ornl

NUREG/CR-5061
ORNL/TM-10663

OAK RIDGE
NATIONAL
LABORATORY

MARTIN MARIETTA

Three-Frequency Eddy-Current Instrument

C. V. Dodd
L. D. Chitwood

Prepared for the
U.S. Nuclear Regulatory Commission
Office of Nuclear Regulatory Research
Under Interagency Agreement DOE 40-551-75

OPERATED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

Available from

Superintendent of Documents
U.S. Government Printing Office
Post Office Box 37082
Washington, D.C. 20013-7982

and

National Technical Information Service
Springfield, VA 22161

NUREG/CR-5061
ORNL/TM-10663
Distribution
Category RF

Metals and Ceramics Division

THREE-FREQUENCY EDDY-CURRENT INSTRUMENT

C. V. Dodd and L. D. Chitwood

Manuscript Completed - January 1988

Date Published - May 1988

Notice: This document contains information of a preliminary nature. It is subject to revision or correction and therefore does not represent a final report.

Prepared for the
U.S. Nuclear Regulatory Commission
Office of Nuclear Regulatory Research
Washington, DC 20555
Under Interagency Agreement DOE 40-551-75

NRC FIN No. B0417

Prepared by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831
operated by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under Contract DE-AC05-84OR21400

CONTENTS

LIST OF FIGURES

THREE-FREQUENCY EDDY-CURRENT INSTRUMENT

C. V. Dodd and L. D. Chitwood

ABSTRACT

A three-frequency eddy-current instrument has been constructed for general multiple property applications, with particular emphasis on light water reactor steam generator tubing examination. A description is given of the overall operating principles of the complete instrument and of the operation of the different modules in the instrument. Also included are wiring and printed circuit diagrams and the necessary computer programs to run the instrument.

INTRODUCTION

Eddy-current measurements are affected by a large number of variations in the properties of a given test, such as conductivity, permeability, distance between the probe and the test specimen, shape of the specimen, the thickness or cladding thickness of the specimen, and defects in the specimen. This sensitivity to so many property variations allows us to use eddy currents for a number of different types of measurements but requires that we eliminate the effect that variations in the other properties have on the variation of the particular property that we wish to measure. In general, this requires that we have as many independent instrument readings as we have test property variations, and we can get independent readings from either multiple frequency or pulsed eddy-current tests. We can get two independent readings (magnitude and phase) at each frequency, or one independent reading at each time interval reading along a pulse. This instrument has been successfully applied to several multiple property tests[1] and the theory and design procedures[2] will not be repeated here. The computer programs are given in the appendices for completeness and because of changes in the programs, computers, and positioners used for earlier versions.

This instrument is an updated version of one originally developed for liquid metal reactor steam generator tubing examination,[3] but it includes an IEEE-488 interface bus and a module to multiplex an array probe with up to 16 inspection coils. The programming of the instrument has been changed so that the data analysis and storage functions have been shifted back to the host computer, while the internal instrument computer is used only for data gathering and transmission. The data analyzing, manipulation, and storage are now done using a high-level language (FORTRAN) which speeds and simplifies this process considerably. The eddy-current instrument, the control computer, and the mechanical positioners all use the standard IEEE-488 bus which simplifies the interfacing to new positioners and controllers.

INSTRUMENTATION

The three-frequency eddy-current instrument generates three discrete frequencies, mixes them together, sends them to an eddy-current probe, separates and amplifies the detected frequencies after modulation by a specimen, and then generates voltages proportional to the magnitude and phase of each frequency. The six voltages are then converted to digital signals and sent to the controlling computer for further analysis.

OVERALL BLOCK DIAGRAM

A block diagram of the instrument is shown in Fig. 1; from the block diagram, we can see that three separate and independent oscillators are used to generate the three frequencies. The signal from each oscillator is mixed with the others through summing resistors, where the amplitude of each frequency can be controlled independently. When driving eddy-current coils, which are inductive circuits, the high frequencies are often transmitted with less attenuation than the lower frequencies. Therefore, to keep the higher frequencies from saturating the receiving circuits, their amplitude is decreased by adjusting the current summing resistors at the mixer. A reference signal from each oscillator is also fed to the phase detectors.

ORNL-DWG 78-16168

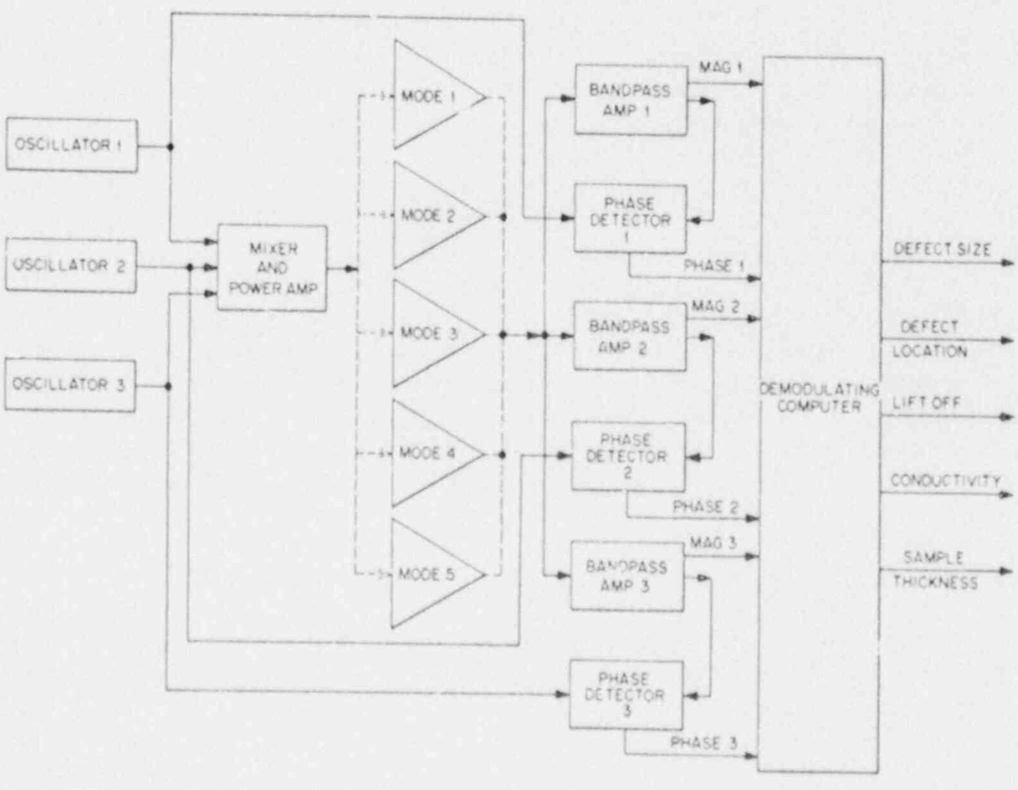


Fig. 1. Block diagram of a three-frequency eddy-current instrument.

The mixed signal is then amplified by a power amplifier and fed to either the probe or calibrator, depending on the mode of operation that is selected. The instrument can be operated in five different modes. These are a reflection mode, a through-transmission mode, an absolute mode, a differential mode, and a calibration mode. The calibration mode can be selected remotely by the controlling computer, so that the calibration can be frequently and automatically checked during the tests. The other four modes refer to the connection of the coils and the arrangement of the coils with respect to the conductor. In addition, the instrument can be operated with a multiplexer that will drive an array with up to 16 coils. The signal from the probe or calibrator is then fed in parallel to three bandpass amplifiers, with each amplifier tuned to the frequency of one of the oscillators. Each bandpass amplifier passes only the tuned frequency to a phase detector, and also generates a dc voltage proportional to the output amplitude of the tuned frequency. Each phase detector measures the phase shift between the reference signal, from one oscillator and the signal through the corresponding bandpass amplifier, and generates a dc voltage proportional to the phase shift. The dc voltages are fed to integrating analog-to-digital converters, which are read by the instrument's computer. The instrument's computer controls the readings, the calibrator, and the multiplexer (if one is used). It in turn is controlled by and transmits its data to the laboratory computer over the IEEE-488 bus (also known as the HPIB bus, the GPIB bus, and the IEC 625 bus). The laboratory computer takes the magnitude and phase data and computes the properties of interest from these data. It will also display the data and can store it for future use. The laboratory computer can be any computer that has this particular bus, such as the IBM 9000, the IBM PC/AT, or the HP 9000. However, it is desirable to have a good FORTRAN 77 language on the computer since the programs are presently written in this language. Also, a real-time operating system is necessary to avoid the loss of data during system interrupts. The programs in this report are for a IBM PC/AT and use Ryan-McFarland FORTRAN linked with an assembly level program, GETKEY, to handle the keyboard I/O.

We will now discuss the operation of the instrument in the different modes. The mode changes may require changes in the connections of the probes, changes in the software commands, some changes in the instrument modules, or a combination of the above.

Absolute Mode of Operation

The absolute mode of operation is diagrammed in Fig. 2. It consists of a single coil with a series dropping resistor. The impedance of the coil determines the magnitude and phase of the voltage developed across the coil, and this in turn is affected by the eddy-current properties of the sample. This is similar to one leg of a bridge circuit, but, rather than having another coil to balance against, it uses values stored in the computer memory. Advantages to using the absolute coil mode of operation are the ease and unambiguousness of the data interpretation, but there are also advantages in using a full bridge mode, which we will now discuss.

ORNL-DWG 78-16165

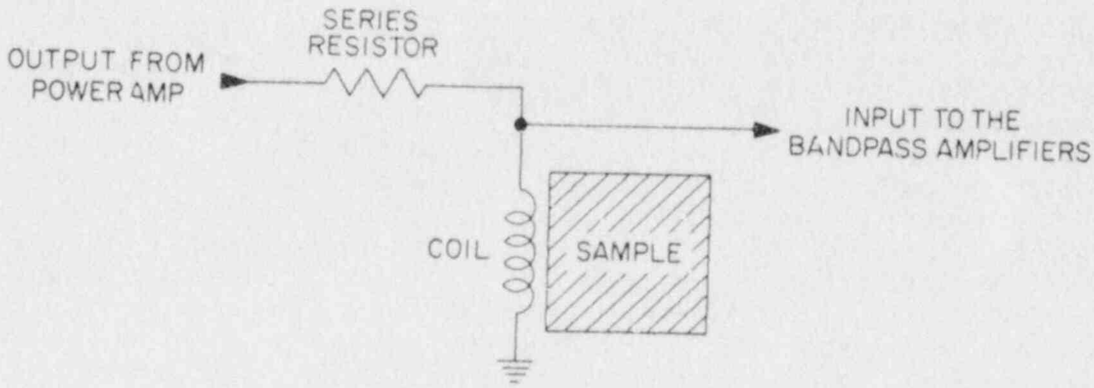


Fig. 2. Absolute mode of operation.

Bridge Mode of Operation

The bridge mode of operation is shown in Fig. 3 and consists of two dropping resistors to supply current to two coils. The coils may be wound in the same directions, so that the mutual inductance adds to the self inductance of each coil, or in opposite directions so that the mutual inductance subtracts from the self inductance. If the coils are in close proximity to each other, as is typical in many differential bobbin probes, the mutual impedance term can be a significant fraction of the coil self inductance, resulting in a lower impedance. It in turn is easier to drive a signal over the cable between the coil and instrument, when the cable is terminated in a low impedance. The resistor at the top of the bridge will help balance the inductive component of voltage developed across the two coils. Sometimes, resistors are also placed in parallel between the coils and ground to help balance the resistive component of the impedance, but it is difficult to balance a circuit such as this over a wide frequency range. This type of circuit is generally operated away from null because the phase detectors cannot measure the voltage phase with zero voltage. The output of the circuit is fed to the bandpass amplifiers and the rest of the circuits in the usual manner. This arrangement has the advantage that impedance differences between the two coils are measured, and the coils can be arranged so that both of them are detecting some of the same features of the same sample, and the differences in what the two coils detect can be amplified. Many of the properties that vary slowly along a sample being scanned will cancel out (if one coil scans behind the other), while small but rapid variations in the coil impedance, such as a defect may produce, can be selectively amplified. On the other hand, we may overlook a significant but slowly varying defect.

ORNL-DWG 78-16164

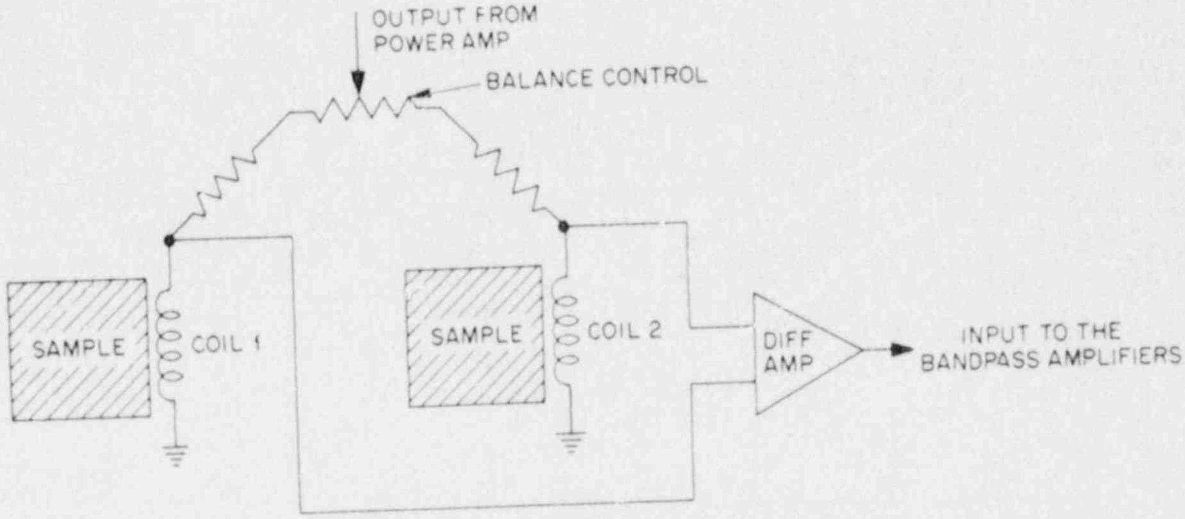


Fig. 3. Bridge mode of operation.

Reflection Mode of Operation

The reflection probe and its connections are shown in Fig. 4. A signal is supplied to the driver coil through an RLC network. The current in the driver coil produces an electromagnetic field that is detected by pickup coils coaxially mounted at either end of the driver coil. The pickup coils are wound in opposite directions and electrically balanced so that no net signal is produced when the probe is in air (away from a conductor). When the probe is placed on a conductor, a signal is produced that can be thought of as a signal reflected from the conductor. The reflection coil can also be considered an induction bridge, with the conducting sample unbalancing the bridge. This type of probe has been well researched[4] and successfully applied to a number of eddy-current problems over the years.[5] It can greatly reduce the effects of liftoff and probe temperature drifts for many applications.

Through-Transmission Mode of Operation

This mode of operation is the most accurate but requires access to both sides of the conductor as shown in Fig. 5. Both coils must be accurately positioned, with the axes aligned for best results. By taking care to insure that all coil and conductor dimensions are accurate, we can make accurate absolute measurements of the electrical and magnetic properties of samples placed between the two coils. Absolute conductivity measurements that agree with the National Bureau of Standards to within 0.01% have been made using this technique.[6]

CHASSIS FOR THREE-FREQUENCY INSTRUMENT

The chassis for the three-frequency instrument is constructed in a standard 133-mm-high (5.25-in.) NIM (Nuclear Instrumentation Module) bin. Two switching power supplies made by Computer Products are used, a ±15 V,

ORNL-DWG 78-16167R

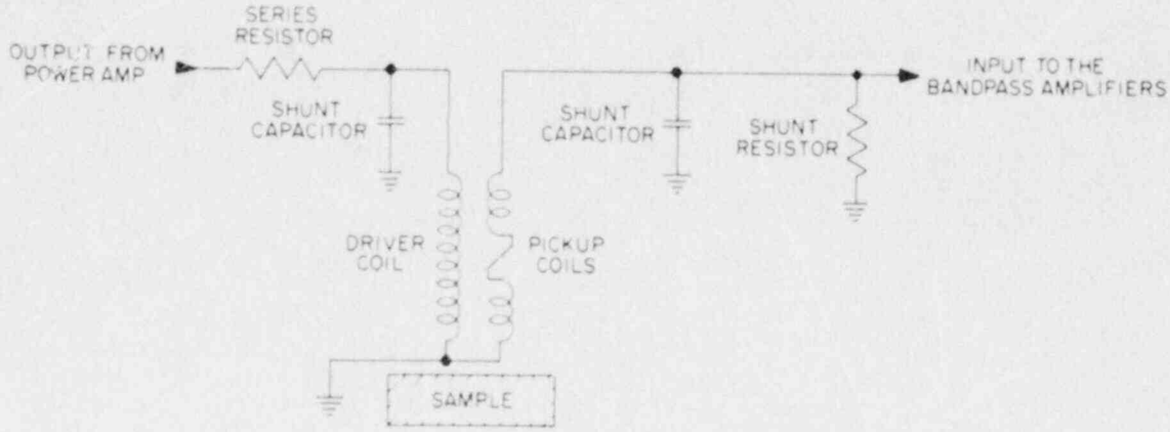


Fig. 4. Reflection mode of operation.

ORNL-DWG 78-16166

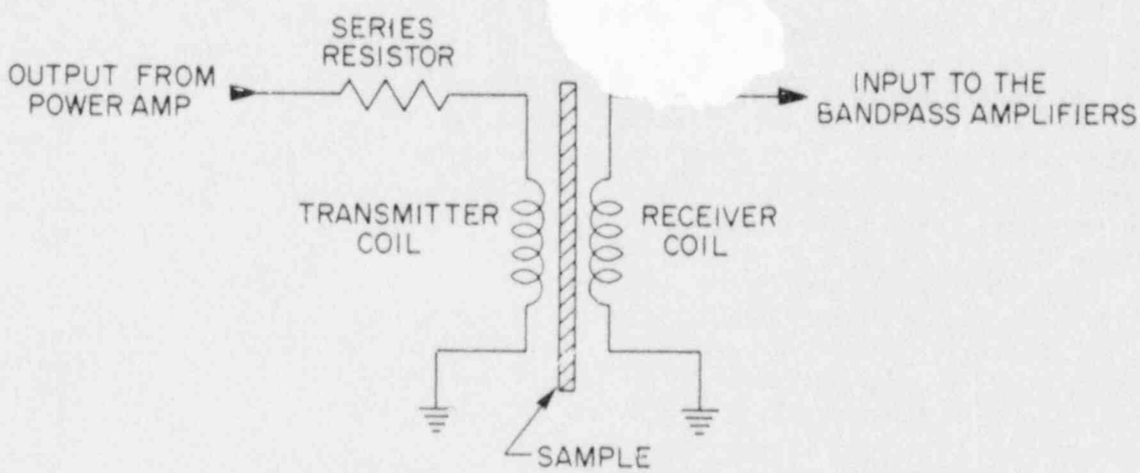


Fig. 5. Through-transmission mode of operation.

model No. HE215 and a 5 V, model No. HE381. Both supplies are adjusted to within 0.05 V of their nominal output voltages. The chassis is wired for an IEE-488 parallel port and an RS232 serial port. The wiring diagram for the chassis is shown in Fig. 6, and a photograph of the chassis with the different modules installed is shown in Fig. 7. The instrument consists of one quadruple-width module (the computer module) and eight single-width modules. The modules are a power amplifier, a calibrator, three bandpass amplifiers, three phase detectors, and the computer.

OSCILLATOR AND POWER AMPLIFIER MODULE FOR THE THREE-FREQUENCY EDDY-CURRENT INSTRUMENT

The oscillator and power amplifier module furnishes three mixed signals and their timing. The signals are low impedance, low distortion, and highly stable both in amplitude and frequency. We shall first discuss the oscillator section of the module and then the power amplifier section.

Oscillator

The module contains three similar Wien Bridge oscillators to generate three separate frequencies. A simplified diagram for a Wien Bridge oscillator is shown in Fig. 8.

The oscillator has two separate sections, the positive feedback leg, which determines the frequency, and the negative feedback leg, which determines the amplitude.

