLAG TO INDICATE TIME IS UP;
*                                CONTINUE...
*
*
WT       SSR    1,12            SENSE THE STATUS OF THE A/D
         BTBS   8,1             LOOP BACK IF CONVERSION IS NOT OVER...
         RHR    1,12            READ THE SAMPLE DATA;
         SRHLS  12,5            SHIFT THE DATA TO NORMALIZE...
         NHI    12,X'3FF'       AND TO GET 10 BITS ONLY...
         STH    12,0-2(5,11)        STORE IT AT "POINTED" LOCATION...
         B      ADLUP           BRANCH BACK FOTR MORE SAMPLES...
*                                THE EXIT WILL BE CONTROLLED BY THE BXLE
*                                INSTRUCTION "B RLD" WHICH IS EXECUTED AFTER
*                                THE POINTER EXCEEDS THE END OF THE BUFFER...
*
*
*
WAIT1    LHI    11,1000
LUP1     SIS    11,1            ADD ONE; LOOP IS OVER IF
         BZR    15              RESULT IS ZERO;
         B      LUP1            ELSE KEEP LOOPING
*
*
RLD      EPSR   3,14            GET OLD PSW BACK...
         L      15,SHEND        LOAD ADDRESS OF THE VALUE STORED
         L      14,0(15)        WHEN TIME RAN OUT;
         SRLS   14,1            AND DIVIDE BY TWO...
         ST     14,0(15)        STORE THIS, THE INTEGER NUMBER OF SAMPLES
*                                TAKEN DURING THE TIME COUNTED...
*
         LM     0,RGSTG         RESTORE THE VALUE OF THE REGISTERS; EXIT..
*
$FORT
C
      RETURN
      END
```

Figure 17- continued

conversions stored is then given by the expression

SHEND/LENGTH;

this plays a role in subsequent numerical processing. Remember that since the A/D is under control of a crystal clock, the samples are taken at regular intervals.

The down counter used to control time events in the remote equipment interface has selectable clock frequencies of 100Hz, and 1, 10 and 100KHz. It is a twenty-four bit counter (two to the twenty-fourth = 1,048,576); hence events from 10 s (one count at 100Khz) to several days (using the 100Hz clock) can be timed. For ATD, the LENGTH of time is assumed to be twenty-eight minutes or less, so that one of the two fastest clocks is selected. For LENGTH less than 166.77 seconds, the 100KHz clock is used (with a resolution of 10$s); for LENGTH greater than 166.77 seconds but less than twenty-eight minutes, the 10 KHz clock is used.

The conversion of LENGTH to the binary number necessary for loading into the down counter takes place in FORTRAN code. After the binary number is formed, it is manipulated to form three eight-bit groups, which are stored in DCHI, DCMID and DCLO. These three groups of eight bits are, respectively, the high-, middle- and low-order bits of the twenty-four bit binary number which will be loaded into the downcounter. The loading of the down counter takes place in the assembly-language portion of ATD described below.

Before the assembly-language section of ATD begins, the factor ISPACE is computed, using the expression

ISPACE=LENGTH/0.5+0.5

ISPACE is an integer which indicates how many samples should be taken from the A/D before one is stored; this serves to, as much as possible, fill the A/D buffer, while maintaining a consistent spacing of the samples stored. The calculation for ISPACE assumes that, ideally, all of the samples in LENGTH seconds will (nearly) fill 8192 halfwords in memory. It also assumes a 16.384KHz sampling rate for the A/D, and so a division of LENGTH by 0.5 (8192/16,384 = 0.5) will produce an ISPACE which will most nearly fill the 8192 halfwords in LENGTH seconds. The quantity 0.5 added to the calculation for ISPACE rounds the value of ISPACE to the nearest integer. In the assembly-language code which reads the A/D, ISPACE samples are taken before a sample is stored.

Before entry into the assembly-language portion of ATD, SHEND is set to NNPTS+99. Thus, if the remote equipment interface and the 7/32 do not communicate properly during the time the A/D samples are taken, SHEND will not have been changed to a value less than NNPTS, as is the case for normal completion of the A/D reads.

A useful feature of the implementation of FORTRAN on the 7/32 is the provision for the use of assembly-language code intermixed with FORTRAN code. This feature is discussed in

Perkin-Elmer publication number 29-536R02, "FORTRAN VI User's Manual", pp. 22-23. A description of assembly-language instructions can be found in Perkin-Elmer publication number 29-405R02, "Model 7/32 Processor User's Manual"; a description of asssembly-language syntax and rules is given in Perkin-Elmer publication B29-640, "Assembly Language Programming (CAL). Descriptions of the I/O instructions needed for interaction with the DIO and the A/D are found in two Perkin-Elmer publications, number 29-477, "Digital Input-Output Controller Programming Manual", and number 29-475, "Analog Input Controller (AIC) Programming Manual".

The assembly language code proceeds as follows: The sixteen 32-bit general purpose registers of the 7/32 are saved in a temporary storage area, RGSTG (an array of 16 fullwords). This is necessary because some of these registers are used by FORTRAN as pointers whose value must be kept intact. Rather than attempting to identify exactly which registers may have significance at a given time, it is good practice to save them all on entry to assembly-language code, and restore them to their original state upon exitting from the assembly-language code. Register 1 is loaded with the address of the A/D on the multiplexor bus of the processor, X'88'. Register 2 is loaded with the value X'D5', and output to the A/D as a command. This value for the command byte shuts off the interrupts from the A/D, and selects the single channel, random access mode for the A/D.

This mode olds the analoq multiplexor of the A/D open to the channel pecified by the WHR instruction to the A/D, described elow. This will effectively eliminate the bus capacitanc of the A/D multiplexor, which otherwise would have to be charged prior to samplinq the voltage on the channel s lected, if the A/D were reading from several channels. Next, registers 2 and 3 are loaded with X'A9' and X'A8', th addresses of the diqital input and output ports, respective y. The halfword X'2909' is written to the diqital output pc t; the code X'909' will be passed via the selection/ solation circuitry to the remote equipment interface; this code will cause the gating of the remote interface status bits to the input port of the DIO, as described n (the section dealing with "mouse room reads").

After the writing of halfwords to the DIO, a wait subroutine is entered via a BAL (Branch and Link) instructic ; the purpose of the wait loop is to make certain that the emote interface has time to respond to the command just issu d, before proceeding with further command outputting

Next, commands for the loadinq of the down counter with the three eiqht-bit groups DCHI, DCMID and DCLO are constructe and output to the remote interface. The commands for the lc ding of the down counter are halfwords of the form X'2*KK', here the hexadecimal digit represented by * is 1,

2 or 3, for the lower-, middle-, and upper-order bits of the down counter, respectively. The eight bit value "KK" corresponds to the eight-bit values contained in DCLO, DCMID or DCHI; these bits are OR'ed into position, with the appropriate value for the * position, and output to the remote interface. Again, the wait subroutine is used to ensure validity of the data.

After the loading of the down counter, some necessary information is calculated for the use of loops in the reading of the A/D. The quantities 0, 2 and NNPTS*2-2 are placed in registers 5, 6 and 7, respectively. These values represent, respectively, the starting, increment and final values of the index register (reg. 5) used in conjunction with register 11 to store the samples taken by the A/D. The quantities 1, 1 and ISPACE are loaded into registers 8, 9 and 10 for the loop which controls the number of samples stored. Register 13 is loaded with the value of CHAN; this value is output to the A/D via the WHR 1,13 instruction in the loop described below, and selects the channel given by the value of CHAN as the analog input to the A/D. Register 14 is then loaded with the value of ISHOT. A branch is taken on this value; if ISHOT is zero, the down counter will be started, but r.f. power is not applied to the transducer; if ISHOT is non-zero, r.f. power will be applied to the transducer for the amount of time specified by LENGTH. The construction of the command to start the down counter, with or without the r.f. power

applied, was given above; this command is output to the remote interface.

Before entry into the loop which controls the reading of the A/D, the addresses of SHEND and ADBUF are loaded into registers 15 and 11, respectively, and a zero is loaded into register 4. Register 4 serves as a flag to the code below; after the end of the time interval controlled by the down counter (as indicated by the changing of the status bits of the down counter), reg. 4 is loaded with a 1 and the current value of the index (reg. 5) of ADBUF is stored in SHEND.

The code which controls the reading of the A/D is contained in a loop which begins with the statement labelled "ADLUP" and ends with the statement labelled "BBACK". The execution of the loop is as follows: The A/D channel to be read is written to the A/D via the statement WHR 1,13 , which selects the channel used for the conversion. Remember that, because of the use of the external clock on the A/D, the actual conversion will take place on the next leading edge of the external clock. Next, the BXLE 8,LR4 instruction increments the value of register 8 by the value of reg. 9 (=1) and compares the result to reg. 10 (ISPACE). If reg. 8 is not yet equal to reg. 10 (ISPACE), a branch is taken around the insturctions LIS 8,0 and BXLE 5,LR4. These two instructions are therfore executed only when ISPACE samples have been taken, and serve to reset the spacing loop, and

increment the index register used for the storage of results, respectively. The incrementing of the pointer in the BXLE 5,LR4 instruction also serves as a check to see if the index has passed its final value; if this is the case, a branch is taken to location RLD, where the return from ATD begins. This is the only exit possible from the A/D read loop, and occurs when NNPTS samples have been placed in the A/D buffer. A check is made to see whether ten samples have been taken, to assure that input lines from the remote interface have settled before the status of the down counter is polled. The statement

```
LR4    LR    4,4
```

serves as a check on the flag which seperates the samples taken during the timed interval from those taken after the time has elapsed. If the flag has not yet been set, a check is made to see if the status bits of the down counter have changed. This is accomplished by reading the digital input port; the status bits of the down counter have been gated to the DIO by outputting the halfword X'2909' to the ouput port of the DIO. If the down counter is counting, the least significant bit of the eight bits read from the remote interface will be zero. The check for this condition is made by reading the input port, and shifting register 12 (into which the input data has been placed by the RHR 1,12 instruction) to the right one place. This instruction places the bit shifted out of the register into the condition code, where it is used by the BNC WT instruction to test whether

the bit was a zero (the branch is taken), or a one (no branch to WT). If the bit was a one, the timed interval is over, and the current value of reg. 5 is loaded into SHEND. The flag register (reg. 4) is then set to 1, to indicate that no further check of the down counter status need be made.

Lastly, the status of the A/D is checked. If the A/D has reached end-of-conversion when the status is checked, BUSY (see p. 3 of Perkin-Elmer publication 29-475, "Analog Input Controller (AIC) Programming Manual") The instruction SSR 1,12 loads the status of the A/D into register 12, with the upper four bits of the eight bit status also copied into the condition code. The instruction BTBS 8,1 checks for the busy bit to be zero. If BUSY is a one, a branch is taken back to the SSR 1,11 instruction; this continues until the conversion is over. When BUSY goes to zero, the data from the A/D is read into the location pointed to by the value of the index in register 5, and a branch is taken once again to the beginning of the A/D loop. Note once again that an exit from this loop is possible only when NNPTS samples have been read. At that time, the instruction B RLD begins the exit sequence. The value of SHEND is adjusted to be the integer number of samples taken during the time the down counter was active. The general purpose registers are then restored to their original state, and execution continues with the FORTRAN return statement.

Subroutine DTASWP provides a means of outputting the contents of a buffer to a selected channel of the digital-to-analog converter (D/A). In addition, a trigger pulse is supplied via the contorl line WT0 (see Perkin-Elmer Publication 29-476, "Analog Output Controller Programming Manual", p. 2-3 in reference to the signal "WRITE-THRU" (WT0)) triggering an oscilloscope used to display buffer contents. A companion routine, DTA, outputs a DC level on a selected channel. Listings for these two programs can be found in Figure 18.

There are four input arguments to DTASWP: DCHAN, POSWT and NNPTS (all INTEGER*4) and BUF (start of an INTEGER*2 array). DCHAN is the channel number of the D/A to which output is to be directed. POSTWT is an integer count used to provide a delay before DTASWP RETURNs. This is useful when successive patterns are to be output (as in a DO loop). The delay time allows an oscilloscope trigger to "arm" before the next buffer is output. A delay of approximately five microseconds per count is effected.

The programming of the D/A is discussed in Perkin-Elmer publication 29-476, "Anaolg Output Controller Programming Manual". All that need be considered here is that the INTEGER*2 array BUF contains data which is in a form compatible with the D/A's data structure. REAL data for the