__Frequency__. We can calculate the voltage at the positive input, e_1, for a given output voltage, V_0:

$$e_1 = \frac{R_2 V_0}{R_1 + j\omega R_1 R_2 C_2 - j/\omega C_1 + R_2 C_2/C_1 + R_2} . \quad (1)$$

The gain will be maximum when the feedback is exactly in phase with the output (if the operational amplifier is ideal). This occurs when the two imaginary terms cancel, or

$$\omega R_1 R_2 C_2 = \frac{1}{\omega C_1} . \quad (2)$$

We can solve for the frequency and get

$$f = \frac{1}{2\pi\sqrt{R_1 C_1 R_2 C_2}} . \quad (3)$$

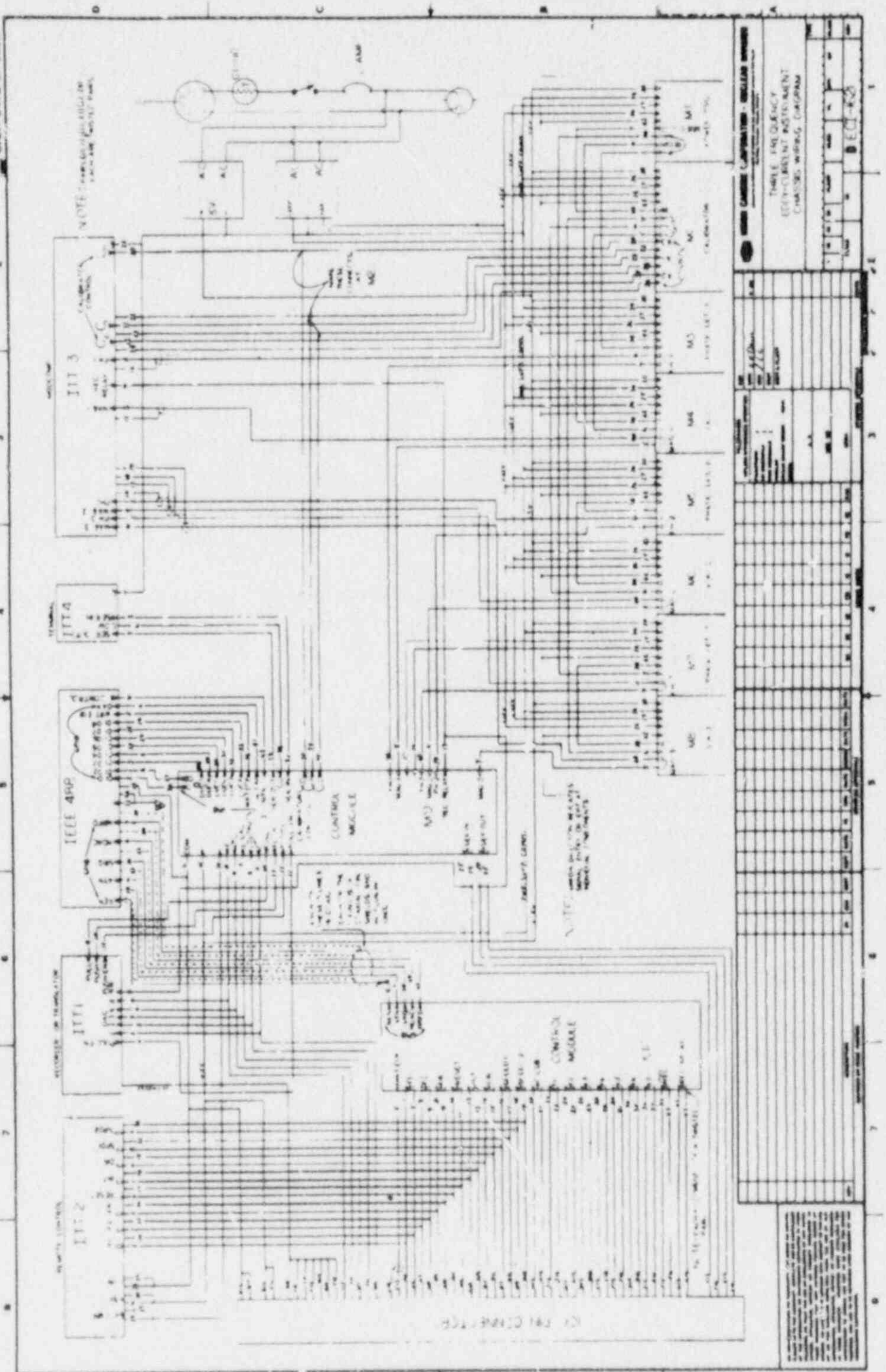


Fig. 6. Circuit diagram of the chassis of the three-frequency eddy-current instrument.

Y189030

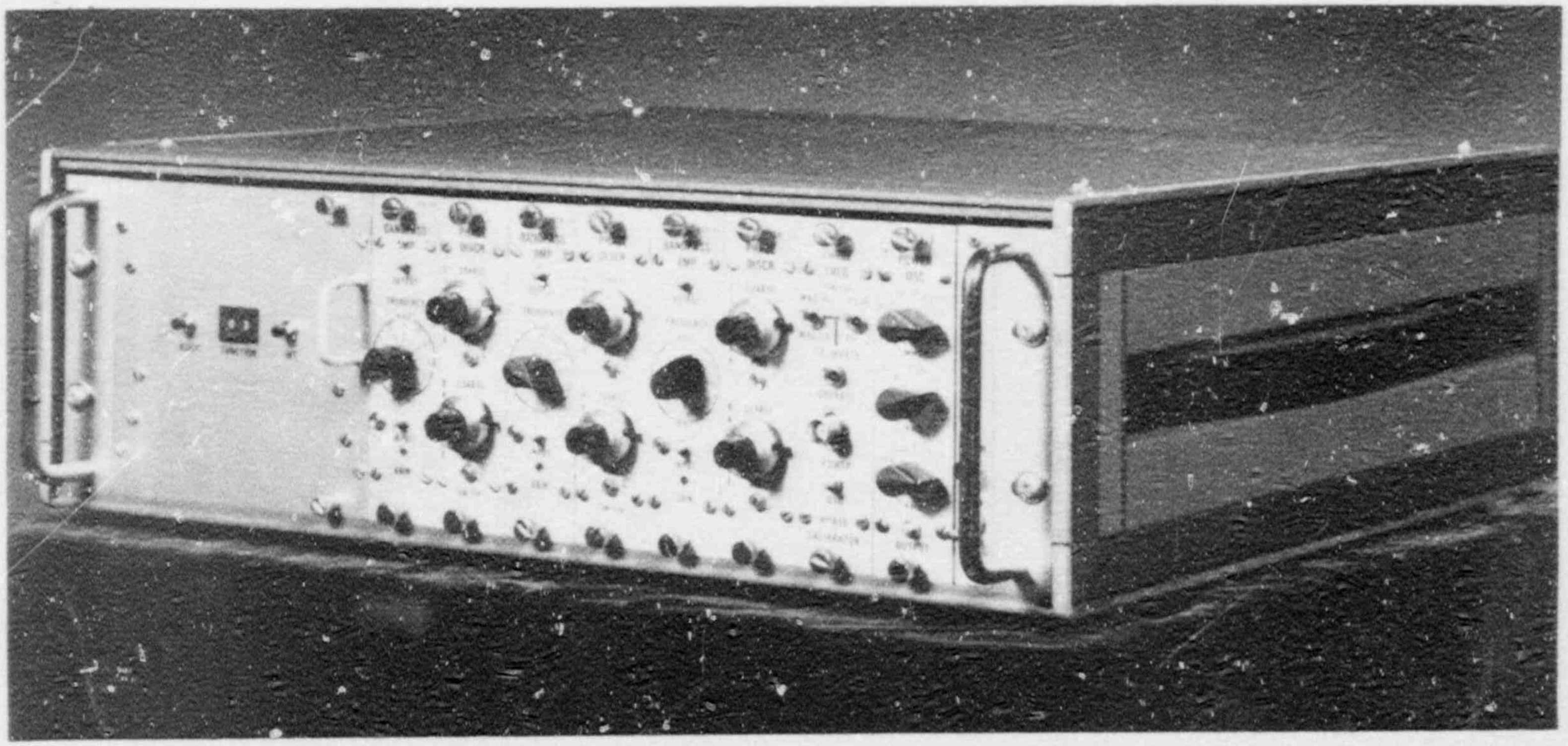


Fig. 7. Three-frequency eddy-current instrument.

ORNL-DWG 78-16751

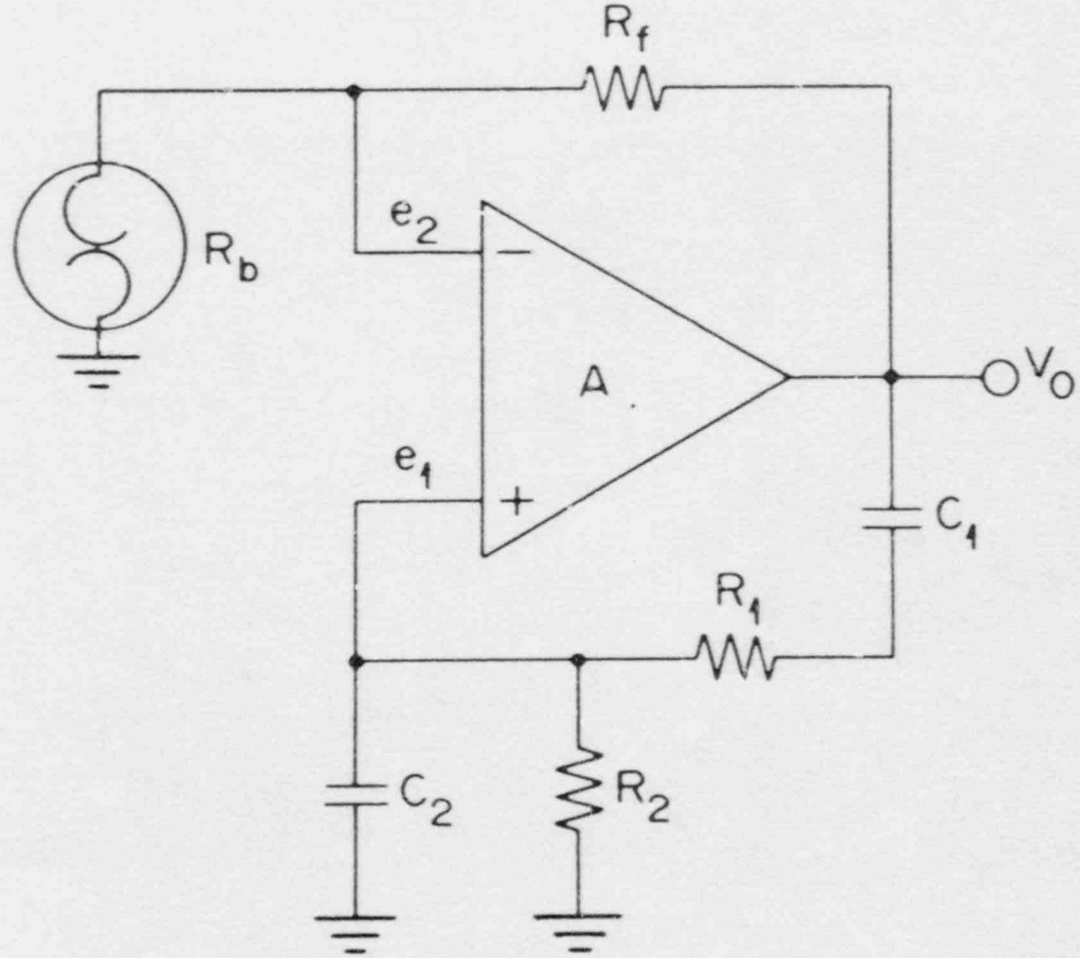


Fig. 8. Simplified diagram for the Wien Bridge oscillator.

If the amplifier is not ideal, but the gain has a single-pole response of the form

$$A = \frac{A_0}{1 + j\omega/\omega_0} \quad , \tag{4}$$

then the approximate equation for the frequency is

$$f \approx \frac{1}{2\pi\sqrt{R_1C_1R_2C_2 + 3C_1R_2/\omega_0}} \; . \tag{5}$$

The measured frequency is less than the frequency calculated by Eq. (3) for operational amplifiers having finite gains, frequency responses, and slew rates. The temperature drift and harmonic distortion both increase as the difference between calculated and actual frequency increases. However, we were able to select operational amplifiers that had good high-frequency response for the high-frequency oscillators and kept the actual frequency within 1% of the frequency calculated by Eq. (3) at 2 MHz. The capacitors used were high-stability, 1000-pF capacitors. To get the low frequencies, large resistances were required for R_1 and R_2. Since the bias for the amplifier must flow through R_2, this resulted in a severe voltage offset for the high-frequency amplifiers, which require a relatively large input current. Therefore, a lower frequency, low-bias-current operational amplifier was used for the low-frequency oscillators.

Amplitude. The amplitude is controlled by the negative feedback part of the circuit.

If we make $R_1 = R_2$ and $C_1 = C_2$, which we did in the circuit, and $\omega R_1R_2C_2 = \frac{1}{\omega C_1}$, Eq. (1) becomes

$$e_1 = V_0/3 \; , \tag{6}$$

and the equation for e_2 is

$$e_2 = R_bV_0/(R_f + R_b) \; . \tag{7}$$

We can also calculate V_0 from the gain multiplied by the signal difference at the input, or

$$(e_1 - e_2)A = V_0 \; . \tag{8}$$

Substituting Eqs. (6) and (7) into Eq. (8) and canceling the V_0 term gives

$$\left(\frac{1}{3} - \frac{R_b}{R_f + R_b}\right) A = 1 \cdot \tag{9}$$

It appears that the circuit is now independent of the output voltage, V_0, but actually the value of R_b depends on V_0. We can solve Eq. (9) for R_b and get

$$R_b = R_f \ (A - 3)/(2A + 3) \ , \tag{10}$$

which is R_b as determined by circuit gain. Also, R_b depends on the temperature of the bulb in the manner

$$R_b = R_0[1 + \alpha(T - T_0)] \ , \tag{11}$$

where R_0 is the resistance at temperature T_0, the temperature coefficient of resistance of the bulb filament is α, and T is the absolute temperature. The bulb is heated by the voltage e_2 until it reaches an equilibrium temperature, T, where the power loss is equal to the power into the bulb.

The power loss will be due to both radiation and conduction. The conduction term is small, since the bulb contains a vacuum and only a small amount is lost through the filament ends. Also, because of the small size of the wire and the relatively low ambient temperature, the bulb absorbs very little radiation from the surroundings. Therefore, the power loss, which is proportional to the power into the bulb, is

$$e_2{}^2/R_b = a_0 T^4 \ . \tag{12}$$

The constant a_0 contains the surface area, emissivity, and other collected factors in the equation. We can use Eqs. (7), (11), and (12) to solve for the voltage out:

$$V_0 = T^2\{R_f + R_0[1 + \alpha(T - T_0)]\} \left\{\frac{a_0}{R_0[1 + \alpha(T - T_0)]}\right\}^{1/2} \cdot \tag{13}$$

Use of Eqs. (10) and (11) to solve for the temperature gives

$$T = \frac{1}{\alpha}\left[\frac{R_f(A - 3)}{R_0(2A + 3)} - 1\right] + T_0 \ , \tag{14}$$

which can be substituted into Eq. (13) to give

$$V_0 = \left\{\frac{1}{\alpha}\left[\frac{R_f(1 - 3/A)}{R_0(2 + 3/A)} - 1\right] + T_0\right\}^2 \left(\frac{a_0 R_f}{2 - 3/A - 9/A^2}\right)^{1/2} \times 3 \ . \tag{15}$$

If we let the gain approach infinity and take the resistance coefficient of tungsten to be 0.005/K, we get

$$V_0 = \left(\frac{R_f - 2R_0}{0.01R_0} + T_0\right)^2 \quad \left(\frac{a_0 R_f}{2}\right)^{1/2} \times 3 \ , \tag{16}$$

and

$$T = \frac{R_f - 2R_0}{0.01R_0} + T_0 \ . \tag{17}$$

The resistance of the bulb at room temperature must be measured with a very low current to avoid heating the filament enough to change the resistance. The output voltage must be measured once, with one set of values to determine a_0. Once these measurements have been performed, different values of R_f can be selected to give different output voltages.

The actual circuits are shown in Fig. 9. The diodes in the negative-feedback leg decrease the time needed to reach stability. With 75-Ω bulbs, the output voltage is about 5 V, peak to peak, and the bulb temperature is about 330°C. The resistor values for a given frequency are shown in Table 1.

The output of each oscillator is fed to the phase-shift network and a power amplifier.

Power Amplifier

The power amplifier is a summing amplifier with a high-gain stage followed by a unity-gain current-amplifier stage. The output is the inverted sum of the three inputs or

$$V_0 = V_1\frac{R_f}{R_1} - V_2\frac{R_f}{R_2} - V_3\frac{R_f}{R_3} \quad , \tag{18}$$

where V_1, V_2, and V_3 are the output voltages of the three oscillators and R_1, R_2, and R_3 are the output resistances of the oscillators. The component layout and lead placement are critical. In Fig. 10 we show the component layout, and in Fig. 11 we show both sides of the printed circuit board.

Table 1. Resistor values (R_f) vs oscillator frequency

Frequency (kHz)	R_f (kΩ)	Frequency (kHz)	R_f (kΩ)
0.20	795.8	50	3.183
0.50	318.3	100	1.592
1.0	159.2	200	0.7958
2.0	79.58	500	0.3183
5.0	31.83	1000	0.1592
10	15.92	2000	0.07958
20	7.958		

BANDPASS AMPLIFIER

The bandass amplifier consists of an input stage, two identical active filter stages, and an output stage. A circuit diagram is shown in Fig. 12. The frequency-determining resistors are the same values as required for the oscillator and are given in Table 1.

Input Stage

The input stage is a single, high-input-impedance, operational amplifier with a variable gain which is adjustable from 1 to 100. The input impedance is set at 1 MΩ by a shunt resistance to ground, and the probe signal is capacitively coupled into the amplifier with a 0.033-μF capacitor.

Active Filter Stages

There are two similar state-variable (sometimes called bi-quad) bandpass filter stages, and a simplified diagram of the filter used is shown in Fig. 13. While the filter can be used as a low-pass, high-pass, or bandpass filter, we used only the bandpass type for this module. The midfrequency is determined by the equation

$$f_0 = \frac{1}{2\pi}\sqrt{\frac{R_4}{R_5}\left(\frac{1}{R_1C_1R_2C_2}\right)} . \tag{19}$$

This equation is very similar to Eq. (3) for the oscillator frequency. The same types and values of capacitance and resistance are used in both circuits so that the drifts between the two tend to track each other. The Q of the circuit is given by

$$Q = \left(1 + \frac{R_6}{R_7} + \frac{R_6}{R_8}\right) \sqrt{\frac{R_4R_1C_1}{R_5R_2C_2}} \Bigg/ \left(1 + \frac{R_4}{R_5}\right) , \tag{20}$$

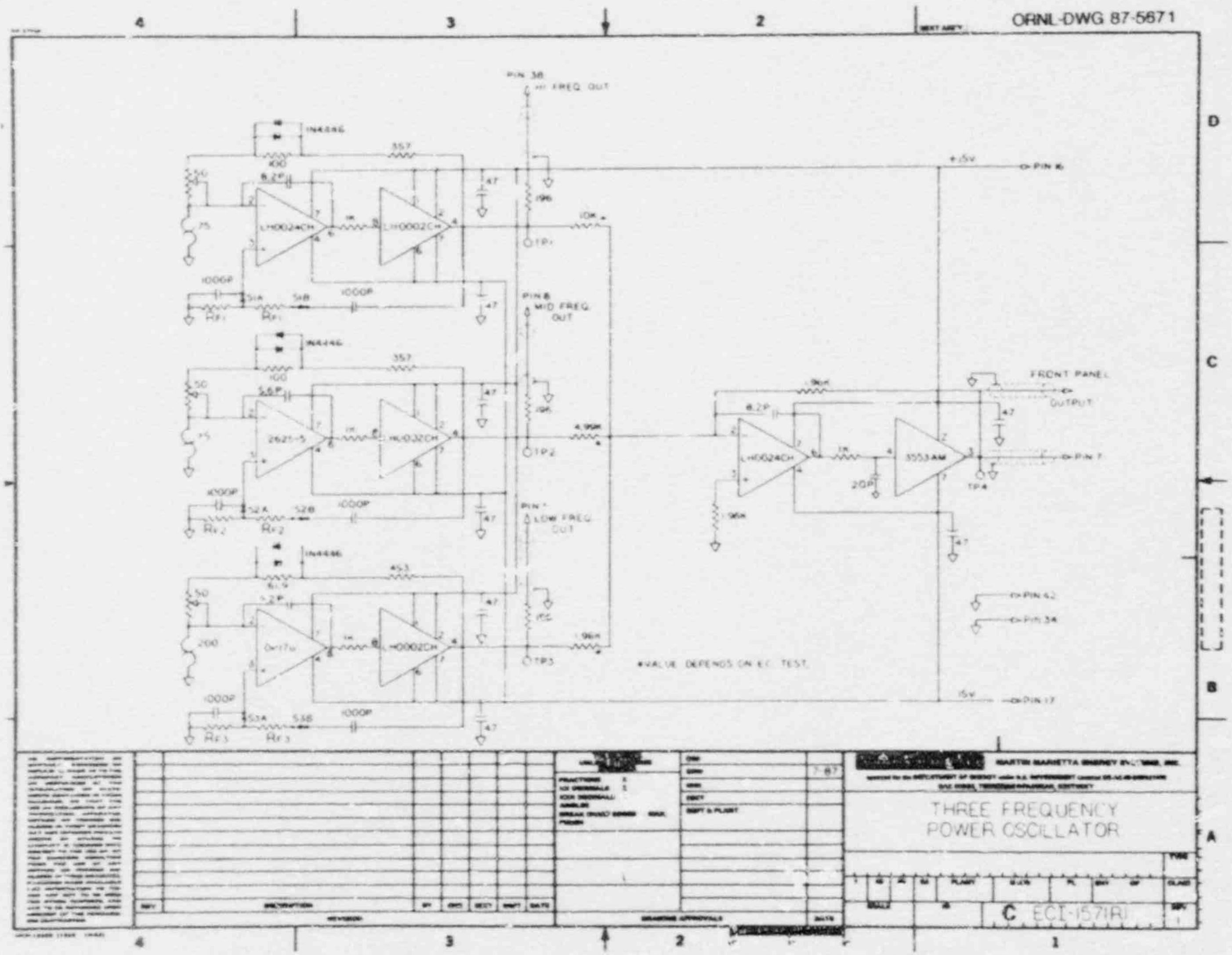


Fig. 9. Circuit diagram of the three-frequency oscillator and power amplifier.