```
      SUBROUTINE DTASWP(DCHAN,POSTWT,BUF,NPTS)
C
      INTEGER DCHAN,POSTWT,NPTS
      INTEGER*2 BUF(1)
      INTEGER RGSTG(16)
C
CC
C
      IF(DCHAN.EG.1)IADDR=Y'98'
      IF(DCHAN.EG.2)IADDR=Y'99'
      IF(DCHAN.NE.1.AND.DCHAN.NE.2)RETURN
C
$ASSM
*
         STM    0,RGSTG
*
         L      1,IADDR              LOAD ADDRESS OF PROPER CHANNEL
*
*
         L      15,POSTWT
         L      4,0(15)              LOAD POST-SWEEP WAIT
*
         L      5,BUF                LOAD STARTING ADDRESS OF BUFFER
*
         L      15,NPTS
         L      6,0(15)              LOAD NUMBER OF VALUES 2/B OUTPUT
         SLLS   6,1                  MULTIPLY BY TWO FOR INDEX LOOP
*                                    (LOADING 1 HALFWORD AT A TIME)
*
         LHI    7,X'98'              LOAD ADDRESS FOR COMMAND
*                                    (ALWAYS THIS ADDRESS)
*
*
         LIS    9,0                  LOAD START COUNT FOR INDEX LOOP
         LIS    10,2                 LOAD INCREMENT VALUE
         LR     11,6                 LOAD FINAL VALUE
*
*                                    WHICH WILL BE CHECKED TO EXIT
*                                    THE "SWEEP" LOOP.
*
SWPST    LHI    8,X'10'              LOAD COMMAND TO RESET "WTO" FLIP-FLOP
         OCR    7,8                  TRIGGER USING "WTO"
*
*
DLOOP    BXH    9,EXIT              INCREMENT INDEX; CHECK;
*                                    BRANCH TO RESET IF ALL VALUES
*                                    HAVE BEEN OUTPUT
*
         LH     12,0-2(5,9)          LOAD VALUE 2/B OUTPUT
         WHR    1,12                 OUTPUT TO D/A
*
*
         B      DLOOP                CONTINUE LOOPING...
*
*
EXIT     LIS    8,0
         OCR    7,8                  RESET "WTO" FLIP-FLOP,
*
         WHR    1,8                  ZERO THE D/A
*
         LR     13,4                 LOAD POST-SWEEP WAIT,
         BAL    14,IWAIT             GO WAIT,
*
         LM     0,RGSTG              RESTORE THE REGISTERS
*
         B      $P999                EXIT VIA FORTRAN "RETURN"
*
IWAIT    SIS    13,1                 SUBTRACT ONE FROM COUNT;
         BZR    14                   RETURN IF COUNTED DOWN TO ZERO;
         B      IWAIT                ELSE, CONTINUE...
*
$FORT
 999  CONTINUE
      RETURN
      END
C
      SUBROUTINE DTA(CHANL,VOLTGE)
      INTEGER CHANL
      INTEGER*2 IVOLT
C
C
      IVOLT=IFIX2(1023.0/10.2375*VOLTGE)*16
C
      CALL DTASWP(CHANL,1,IVOLT,1)
C
      RETURN
      END
```

Figure 18 DTASWP,DTA

voltage range 10.24v can be constructed using a statement of the form

```
BUF(I)=IFIX2(1023.0/10.2375*VOLTGE)*16
```

where VOLTGE is a REAL variable corresponding to the desired output voltage.

When DTASWP is called, a check is made to ensure a proper channel number (ch. 1 or ch. 2). An invalid channel number forces a RETURN with no action taken. Otherwise, NNPTS halfwords are output from the array BUF. Before the first value is output, the trigger line WT0 is taken high (5v); after all values have been output, WT0 is taken low (0v), and the above-mentioned delay counted out before DTASWP RETURNs. Data is output at a rate of one value/12.5 s, approximately.

Subroutine DTA accepts two input arguments, CHANL (INTEGER*4) and VOLTGE (REAL*4). The voltage value given by VOLTGE is output on the selected channel (CHANL) of the D/A by converting VOLTGE to the form required by DTASWP and calling DTASWP with a buffer length of one.

Subroutine READMS obtains data from the remote equipment interface regarding the current position of the transducer relative to the thermocouple (X, Y, Z), the current setting of the variable capacitor used to control intensity (CS) and the current count and status for the down counter used for motion and intensity control, as well as the application of ultasound itself. A listing of READMS can be found in Figure 19. There are no input arguments for READMS; output is placed in the labelled common block READMS.

The above-mentioned data appears at the digital input port via the selection/isolation circuitry in groups of eight bits after the writing of a command halfword to the remote equipment interface. This halfword has the hexadecimal form X'29aa' where "aa" represents the number associated with a given group of eight bits, as given below.

Command	Meaning of data gated
2900	X-pos high order digits (BCD)
2901	X-pos low order digits "
2902	Y-pos high " " "
2903	Y-pos low " " "
2904	Z-pos high " " "
2905	Z-pos low " " "
2906	CS-pos high order bits (binary)
2907	CS-pos low order bits (4): X,Y,Z signs
2908	Value of last eight bits sent to remote equipment
2909	Status of down counter
290A	Low order byte of down counter
290B	Middle " " " "
290C	High " " " "

```
      SUBROUTINE READMS
      INTEGER*2 BUF(16),RGSTG(32),XFF
      COMMON /READMS/ IX,IY,IZ,ICS,IDAT,ISTAT,IDC
      DATA XFF/X'FF'/
C
$ASSM
          STM    0,RGSTG          STORE THE REGISTERS
          LIS    15,0             ZERO INDEX REGISTER
*
          LHI    1,X'A8'          LOAD ADDRESS OF DIGITAL OUTPUT
          LHI    2,X'A9'          LOAD ADDRESS OF DIGITAL INPUT
          LHI    3,X'C0'          LOAD COMMAND TO TURN OFF INTERRUPTS
*
          OCR    1,3              TURN OFF INTERRUPTS - OUTPUT
          OCR    2,3              TURN OFF INTERRUPTS - INPUT
          RHR    2,4              DUMMY READ TO SET STATUS
          LHI    4,X'2000'        COMMAND TO TURN ON INTERFACE
          WHR    1,4              TURN IT ON...
*
          LIS    3,0              START VALUE,
          LIS    4,1              INCREMENT
          LIS    5,15             AND FINAL VALUE OF LOOP CONTROL
*
LUP       LHI    6,X'2900'        MASK FOR READ FROM
*                                 REMOTE EQUIPMENT INTERFACE...
          OR     6,3              OR IN BIT GROUP DESIRED...
          BAL    7,DWAIT          WAIT FOR LINES TO SETTLE...
          WHR    1,6              WRITE ENABLE FOR DESIRED INFO...
          BAL    7,IWAIT          WAIT FOR LINES TO SETTLE...
          RHR    2,6              INPUT DATA...
          STH    6,BUF(15)        STORE IT...
          AIS    15,2             INCREMENT INDEX REGISTER...
*
          BXLE   3,LUP            CHECK TO SEE IF ALL GROUPS
*                                 HAVE BEEN INPUT...
*
          LM     0,RGSTG          HERE IF WE'RE DONE...
          B      $P200            BRANCH TO FORTRAN STMNT 200
*
DWAIT     LIS    8,0              SIMPLE DELAY LOOP...
          LIS    9,1
          LHI    10,100
DLUP      BXLE   8,DLUP
          BR     7
*
IWAIT     LIS    8,0              SIMPLE DELAY LOOP...
          LIS    9,1
          LHI    10,100
ILUP      BXLE   8,ILUP
          BR     7
*
$FORT
  200 CONTINUE
C
      IXHI=BUF(1)
      IXLO=BUF(2)
      IYHI=BUF(3)
      IYLO=BUF(4)
      IZHI=BUF(5)
      IZLO=BUF(6)
      ICSHI=BUF(7)
      ICSLO=BUF(8)
      IDAT=BUF(9)
      ISTAT=IAND2(BUF(10),XFF)
      IDCLO=IAND2(BUF(11),XFF)
      IDCMID=IAND2(BUF(12),XFF)
      IDCHI=IAND2(BUF(13),XFF)
C
C
      IX=100*IXHI-600*(ISHFT(IXHI,-4))+IXLO-6*(ISHFT(IXLO,-4))
      IXSIGN=IAND(ICSLO,Y'2')
      IF(IXSIGN.EQ.0)IX=-IX
C
      IY=100*IYHI-600*(ISHFT(IYHI,-4))+IYLO-6*(ISHFT(IYLO,-4))
      IYSIGN=IAND(ICSLO,Y'4')
      IF(IYSIGN.EQ.0)IY=-IY
C
      IZ=100*IZHI-600*(ISHFT(IZHI,-4))+IZLO-6*(ISHFT(IZLO,-4))
      IZSIGN=IAND(ICSLO,Y'1')
      IF(IZSIGN.EQ.0)IZ=-IZ
C
      ICS=ISHFT(ICSHI,5)
      ICSLO=ISHFT(ICSLO,-3)
      ICS=IOR(ICS,ICSLO)
C
      IDC=ISHFT(IDCHI,8)
      IDC=IOR(IDC,IDCMID)
      IDC=ISHFT(IDC,8)
      IDC=IOR(IDC,IDCLO)
C
      RETURN
      END
```

Figure 19 READMS

An assembly-language block is executed to obtain the raw data from the interface. This data is then converted to integer form and placed in the common block READMS to allow access to the user via FORTRAN. The assembly-language portion consists of a loop executed sixteen times whereby halfword commands of the above-mentioned form are output. The "aa" portion is incremented from "00" to "0F", gating the different "bit groups" from the remote interface to the digital input port, where they are subsequently read. At present, the groups "0D" to "0F" are unused; though the program asks for this data from the interface, anything read thereafter is disregarded.

Since data read from the digital input port is always read via a halfword read statement, the eight bit groups are stored right-justified in successive halfword locations in memory. The data is then manipulated via FORTRAN to produce seven INTEGER*4 variables (IX, IY, IZ, ICS, IDAT, ISTAT, IDC) whose values correspond, respectively, to the X, Y, and Z positions of the transducer, the binary output of the CS encoder, the last data written to the remote interface, the status of the down counter, and the current count of the down counter (24-bit binary).

Note that the value of the integer variable corresponding to X, Y and Z is actually one thousand times the current position. For example, if IX is -1987, the

current X-position is -1.987 . The most significant bit read for the status of the down counter is a zero whenever the down counter is inactive; this bit can be obtained using a mask of Y'80' with the variable ISTAT.

Converted values are stored in the labelled common block READMS and the subroutine RETURNs.

LEAST SQUARES PROGRAMMING

The program for calculating the least squares fit from the A/D data is taken from Numerical Computing: An Introduction , by L.F. Shampine and R.C. Allen, Jr. . A listing can be found in Figure 20. It is based on a method which allows selection of an arbitrary degree for the polynomial used in the approximation of the data, while maintaining high precision in the numerical manipulation of the data. The method itself is described in section II.1.2 of the text. (pp. 43-51) A listing and description of two FORTRAN subprograms implementing the method can be found on pp. 230-234 of the text. The programs are well commented, and can be easily used without studying the mathematics involved. A listing of the two subprograms, LEAST and EVAL, can be found in Figures 20 and 21, respectively. This listing includes several modifications discussed below.

The subprogram LEAST performs the actual least squares fit calculations and outputs an array of coefficients. These coefficients are used in a recursive manner to obtain the actual value of the fit for an arbitrary value of the independent variable. The actual evaluation of the fit takes place in the subprogram EVAL.

In order to use these programs for the system being described, several modifications to the code supplied by