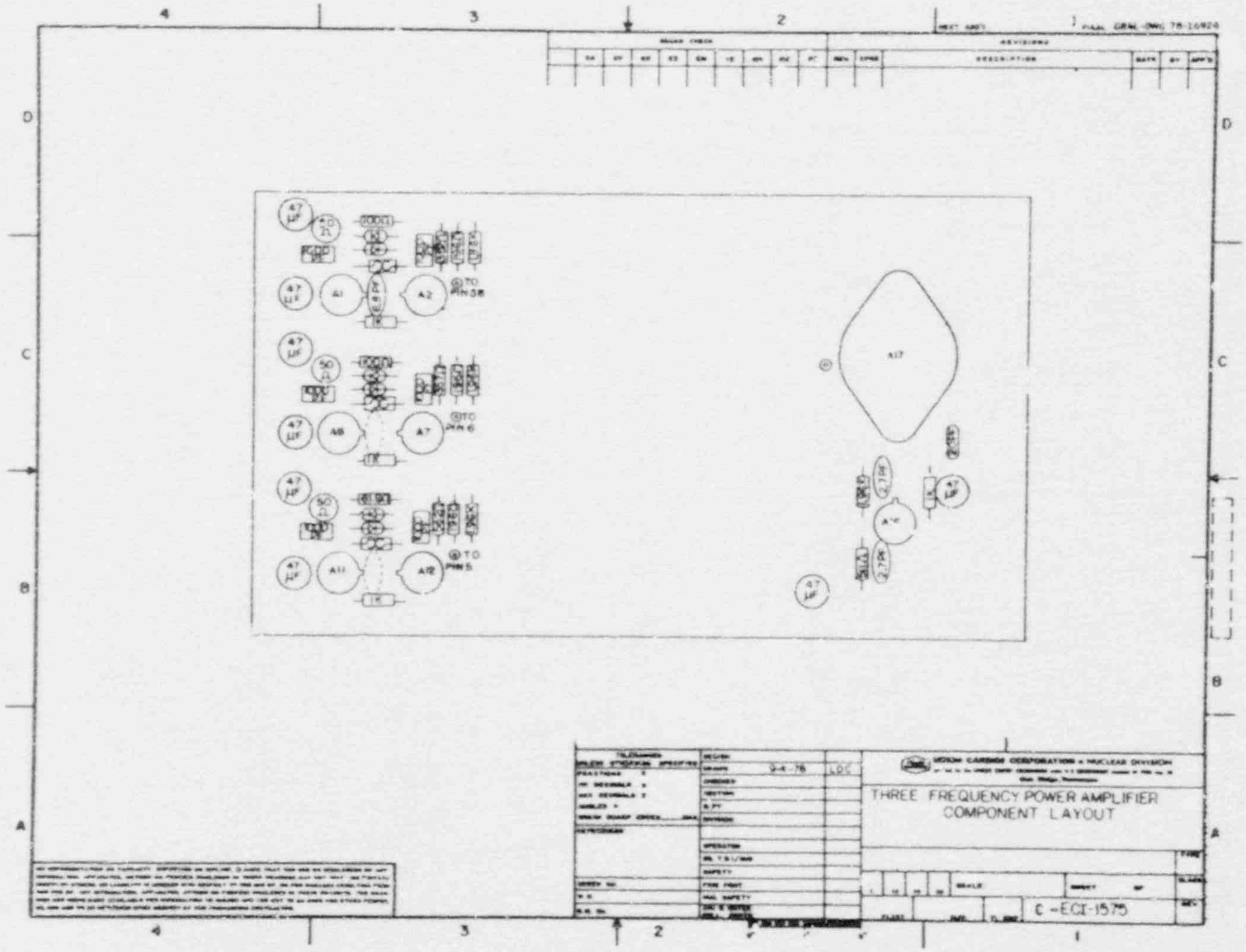
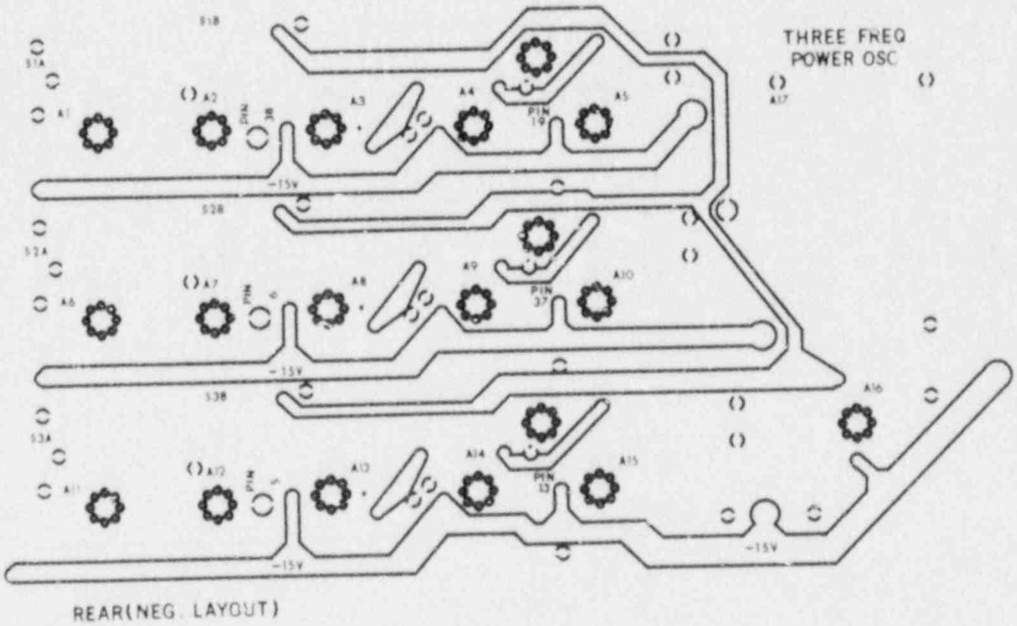


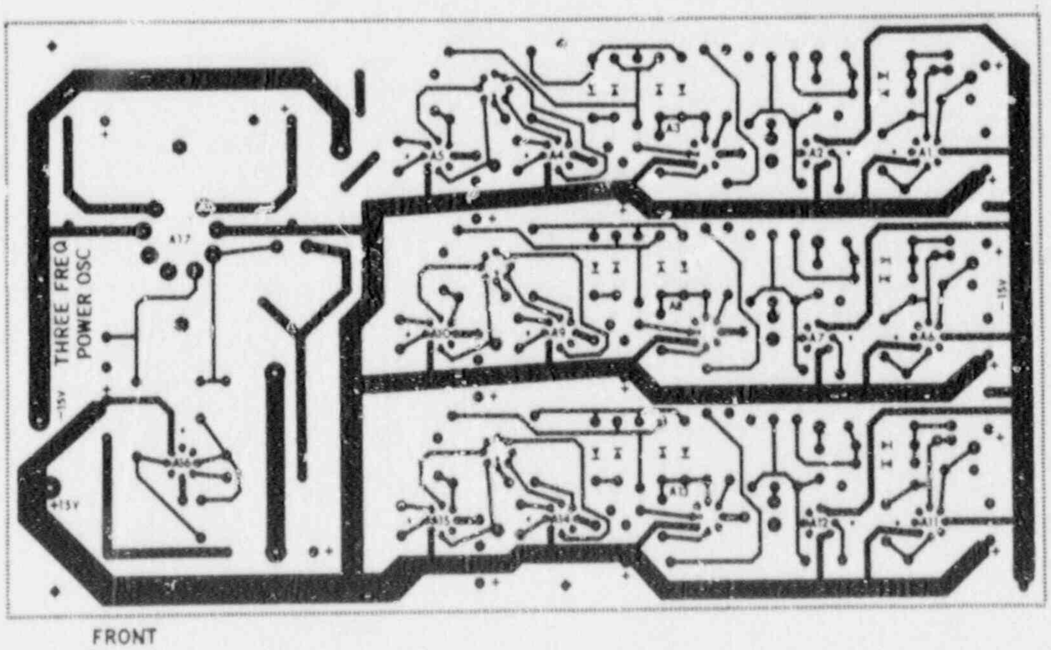
Fig. 10. Component layout of oscillator and power amplifier.

ORNL-DWG 78-16227



(*a*)

ORNL-DWG 78-16226



(*b*)

Fig. 11. Printed circuit board of the three-frequency oscillator and power amplifier. (*a*) Component side. (*b*) Reverse side.

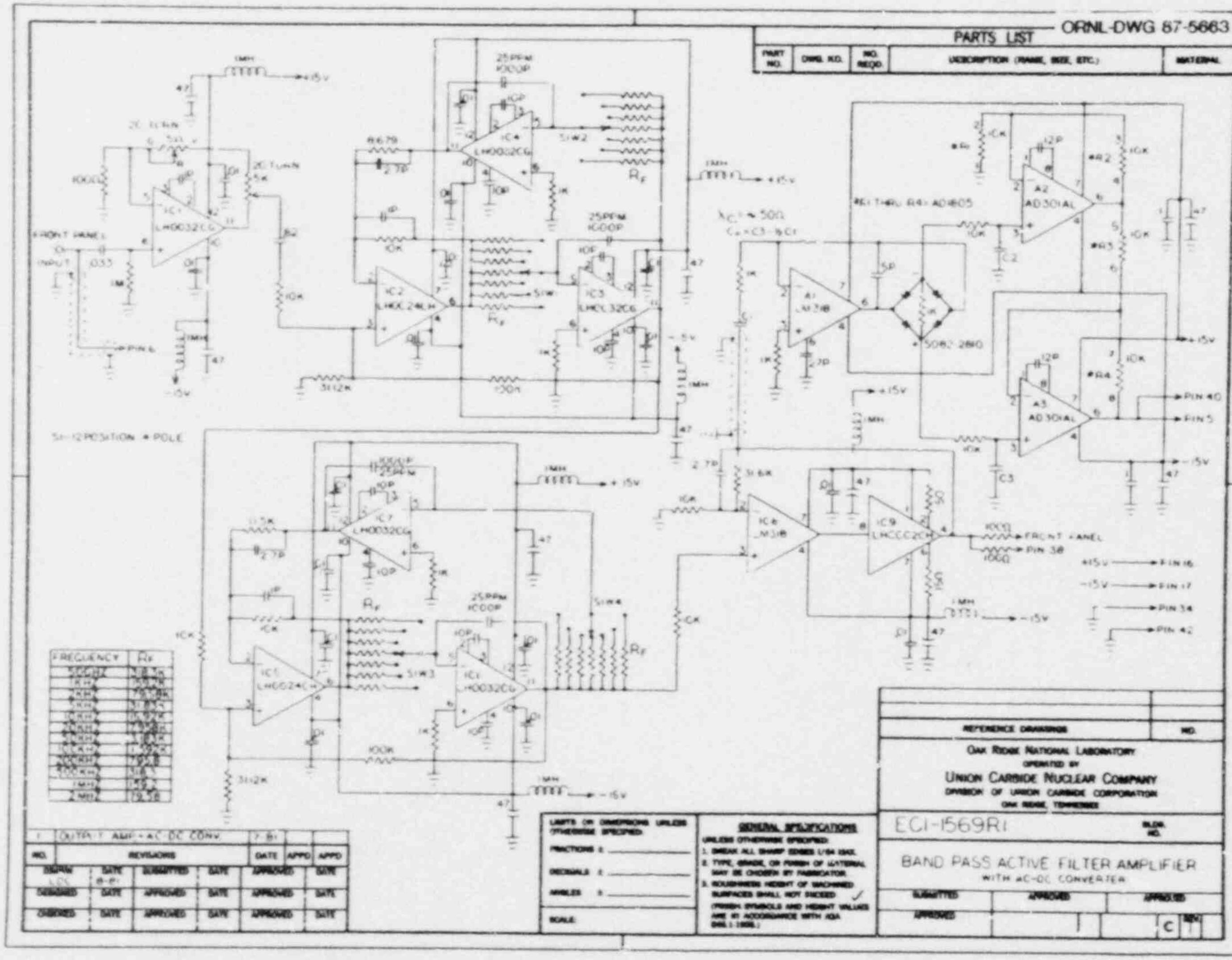


Fig. 12. Circuit diagram of the bandpass amplifier.

ORNL-DWG 78-16752

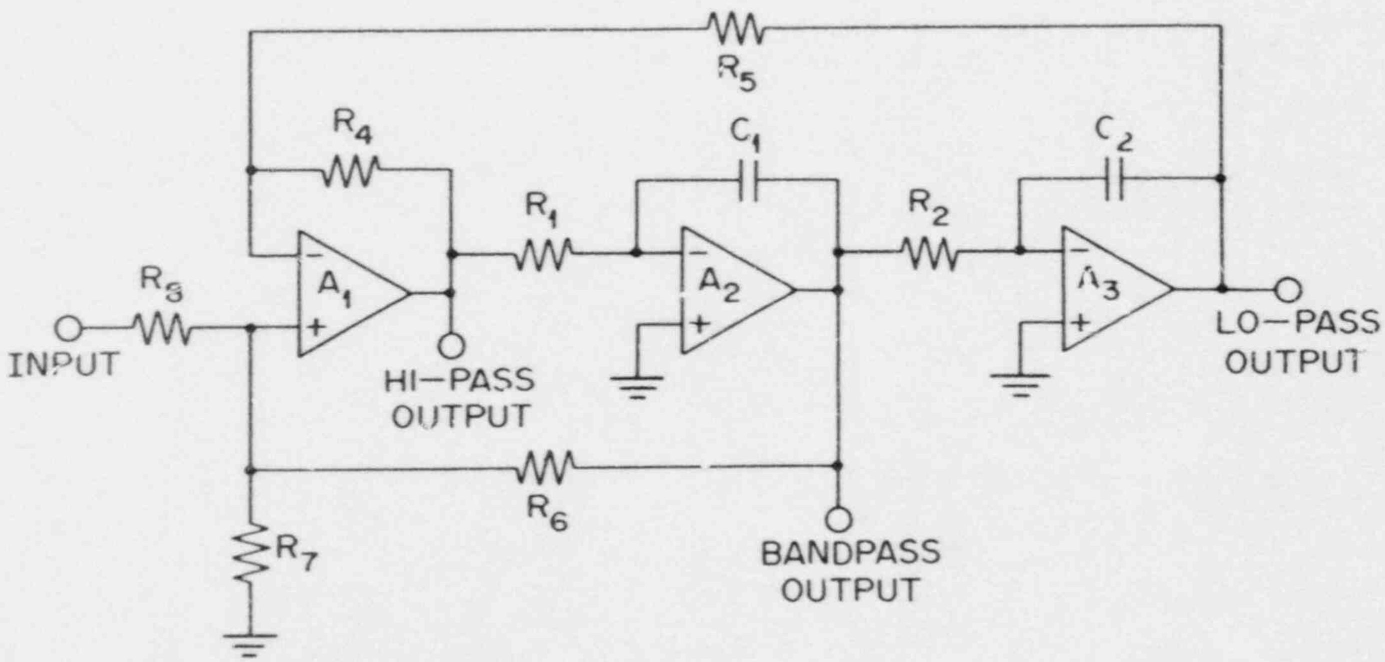


Fig. 13. Simplified diagram of the state variable filter.

and the gain by

$$\text{Gain} = -R_6/R_8 \ . \tag{21}$$

In the actual design of the filter we have chosen R_1 equal to R_2, and these are the only components that are switched as the frequency is changed. Both C_1 and C_2 are 1000 pF high-stability capacitances. We have set R_4 to 10 kΩ, and R_5 is varied to change the "pole location," depending on the type of filter we want. We have chosen R_6 equal to 100 kΩ and R_8 equal to 10 kΩ so that each filter section has a gain of 10.

The value of R_7 is chosen from the Q of the filter stage, by use of the equation

$$R_7 = \frac{R_6}{\dfrac{Q(1 + R_4/R_5)}{(R_4/R_5)^{1/2}} - 1 - R_6/R_8} \ . \tag{22}$$

The maximum gain, $-R_6/R_8$, is limited to a value that makes R_7 positive.

The type of filter response desired determines the pole placement and the Q of the individual sections. The type filter chosen is a two-pole Butterworth with an overall Q of 5. This type of filter has the maximum attenuation with no ripple in the bandpass region, and the phase shift through this region is linear. The values of Q_1 and f_1 for each filter section were determined from Tables 1 and 3 in ref. 7. The first stage had $f_1 = 0.8679f_0$ and $Q_1 = 7.0888$, and the second stage had $f_2 = 1.1522f_0$ and $Q_2 = 7.0888$.

An analysis of the circuit, using nonideal operational amplifiers, gives the transfer function as

$$\frac{V_{BP}}{V_{in}} = -\frac{R_6}{R_8} \Bigg/ \left[1 + \frac{(Z_1+R_1+R_1A_2)\ (R_3+R_4+R_3A_1)}{Z_1F_2(R_3+R_5)A_1A_2} + \frac{R_4Z_2A_2}{(R_4+R_5)F_2\ (Z_2+R_2+R_2A_3)}\right] \tag{23}$$

where $F_2 = \dfrac{1}{1 + R_6/R_8 + R_6/R_7}$, $Z_1 \equiv \dfrac{1}{j\omega C_1}$, $Z_2 \equiv \dfrac{1}{j\omega C_2}$.

The gains of each stage, A_1, A_2, and A_3, are complex functions of the frequency. If we assume a perfect operational amplifier with infinite gain, the transfer function becomes

$$\frac{V_{BP}}{V_{in}} = \frac{j\omega \text{ Gain } \omega_0/Q}{-\omega + (j\omega\omega_0/Q) + \omega_0^2} \ , \tag{24}$$

where $\omega = 2\pi f$, and the equations for ω_0, Q, and gain are given in Eqs. (19), (20), and (21).

The calculated relative magnitude and phase of the output of the two filter sections as functions of frequency are shown in Fig. 14. The response shown here is for ideal filters and is the same as the measured response at low frequencies. At the higher frequencies, both the measured and calculated responses show peaking and will even oscillate if the actual circuit components are too far apart. The overall gain of both stages of the filter is 50 at f_0, and the phase shift is 8.125° for a 1% frequency change. In order to maintain the 0.01° accuracy on phase shift measurements, the differential frequency drift between the oscillator and bandpass amplifier must be less than 0.0012%, or 12 ppm. The total measured drift is about 0.04° phase shift/°C. The output at $0.1f_0$ is only 0.0004 times the output at f_0, and at $0.5f_0$ and $2f_0$ the output is 0.018 times the f_0 values.

Output Stage

The output stage has a gain of 2 and can drive a low-impedance load with very little distortion. The overall gain of the bandpass amplifier module can be varied from 100 to 10,000. The amplifier operates up to a frequency of 2 MHz, with the performance falling off above 800 kHz.

Alternating Current-to-Direct Current Converter

The ac-to-dc converter drives a full-wave bridge that is in the feedback loop of a wideband operational amplifier. The diode forward voltage drop and its drift with temperature are eliminated by use of this circuit. The differential amplifier output furnishes a gain of 2, and the dc voltage out is a linear function of the RMS input voltage of the form

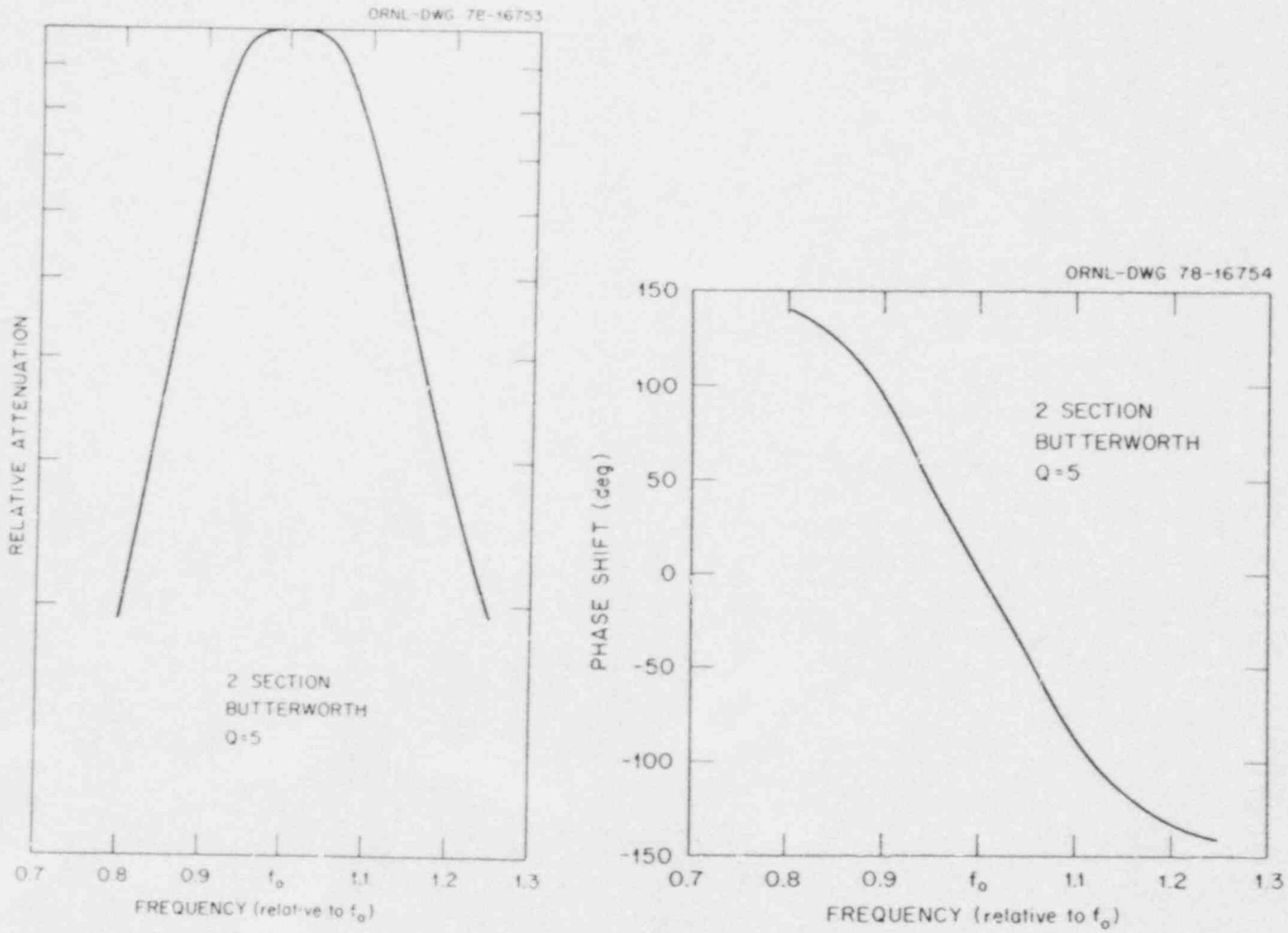
$$V_{out} = 0.062 + 1.902V_{RMS} . \qquad (25)$$

The equation if linear to within about ±0.003 from 0.5 to 4.0 MRS input volts. The temperature coefficient is less than 40 ppm/°C over the entire frequency range.