```
CC     ALL COMMENTS WHICH BEGIN WITH A "CC" IN COLUMN 1
CC     HAVE BEEN ADDED TO THE ORIGINAL LISTING
CC     FOR THIS PROGRAM - THEY INDICATE CHANGES
CC     MADE TO ACCOMODATE THE USE OF THIS PROGRAM
CC     AT THE BIOACOUSTICS RESEARCH LABORATORY
CC     AT THE UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN.
CC
CC     THE REMOVAL OF THESE COMMENTS AND THE ADDITIONAL
CC     CODE (IF ANY) FLAGGED BY THEM WILL YIELD THE
CC     ORIGINAL CODE FOR "LEAST"
CC
CC
CC THE FOLLOWING LEAST SQUARES PROGRAM IS TAKEN
CC FROM NUMERICAL COMPUTING: AN INTRODUCTION,
CC BY SHAMPINE, LAWRENCE F. AND RICHARD C. ALLEN, JR.
CC 1973: W. B. SAUNDERS COMPANY, PHILADELPHIA, PA.
CC
      SUBROUTINE LEAST(M, X, F, W, EPS, MAXDEG, NDEG, ARRAY, R)
CC
CC     THE ARRAYS USED BY LEAST HAVE BEEN PLACED IN
CC     THE COMMON BLOCK "CRUNCH", TO SPEED UP PROCESSING
CC
CC     THEREFORE, THE NUMBER OF PARAMETERS PASSED
CC     TO LEAST IS FOUR, AND THE ABOVE
CC     SUBROUTINE DECLARATION IS AMENDED TO THAT
CC     GIVEN BELOW
CC
      SUBROUTINE LEAST(M, EPS, MAXDEG, NDEG)
CC
C
C THE SUBROUTINE LEAST AND THE FUNCTION EVAL CALCULATE
C THE LEAST SQUARES POLYNOMIAL APPROXIMATION TO A SET
C OF DATA SPECIFIED BY THE ARRAY OF M NODES, X, WITH
C CORRESPONDING FUNCTION VALUES AND WEIGHTS THE ARRAYS F
C AND W, RESPECTIVELY. THE WEIGHTS MUST ALL BE POSITIVE. THE
C POLYNOMIAL IS DETERMINED IN LEAST AND EVALUATED IN EVAL.
C ON INPUT EPS THE DESIRED WIGHTED RMS ERROR. THE
C CODE INCREASES THE DEGREE OF THE FIT IN AN ATTEMPT TO
C MEET THIS ERROR REQUEST. ON RETURN EPS IS SET TO
C THE WEIGHTED RMS ERROR OF THE FIT. BECAUASE EPS IS USED
C FOR BOTH INPUT AND OUTPUT, IT MUST BE A VARIABLE IN
C IN THE CALLING PROGRAM. MAXDEG IS THE HIGHEST DEGREE
C ALLOWED AND MUST BE LESS THAN OR EQUAL TO (M-1). THE
C ACTUAL DEGREE OF THE FIT IS RETURNED IN NDEG. TO FORCE
C THE CODE TO USE THE PARTICULAR DEGREE MAXDEG, SET
C EPS NEGATIVE ON INPUT. THE DOUBLE PRECISION VECTOR
C R OF M WORDS OUTPUTS THE DOUBLE PRECISION VALUESOF
C THE POLYNOMIAL FIT AT EACH OF THE DATA POINTS X(I).
C THE VECTOR "ARRAY" SPECIFIES THE ORTHOGONAL POLYNOMIAL
C FIT AND PROVIDES WORKING STORAGE. THE DIMENSION OF
C ARRAY IN THE CALLING PROGRAM MUST BE AT LEAST 2*M+3*MAXDEG.
C THE ARRAYS X, F, W, ARRAY AND R MUST BE DIMENSIONED IN THE CALLING PROGRAM.
C
CC     DIMENSION X(1), F(1), W(1), ARRAY(1)
CC
CC     BECAUSE OF THE ABOVE-MENTIONED USE OF A COMMON
CC     BLOCK FOR THE ARRAYS X, F, W, ARRAY AND R, THE
CC     DIMENSION STATEMENT ABOVE IS AMENDED TO:
CC
      DIMENSION X(600), F(600), ARRAY(1245)
CC
CC     NOTE ALSO THAT BECAUSE WEIGHTS ARE ALWAYS
CC     EQUAL TO 1.0, THE USE OF THE ARRAY "W", WHICH
CC     WOULD CONTAIN THE POSITIVE WEIGHTS OF THE
CC     FUNCTION VALUES, CAN BE REDUCED TO THE USE
CC     OF THE CONSTANT "1.0" IN PLACE OF REFERENCES
CC     TO "W(I)"; THIS CHANGE HAS BEEN MADE FOR ALL
CC     OF THE FOLLOWING CODE; ALL OCCURENCES OF
CC     THE VARIABLE "WGHT" BELOW INDICATE A
CC     REPLACEMENT OF A REFERNCE TO "W(I)" WITH THE
CC     VARIABLE "WGHT", WHICH IS ASSIGNED A VALUE
CC     OF 1.0 IN THE DATA STATEMENT BELOW
CC     AND THE FOLLOWING DIMENSIONS ADDED; THEY ARE USED
CC     BY OTHER SUBROUTINES WHICH OPERATE ON PROCESSED
CC     DATA...
CC
      DIMENSION TF(11), TT(11), TS(11), ALPH(11),
     1          DTF(11), DTS(11), DTR(11)
CC
CC     IN ADDITION, THE FOLLOWING VARIABLES MUST BE
CC     DECLARED AS REAL:
CC
      REAL MEAN, MSLOPE
CC
CC     DOUBLE PRECISION R(1), SUM, CK, TEMP
CC
CC     NEXT, THE DIMENSION STATEMENT GIVEN ABOVE
CC     IS AMENDED TO:
CC
      DOUBLE PRECISION R(600), SUM, CK, TEMP
CC     FINALLY, THE "COMMON" STATEMENT
CC
C
      COMMON/CRUNCH/ X(600), F(600), W(600), ARRAY(1245),
     1               TF(11), TT(11), TS(11), ALPH(11),
     2               DTF(11), DTS(11), DTR(11), R(600),
     3               MEAN, MSLOPE
```

Figure 20 LEAST

```
C
C  INITIALIZE STORAGE AND CONSTANTS.
CC
      DATA WGHT/1.0/
CC
      IB=MAXDEG+1
      IBL2=MAXDEG-1
      IC=IB+IBL2
      IOL1=IC+MAXDEG
      I1L1=IOL1+M
      RM=M
      TOL=RM*EPS**2
C
C  CALCULATE CONSTANT FIT
C
      NDEG=0
      S=0.0
      SUM=0.D0
      DO 1 I=1,M
      S=S+WGHT
      SUM=SUM+DBLE(WGHT)*DBLE(F(I))
 1    CONTINUE
      RNO=S
C
C  CK IS THE COEFFICIENT C(0) HERE
C
      CK=SUM/RNO
      ARRAY(IC)=CK
      ERROR=0.0
      DO 2 I=1,M
      R(I)=CK
      ERROR=ERROR+WGHT*SNGL(CK-DBLE(F(I)))**2
 2    CONTINUE
      IF(NDEG.EQ.MAXDEG)GO TO 14
      IF(EPS.LT.0.0)GO TO 3
      IF(ERROR.LE.TOL)GO TO 14
C
C  CALCULATE LINEAR FIT
C
   3  NDEG=1
      ES=ERROR
      SUM=0.D0
      DO 4 I=1,M
      SUM=SUM+DBLE(WGHT)*DBLE(X(I))
4     CONTINUE
C
C  CALCULATE A(1).
C
      ARRAY(1)=SUM/RNO
C
C  CALCULATE Q1(.).
C
      S=0.0
      SUM=0.D0
      DO 5 I=1,M
CC
CC    SPECIAL SUBSCRIPT EVALUATION...
CC
      I1L1PI=I1L1+I
      ARRAY(I1L1PI)=X(I)-ARRAY(1)
      S=S+WGHT*ARRAY(I1L1PI)**2
      TEMP=DBLE(F(I))-R(I)
      SUM=SUM+DBLE(WGHT)*DBLE(ARRAY(I1L1PI))*TEMP
 5    CONTINUE
      RN1=S
C
C  CK IS THE COEFFICIENT C(1) HERE
C
      CK=SUM/RN1
      ARRAY(IC+1)=CK
C
C  CALCULATE THE VALUE OF THE FIT AT THE DATA POINTS AND
C  ALSO THE RMS ERROR.
C
      ERROR=0.0
      DO 6 I=1,M
CC
CC    SPECIAL SUBSCRIPT EVALUATION...
CC
      I1L1PI=I1L1+I
      R(I)=R(I)+CK*DBLE(ARRAY(I1L1PI))
      ERROR=ERROR+WGHT*SNGL(R(I)-DBLE(F(I)))**2
 6    CONTINUE
```

Figure 20- continued

```
      IF(ERROR.GT.ES.AND.EPS.GE.0.0)GO TO 12
      IF(NDEG.EQ.MAXDEG)GO TO 14
      IF(ERROR.LE.TOL.AND.EPS.GE.0.0)GO TO 14
      DO 7 I=1,M
CC
CC    SPECIAL SUBSCRIPT EVALUATION...
CC
      IOL1PI=IOL1+I
      ARRAY(IOL1PI)=1.0
 7    CONTINUE
      NDEG=2
      K=2
C
C GENERAL FIT
C
   8 ES=ERROR
C
C CALCULATE B (K).
C
CC
CC    SPECIAL SUBSCRIPT EVALUATION...
CC
      IBL2PK=IBL2+K
      ARRAY(IBL2PK)=RN1/RNO
C
C CALCULATE A(K).
      SUM=0.DO
      DO 9 I=1,M
CC
CC    SPECIAL SUBSCRIPT EVALUATION...
CC
      I1L1PI=I1L1+I
      SUM=SUM+DBLE(WGHT)*DBLE(X(I))*DBLE(ARRAY(I1L1PI))**2
 9    CONTINUE
      ARRAY(K)=SUM/RN1
C
C CALCULATE GK(.) OVERWRITING ON GK-2(.).
C
      S=0.0
      SUM=0.DO
      DO 10 I=1,M
CC
CC    SPECIAL SUBSCRIPT EVALUATION...
CC
      I1L1PI=I1L1+I
      IOL1PI=IOL1+I
      IBL2PK=IBL2+K
      ARRAY(IOL1PI)=(X(I)-ARRAY(K))*ARRAY(I1L1PI)
     1   -ARRAY(IBL2PK)*ARRAY(IOL1PI)
      S=S+WGHT*ARRAY(IOL1PI)**2
      TEMP=DBLE(F(I))-R(I)
      SUM=SUM+DBLE(WGHT)*DBLE(ARRAY(IOL1PI))*TEMP
 10   CONTINUE
      RNO=RN1
      RN1=S
C
C SWAP INDICES SO I1 REFERS TO STORAGE OF GK(.)
C AND IO TO GK-1(.).
C
      IT=IOL1
      IOL1=I1L1
      I1L1=IT
C
C CK IS THE COEFFIFCIENT C(K) HERE
C
      CK=SUM/RN1
CC
CC    SPECIAL SUBSCRIPT EVALUATION...
CC
      ICPK=IC+K
      ARRAY(ICPK)=CK
C
```

Figure 20- continued

```
C CALCULATE THE VALUE OF THE FIT AT THE DATA POINTS AND
C ALSO THE RMS ERROR.
C
      ERROR=0.0
      DO 11 I=1,M
CC
CC    SPECIAL SUBSCRIPT EVALUATION...
CC
      I1L1PI=I1L1+I
      R(I)=R(I)+CK*DBLE(ARRAY(I1L1PI))
      ERROR=ERROR+WGHT*SNGL(R(I)-DBLE(F(I)))**2
 11   CONTINUE
      IF(ERROR.GT.ES.AND.EPS.GE.0.0)GO TO 12
      IF(NDEG.EQ.MAXDEG)GO TO  14
      IF(ERROR.LE.TOL.AND.EPS.GE.0.0)GO TO 14
      NDEG=NDEG+1
      K=K+1
      GO TO 8
C
C HERE IF ERROR INCREASED ON RAISING DEGREE.
C
   12 NDEG=NDEG-1
      ERROR=ES
      DO 13 I=1,M
CC
CC    SPECIAL SUBSCRIPT EVALUATION...
CC
      I1L1PI=I1L1+I
      R(I)=R(I)-CK*DBLE(ARRAY(I1L1PI))
 13   CONTINUE
C
C EXIT.
C
   14 EPS=SQRT(ERROR/RM)
      RETURN
      END
```

Figure 20- continued

```
      SUBROUTINE EVAL(YY,N,FIT,SLOPE)
C
C  SUBROUTINE EVAL EVALUATES THE ORTHOGONAL POLYNOMIAL
C FIT COMPUTED BY LEAST AND SPECIFIED BY THE VECTOR
C ARRAY.  THE FIT OF DEGREE N IS EVALUATED AT THE
C ARGUMENT YY.  N MUST BE LESS THAN OR EQUAL TO N AS
C  RETURNED FROM LEAST.  LEAST IS CALLED ONLY ONCE FOR
C EACH FIT, BUT EVAL IS CALLED ONCE FOR EACH ARGUMENT
C AT WHICH WE REQUIRE THE VALUE OF THE FIT.  MAXDEG MUST
C HAVE THE SAME VALUE AS IN THE CALL TO LEAST.
CC
CC THE CODE BELOW HAS BEEN MODIFIED TO INCLUDE
CC CALCULATION OF THE FIRST DERIVATIVRE OF THE
CC FIT.  THE VALUE OF THE FIT IS RETURNED IN THE
CC ARGUMENT FIT AND THE VALUE OF THE DERIVATIVE
CC IS RETURNED IN THE ARGUMENT SLOPE.  THE USE
CC OF THE COEFFICIENTS IN ARRAY TO COMPUTE AN
CC ARBITRARY DERIVATIVE IS DISCUSSED AS AN EXERCISE
CC IN SHAMPINE AND ALLEN.
CC
C/COMST
C
      REAL TX(600),ADF(600),ARRAY(1245)
      REAL TF(11),TT(11),TS(11),ALPH(11)
      REAL DTF(11),DTS(11),DTR(11)
      REAL MEAN,MSLOPE
      DOUBLE PRECISION R(600)
C
      COMMON/CRUNCH/ TX(600),ADF(600),ARRAY(1245),
     1               TF(11),TT(11),TS(11),ALPH(11),
     2               DTF(11),DTS(11),DTR(11),R(600),
     3               MEAN,MSLOPE
C
      INTEGER SHEND,ASHEND
C
      COMMON /NINFO/ NRNGE,NNPTS,NNUSED,IDEG,MAXDEG,NDESRD,NSINBK,
     1     SHEND,ASHEND,ICODE
C
C/COMEND
C
      IB=MAXDEG+1
      IC=MAXDEG+IB-1
C
C EVALUATE N=0,1 AS SPECIAL CASES.
C
      IF(N.GT.0)GO TO 1
      FIT=ARRAY(IC)
      SLOPE=0.0
      RETURN
1     IF(N.GT.1)GO TO 2
      FIT=ARRAY(IC)+ARRAY(IC+1)*(YY-ARRAY(1))
      SLOPE=ARRAY(IC+1)
      RETURN
C
C GENERAL RECURRENCE RELATION
C
CC
CC    SPECIAL SUBSCRIPT EVALUATION...
CC
    2 ICPN=IC+N
      DKP2=ARRAY(ICPN)
CC
CC    SPECIAL SUBSCRIPT EVALUATION...
CC
      ICPNM1=IC+N-1
      DKP1=ARRAY(ICPNM1)+(YY-ARRAY(N))*DKP2
      DPKP2=0.0
      DPKP1=DKP2
      NL2=N-2
      IF(NL2.LT.1)GO TO  4
      DO 3 L=1,NL2
      K=1+NL2-L
CC
CC    SPECIAL SUBSCRIPT EVALUATION...
CC
      ICPK=IC+K
      IBPK=IB+K
      DK=ARRAY(ICPK)+(YY-ARRAY(K+1))*DKP1
     1   -ARRAY(IBPK)*DKP2
      DPK=DKP1+(YY-ARRAY(K+1))*DPKP1
     1   -ARRAY(IBPK)*DPKP2
      DKP2=DKP1
      DKP1=DK
      DPKP2=DPKP1
      DPKP1=DPK
    3 CONTINUE
      FIT=ARRAY(IC)+(YY-ARRAY(1))*DKP1
    4   CONTINUE
     1   -ARRAY(IB)*DKP2
      SLOPE=DKP1+(YY-ARRAY(1))*DPKP1
     1   -ARRAY(IB)*DPKP2
      RETURN
      END
```

Figure 21 EVAL

Shampine and Allen were made: The first involves those statements where array subscripts are functions of two variables, or of two variables and a constant. Subscript expressions using more than one variable are incompatible with the implementation of FORTRAN on the 7/32. To circumvent this, these subscript expressions are evaluated prior to use, and assigned to a dummy variable. This dummy variable is then used as a subscript in the offending statements. All occurences of these statements are flagged in the listing given in Appendix I with a comment line of the form

```
CC        SPECIAL SUBSCRIPT EVALUATION...
```

The second modification is the use of common block variables to communicate between the main program and LEAST. This results in a significant decrease in program size, and a corresponding increase in execution speed. Common block variables are flagged in the listing. Note also that the weights needed by LEAST are all set to 1.0, and this is reflected in that all references to W(I) have been changed to a weight factor which is initialized to 1.0. The last modification applies only to the subprogram EVAL. EVAL has been changed from a function subprogram to a subroutine. This was done because it was desired to obtain two values for each value of the independent variable, that of the fit and that of the derivative of the fit. (A function subprogram can return only one value.) The calculation of an arbitrary

derivative of the fitted function using the aforementioned array of coefficients is given as an excercise (Section II.1.2, ex. 2, p. 52) in Shampine and Allen; the solution to this exercise is found on page 184, and defines a recurrence relation for the derivative. Since only the first derivative is of interest in this work, it is the only one which has been programmed.

Subroutine CALC performs the calculation of data which is displayed on the oscilloscope. A listing of CALC can be found in Figure 22. The data is stored in a labelled common block, SCOP, and if desired, it can be plotted on the line printer through the command {P}.

There is one input argument to CALC, INUM, an INT*4 variable. Also needed by CALC is the variable ICODE, contained in the labelled common block NINFO. INUM is the number of data points to be averaged in calculating the slope of tangents to a fitted polynomial. ICODE is a variable under operator control which is the sum of the codes given below.

1 Instantaneous slope of fitted curve...

2 Fitted curve...

4 A/D data as recorded during ultrasound...

8 Test pattern used to set up oscilloscope...

16 Slope of fitted curve at t=0.5 displayed as a tangent at t=0.5

When the operator enters ICODE (variable 29), the sum of the above codes indicates the display(s) desired. The appropriate calculations are carried out, and logical variables set to indicate to the main program which displays are desired. Output is in a form compatible with DTASWP, the

```
C
      SUBROUTINE CALC(INUM)
      INTEGER*2 DTAFIX, IDTAFX
C/COMST
      REAL LENGTH, INTENS, ICAL, INFO(40), ISUBS, NPTS, NUSED
C
      INTEGER HEAD(40,6), AINFO(7,6), APTR(40), STATUS
C
      INTEGER*2 WCUR(40,4), CMDT(40,2)
C
      COMMON/REFRSH/HEAD(40,6), WCUR(40,4), AINFO(7,6), APTR(40), INFO(40),
     1  STATUS, CMDT(40,2)
C
C
      INTEGER*2 ADBUF(9000)
C
      COMMON/ATD/ ADBUF(9000)
C
C
      LOGICAL TSTOUT, CROUT, SLPOUT, TANOUT, CALCD
      LOGICAL SHTOUT, INFOUT, DATOUT, SUMOUT, MAKCHK, AVDONE, CRNCHD
C
      COMMON /CNTRL/ TSTOUT, CROUT, SHTOUT, SLPOUT, TANOUT, CALCD,
     1    INFOUT, DATOUT, SUMOUT, MAKCHK, AVDONE, CRNCHD
C
C
      REAL TX(600), ADF(600), ARRAY(1245)
      REAL TF(11), TT(11), TS(11), ALPH(11)
      REAL DTF(11), DTS(11), DTR(11)
      REAL MEAN, MSLOPE
      DOUBLE PRECISION R(600)
C
      COMMON/CRUNCH/ TX(600), ADF(600), ARRAY(1245),
     1               TF(11), TT(11), TS(11), ALPH(11),
     2               DTF(11), DTS(11), DTR(11), R(600),
     3               MEAN, MSLOPE
C
C
C
      INTEGER*2 ISBUF(600), ICRBUF(600), ISLBUF(600)
      INTEGER*2 TANBUF(600)
      COMMON /SCOP/ ISBUF(600), ICRBUF(600), ISLBUF(600), TANBUF(600)
C
      INTEGER SHEND, ASHEND
C
      COMMON /NINFO/ NRNGE, NNPTS, NNUSED, IDEG, MAXDEG, NDESRD, NSINBK,
     1    SHEND, ASHEND, ICODE
C
C
      EQUIVALENCE (INFO(6),VCAL),(INFO(7),ICAL),(INFO(8),TSENS),
     1 (INFO(9),TDIAM),(INFO(10),ATTEN),(INFO(11),RCPK),(INFO(12),GAIN),
     2 (INFO(13),FILTER),(INFO(14),LENGTH),(INFO(38),INTENS),
     3 (INFO(16),X),(INFO(17),Y),(INFO(18),Z),(INFO(20),SHOT),
     4 (INFO(25),ALPHA),(INFO(26),NPTS),(INFO(27),NUSED),
     5 (INFO(28),DEG),(INFO(29),EPS),(INFO(34),TDEPTH),(INFO(35),V),
     6 (INFO(36),CS),(INFO(37),ISUBS),(INFO(39),RNGE)
     7 ,(INFO(40),DESRD),(INFO(22),RMSERR)
     8  ,(INFO(15),SINBK),(INFO(19),CODE),(INFO(21),CSCS)
C/COMEND
C
C
      TSTOUT=.FALSE.
      SHTOUT=.FALSE.
      CROUT=.FALSE.
      SLPOUT=.FALSE.
      TANOUT=.FALSE.
      ITEMP=IAND(ICODE,Y'10')
      IF(ITEMP.NE.0)TANOUT=.TRUE.
      ITEMP=IAND(ICODE,Y'8')
      IF(ITEMP.NE.0)TSTOUT=.TRUE.
      ITEMP=IAND(ICODE,Y'4')
      IF(ITEMP.NE.0)SHTOUT=.TRUE.
      ITEMP=IAND(ICODE,Y'2')
      IF(ITEMP.NE.0)CROUT=.TRUE.
      ITEMP=IAND(ICODE,Y'1')
      IF(ITEMP.NE.0)SLPOUT=.TRUE.
C
```

Figure 22 CALC