The layout of the amplifier is critical, and the printed circuit is shown in Fig. 15. The tuning resistors of the amplifier are mounted on a four-way rotary switch with shielding between the stages. The component layout is shown in Fig. 16.

PHASE DETECTOR

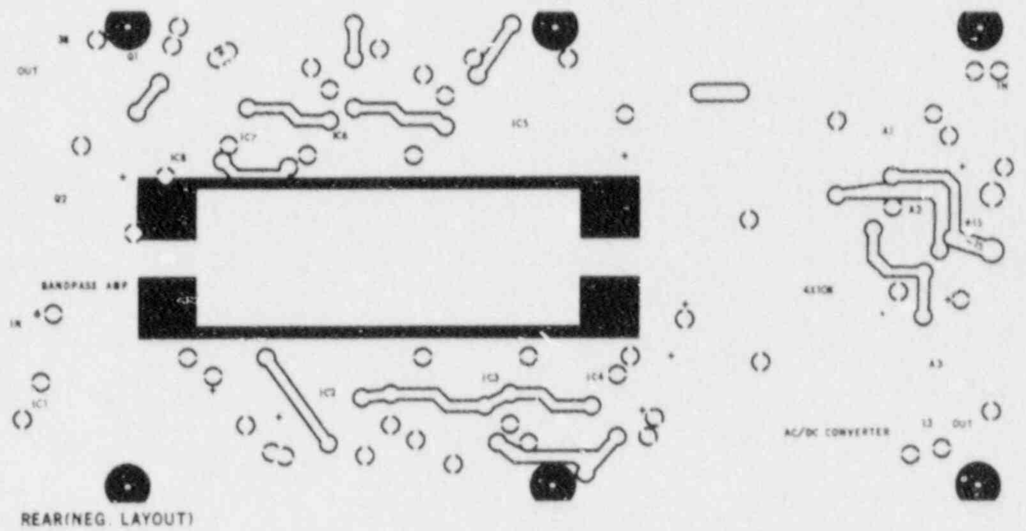
The phase detector has an output voltage that is proportional to the difference between an arbitrary reference phase and the phase shift between the oscillator signal and the bandpass amplifier signal. It consists of two voltage height discriminators that produce pulses when the input signals pass through a set level, pulse-shaping circuits, a flip-flop that turns on when one pulse is received and off with the next pulse, a low-pass filter to integrate the flip-flop output, and an



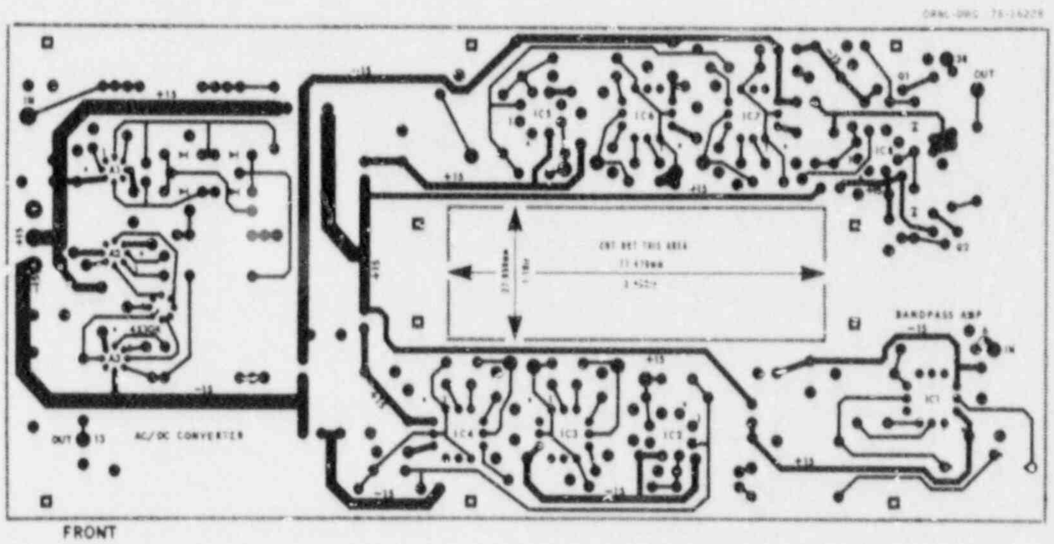
(*a*) (*b*)

Fig. 14. Characteristics of the bandpass amplifier. (*a*) Relative attenuation. (*b*) Phase shift.

ORNL-DWG 78-16229



(*a*)



(*b*)

Fig. 15. Bandpass amplifier board. (*a*) Component side. (*b*) Reverse side.

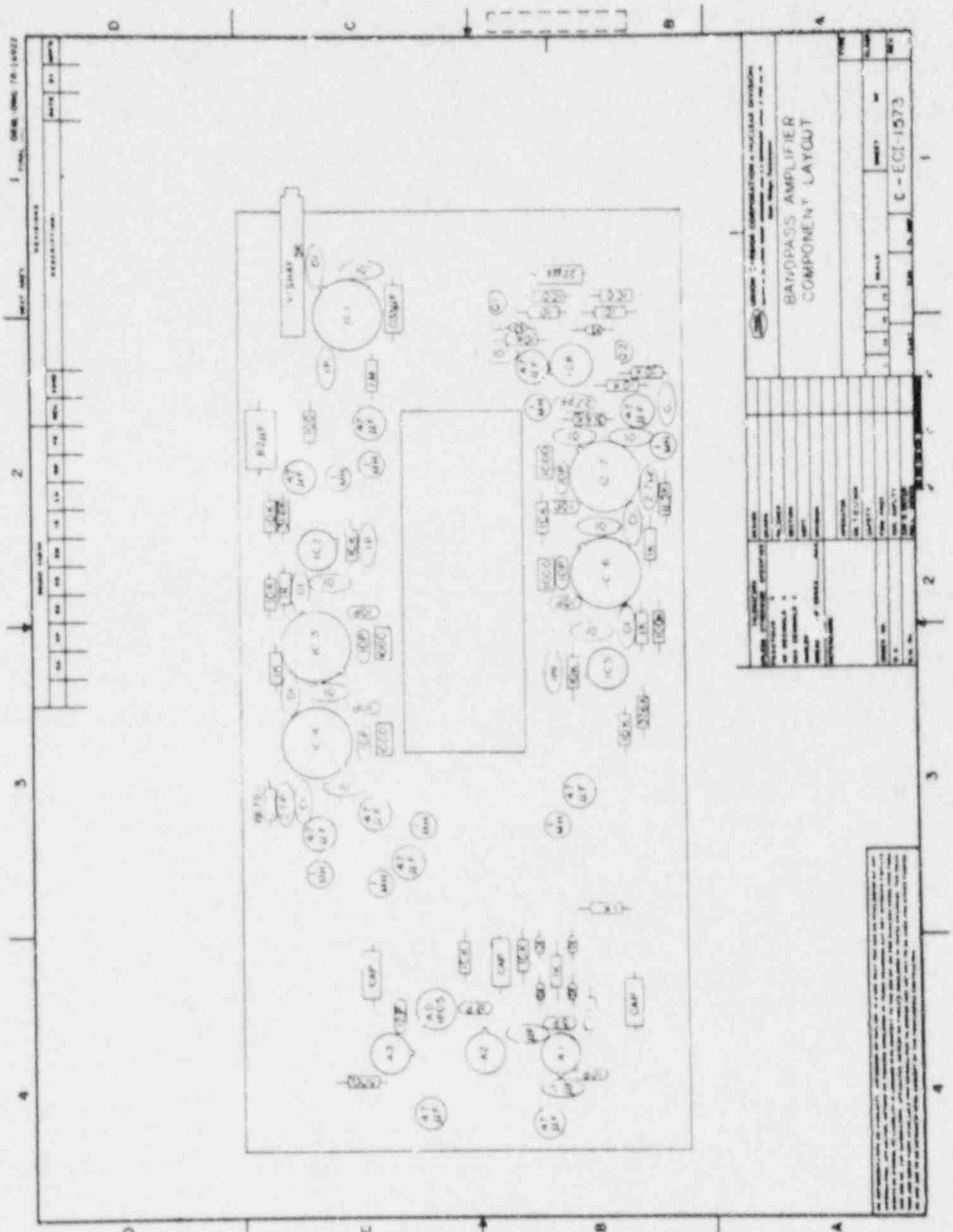


Fig. 16. Component layout of the bandpass amplifier board.

adjustable offset voltage to offset the phase shift. The circuit diagram of the phase detector is shown in Fig. 17 and the frequency-determining resistors for the low-pass filter are given in Table 2.

Voltage Height Discriminators

The two voltage height discriminators are very similar, one for the oscillator signal and one for the bandpass amplifier signal. The voltage discriminator is basically a differential amplifier with a constant current source in each emitter and a 2.2-mA tunnel diode in each collector. When the input signal reaches the voltage level set by the variable resistance in the other side of the differential amplifier, the tunnel diode will switch from its low state to its high state. The variable resistance for the bandpass amplifier circuit is mounted on the front panel and can be adjusted over a ±5 V range, although it is usually set for zero crossover voltage for multiple-frequency operations. The oscillator signal circuit is always adjusted for zero crossover, and there is a tunnel diode in each collector, so that the phase shift can be measured from zero or 180°. This must be used in some cases to keep the output signal from changing by 360° when the two inputs are very close together in time. This circuit is very sensitive and stable up to 5 MHz. The differential amplifiers keep the voltage drift to less than 2 μV/°C, and the circuit has essentially no voltage gain, which reduces capacitance effects.

The signal from the tunnel diodes is amplified by two temperature-controlled pulse transistors in a can that contains its own oven. These pulses are then fed to either side of a flip-flop circuit.

Flip-Flop Circuit

The flip-flop consists of a dual transistor pair in a single can with a very accurate constant-current source in the emitters. Both collector resistors and the base bias-resistor strings have been chosen to minimize the changes in gain and base-to-emitter voltage with temperature. The flip-flop operates in the unsaturated mode to achieve high-speed switching. The collector resistors are precision metal-film resistors mounted in the same case for thermal tracking.

Balance Control

The balance voltage is used as a "phase offset" for the flip-flop voltage and can be adjusted to cover the entire range of flip-flop output voltages. It also uses the voltage drop across fine and coarse potentiometers mounted on the front panel and fed by a constant-current source. The reference voltage for the constant-current sources and the upper voltage levels for both the flip-flop and balance control are regulated by a 723 voltage-regulator integrated circuit. Since only the difference between these two voltages is measured, small drifts in the voltage-regulator output tend to cancel. The output of the balance voltage is fed to a unity gain amplifier to furnish a high current level.

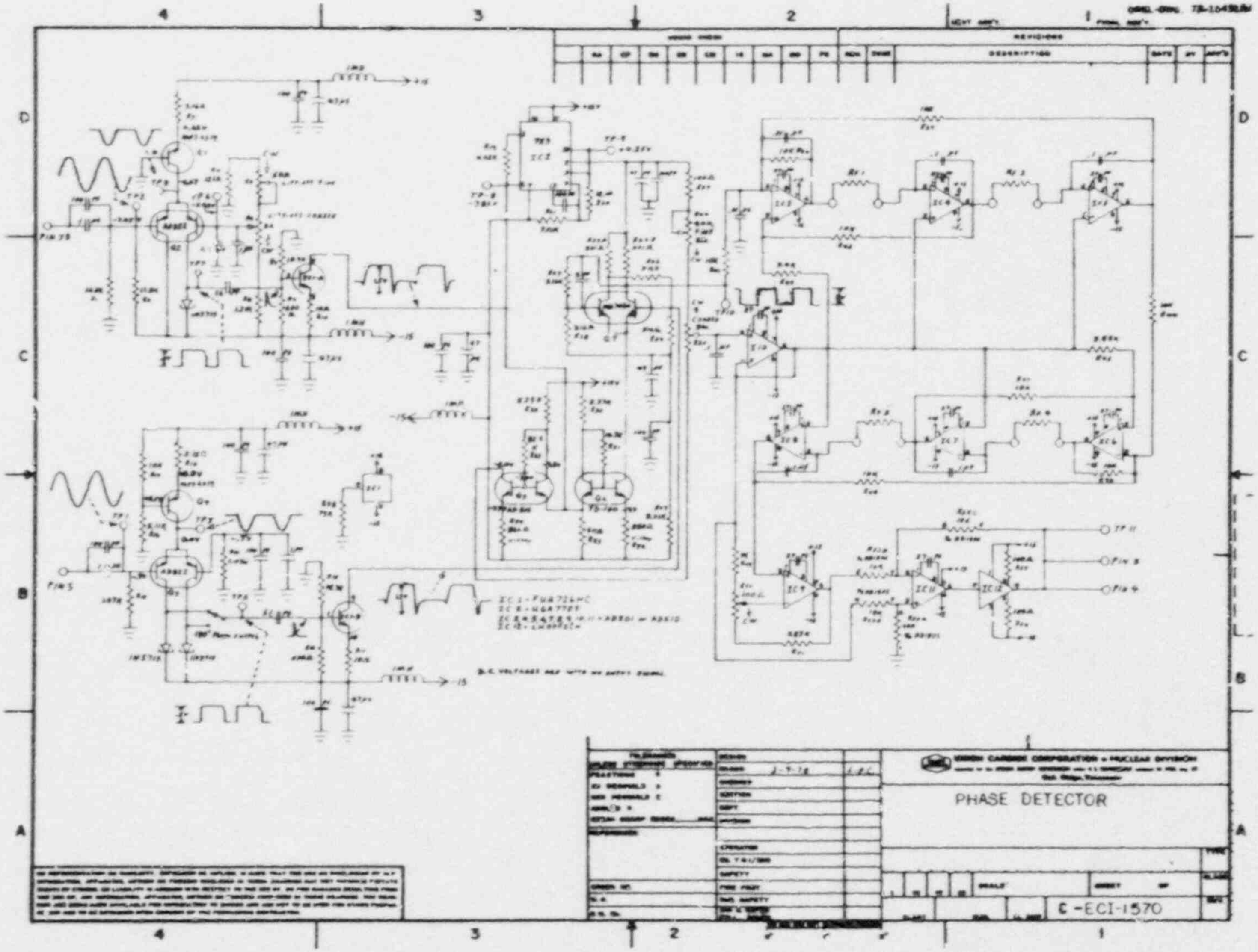


Fig. 17. Circuit diagram of the phase detector.

Table 2. Resistance, Rf, to determine the cutoff frequency of the low-pass filter

R_f (kΩ)	Frequency (Hz)	Switch	R_f (kΩ)	Frequency (Hz)	Switch
795.8	2	1	15.92	100	6
318.3	5	2	7.958	200	7
159.2	10	3	3.183	500	8
79.58	20	4	1.592	1000	9
31.83	50	5	0.7958	2000	10

Low-Pass Filter

A low-pass filter is used to integrate the signal from the flip-flop. The design study of this filter is similar to the bandpass filter and will not be repeated here. The response of the filter with frequency is shown in Fig. 18. The value of f_0 can be chosen by varying the resistors R_{f_1}, R_{f_2}, R_{f_3}, and R_{f_4} as shown in Table 2. The filter has a gain of 5. The output of the filter is fed to an adjustable gain amplifier, which is set to give 1 V out for 10° phase shift. The signal is next fed to a zero-restoring and buffer-output circuit, which sets the signal level to zero when the average flip-flop and the balance voltages are equal.

The printed circuit layout and the component layout for the phase detector module are shown in Figs. 19 and 20.

Calibrator and Multiplexer

The phase calibrator is a passive R-L-C circuit that will vary the amplitude and the phase independently. A simplified circuit diagram of the phase-shift network is shown in Fig. 21. The inductance, L_1, and resistance, R_3, of a stable toroidal coil are measured, and the tuning capacitance, C_1, is calculated from the equation

$$C_1 = 2/(\omega^2 L_1) \qquad (26)$$

The output voltage for the phase shift network is

$$V_{\text{out}} = [R_4 V_0(\omega L_1 R_2 - R_3/\omega C_1) - jR_4 V_0(R_2 R_3 + L_1/C_1)]$$

$$\div \left\{(R_1 R_3 + j\omega L_1 R_1)\left[\omega C_3 R_2 R_4 - \frac{1}{\omega} - \left(\frac{1}{C_1} + \frac{1}{C_2}\right) - jR_2 - jR_4\left(\frac{C_3}{C_1} + \frac{C_3}{C_2} + 1\right)\right] + \left[(R_1 + R_3)R_2 + \frac{L_1}{C_1} + j\left(\omega L_1 R_2 - \frac{R_1 + R_3}{\omega C_1}\right)\right]\left[\frac{-1}{\omega C_4} - jR_4\left(\frac{C_3}{C_4} + 1\right)\right]\right\}$$