```
      IF(TSTOUT)GOTO 200
C
      SUMF=0.0
      SUMS=0.0
      TY=0.5-(INUM-1)*0.005-0.01
      DO 10 I=1,INUM
      TY=TY+0.01
      CALL EVAL(TY,MAXDEG,FIT,SLOPE)
      SUMF=SUMF+FIT
      SUMS=SUMS+SLOPE
 10   CONTINUE
      T05=SUMF/INUM
      SLP05=SUMS/INUM
      YORG=T05-0.5*SLP05
      DO 100 I=1,NNPTS,15
C
      J=(I-1)/15+1
C
C
      ISBUF(J)=IDTAFX(ADBUF(I))
C
C
      TY=FLOAT(I-1)*LENGTH/SHEND
      CALL EVAL(TY,MAXDEG,FIT,SLOPE)
C
      TANBUF(J)=DTAFIX((I-1)*LENGTH/SHEND*SLP05+YORG)
      IF(I.GT.SHEND)GOTO 15
      ICRBUF(J)=DTAFIX(FIT)
      ISLBUF(J)=DTAFIX(SLOPE)
      GOTO 100
 15   CONTINUE
      ICRBUF(J)=IDTAFX(0)
      ISLBUF(J)=IDTAFX(0)
 100  CONTINUE
      GOTO 99
 200  CONTINUE
      DO 210 I=1,600
C
C
      ISBUF(I)=DTAFIX(I/600.0*10.0)
      IF(SHTOUT)ISBUF(I)=IDTAFX(ADBUF(10*I))
      ICRBUF(I)=DTAFIX((600-I)/600.*10.0)
      ISLBUF(I)=DTAFIX(5.0)
 210  CONTINUE
 99   CONTINUE
      CALCD=.TRUE.
      RETURN
      END
C
      INTEGER*2 FUNCTION IDTAFX(ITEMP)
C
      IDTAFX=ISHFT(ITEMP,4)
C
      RETURN
      END
C
C
      INTEGER*2 FUNCTION DTAFIX(VOLTGE)
C
      DTAFIX=IFIX2(VOLTGE*1023.0/10.2375*16)
C
      RETURN
      END
```

Figure 22- continued

program used to display data from a buffer to the oscilloscope. Six hundred points are stored in each buffer.

Subroutine GETDAT performs the calculations necessary to yield the ultrasonic absorption coefficient from the fitted polynomial. A listing for GETDAT can be found in Figure 23. There is one input argument for GETDAT, and output is directed to the labelled common block CRUNCH. The input argument, ITYPE, determines the degree for which the data will be evaluated. For ITYPE = 1, a linear fit is assumed, as in baseline calculations (see section on subroutine BASELN). For ITYPE = 2, the degree of the fit ordered by the operator is used.

The calculations are straightforward. EVAL is used to obtain the values of the fitted function and its first derivative at eleven equally spaced times from t=0 to t=LENGTH. The value of the fit corresponds to a voltage; this is multiplied by the factor TSG (equal to 100/TSENS*GAIN) to yield the equivalent temperature. This applies also to the slope calculated from the fit; it is in turn multiplied by the factor RCPK/(2.0*ISUBS) (as in equation 1 of chapter 1) to yield the absorption coefficient. This calculation is carried out for each value of time; in practice, only the value for t=0.5 is of interest. The mean value of voltage and voltage slope is computed. It is displayed during baseline calculations as part of the "learning" data upon which criteria for continuation of automated experiments will be based. As mentioned in chapter 3, large temperature drifts preclude accurate measurements.

```
C
      SUBROUTINE GETDAT(ITYPE)
C/COMST
C
      REAL LENGTH, INTENS, ICAL, INFO(40), ISUBS, NPTS, NUSED
C
      INTEGER HEAD(40,6), AINFO(7,6), APTR(40), STATUS
C
      INTEGER*2 WCUR(40,4), CMDT(40,2)
C
      COMMON/REFRSH/HEAD(40,6), WCUR(40,4), AINFO(7,6), APTR(40), INFO(40),
     1  STATUS, CMDT(40,2)
C
C
C
      REAL TX(600), ADF(600), ARRAY(1245)
      REAL TF(11), TT(11), TS(11), ALPH(11)
      REAL DTF(11), DTS(11), DTR(11)
      REAL MEAN, MSLOPE
      DOUBLE PRECISION R(600)
C
      COMMON/CRUNCH/ TX(600), ADF(600), ARRAY(1245),
     1               TF(11), TT(11), TS(11), ALPH(11),
     2               DTF(11), DTS(11), DTR(11), R(600),
     3               MEAN, MSLOPE
C
      INTEGER SHEND, ASHEND
C
      COMMON /NINFO/ NRNGE, NNPTS, NNUSED, IDEG, MAXDEG, NDESRD, NSINBK,
     1     SHEND, ASHEND, ICODE
C
C
      EQUIVALENCE (INFO(6),VCAL), (INFO(7),ICAL), (INFO(8),TSENS),
     1 (INFO(9),TDIAM), (INFO(10),ATTEN), (INFO(11),RCPK), (INFO(12),GAIN),
     2 (INFO(13),FILTER), (INFO(14),LENGTH), (INFO(38),INTENS),
     3 (INFO(16),X), (INFO(17),Y), (INFO(18),Z), (INFO(20),SHOT),
     4 (INFO(25),ALPHA), (INFO(26),NPTS), (INFO(27),NUSED),
     5 (INFO(28),DEG), (INFO(29),EPS), (INFO(34),TDEPTH), (INFO(35),V),
     6 (INFO(36),CS), (INFO(37),ISUBS), (INFO(39),RNGE)
     7 ,(INFO(40),DESRD), (INFO(22),RMSERR)
     8 ,(INFO(15),SINBK), (INFO(19),CODE), (INFO(21),CSCS)
C/COMEND
      IF(ITYPE.EQ.1)IIDEGR=1
      IF(ITYPE.EQ.2)IIDEGR=MAXDEG
C
      TSG=100.0/(TSENS*GAIN)
      FTOT=0.0
      STOT=0.0
C
      DO 10 I=1,11
C
      TY=FLOAT(I-1)*LENGTH/10.0
      CALL EVAL(TY,IIDEGR,FIT,SLOPE)
C
      TF(I)=FIT
      TT(I)=FIT*TSG
      IF(I.GT.1.AND.I.LT.11)FTOT=FTOT+FIT
      TS(I)=SLOPE
      ALPH(I)=(RCPK/(2.0*ISUBS))*TSG*SLOPE
      IF(I.EQ.6)ALPHA=ALPH(I)
      IF(I.GT.1.AND.I.LT.11)STOT=STOT+SLOPE
      IF(ITYPE.EQ.3)ALPH(I)=ALPH(I)/NDESRD
  10  CONTINUE
C
      MEAN=FTOT/9.0
      MSLOPE=STOT/9.0
C
      RETURN
      END
```

Figure 23 GETDAT

Subroutine BASELN is used to "draw" a baseline for ensuing measurements. That is, a fit of degree one is applied to data taken without ultrasound on, and this data processed to show the magnitude of temperature drifts in the specimen under study. The mean value of the voltage obtained from the thermocouple amplifier output is used to provide the offsetting voltage for measurements in physiological saline. The reasons for this offset were discussed in chapter 2. A listing of BASELN is given in Figure 24. Note that BASELN contains the subroutine OUTDAT as an entry; this subroutine is responsible for the outputting of calculated results for measurements.

There are two input arguments to BASELN, ICHAN and IOUT, both INT*4 variables. The output variables RMEAN and RSLOPE are no longer used. ICHAN is the channel of the digital-to-analog converter which will be set to the average voltage for a one second frame of the thermocouple amplifier output. IOUT is a variable used by OUTDAT to determine the logical unit to which data output is to be directed. At present, this would be either the terminal (logical unit 3) or the printer (logical unit 10).

The program proceeds in two steps: first, the voltage at the output of the thermocouple amplifier is digitized for a period of LENGTH seconds (LENGTH is a common variable). A fit of degree one is performed, and the average value of the

```
      SUBROUTINE BASELN(RMEAN,RMSLPE,ICHAN,IOUT)
C
      LOGICAL DONLY
      INTEGER STEP
C/COMST
      REAL LENGTH,INTENS,ICAL,INFO(40),ISUBS,NPTS,NUSED
C
      INTEGER HEAD(40,6),AINFO(7,6),APTR(40),STATUS
      INTEGER*2 WCUR(40,4),CMDT(40,2)
      COMMON/REFRSH/HEAD(40,6),WCUR(40,4),AINFO(7,6),APTR(40),INFO(40),
     1  STATUS,CMDT(40,2)
C
      REAL TX(600),ADF(600),ARRAY(1245)
      REAL TF(11),TT(11),TS(11),ALPH(11)
      REAL DTF(11),DTS(11),DTR(11)
      REAL MEAN,MSLOPE
      DOUBLE PRECISION R(600)
C
      COMMON/CRUNCH/ TX(600),ADF(600),ARRAY(1245),
     1               TF(11),TT(11),TS(11),ALPH(11),
     2               DTF(11),DTS(11),DTR(11),R(600),
     3               MEAN,MSLOPE
C
      INTEGER SHEND,ASHEND
      COMMON /NINFO/ NRNGE,NNPTS,NNUSED,IDEG,MAXDEG,NDESRD,NSINBK,
     1    SHEND,ASHEND,ICODE
C/COMEND
C
      DONLY=.FALSE.
C
      STEP=1
    1 CONTINUE
      CALL ATD(ADBUF,LENGTH,6000,NOSHOT,ICHAN,SHEND)
      CALL SETUP(6000,200)
      CALL LEAST(200,TX,ADF,W,EPS,MAXDEG,1,ARRAY,R)
      GOTO 10
C
      ENTRY OUTDAT
      DONLY=.TRUE.
   10 CONTINUE
      CALL GETDAT(ICHAN)
      IF(DONLY)GOTO 20
      IF(STEP.EQ.2)CALL WAIT(5,2,ISTAT)
   20 CONTINUE
      IF(IOUT.EQ.3)CALL CLEAR
      IF(ICHAN.EQ.2)GOTO 21
      WRITE(IOUT,34)
   34 FORMAT(/T5,'VOLTAGE',T24,'TEMP',T43,'DEG/SEC',
     1          T60,'TIME')
      DO 4 I=1,11
      TIME=FLOAT(I-1)*LENGTH/10.0
      IF(I.EQ.1.OR.I.EQ.6.OR.I.EQ.7)WRITE(IOUT,35)
   35 FORMAT(' ')
      WRITE(IOUT,36) TF(I),TT(I),TS(I),TIME
   36 FORMAT(3(G15.6,4X),3X,F5.2)
    4 CONTINUE
      WRITE(IOUT,37) MEAN,MSLOPE
   37 FORMAT(/'MEAN VOLTAGE= ',F8.3,2X,'MEAN SLOPE= ',
     1          F8.3)
      IF(DONLY)RETURN
      GOTO 8
   21 CONTINUE
      WRITE(IOUT,31)
   31 FORMAT(/T5,'VOLTAGE',T24,'TEMP',T43,'DEG/SEC',
     1          T60,'ALPHA',T74,'TIME')
      DO 5 I=1,11
      TIME=FLOAT(I-1)*LENGTH/10.0
      IF(I.EQ.1.OR.I.EQ.6.OR.I.EQ.7)WRITE(IOUT,35)
      WRITE(IOUT,32) TF(I),TT(I),TS(I),ALPH(I),TIME
   32 FORMAT(4(G15.6,3X),F5.2)
    5 CONTINUE
      WRITE(IOUT,33) SHEND
   33 FORMAT(/'SHEND= ',I5)
      IF(DONLY)RETURN
    8 CONTINUE
      IF(ICHAN.EQ.2)RETURN
      IF(STEP.EQ.2)GOTO 6
      CALL DTA(1,MEAN)
      CALL BLANK(21,21)
      WRITE(3,30) MEAN
   30 FORMAT('OUTPUTTING VOLTAGE OF ',F8.3)
      CALL BLANK(23,24)
      WRITE(3,38)
   38 FORMAT('RECHECKING NULL....')
      STEP=2
      GOTO 1
    6 CONTINUE
      CALL BLANK (21,21)
      WRITE(3,39)
   39 FORMAT('NULLING PROCEDURE COMPLETE....')
      RETURN
      END
```

Figure 24 BASELN

fit used as a voltage to which the selected channel of the D/A is set. Data is output, and digitization repeated, with a second calculation of the mean voltage to indicate whether the amplifier is maintaining a positive output voltage, a requirement for proper operation of the analog-to-digital converter used in the experiment.

Program steps are communicated to the operator, and indication made when the program is finished. At present, this program is being used to gain experience with automated techniques, and will require some modification to be compatible with self-running programs.

Subroutine AVG takes data from succesive thermocouple responses and adds them together, forming in the end an averaged response. Figure 25 shows a listing of AVG. This will be useful for low-intensity measurements; it should be noted that changes in the main program and the addition of real-time displays forced a temporary halt to work on this routine. Additonal memory is required to run the averaging program, and this will not, unfortunately, be added until after the completion of this thesis. Therefore, it should be kept in mind that modifacations are necessary to provide the operator with the averaging feature.

Initially, the averaging buffer is cleared and a note made of the intensity at which measurements begin - this must remain constant, and a check is made before each sample is added to the buffer to ensure this. The number of the shots to be added together is given by the program variable NSINBK. Successive samples are added to the buffer until NSINBK samples have been recorded; thereafter, attempts to add more samples results in a warning "beep" to the operator. The DC component of each measurement is subtracted before addition to the buffer, so that all signals are assumed to start from zero voltage.

```
      SUBROUTINE AVG
C
      INTEGER SHOOT,NOSHOT,IBELL
C
C/COMST
C
      REAL LENGTH,INTENS,ICAL,INFO(40),ISUBS,NPTS,NUSED
C
      INTEGER HEAD(40,6),AINFO(7,6),APTR(40),STATUS
C
      INTEGER*2 WCUR(40,4),CMDT(40,2)
C
      COMMON/REFRSH/HEAD(40,6),WCUR(40,4),AINFO(7,6),APTR(40),INFO(40),
     1  STATUS,CMDT(40,2)
C
C
      INTEGER*2 ADBUF(9000)
C
      COMMON/ATD/ ADBUF(9000)
C
C
      LOGICAL TSTOUT,CROUT,SLPOUT,TANOUT,CALCD
      LOGICAL SHTOUT,INFOUT,DATOUT,SUMOUT,MAKCHK,AVDONE,CRNCHD
C
      COMMON /CNTRL/ TSTOUT,CROUT,SHTOUT,SLPOUT,TANOUT,CALCD,
     1   INFOUT,DATOUT,SUMOUT,MAKCHK,AVDONE,CRNCHD
C
C
      INTEGER SHEND,ASHEND
C
      COMMON /NINFO/ NRNGE,NNPTS,NNUSED,IDEG,MAXDEG,NDESRD,NSINBK,
     1    SHEND,ASHEND,ICODE
C
C
      EQUIVALENCE (INFO(6),VCAL),(INFO(7),ICAL),(INFO(8),TSENS),
     1 (INFO(9),TDIAM),(INFO(10),ATTEN),(INFO(11),RCPK),(INFO(12),GAIN),
     2 (INFO(13),FILTER),(INFO(14),LENGTH),(INFO(38),INTENS),
     3 (INFO(16),X),(INFO(17),Y),(INFO(18),Z),(INFO(20),SHOT),
     4 (INFO(25),ALPHA),(INFO(26),NPTS),(INFO(27),NUSED),
     5 (INFO(28),DEG),(INFO(29),EPS),(INFO(34),TDEPTH),(INFO(35),V),
     6 (INFO(36),CS),(INFO(37),ISUBS),(INFO(39),RNGE)
     7 ,(INFO(40),DESRD),(INFO(22),RMSERR)
     8  ,(INFO(15),SINBK),(INFO(19),CODE),(INFO(21),CSCS)
C/COMEND
C
      LOGICAL AVDONE
      DATA IBELL/Y'07000000'/,SHOOT/1/,NOSHOT/0/
C
      IF(AVDONE)GOTO 7
      GOTO 2
C
      ENTRY INITAV
C
      NSINBK=0
      SINBK=NSINBK
      SINTNS=INTENS
      AVDONE=.FALSE.
      ASHEND=0
C
      DO 1 I=1,NNPTS
      AVBUF(I)=0
    1 CONTINUE
C
    2 CONTINUE
C
      NSINBK=NSINBK+1
      SINBK=NSINBK
      CALL REFR(15,15)
C
      IF(INTENS.EQ.SINTNS)GOTO 4
      WRITE(3,3)IBELL
    3 FORMAT('CHANGE OF INTENSITY DURING',
     1  ' AVERAGING - SHOT ABORTED',A1)
      RETURN
C
    4 CONTINUE
C
      CALL BASELN(MEAN,MSLOPE,2,3)
      IIORG=MEAN*1023.0/10.2375
C
      CALL ATD(ADBUF,LENGTH,NNPTS,SHOOT,2,SHEND)
C
      DO 5 I=1,NNPTS
      AVBUF(I)=AVBUF(I)+ADBUF(I)-IIORG
    5 CONTINUE
C
      IF(NSINBK.NE.NDESRD)RETURN
C
C
    7 CONTINUE
      CALL BLANK(19,19)
      WRITE(3,8)IBELL
    8 FORMAT('AVERAGING COMPLETE - READY',
     1  ' FOR CRUNCH',A1)
C
      AVDONE=.TRUE.
      ASHEND=ASHEND/FLOAT(NDESRD)
C
C
      RETURN
      END
```

Figure 25 AVG

Subroutine REFR controls the outputting of the variables used in the absorption program to the screen of the terminal used by the operator. In order to facilitate changing of the format of the screen, actual placement of variables, and their associated headings, is controlled by information stored in arrays. This allows changing the format of the screen without recompiling the program. The information in the arrays is read from files on the disk by the main program. A listing of REFR can be found in Figure 26.

There are two input arguments (K and L) to REFR, and no output arguments. A DO loop is executed, for I=K,L; the following sequence is executed within the DO loop. The INTEGER*2 variable is assigned to the current value of the DO parameter I. The following information is obtained from row "I2" of array WCUR:

column 1 - line number on screen where info is found.

column 2 - column position of beginning of the heading for the information. (1-80)

column 3 - column position of the information to be printed.

column 4 - type of format used.

The types of formats available are, with their one-digit format numbers:

1 - heading only
2 - alphanumeric information (24 characters)
3 - heading + information in F8.3 format
4 - heading + information in G15.7 format

```
      SUBROUTINE REFR(K,L)
      INTEGER DDATE(3),ITME(2),TBUF(33)
      INTEGER CCURH,CCURI
C/COMST
C
      REAL LENGTH,INTENS,ICAL,INFO(40),ISUBS,NPTS,NUSED
C
      INTEGER HEAD(40,6),AINFO(7,6),APTR(40),STATUS
C
      INTEGER*2 WCUR(40,4),CMDT(40,2)
C
      COMMON/REFRSH/HEAD(40,6),WCUR(40,4),AINFO(7,6),APTR(40),INFO(40),
     1  STATUS,CMDT(40,2)
C
C
      INTEGER PBLK(5),PBLK1(5),LU,CRTYPE
C
      INTEGER*2 CMD(40)
C
      COMMON /KEYBD/ PBLK(5),PBLK1(5),CMD(40),LU,CRTYPE
C
      COMMON /NINFO/ NRNGE,NNPTS,NNUSED,IDEG,MAXDEG,NDESRD,NSINBK,
     1    SHEND,ASHEND,ICODE
C
C
      EQUIVALENCE (INFO(6),VCAL),(INFO(7),ICAL),(INFO(8),TSENS),
     1 (INFO(9),TDIAM),(INFO(10),ATTEN),(INFO(11),RCPK),(INFO(12),GAIN),
     2 (INFO(13),FILTER),(INFO(14),LENGTH),(INFO(38),INTENS),
     3 (INFO(16),X),(INFO(17),Y),(INFO(18),Z),(INFO(20),SHOT),
     4 (INFO(25),ALPHA),(INFO(26),NPTS),(INFO(27),NUSED),
     5 (INFO(28),DEG),(INFO(29),EPS),(INFO(34),TDEPTH),(INFO(35),V),
     6 (INFO(36),CS),(INFO(37),ISUBS),(INFO(39),RNGE)
     7 ,(INFO(40),DESRD),(INFO(22),RMSERR)
     8  ,(INFO(15),SINBK),(INFO(19),CODE),(INFO(21),CSCS)
C/COMEND
      DATA LDIN/Y'7E110000'/
      IF(K.LT.1)K=1
      IF(L.GT.40)L=40
C
      CALL DATE(DDATE)
      ENCODE(TBUF,101) DDATE(2),DDATE(3),DDATE(1)
 101  FORMAT(2(I2,'/'),I2)
      AINFO(6,1)=TBUF(1)
      AINFO(6,2)=TBUF(2)
C
      CALL ICLOCK(1,ITME)
      AINFO(7,1)=ITME(1)
      AINFO(7,2)=ITME(2)
      DO 12 I=K,L
      I2=I
      ICOL=(WCUR(I2,2)-1)*256
      CCURH=LDIN+ICOL+(WCUR(I2,1)-1)
      ICOL=(WCUR(I2,3)-1)*256
      CCURI=LDIN+ICOL+(WCUR(I2,1)-1)
      LCUR=Y'0'
      L22=Y'0'
      IFPTR=WCUR(I2,4)
      GOTO(40,41,42,43,44,12),IFPTR
 40   WRITE(3,950) LCUR,CCURH,(HEAD(I2,J),J=1,3),L22
 950  FORMAT(2A4,3A4,A4)
      GOTO 12
 41   IPTR=APTR(I2)
      WRITE(3,951) LCUR,CCURH,(HEAD(I2,J),J=1,6),CCURI,
     1   (AINFO(IPTR,KK),KK=1,6),L22
 951  FORMAT(2A4,6A4,A4,6A4,A3)
      GOTO 12
 42   WRITE(3,952) LCUR,CCURH,(HEAD(I2,J),J=1,6),CCURI,
     1    INFO(I2),L22
 952  FORMAT(2A4,6A4,A4,F8.3,A3)
      GOTO 12
 43   WRITE(3,953) LCUR,CCURH,(HEAD(I2,J),J=1,6),CCURI,
     1    INFO(I2),L22
 953  FORMAT(2A4,6A4,A4,F12.3,A3)
      GOTO 12
 44   IINFO=INFO(I2)
      WRITE(3,954) LCUR,CCURH,(HEAD(I2,J),J=1,6),CCURI,
     1    IINFO,L22
 954  FORMAT(2A4,6A4,A4,I5,A3)
 12   CONTINUE
      RETURN
      END
```

Figure 26 REFR

5 - heading + information in I5 format
6 - this code suppresses all action for this variable

The method used for cursor addressing with the Hazeltine Model 1500 Video Display terminal which provides operator control of experiments is discussed in the reference manual for the terminal, Hazeltine publication number HI-1056A. To position the cursor, a "lead-in" code (hex "7E") is output to the terminal. This signals the terminal that a special function follows; if the next character is a hex "11", cursor addressing is indicated. After these two bytes have been output, two more bytes are sent to the terminal, these being the bytes controlling the column and line where the cursor is to be positioned, respectively. The actual byte sent for either the column or the line is the binary equivalent less one of the desired position.

The sequences for the line and columns desired are computed from row "I2" of WCUR. A branch is then taken via a computed GOTO statemnent to a WRITE statement which will output the infromation relating to the variable "I2" in the format specified by column four of the entry "I2" in WCUR. Note that a variable need not exist; it is possible to output a heading alone (format 1). Note also that since a heading is output before any information, trailing blanks in the 24 character heading serve to erase previously written information before the updated infomation is output. For

alphanumeric infomation (format 2), a row of the array AINFO is output (24 characters); the particular row used is obtained from the "I2" entry of the INTEGER*2 vector APTR.

Subroutine PRNTSC outputs to the printer a copy of the screen of the terminal used by the operator. A listing for PRNTSC can be found in Figure 27. There are no input or output arguments for PRNTSC, but the COMMON block named REFRSH is used, as in the code for subroutine REFR.

The screen of the operator terminal consists of twenty-four lines, each containing eighty characters. As in REFR, the placement of headings and program information is controlled by the data contained in the arrays HEAD, WCUR, and APTR. Since the printer outputs on a line-by-line basis, a search is made to determine if any information should be included in the line being output; the process is repeated for all twenty-four lines.

Recall that line numbers for information and/or headings are contained in column one of the array WCUR; each time a new line is output, column one of all rows of WCUR is checked to see if it agrees with the line number being output. If there is agreement, the follwing takes place: In a manner which directly corresponds to the method used in REFR, a computed GOTO is executed, using column four of WCUR as its argument. Column four of WCUR contains a one digit format code as for REFR; the branch is taken to statements which handle each code separately. The action of these statements for a given format code is based on the FORTRAN feature ENCODE. ENCODE provides a means of data manipulation under