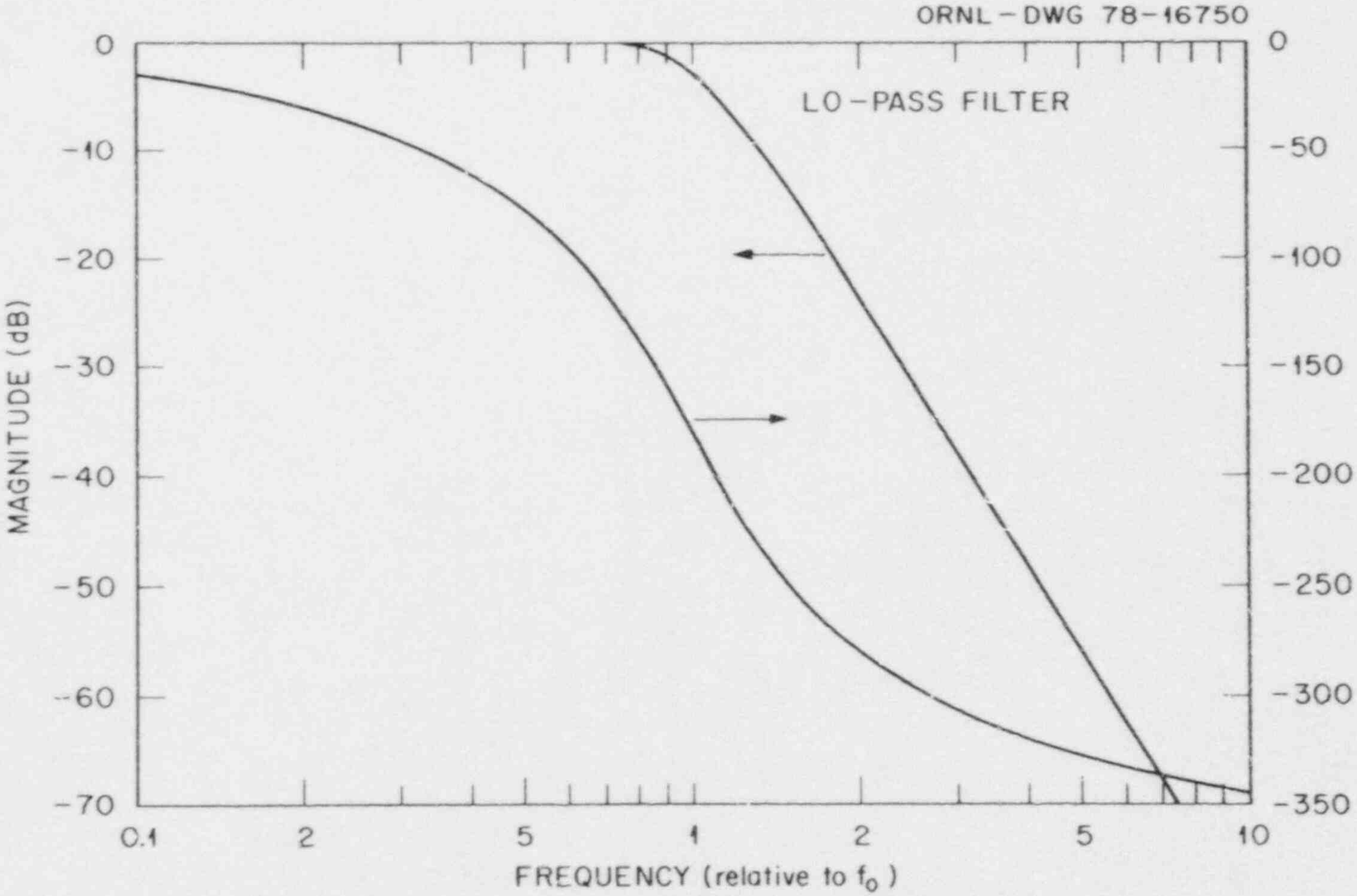
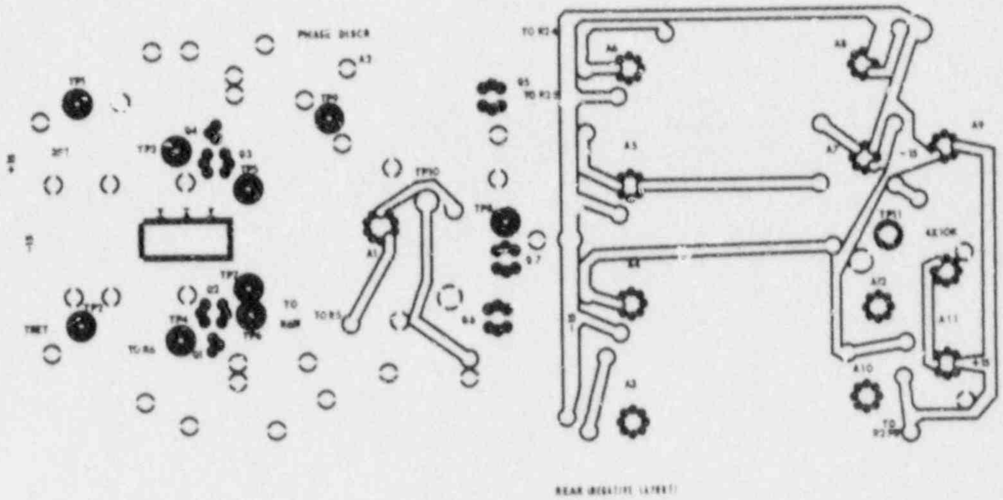
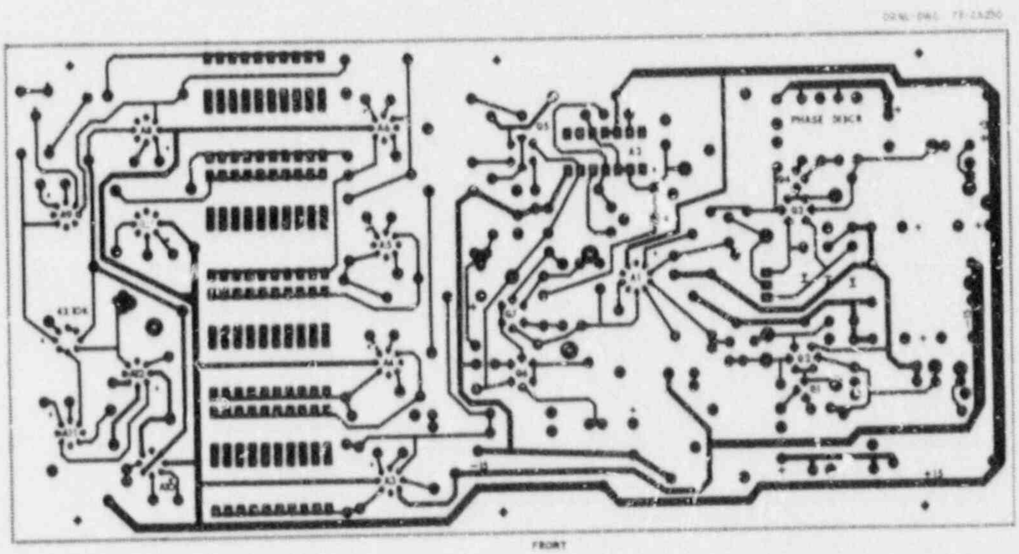


Fig. 18. Magnitude and phase of the output voltage of the low-pass filter.

ORNL-DWG 78-16231



(*a*)



(*b*)

Fig. 19. Phase detector board. (*a*) Component side. (*b*) Reverse side.

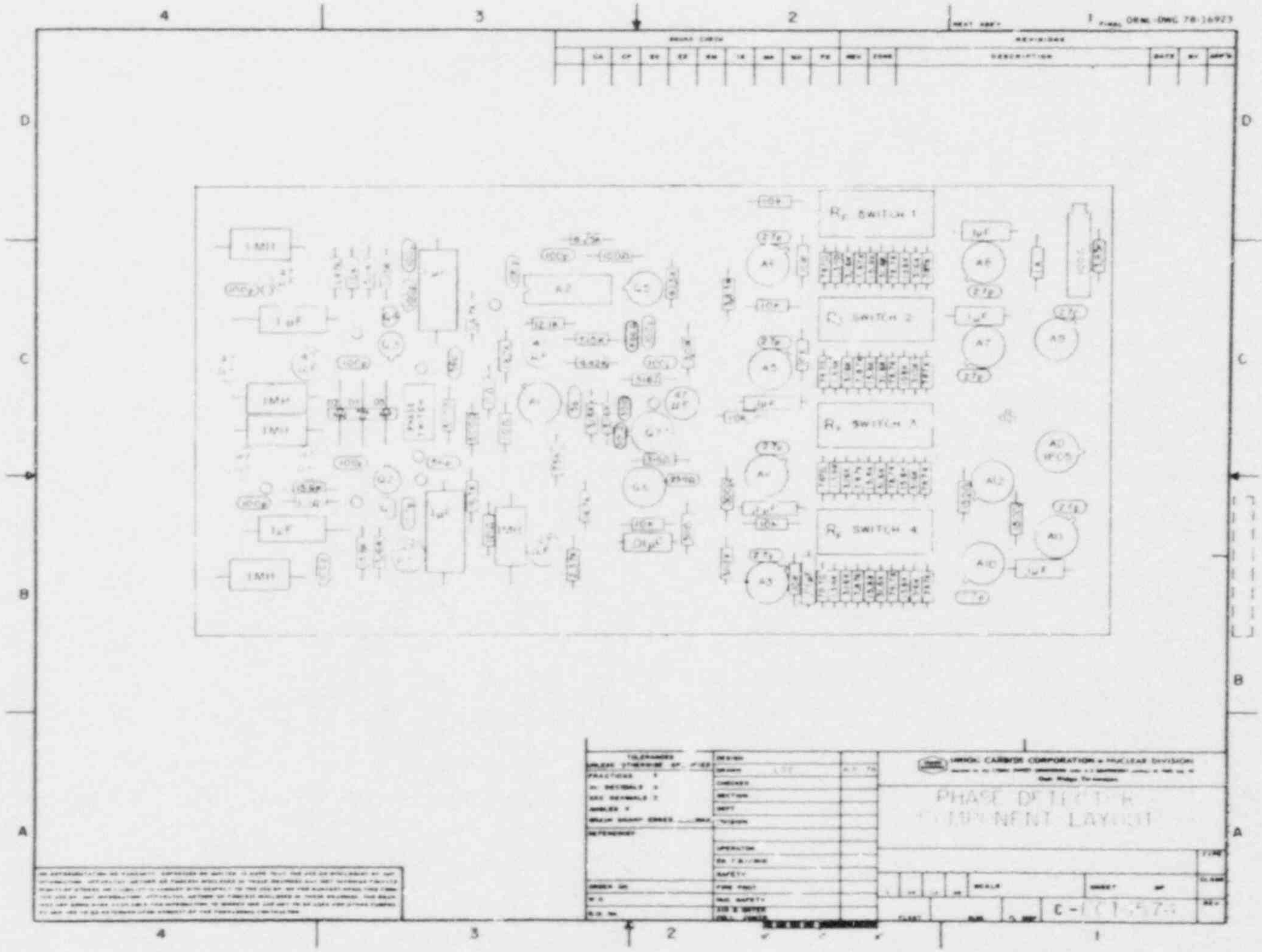
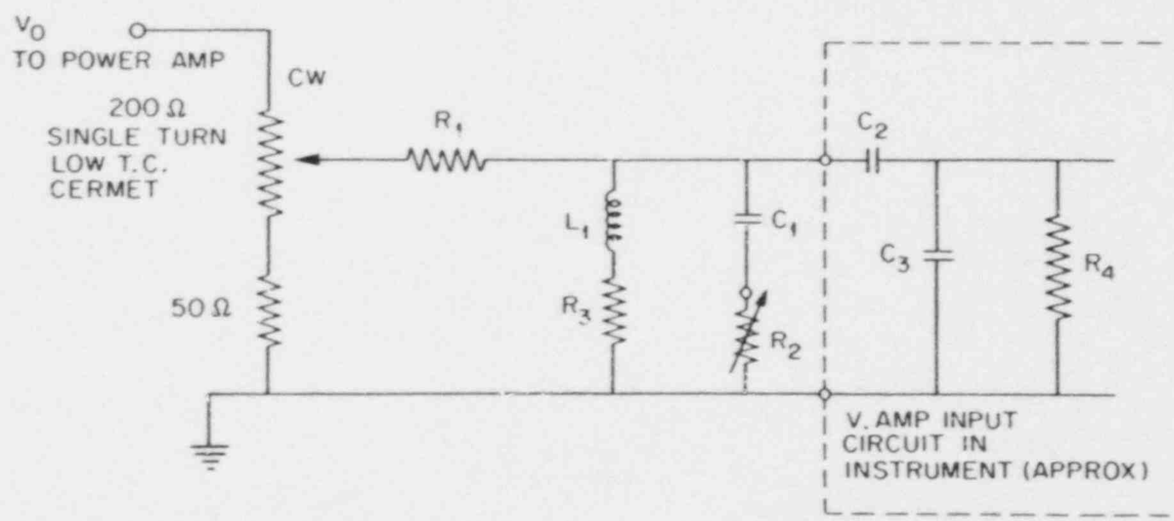


Fig. 20. Component layout of the phase detector board.

ORNL-DWG 78-16749



LINE NO.	SYMBOL	EXPLANATION
100	R_1	SERIES RESISTANCE IN INPUT (Ω)
110	R_3	SERIES RESISTANCE IN INDUCTOR (Ω)
120	R_4	V. AMP INPUT RESISTANCE (Ω)
130	L_1	INDUCTANCE (HENRIES)
150	C_2	V. AMP COUPLING CAPACITANCE (FARADS)
160	C_3	V. A[illegible] + CABLE + CIRCUIT SHUNT CAP.
170	F	FREQUENCY (HERTZ)
180	V_0	POWER AMP INPUT VOLTAGE (VOLTS)
(CALCULATED)	C_1	CAPACITANCE (FARADS)
(CALCULATED)	R_2	RESISTANCE (Ω)

Fig. 21. Simplified diagram of the phase-shift and attenuator network.

This expression is evaluated with the computer program PHSNWK, which is listed in Appendix G. The program will take a given phase shift and adjust the value of R_2 until the desired phase shift is obtained. The circuit diagram for the actual three-frequency calibrator is shown in Fig. 22. The layout is critical, and the transmitting and receiving circuits should be isolated from each other, particularly with the switch in the operate position. The instrument readings can be switched between the calibrator and the probe without removing the signal to either.

The operating mode of the phase calibrator is controlled by the microcomputer which, in turn, is controlled by the second byte sent by the system control computer, either the IBM PC/AT or the IBM 9000. Values between 0 and 3 are all calibrate modes and select the high and low magnitudes and phases. A value of 4 is the normal operate mode for non-multiplexed probes, and values of 6 and 4 are used to toggle the multiplexer board. The value of 5 is used to reset the multiplexer to the zero or initial state, and this is done at the end of every reading cycle. The circuit diagram for the multiplexer board for the three-frequency eddy-current instrument is shown in Fig. 23. The printed circuit board is shown in Fig. 24 and the component layout is shown in Fig. 25. The particular model is for a 16-coil array, although 8-coil multiplexers have also been designed and constructed. The multiplexer is toggled from coil to coil by the switching of the signal from the calibrator board between the high and low state as previously described. This board is fitted in the probe-drive cable, next to the probe so that the coil leads are short. The switching order of the coils is not in order around the circumference. This is done to minimize the interaction between the coils. Since the coils are arranged in two rings around the probe, coils in alternate rings are switched in sequence. This is taken into account in the reading and display programs where the readings are put back in the proper sequence.

COMPUTER MODULE

The computer module is a quadruple-width module that consists of the COMP9B microcomputer board, the IEEE-488 board that plugs into the microcomputer board, and the analog-to-digital converter board. The wiring diagram for the computer module is shown in Fig. 26. The COMP9B microcomputer board is a modification of the COMP9 microcomputer, described in another report.[8] The circuit diagram for the revised board is shown in Fig. 27. Since only the I/O portion of the board has been changed, only this portion of the computer board is shown. The printed circuit layouts for the COMP9B are shown in Fig. 28 and Fig. 29, and the component layout is shown in Fig. 30. The revised board also requires that the computer program to run the board be changed, and the new program, COMP9B, is listed in Appendix A. The circuit diagram for the IEEE-488 board is shown in Fig. 31, with the board layout shown in Fig. 32 and the component layout shown in Fig. 33. This board plugs into the third I/O port of the COMP9B board.

The analog-to-digital converter board consists of eight integrating analog-to-digital converters and the decoding circuitry to allow the computer to address each of them individually. The computer generates a common start pulse that starts the conversion in all converters. The busy

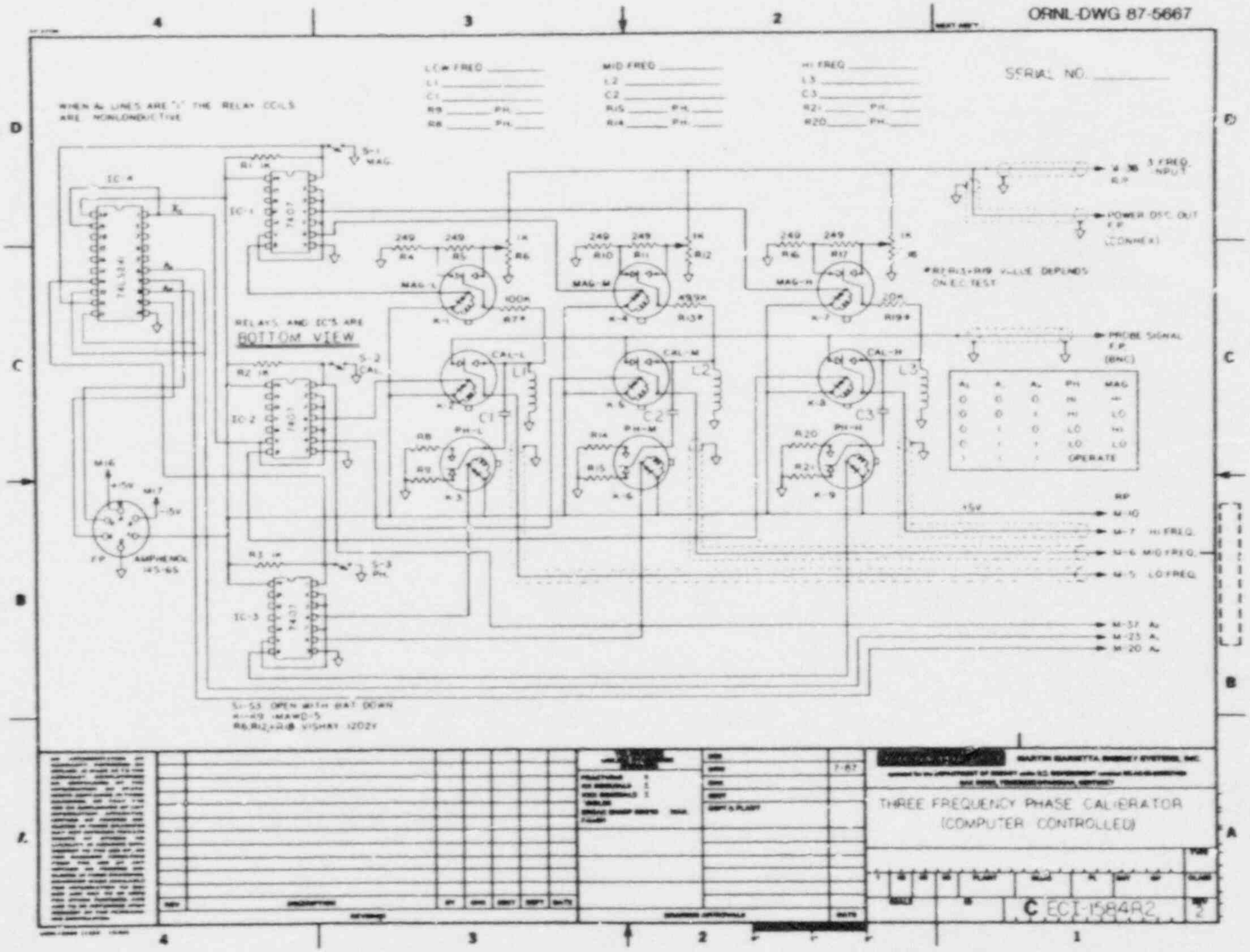


Fig. 22. Circuit diagram for the three-frequency calibrator and multiplexer.

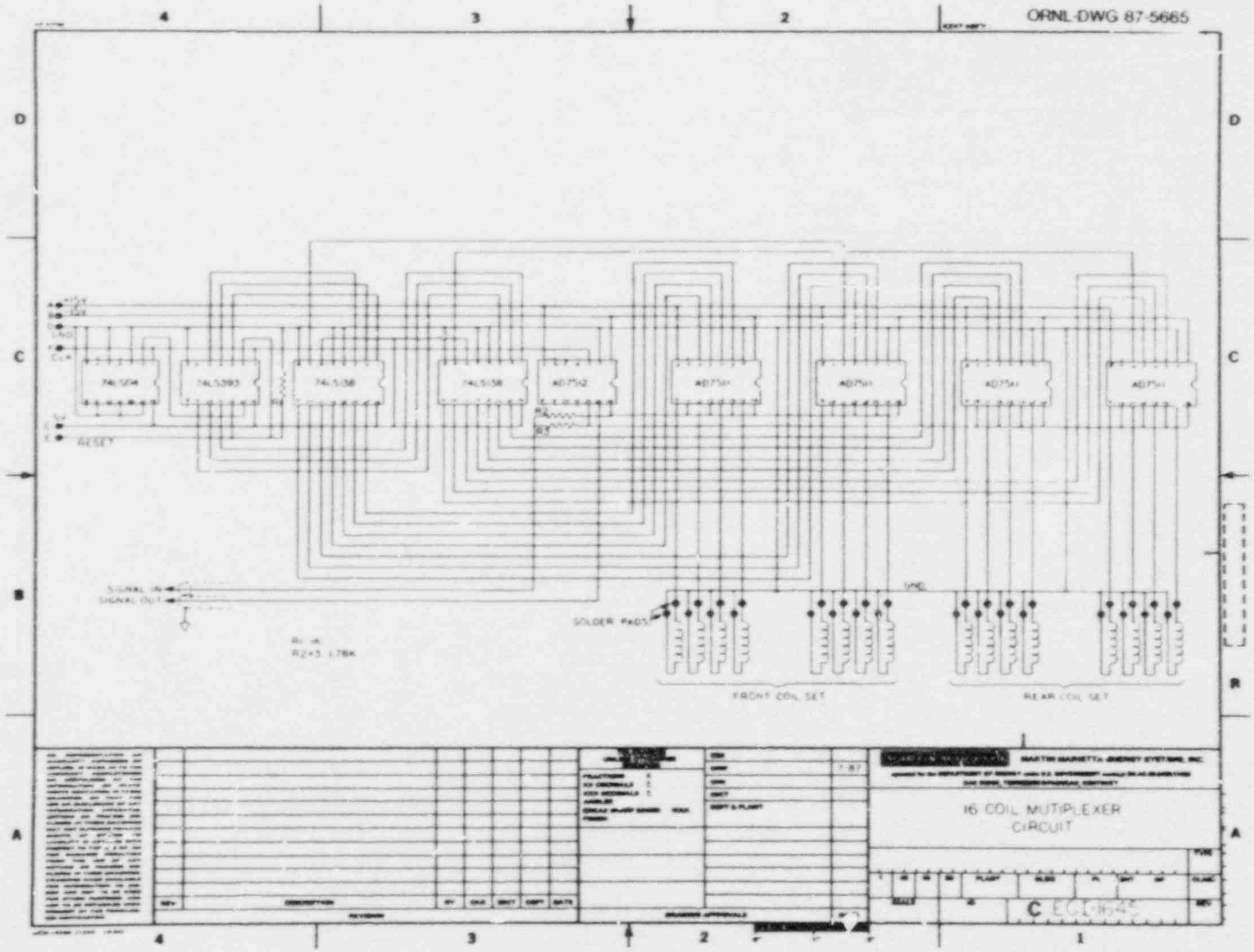


Fig. 23. Probe multiplexer circuit diagram.