```
$ARGC
      SUBROUTINE PRNTSC
      INTEGER LINE(33),FMT(33)
C/COMST
C
      REAL LENGTH,INTENS,ICAL,INFO(40),ISUBS,NPTS,NUSED
C
      INTEGER HEAD(40,6),AINFO(7,6),APTR(40),STATUS
C
      INTEGER*2 WCUR(40,4),CMDT(40,2)
C
      COMMON/REFRSH/HEAD(40,6),WCUR(40,4),AINFO(7,6),APTR(40),INFO(40),
     1  STATUS,CMDT(40,2)
C
C/COMEND
      WRITE(10,907)
  907 FORMAT(/)
      DO 5 J=1,24
C
      DO 10 K=1,33
   10 LINE(K)=Y'20202020'
C
      DO 12 I=1,40
C
      I2=I
      LCUR=WCUR(I2,1)
      IF(LCUR.NE.J)GOTO 12
C
      CCURH=WCUR(I2,2)
      CCURI=WCUR(I2,3)
      IFPTR=WCUR(I2,4)
C
      GOTO(40,41,42,43,44,12),IFPTR
C
   40 CONTINUE
      ENCODE(FMT,100) CCURH
  100 FORMAT('(T',I2,',3A4)')
      ENCODE(LINE,FMT) (HEAD(I2,M),M=1,3)
      GOTO 12
C
   41 CONTINUE
      IPTR=APTR(I2)
      ENCODE(FMT,110) CCURH,CCURI
  110 FORMAT('(T',I2,',6A4,T',I2,',6A4)')
      ENCODE(LINE,FMT) (HEAD(I2,M),M=1,6),(AINFO(IPTR,M),M=1,6)
      GOTO 12
C
   42 CONTINUE
      ENCODE(FMT,120) CCURH,CCURI
  120 FORMAT('(T',I2,',6A4,T',I2,',F8.3)')
      ENCODE(LINE,FMT) (HEAD(I2,M),M=1,6),INFO(I2)
      GOTO 12
C
   43 CONTINUE
      ENCODE(FMT,130) CCURH,CCURI
  130 FORMAT('(T',I2,',6A4,T',I2,',G15.7)')
      ENCODE(LINE,FMT) (HEAD(I2,M),M=1,6),INFO(I2)
      GOTO 12
C
   44 CONTINUE
      IINFO=INFO(I2)
      ENCODE(FMT,140) CCURH,CCURI
  140 FORMAT('(T',I2,',6A4,T',I2,',I8)')
      ENCODE(LINE,FMT) (HEAD(I2,M),M=1,6),IINFO
C
   12 CONTINUE
C
      WRITE(10,150) (LINE(I),I=1,20)
  150 FORMAT(20A4)
C
    5 CONTINUE
      RETURN
      END
```

Figure 27 PRNTSC

Subroutine BLANK clears selected lines from the screen. A listing for BLANK can be found in Figure 28. There are two input arguments to BLANK (K and L, both INT*4), and output is in the form of action taken to the screen. A DO loop is executed for I=K,L and lines of the screen between line K and line L are erased. If K=L, one line is erased. The cursor is then positioned at line K, so that subsequent output via WRITE statements will fill the blanked portion of the screen.

The blanking of a line is effected by positioning the cursor at the beginning of the desired line, and outputting the hex sequence "7E0F0C". Cursor positioning is as for REFR. After the DO loop is executed, a single cursor position command is issued to place the cursor at the beginning of the portion of the screen just blanked (i.e, at line K).

Subroutine CLEAR clears the screen of all characters. A listing for CLEAR is also found in Figure 27. The screen is cleared by outputting the hex sequence "7E1C". This command also returns the cursor to the "home" position, i.e., column 1 of line 1.

Subroutine IBELL outputs an audible signal to the operator. A listing of IBELL can be found in Figure 27. The standard ASCII character for "BELL", i.e., hex "07" is output to the terminal.

```
      SUBROUTINE BLANK(J,K)
      INTEGER BUF(3),PBLK(5)
      EQUIVALENCE (BUF(1),LNE),(BUF(2),ICLR),(BUF(3),L22)
      EQUIVALENCE (BUF(4),ICLR2)
      DATA LDIN/Y'7E110000'/
      DATA ICLR/Y'007E0F0C'/
      DO 1 I=J,K
      LINE=I
      LNE=LDIN+0+(LINE-1)
      CALL SYSIO(PBLK,Y'29',3,LNE,8,0)
 1    CONTINUE
      GOTO 2
      ENTRY CPOS
 2    CONTINUE
      LNE=LDIN+0+J-1
      CALL SYSIO(PBLK,Y'29',3,LNE,8,0)
      RETURN
      END
C
C
      SUBROUTINE CLEAR
      DATA ICLR/Y'7E1C0000'/
      WRITE(3,900) ICLR
 900  FORMAT(A4)
      RETURN
      END
C
      SUBROUTINE IBELL
      DATA IBELL/Y'07000000'/
      WRITE(3,100)IBELL
 100  FORMAT(A1)
      RETURN
      END
C
C
      SUBROUTINE WONKB
C
      INTEGER PBLK(5),PBLK1(5),LU
      INTEGER BUF(6)
      INTEGER*2 CMD(40)
C
      COMMON /KEYBD/ PBLK(5),PBLK1(5),CMD(40),LU
C
      DATA (BUF(I),I=1,5)/'  << ENTER COMMAND'/
      DATA BUF(6)/Y'0D070000'/
C
      CALL BLANK(23,24)
      CALL SYSIO(PBLK1,Y'29',3,BUF(1),24,0)
      CALL WAIT(500,1,ISTAT)
      PBLK(1)=-1
      ILU=5
      NBYTES=1
      CALL SYSIO(PBLK,Y'42',ILU,CMD,NBYTES,0)
      RETURN
      ENTRY KBCA
      CALL SYSIO(PBLK1,Y'80',5,CMD,1,0)
      RETURN
      END
```

Figure 28 BLANK, CLEAR, IBELL and WONKB

Subroutine WONKB provides a means of continuously updating the time on the screen, while waiting for a command input. This is not possible using standard FORTRAN READ and Write statements. The reason for this is that the I/O calls generated at the system level by READ and WRITE are of the "halt I/O" type. This means that program execution is delayed until the I/O operation requested is completed; no further input or output is possible with the device just specified for the I/O. It should be noted that while a constant refreshing of the time on the screen is not necessary for the proper execution of the program, the code described below is an adaptation of the code used to provide the emergency shut-off of equipment, and was obtained with little expense in programming time. A listing for WONKB can be found in Figure 27.

In operation, a prompt is issued to the screen for a command via a standard WRITE statement. A call to the real-time subroutine SYSIO then issued. SYSIO provides a FORTRAN user with direct access to operating-system level I/O calls. A discussion of SYSIO can be found in Perkin-Elmer publication number 29-564R01, "FORTRAN VI 32-BIT RUN-TIME LIBRARY REAL TIME EXTENSIONS", pp. 3-28 - 3-29. A discussion of the procedures involved in system-level I/O operations can be found in Perkin-Elmer publication number S29-613R03, ",OS/32 Programmer Reference Manual", ch.5, sect. 1-3, pp. 5-1 - 5-26.

For the purposes of this discussion, the following should suffice: A read request is issued to logical unit 3 with the "proceed I/O" bit of the function code for the request set (This is bit six). As a result, after the read request is issued, program execution continues with the next statement. A short wait is counted out before RETURNing to allow the operating system time to set up the read. After a return to the main program, the status of the read request can be obtained by checking the contents of the parameter block against Y'42030000'; when the first word of the parameter block is equal to this, a key has been pressed, and the character from the keyboard is the first byte of the array CMD. The normal use of WONKB would thus be to call WONKB, then to attend to whatever functions are desired, periodically checking the status of the request to determine whether command action should be taken.

ENTRY KBCA provides a means of cancelling the read request, necessary if one desires to update the terminal screen with information. This is so because the operating system does not support a division of a logical unit into separate read/write units.

the control standard FORMAT statements, with the output of the formatter loaded into a user-specified buffer. A discussion of ENCODE can be found in Perkin-Elmer publication number B29-540R02, "FORTRAN VI Reference Manual", pp. 50-51. For each piece of information determined to lie in the line being output, ENCODE is called twice; the first call forms a FORMAT statement in memory, and the second call uses this FORMAT statement to effect the positioning of the information and/or heading in a buffer. After all the necessary information has been placed in the buffer, the buffer is output to the printer. The "T" format (tab) is used in conjunction with the two digit numbers contained in columns 3 and 4 of the array WCUR to control the placement of the heading and information, respectively. The format and content of the information and/or heading associated with a row of WCUR is the same as in REFR, the only difference being the use of the "T" format to effect the positioning of characters in the output (as opposed to the cursor addressing which is a part of REFR). When all twenty-four lines have been output to the printer, PRNTSC executes a RETURN.

Subroutine PLTDOT is used to provide an easy means for plotting on the Centronix line printer which is a peripheral to the 7/32. A listing for PLTDOT can be found in Figure 29. Single FORTRAN calls to the ENTRYs CLEARL, SETDOT and OUTLIN will clear the buffer used for the current line being output, set a given dot in the line to be output, and actually print the output line, respectively. Note that PLTDOT itself is never called; for ease of use, the entries have been labeled with abbreviations describing their function; PLTDOT serves to tell users of the 7/32 library of programs what this group of subprograms does.

The plotting mode of the Printronix 300 line printer is described in two Printronix publications, "Applications Manual" (part number 102487), p. 7, and Application Note 102370, "Using the Plot Mode". In essence, the use of the byte X'05' anywhere in the 133-byte buffer being output by the printer causes the printer to print a single line of dots. The dots to be printed are selected by certain bits in the output buffer; this is described below. Normally, the printer advances 9 dot positions during the printing of one line; ASCII characters are printed in 7x9 dot matrices using ROM-generated information.

In the plot mode, each byte in the output buffer has the structure shown in Figure 30. Bit 1 of a byte must be set to indicate to the printer that this byte is to be interpreted

```
      SUBROUTINE PLTDOT(LINE,IDOT)
      INTEGER LINE(33)
      ENTRY CLEARL
      DO 5 I=1,33
5     LINE(I)=Y'40404040'
      RETURN
      ENTRY SETDOT
      IF(IDOT.LT.1.OR.IDOT.GT.750)IDOT=760
      IWD=IDOT/24+1
      IBIT=MOD(IDOT,24)
      IF(IBIT.NE.0) GOTO 6
      IBIT=26
      IWD=IWD-1
      GOTO 8
6     IMOD=MOD(IBIT,6)
      IBYTEE=IBIT/6
      IF(IMOD.EQ.0) IBYTEE=IBYTEE-1
      IBIT=8*IBYTEE+2+MOD(6-IMOD,6)
8     CALL BSET(LINE(IWD),IBIT)
      RETURN
      ENTRY OUTLIN
      IX05=Y'05000000'
      WRITE(10,100) (LINE(J),J=1,32),IX05
100   FORMAT(32A4,A2)
      RETURN
      END
$ARGC
```

Figure 29 PLTDOT

as plot-mode data. Bit 0 is left reset; it has, however, no effect on the output. Each byte controls the printing of six dot positions in the line being output. In all, 792 dot positions exist; some difficulty was experienced in programming for all 792 dots, hence, the number of positions accessible through SETDOT is limited to 768.

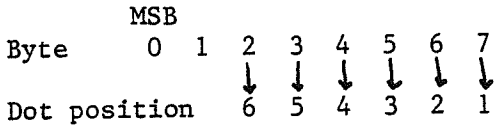


Figure 30 - Structure of byte for plot mode output.

The call to CLEARL is of the form

```
CALL CLEARL(LINE)
```

where LINE represents an array of 33 fullwords to be used as a buffer by SETDOT and OUTLIN. CLEARL sets all bytes of the buffer to X'40', bit 1 being set, as required for plotting. This initializes the buffer; subsequent calls to SETDOT will set the bits controlling desired dot positions to be plotted.

Calls to SETDOT are of the form

```
CALL SETDOT(LINE,IDOT)
```

where LINE is the same buffer used in the call to CLEARL, and IDOT is an INTEGER *4 variable whose value gives the position of the dot which the user wants plotted. CLEARL is called once prior to plotting a line, but SETDOT may be called as many times as desired; more than one dot may be plotted in a given line.

Modular arithmetic is used to compute to position of the bit which controls the printing of the dot numbered IDOT. A FORTRAN language extension, subroutine BSET, is used. BSET is called in the form

```
CALL BSET(WORD,BIT)
```

where WORD is the variable to be modified, and BIT specifies the bit to be set to a binary 1 by this call. Bits are numbered from left to right beginning with bit 0. For an explanation of BSET, refer to Perkin-Elmer publication 29-563R01, "FORTRAN VI 32-Bit Run Time Library Language Extensions", pp. 8-9.

The arithmetic to determine the particular word and bit controlling the printing of the desired dot position follows: The word in which the bit position controlling the desired dot is obtained by the INTEGER calculation

```
IWD=IDOT/24+1
```

The division by 24 stems from the facts that each word has four bytes, and each byte controls six dot positions. Addition of one to IDOT/24 assures that the proper element of

the array LINE will be addressed. This formula correctly determines that element, unless IDOT is an integer multiple of 24. In that case, one must be subtracted from IWD, since the integer calculation IDOT/24, without the addition of one, is sufficient to identify the element of LINE controlling position IDOT. Note here that the bit position corresponding to the 24th dot (i.e., the rightmost in a group of 24) is bit 26,as shown in Figure 31.

Bit	0...7	8...15	16...23	24...31
Dots	6...1	12...7	18...13	24...19

Figure 31 - String of bytes to be output in plot mode.

Thus, in the case where IDOT is an integer multiple of 24, 1 is subtracted from IWD and 26 is assigned to IBIT; BSET is then called using the form

```
CALL BSET(LINE(IWD),IBIT)
```

This sets the proper bit to print dot IDOT in this case.

For the case where IDOT is not an integer multiple of 24, the following applies to the calculation of IBIT: (Remember that IWD has, in this case, been correctly

determined.) IBIT1 is set equal to IDOT mod 24, i.e., the remainder when IDOT is divided by 24. IBYTEE is then set to equal to the INTEGER result of IBIT1/6. IBYTEE is then equal to the number of bytes in the element of LINE given by IWD which must be skipped to find the byte in which the bit controlling the dot position given by IDOT is found. The sole exception to this rule occurs when IDOT is an integer multiple of six. In that case, one must be subtracted from IBYTEE to give the correct number of bytes to be skipped; the situation is analogous to that for the calculation of IWD, and is a consequence of INTEGER arithmetic rules. IBIT is calculated using the equation

IBIT=8*IBYTEE+2+MOD(6-IMOD,6)

The first term, 8*IBYTEE, adds 8 bits to the count of the bit position, for every full group of six dot positions contained in IDOT mod 24. This effectively skips the bits controlling the groups mentioned. Two is added to this because bit positions 0 and 1 of a given byte are always binary 0 and 1, respectively, and must also be skipped. The last term, MOD(6-IMOD,6), produces the result 6-IMOD, unless IMOD is zero (which only occurs when IDOT is an integer multiple of six.). This is done because bit 7 (the rightmost bit) in a byte controls the leftmost dot in the group of dots under control of the byte. If IMOD is zero, the last term in the equation for IBIT is also zero; bit number 2 of the selected byte will be set, forcing the printing of the rightmost (sixth) dot in the group controlled by this byte.

When all of the dot locations have been set up by calls to SETDOT, OUTLIN is called in the form

```
CALL OUTLIN(LINE)
```

to output the contents of the buffer LINE, with the special plot mode character, X'05', apppended to the buffer. This causes the plotting of the current line.

In general, a call to OUTLIN would be followed by a call to CLEARL, to ready the buffer for printing a new line; in certain circumstances, however, this would not be the case (for example, the plotting of hash marks usually involves outputting duplicate lines), and thus the two functions have been left seperate.

It is good practice to output a blank line when finished with a plot, as otherwise the last line plotted might intefere with ASCII printing in the normal mode (See p. 11 of the Application Note).

Subroutine TOF causes the printer to eject to top-of-form. A listing for TOF is included in Figure 32. A standard FORTRAN format convention is used to accomplish this; FORMAT(1H1) is used in a WRITE statement.

Subroutine HASHMK outputs a series of hashmarks using the plot mode of the line printer. A listing for HASHMK can be found in Figure 32. There are no input or output arguments for HASHMK. Subroutine TOF is called to eject to top-of-form on the printer, then a heading is printed using ASCII characters, and next, eleven values of temperature (from 0.0 to 1.0 by tenths) are output as characters, with spacing controlled to place the values as close as possible to their corresponding hashmarks. The origin (0.0) is placed near dot position 150 (see section on plotting subs); the temperature measurement upon which this system is based is a differential measurement, and drifts occuring in the temperature bath containing the object under study sometimes give rise to "negative" relative temperatures. The shifting of the origin allows the printing of these (usually minor) drifts.

The hashmarks are then printed in the following manner: Twenty-one lines are printed in the plot mode. In the first fourteen lines, a dot is printed every 50 dot positions, from dot position 50 to dot position 750 ("major" hashmarks). In the next six lines, the same dot positions are printed and in

```
      SUBROUTINE TOF
      WRITE(10,100)
 100  FORMAT(1H1)
      RETURN
      END
C
$ARGC
      SUBROUTINE HASHMK
      REAL H(11)
      INTEGER LINE(33)
C
      DO 2000 I=1,11
      H(I)=FLOAT(I-1)*0.1
 2000 CONTINUE
      WRITE(10,2010) (H(I),I=1,11)
 2010 FORMAT(T22,6(F5.1,3X,F5.1,4X))
      CALL CLEARL(LINE)
      CALL OUTLIN(LINE)
      DO 2020 I=1,20
      IF(I.LT.14)GOTO 2030
      DO 2040 J=50,750,10
      CALL SETDOT(LINE,J)
 2040 CONTINUE
 2030 DO 2050 K=50,750,50
      CALL SETDOT(LINE,K)
 2050 CONTINUE
      CALL OUTLIN(LINE)
      CALL CLEARL(LINE)
 2020 CONTINUE
      DO 2060 I=50,750
      CALL SETDOT(LINE,I)
 2060 CONTINUE
      CALL OUTLIN(LINE)
      CALL CLEARL(LINE)
      CALL OUTLIN(LINE)
      RETURN
      END
```

Figure 32 TOF and HASHMK

addition, every tenth dot position as well ("minor" hashmarks). In the next line of output, all dot positions from 50 to 150 are printed.

A blank line is then output in the plot mode, to avoid possible interference with ASCII characters (see last line of section on PLTDOT). The "major" hashmarks (every 50th dot position) thus correspond to one tenth of a degree Celsius; proper scaling must be effected in a program using these hashmarks, to ensure accuracy of representation.

Subroutine PLTSHT outputs a series of hashmarks using HASHMK and follows with a plot of the curves currently being displayed on the oscilloscope. Figure 33 shows a listing of PLTSHT. There are two input arguments to PLLTSHT; the first is ignored at present. The second, IORG (INT*4) is used to prevent any DC shift in the output of the thermocouple amplifier from being plotted. Output takes place on the line printer.

The same buffers used for display by subroutine DTASWP are used by PLTSHT; the logical variables SHTOUT, CROUT, SLPOUT and TANOUT generated by CALC are used to determine which curves will be plotted. During a plot, the variable IORG is subtracted from the contents of each buffer to shift all curves to an origin of 150, for compatibility with HASHMK and to eliminate the effects of DC shifts on the output.

```
C
$ARGC
      SUBROUTINE PLTSHT(NSPACE, IORG)
      INTEGER*2 IORG
      LOGICAL ATDPLT
      INTEGER LINE(33)
C/COMST
      INTEGER*2 ADBUF(9000)
C
      COMMON/ATD/ ADBUF(9000)
C
C
      LOGICAL TSTOUT, CROUT, SLPOUT, TANOUT, CALCD
      LOGICAL SHTOUT, INFOUT, DATOUT, SUMOUT, MAKCHK, AVDONE, CRNCHD
C
      COMMON /CNTRL/ TSTOUT, CROUT, SHTOUT, SLPOUT, TANOUT, CALCD,
     1    INFOUT, DATOUT, SUMOUT, MAKCHK, AVDONE, CRNCHD
C
C
C
C
C
      INTEGER*2 ISBUF(600), ICRBUF(600), ISLBUF(600)
      INTEGER*2 TANBUF(600)
      COMMON /SCOP/ ISBUF(600), ICRBUF(600), ISLBUF(600), TANBUF(600)
C
      INTEGER SHEND, ASHEND
C
      COMMON /NINFO/ NRNGE, NNPTS, NNUSED, IDEG, MAXDEG, NDESRD, NSINBK,
     1     SHEND, ASHEND, ICODE
C
C/COMEND
      ATDPLT=.FALSE.
      GOTO 7
      ENTRY PLTATD
      ATDPLT=.TRUE.
 7    CONTINUE
      CALL HASHMK
      DO 10 I=1,600
      CALL CLEARL(LINE)
      ITEM1=ISBUF(I)/16+150-IORG
      IF(ATDPLT)ITEM1=ADBUF(I)+150-IORG
      ITEM2=ICRBUF(I)/16+150-IORG
      IF(I.GT.SHEND/15)ITEM2=150
      ITEM3=ISLBUF(I)/16+150-IORG
      IF(I.GT.SHEND/15)ITEM3=150
      ITEM4=TANBUF(I)/16+150-IORG
      ITEM1=IAND(ITEM1,Y'3FF')
      ITEM2=IAND(ITEM2,Y'3FF')
      ITEM3=IAND(ITEM3,Y'3FF')
      ITEM4=IAND(ITEM4,Y'3FF')
      IF(ATDPLT)GOTO 101
      IF(SHTOUT)CALL SETDOT(LINE,ITEM1)
      IF(CROUT)CALL SETDOT(LINE,ITEM2)
      IF(SLPOUT)CALL SETDOT(LINE,ITEM3)
      IF(TANOUT)CALL SETDOT(LINE,ITEM4)
      GOTO 102
X     WRITE(6,9099)ITEM1,ITEM2,ITEM3,ITEM4
X9099 FORMAT(4I8)
 101  CONTINUE
      CALL SETDOT(LINE,ITEM1)
 102  CONTINUE
      CALL SETDOT(LINE,150)
      CALL OUTLIN(LINE)
 10   CONTINUE
      CALL CLEARL(LINE)
      CALL OUTLIN(LINE)
      RETURN
      END
```

Figure 33 PLTSHT