ORNL-DWG 87-5675

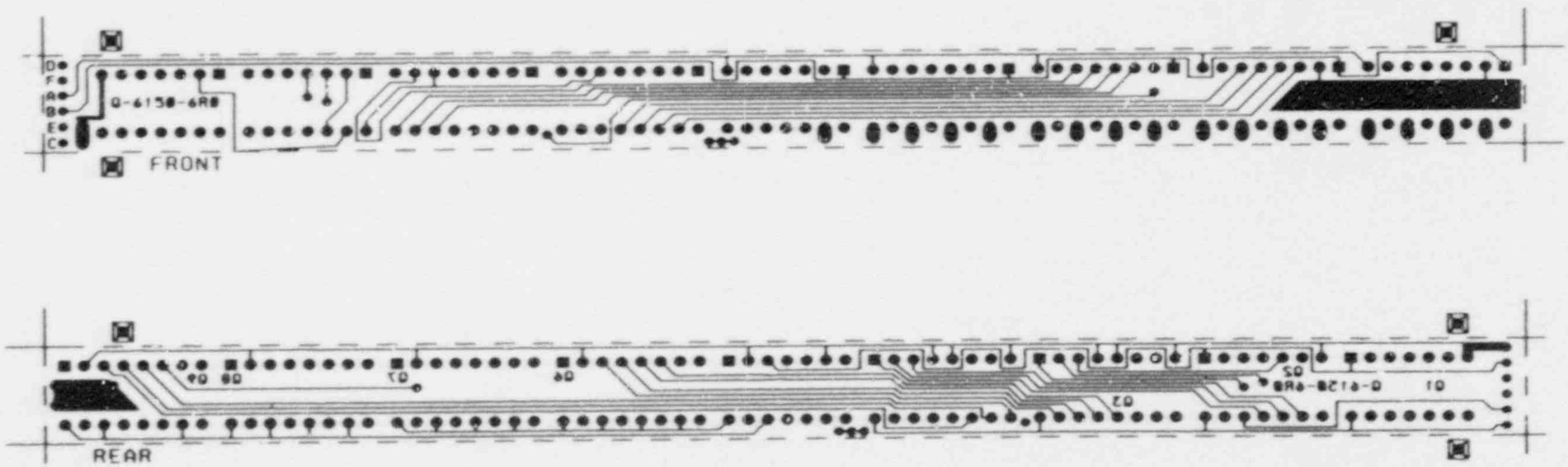


Fig. 24. Probe multiplexer printed circuit board. (*a*) Component side. (*b*) Reverse side.

ORNL-DWG 87-5676

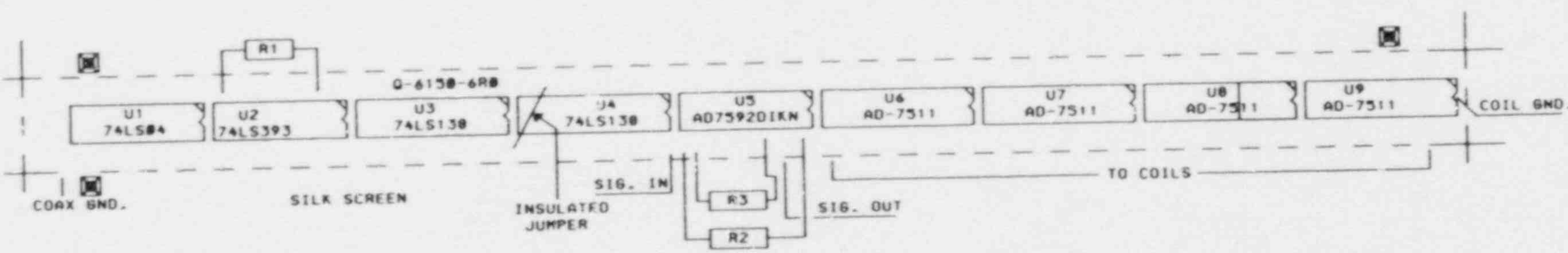


Fig. 25. Probe multiplexer component layout.

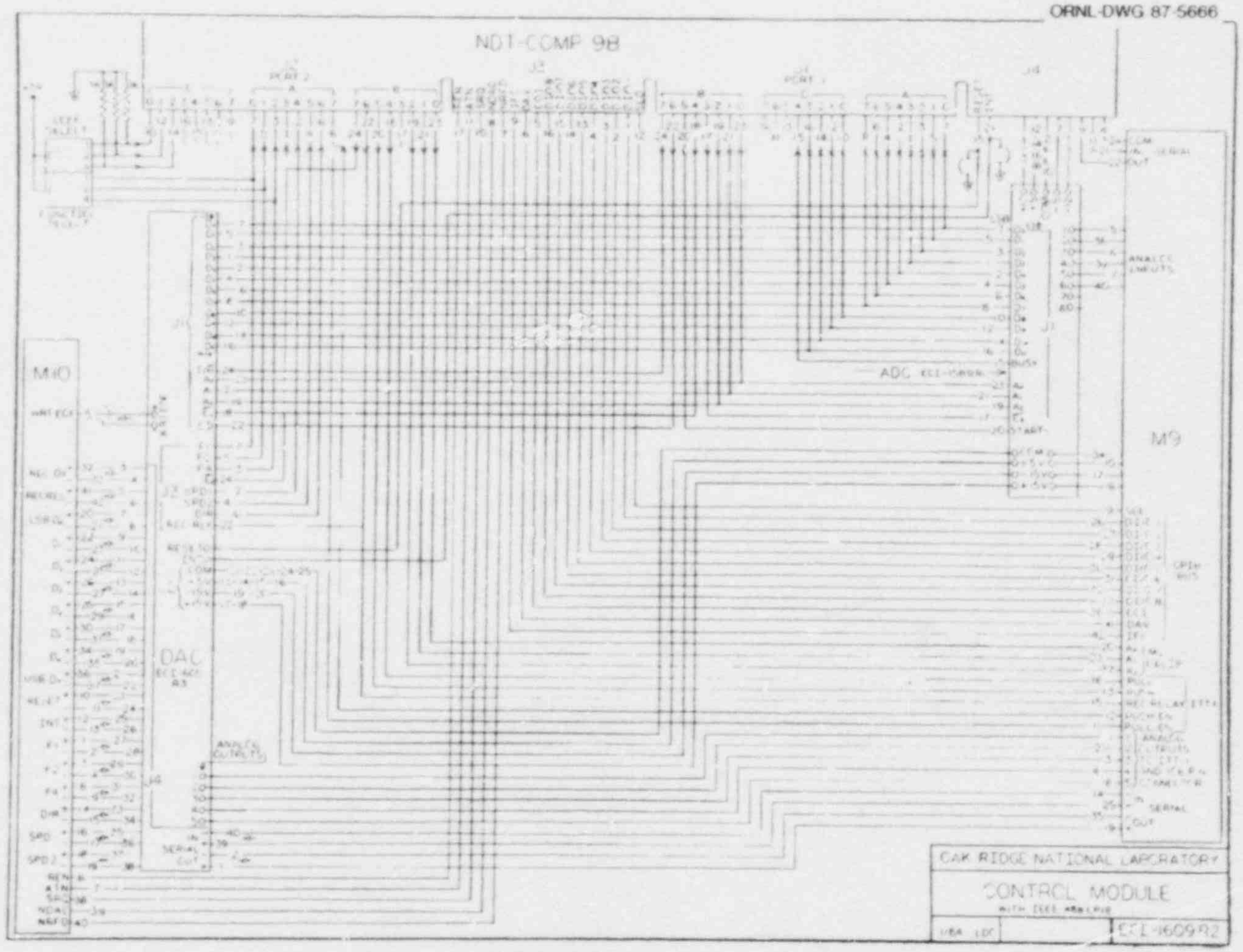


Fig. 26. Computer module wiring diagram with IEEE-488 bus.

ORNL-DWG 87-5669

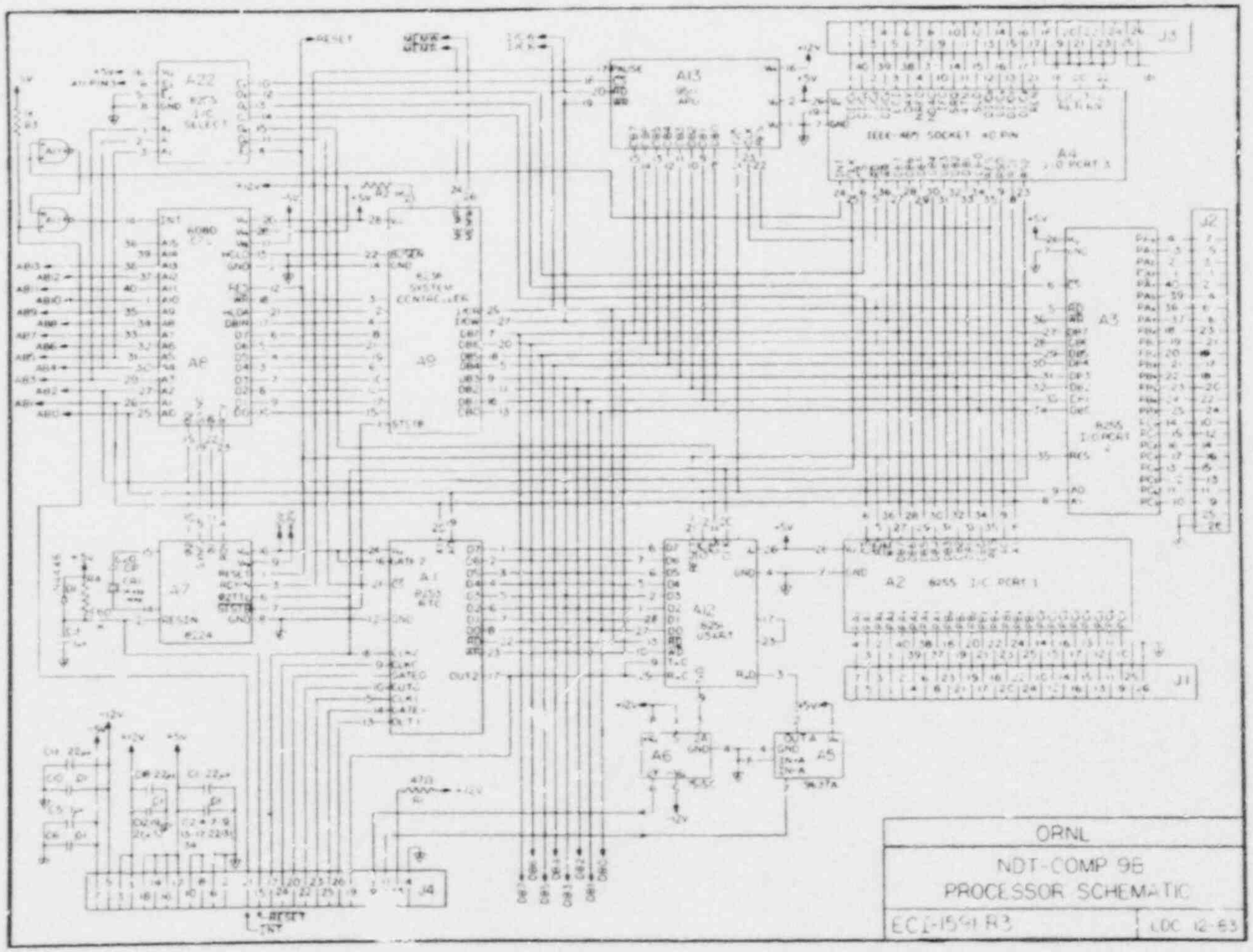


Fig. 27. Circuit diagram for I/O portion of COMP9B microcomputer.

ORNL-DWG 87-5677

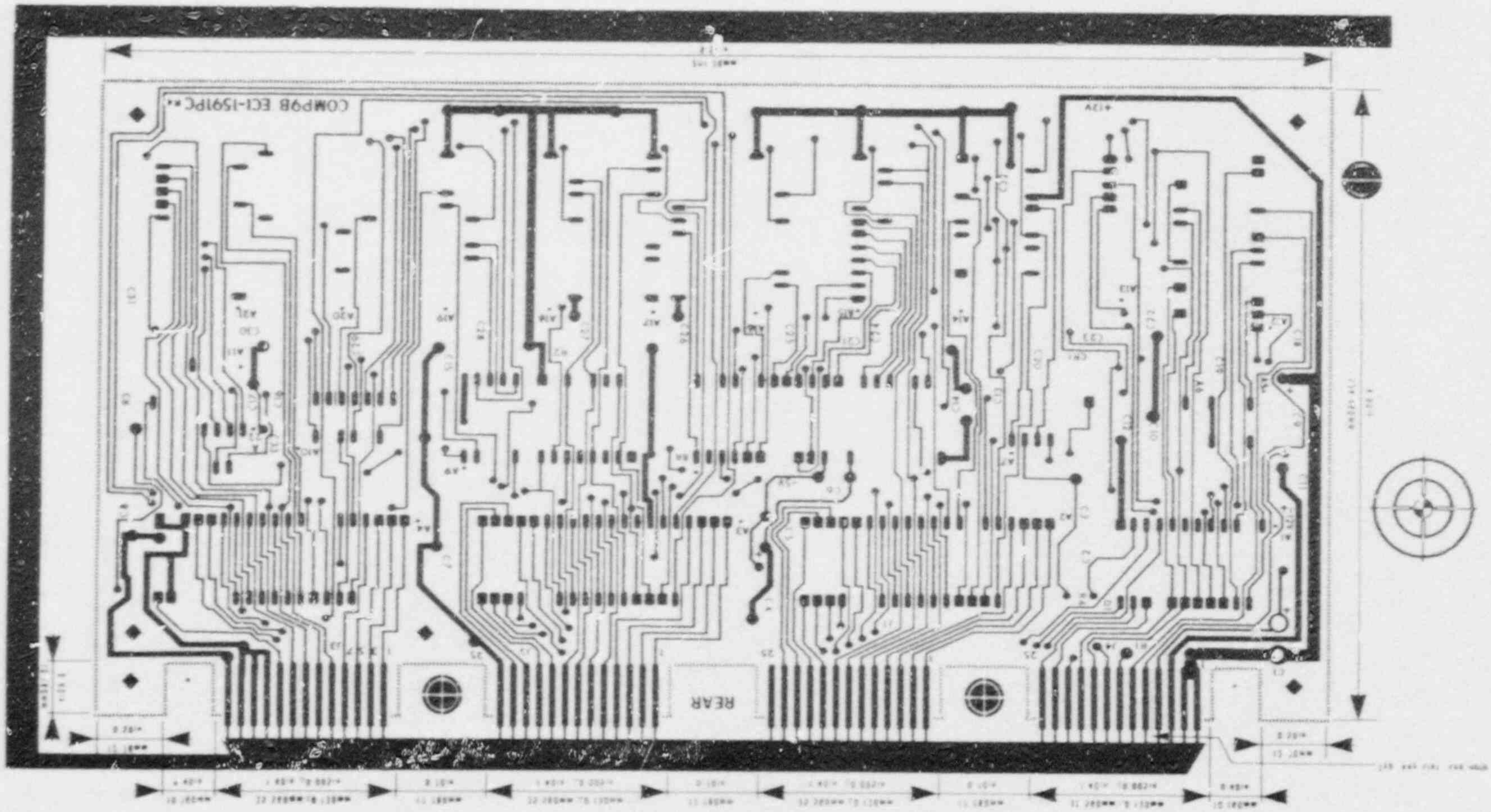


Fig. 28. Printed circuit board for the COMP9B microcomputer, component side.

ORNL-DWG 87-5678

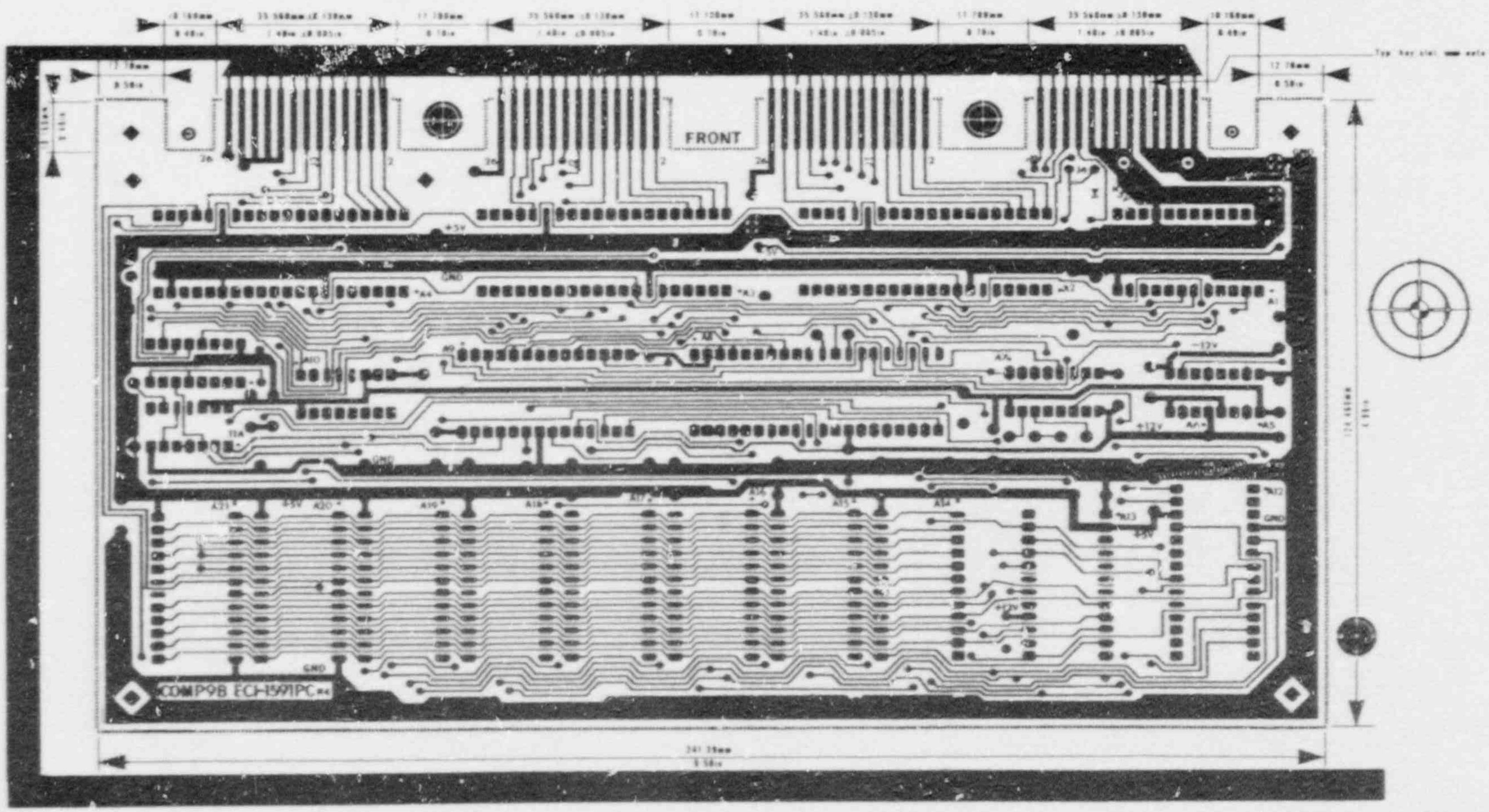


Fig. 29. Printed circuit board for the COMP9B microcomputer, reverse side.

ORNL-DWG 87-5679

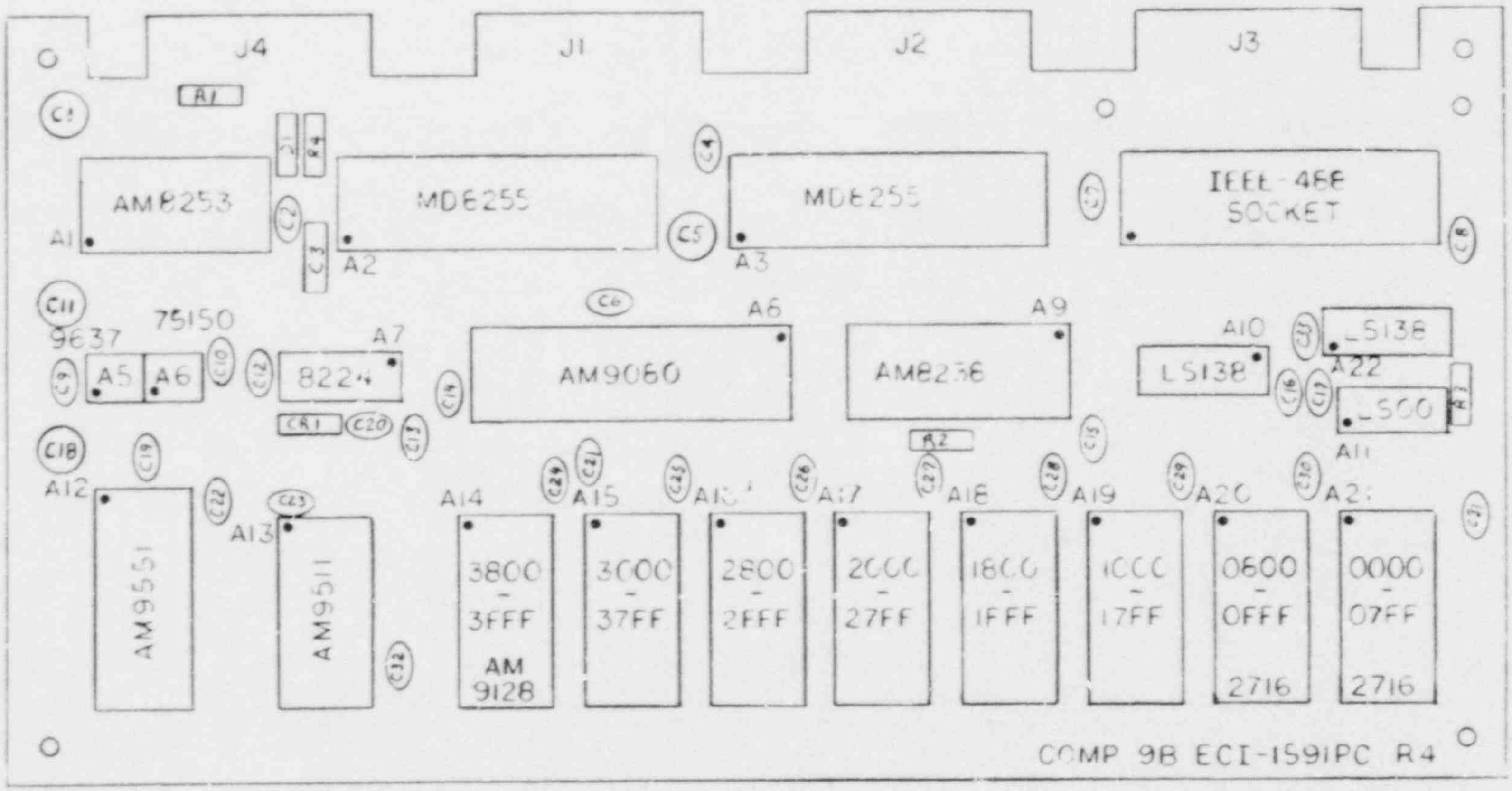


Fig. 30. Component layout for the COMP9B microcomputer.

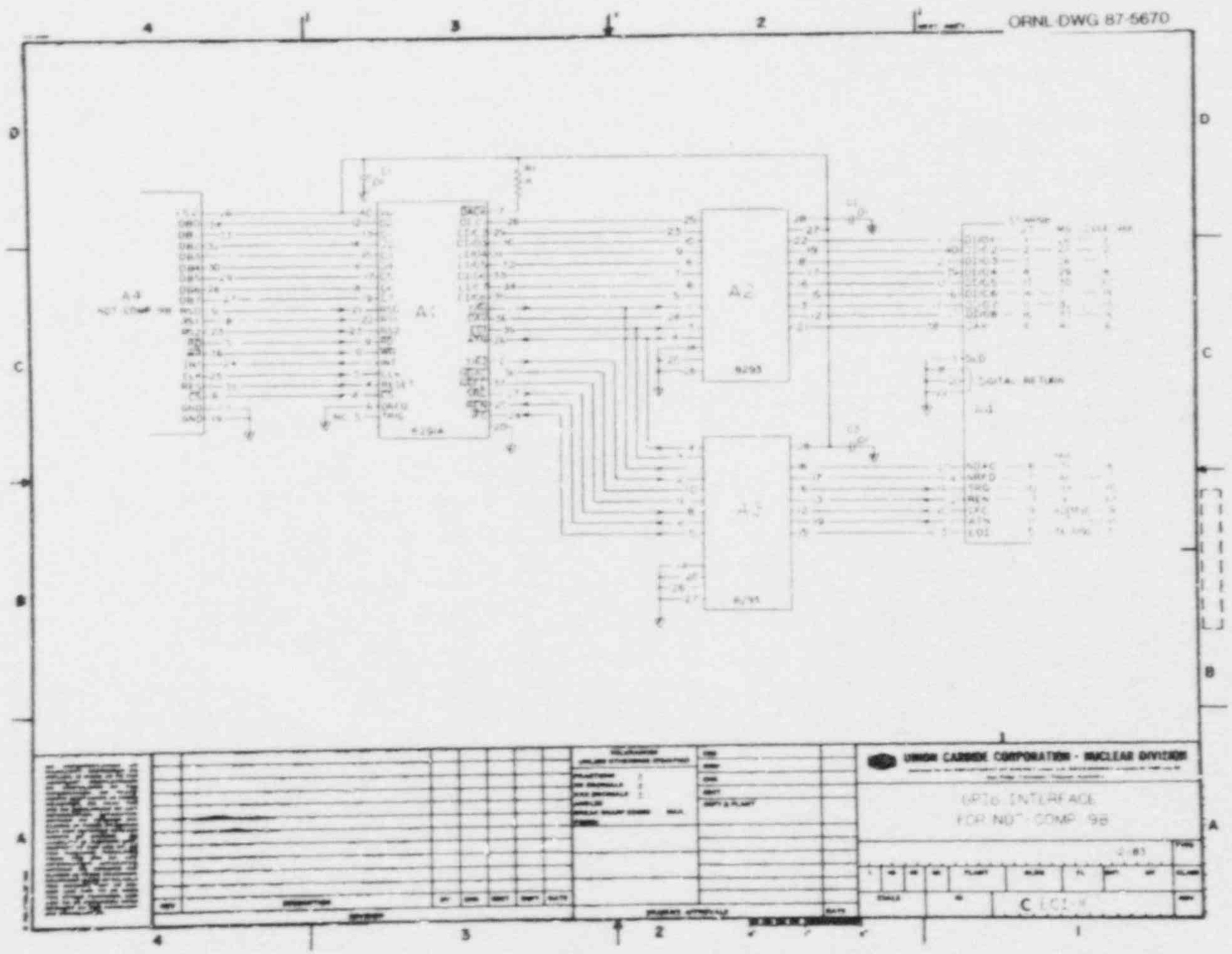
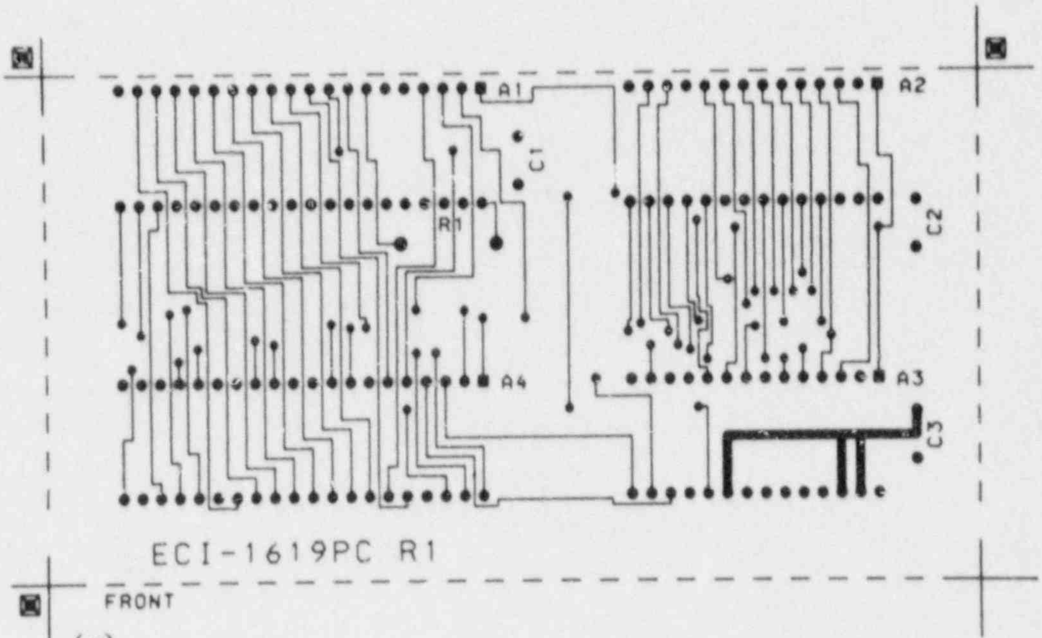


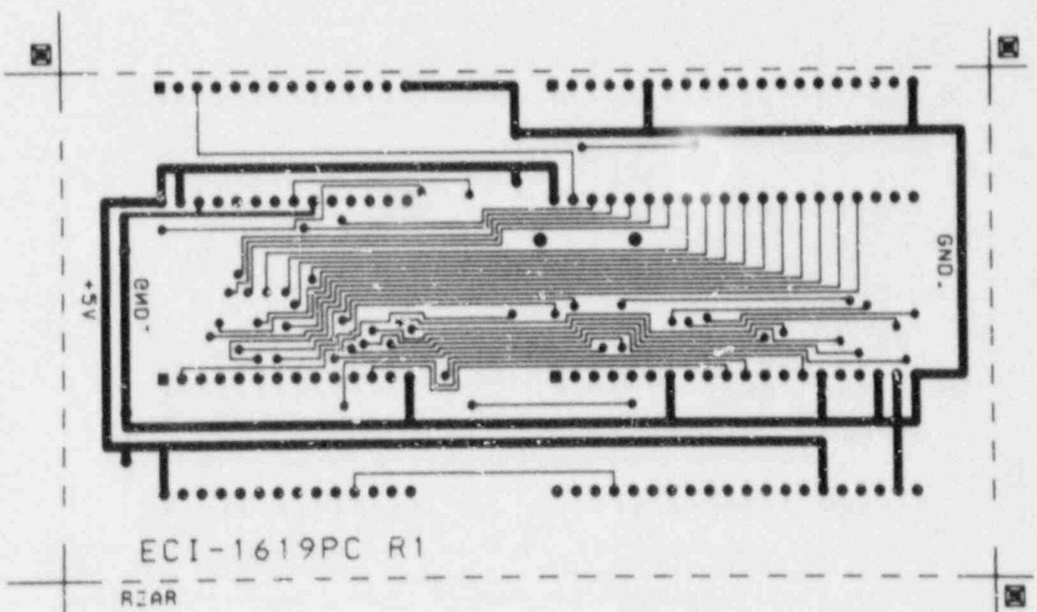
Fig. 31. Circuit diagram for the IEEE-488 board.

ORNL-DWG 87-13994



(*a*)

ORNL-DWG 87-13995



(*b*)

Fig. 32. Printed circuit board for the IEEE-488 board. (*a*) Component side. (*b*) Reverse side.

ORNL-DWG 87-13996

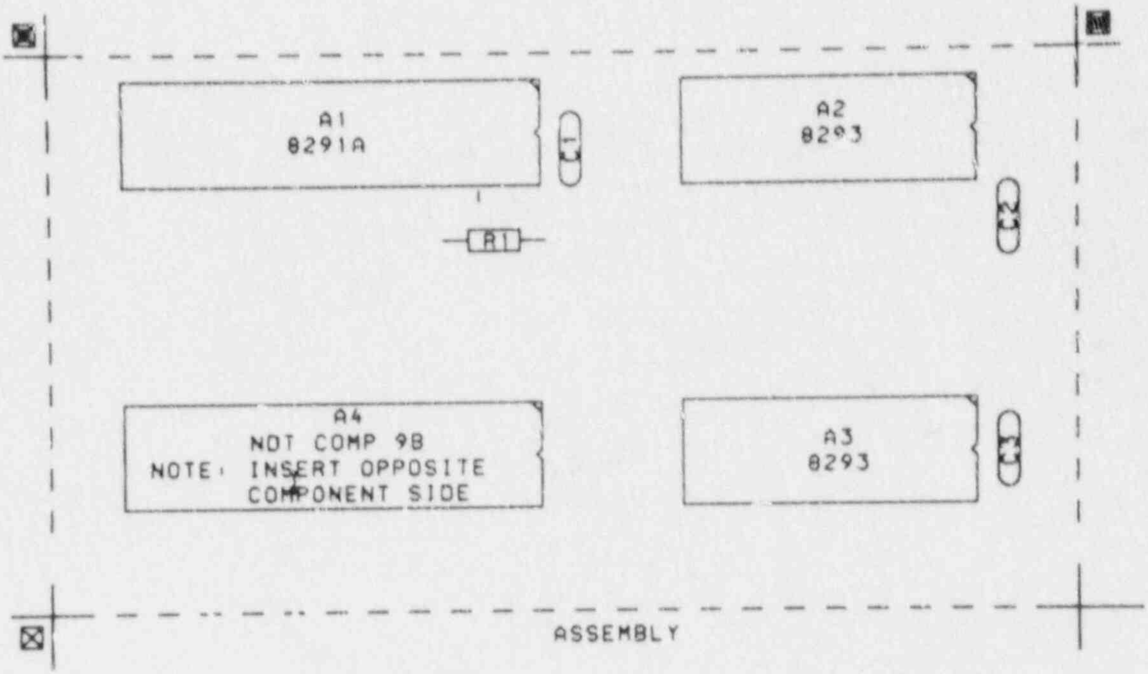


Fig. 33. Component layout for the IEEE-488 board.

line on all of the converters is logically OR-ed together so that a common busy line goes low when they have all completed their conversions. The integration time of each converter is controlled by a single resistor and can be varied from 10 to 20 milliseconds. The board has been operated with all eight converters for some special problems, but the normal mode of operation is with six converters, one for each magnitude and phase at each of the three frequencies. The circuit diagram for the board is shown in Fig. 34 and the printed circuit layout is shown in Fig. 35. The component layout is shown in Fig. 36.

INSTRUMENT OPERATION INSTRUCTIONS

While much of the operation procedure is done by the software programs, there are still a few set-up operations that must be performed manually. These include the selection of operating frequencies, the selection of signal amplitudes and amplifier gains, the selection and connection of probes, the selection of calibration values and ranges, and the set-up of phase detector levels and offsets. We will cover the set-up of the modules individually and in the order they should be performed. Although these detailed instructions may sound laborious and time consuming, they can usually be performed in a matter of hours and in general will not require repeating except when major changes occur in the inspection problem.

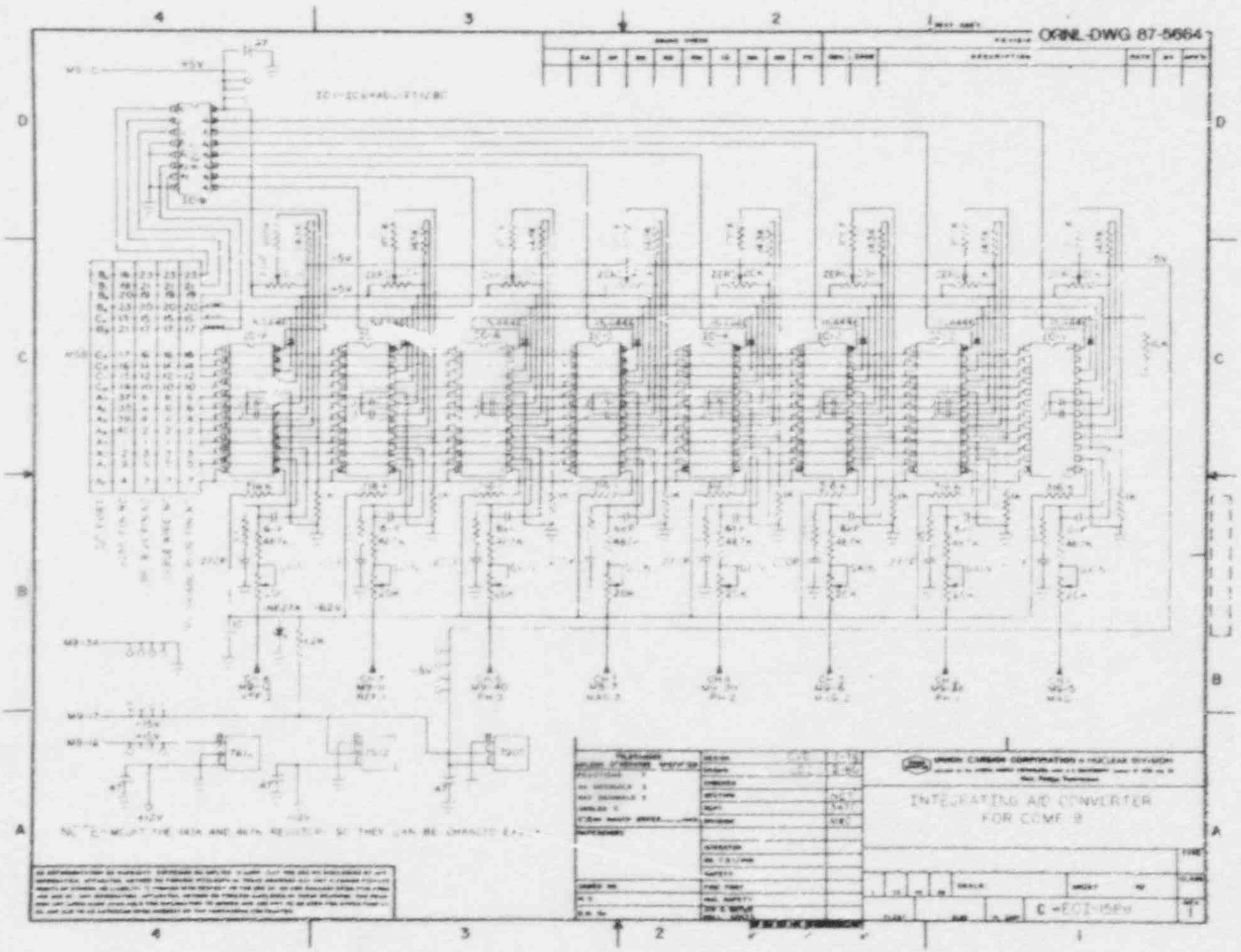
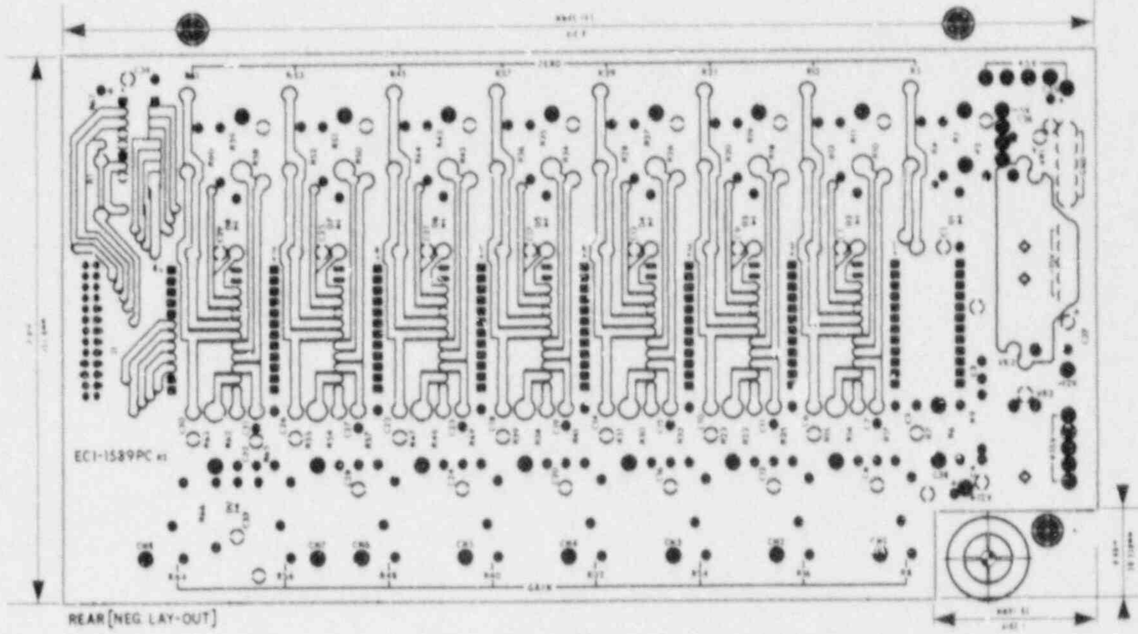


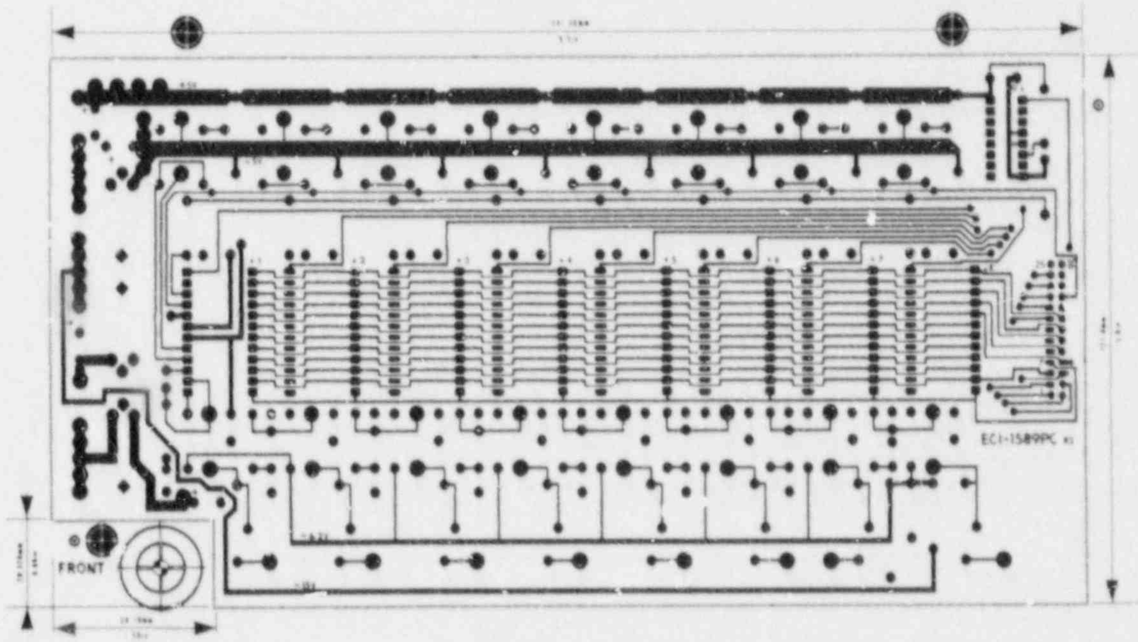
Fig. 34. Circuit diagram for the analog-to-digital converter board.

ORNL-DWG 87-5672



(*a*)

ORNL-DWG 87-5673



(*b*)

Fig. 35. Printed circuit board for the analog-to-digital converter board. (*a*) Component side. (*b*) Reverse side.

ORNL-DWG 87-5674

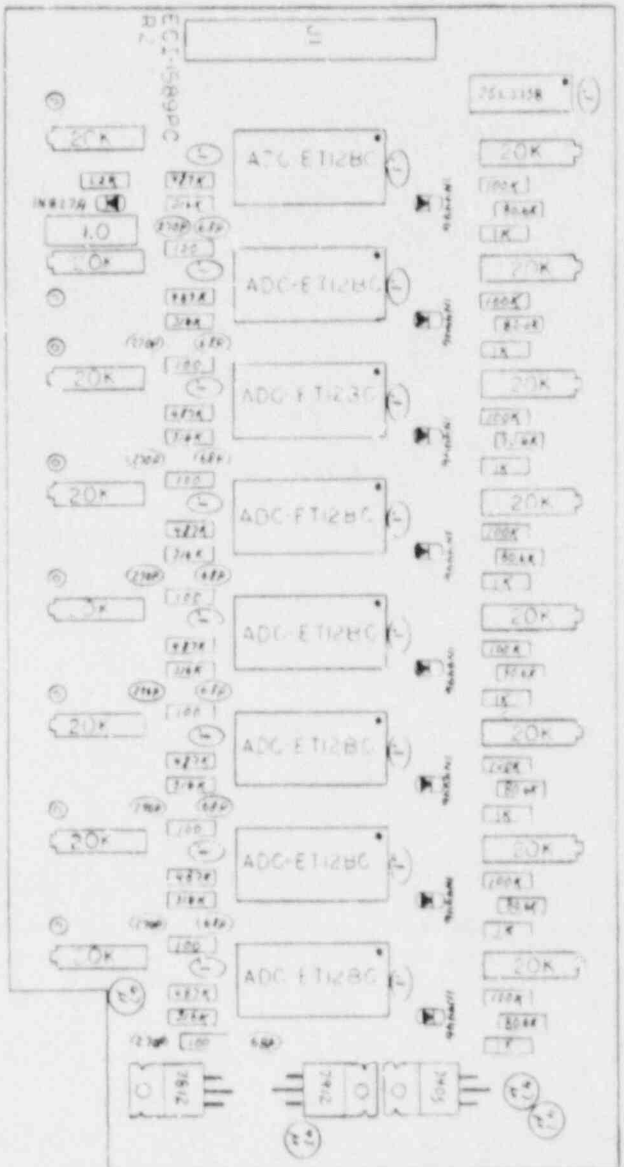


Fig. 36. Component layout for the analog-to-digital converter board.

POWER AMPLIFIER SET-UP

The three operating frequencies are selected using the three rotary switches of the front panel of the power-amplifier module. The operating frequencies, if not known from experience, should be selected by running one of the multiple property analytical-design programs[9,10] or running experimental design studies.[11] The highest frequency should be set using the top switch on the module, the middle frequency set using the middle switch, and the lowest frequency set using the bottom switch. Since the higher frequencies are usually transmitted better than the lower frequencies, the amplitude of the higher frequency is adjusted to be less than the amplitude of the lower frequency by using a larger mixing resistor than is used for the lower frequency. These resistors may need to be changed for a new test, depending on the particular test properties. This is determined by trial and error. In Fig. 37(*a*), we show the multifrequency output of the power amplifier, as measured at the front panel of the power amplifier. Note that the amplitude of the low frequency is greater than the high frequency. After the signal passes through a reflection probe, the low frequencies are attenuated more than the high frequencies, and the high frequencies are then greater than the low frequencies, as shown in Fig. 37(*b*).

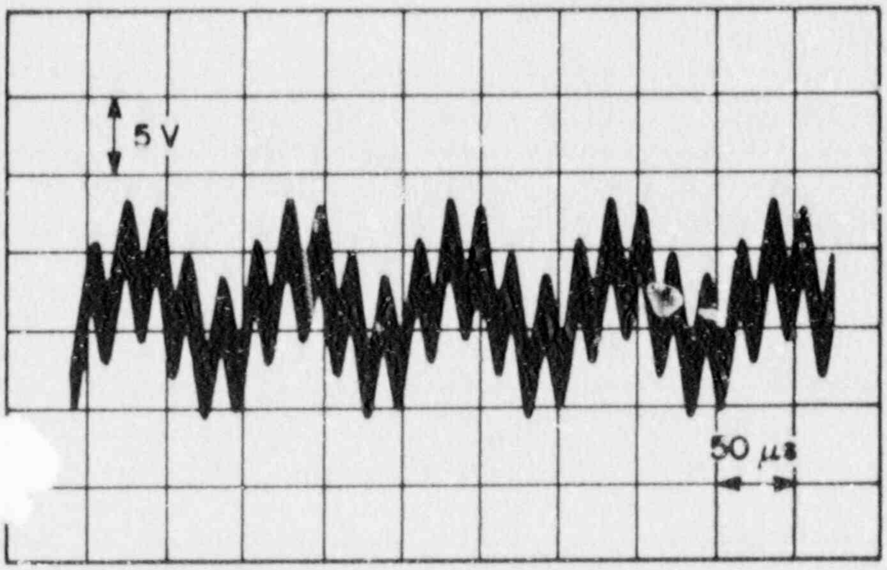
BANDPASS AMPLIFIER SET-UP

The three bandpass amplifiers should be switched to the proper operating frequency, with the amplifier on the right switched to the lowest frequency, the middle amplifier switched to the middle frequency, and the amplifier on the left switched to the highest frequency. The probe should be connected to the proper connections of the calibrator module, and the calibrator module should be switched to the operate position. The probe should be placed on the sample that produces the largest magnitude signal (some experimentation may be necessary to determine this sample). The gain of each bandpass amplifier should be adjusted to a value of about 4.4 V dc, as measured by running the program DIG3. The purpose is to insure that the signal from the probe will be large enough to get a reliable magnitude and phase reading without saturating the bandpass amplifier or the phase detector.

CALIBRATOR AND PHASE DETECTOR SET-UP

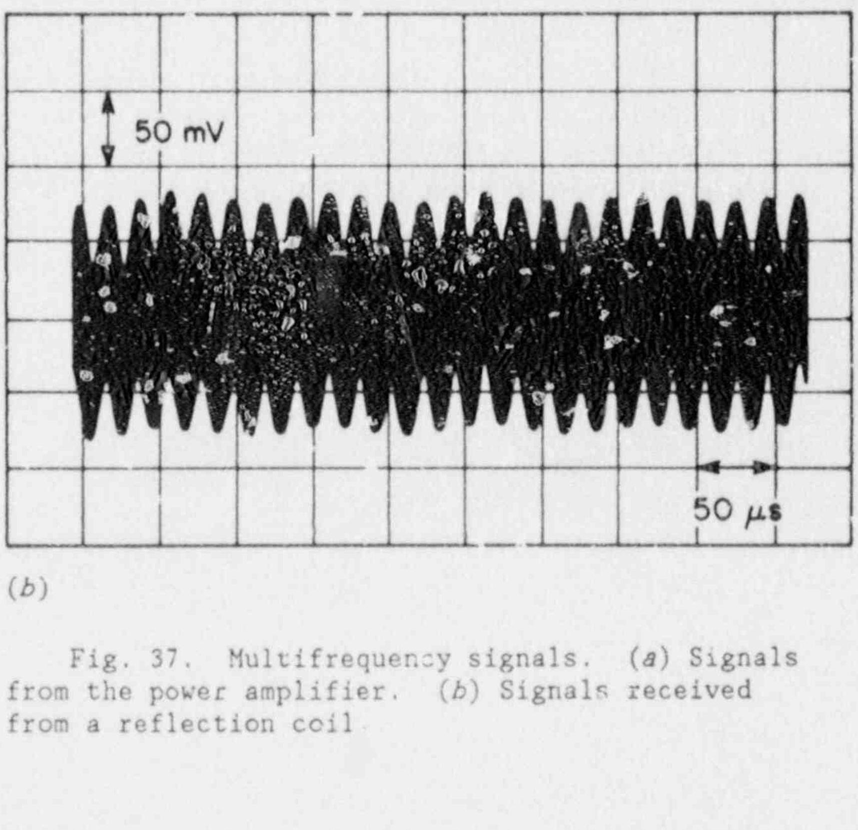
The next two modules must be set up on an interactive basis. The object is to use the calibrator to adjust the phase detector to measure phase shift when the sine wave passes through zero and to determine the amount of balance or offset required to keep the detector output in the measurement range of the analog-to-digital converters. On the other hand, the measurements made by the phase detectors are needed to determine what changes are needed in the calibrator phase shift range to match the range of phase shift encountered by the probe during the test. The analog-to-digital converter can operate over a range of 0.0 to 5.0 V, which represents a phase shift of 50°, more than enough for most tests. The initial values of phase shift can be estimated by measuring the phase

ORNL-DWG 78-16756



(a)

ORNL-DWG 78-16755



(b)

Fig. 37. Multifrequency signals. (*a*) Signals from the power amplifier. (*b*) Signals received from a reflection coil

shift range with an oscilloscope and determining the initial resistor values from the program PHNWRK (see Appendix G). These values installed in the calibrator (see Fig. 22, R8 and R9, R14 and R15, R20 and R21) should be close enough to proceed to the next step. With the calibrator in the calibrate position, and the magnitude switch in the high position, the amplitude must be adjusted for each frequency using the potentiometers R6, R12, and R18. The magnitude at each frequency should read 4.5 V when measured by running the program DIG3. The lift-off setting of the phase detectors should be adjusted so that the high phase shift value does not change as the magnitude is switched from maximum to minimum. The low phase shift may change a small amount. At this point, an adjustment of the balance setting on the phase detector must be made so that the high phase reads about 4.5 V; then the calibrator switch can be set to operate to scan the samples. The value of the measured phase should fall between the high and low phase of the calibrator. If not, the measured phases are used to determine how much the calibrator range should be shifted and to compute and install a new set of resistors; then the last part of the calibration process must be repeated. The magnitude and phase readings need only to be adjusted to within a few millivolts of the desired voltages, using first the coarse and then the fine lift-off and balance dials.

INSTRUMENT OPERATION AND PROGRAM USE

Once the instrument is set up, it may then be used to perform a variety of measurements. In general, we will first perform a series of calibration readings. These should be done on a set of standards that are representative of the range of measurements that the instrument will encounter in the actual test. The development of the standards that span the range of variation in test properties that will be encountered is the most difficult part of the process of developing a multifrequency, multiproperty eddy-current test. Since eddy-current readings are such a nonlinear function of test property variations, the array of standard readings must contain enough intermediate values to adequately define the functions used. The program BIGRDG (See Appendix C) is used to take the readings. It will also control a three-axis positioner so that the probe can be positioned with respect to the standards. An input data file must contain the positions of the axis and the properties of the standards at each position. Errors in the standards will result in errors in the eddy-current readings made during the actual inspection. BIGRDG positions the standards, makes the readings, and stores the instrument readings and test properties that produced these readings in a data file. A set of readings may consist of several thousand different property combinations, and the positions and readings are usually repeated three times to average out any small error due to standard position.

Next, the program BIGFIT (see Appendix D) is run to take the reading data and the property data and compute a least squares set of coefficients that will allow the properties to be computed from nonlinear functions of the readings. The operator selects the type of nonlinear function, the degree of the polynomial, and the number of cross terms between the magnitude and the phase on an interactive basis. BIGFIT will compute the coefficients and two error terms, one for how well the coefficients fit the readings to the properties and the other for how much a drift in the readings will change the calculated property. When the errors are small enough to satisfy the operator, he can save the coefficients in a data file for the next program to run. BIGFIT will also take additional readings from the instrument and compute the readings on a real-time basis. The program PLTRDG (see Appendix E) can then be run to make programmed motion scans on the standards or different parts using the laboratory positioners. The computed value of the properties is graphed out on a printer as the sample is scanned. Versions of this program have been run using faster scanning devices and disk or tape storage media to speed up the inspection rate. For the faster programs, the coefficients and the nonlinear polynomial equations that contain them are coded directly into the programs using factoring to reduce the number of multiplications required. This small extra amount of programming greatly increases the computation speed and reduces the size of the programs.

An example of a major application has been the development of in-service inspections for steam generator tubing for light water reactors. That system incorporated fast scanning probe drives and the necessary data collection for real-time analysis of the results. As discussed at the beginning of the calibration section, once the data and calibration have been established with the BIGRDG, BIGFIT, and PLTRDG programs, the inspection system is ready to perform many extensive examinations over a long period of time unless basic changes occur, such as major changes in the tube dimensions or the discovery of a new unexpected property in the tubing. Verification of the proper operation of the system can be confirmed by passing the probe through standards with known artifacts. This is normally done at the end of each tube scan, and slight changes in the calibration that may occur due to temperature variations or the use of a new probe can be corrected mathematically.

SUMMARY AND CONCLUSIONS

A three-frequency eddy-current instrument has been developed that allows the performance of multiple property eddy-current examinations, such as steam generator tubing inspections. This instrument, along with the associated software, allows this complex type of problem to be solved. The use of the IEEE-488 interface bus and the PC/AT as the controlling computer allows the instrument to be applied with standard, well-documented, and inexpensive accessory equipment.

ACKNOWLEDGMENTS

The authors express their appreciation to Wayne King and Ott Smith for their help in laying out the printed circuit boards, to N. W. McCoy for his help in constructing the instrument, to K. V. Cook and B. E. Foster for reviewing the manuscript, and to Mary R. Upton for typing the manuscript.

REFERENCES

1. J. H. Smith, C. V. Dodd, and L. D. Chitwood, *Multifrequency Eddy Current Inspection of Seam Weld in Steel Sheath*, ORNL/TM-9470, April 1985.

2. C. V. Dodd and W. E. Deeds, "Design Considerations for Multiple Frequency Eddy-Current Tests," pp. 121–23 in "Quantative NDE in the Nuclear Industry," *Proceedings of the Fifth International Conference on Nondestructive Evaluation in the Nuclear Industry, San Diego, California, May 10-13, 1982*, ed. R. B. Clough, American Society for Metals, Metals Park, Ohio, 1983.

3. C. V. Dodd and L. D. Chitwood, *Three-Frequency Eddy-Current Instrument for Multiple Property Problems*, ORNL-5495, March 1979.

4. C. V. Dodd et al., *The Analysis of Reflection Type Coils for Eddy-Current Testing*, ORNL-TM-4107, April 1973.

5. C. V. Dodd and W. A. Simpson, *Thickness Measurements Using Eddy-Current Techniques*, ORNL-TM-3712, March 1972.

6. C. V. Dodd and W. E. Deeds, "Absolute Eddy-Current Measurement of Electrical Conductivity," pp. 387-94 in *Rev. Prog. Quant. Nondstr. Eval.* (Proc. AF/DARPA Rev. Progr. Quant. NDE, Boulder, Colo., August 2-7, 1981), vol. 1, ed. D. O. Thompson and D. E. Chimenti, Plenum Publishing Corporation, 1982.

7. *Universal Active Filter*, Application Note UAF41, Burr-Brown Research Corp., Tucson, Ariz., September 1976.

8. C. V. Dodd, and R. F. Cowan, *The NDT-COMP9 Microcomputer*, NUREG/CR-158; ORNL/NUREG/TM-390, September 1980.

9. W. E. Deeds, C. V. Dodd, and G. W. Scott, *Computer-Aided Design of Multifrequency Eddy-Current Tests for Layered Conductors with Multiple Property Variations*, ORNL/TM-6858, October 1979.

10. W. E. Deeds and C. V. Dodd, *Multiple Property Variations in Coaxial Cylindrical Conductors Determined with Multiple-Frequency Eddy Currents*, NUREG/CR-0967, ORNL/NUREG/TM-335, November 1979.

11. C. V. Dodd, L. M. Whitaker, and W. E. Deeds, *An Eddy-Current Laboratory Test System Using Commercial Equipment*, ONRL-6366, April 1987.

APPENDIX A

The COMP9B PROGRAM

The COMP9B program is very similar to the COMP9 program with one very important exception. When execution first starts, it transfers to the port set-up subroutine, PRTSU, which is in the second PROM. This subroutine sets all the ports and supplies the other initial values for the particular job that will be run. After the execution of the subroutine, the monitor log-on message is printed on a CRT (if one is connected), and the program waits for either a monitor command to be typed or an interrupt to be received over the IEEE-488 bus. The interrupt will transfer control to the first location of the second PROM, where the type will be determined and executed. Since the port set-up must be run prior to the IEEE-488 interrupts, the COMP9B program will not work properly by itself without a PROM in the second position. A listing of the program COMP9B follows:

```
        TITLE   'NDT-COMP9B MONITOR ASSEMBLY VER. 3 APRIL 1983'
;            PORTS ARE SET FOR GPIB BOARD,ALSO PORT SETUP CALL
        NAME    COMP9
        LIST    B,G,O,T ; LIST SYMBOL TABLE ONLY - PUT SYMBOLS
                        ; INTO OBJECT MODULE TO ALLOW CREATION
                        ; OF A LOAD MAP
;       NLIST   I,M,S,X ; DO NOT LIST SOURCE TEXT
 PUBLIC         CI,CNVBN,CO,COPDT,CROUT,DCMD,ECHO,ERROR,FRET
 PUBLIC         GCMD,GETCH,GETCM,GETHX,GETNM,HILO,ICMD,NMOUT
 PUBLIC         GETC,PRVAL,SCMD,VALDG,VALDL,XCMD,ZEROM,SGNON
 EXTRN        PRTSU
        CSEG            ; USE RELOCATABLE CODE COUNTER
;
;
;*****************************************************************
;*****************************************************************
;
;
;           NDT-COMP9 MONITOR
;
;           AUTHORS C. V. DODD AND R. F. COWAN
;
;           NONDESTRUCTIVE TESTING GROUP
;           METALS AND CERAMICS DIVISION
;           OAK RIDGE NATIONAL LABORATORIES
;           OAK RIDGE, TENNESSEE 37831
;
;           NOTE:  THE COMP9 MONITOR WAS ADAPTED FROM THE COMP8,
;                  WHICH IS A MODIFIED VERSION OF THE INTEL 8080A
;                  BOARD MONITOR.
;
;
;
;*****************************************************************
;*****************************************************************
;
;
;
;
```

```
;*****************************************************************
;
;
;
;          SYMBOL DEFINITIONS
;
;
;
;*****************************************************************
;
;
BRCHR  EQU     1BH      ; ASCII CODE FOR BREAK CHAR (ESCAPE)
BRTAB  EQU     3FAH     ; LOCATION OF START OF BRANCH TABLE IN PROM
CMD    EQU     27H      ; COMMAND INSTRUCTION FOR USART INITIALIZATION
CNCTL  EQU     0C7H     ; CONSOLE (USART) CONTROL PORT
CNIN   EQU     0C6H     ; CONSOLE INPUT PORT
CNOUT  EQU     0C6H     ; CONSOLE OUTPUT PORT
CONST  EQU     0C7H     ; CONSOLE STATUS INPUT PORT
CR     EQU     0DH      ; ASCII CODE FOR CARRIAGE RETURN
DATA   EQU     16*1024-256 ; START OF MONITOR RAM USAGE AT
                           ; (MEM SIZE IN K)*1K-256 BYTES
ESC    EQU     1BH      ; ASCII CODE FOR BREAK CHAR
HCHAR  EQU     0FH      ; MASK TO SELECT LOWER HEX CHAR FROM BYTE
INVRT  EQU     0FFH     ; MA[illegible]K TO INVERT HALF BYTE FLAG
LF     EQU     0AH      ; A[illegible]I CODE FOR LINE FEED
LOWER  EQU     0        [illegible]OTES LOWER HALF OF BYTE IN ICMD
;LSGNON  --  LENGTH OF [illegible]ON MESSAGE - DEFINED LATER
MODE   EQU     0CEH     ; MODE SET FOR USART INITIALIZATION
;NCMDS  --  NUMBER OF VALID COMMANDS - DEFINED LATER
NLSB   EQU     00DH     ; LSB OF RTC DIVIDE COUNT
NMSB   EQU     00H      ; MSB OF RTC DIVIDE COUNT
NEWLN  EQU     0FH      ; MASK FOR CHECKING MEMORY ADDR DISPLAY
PRTY0  EQU     07FH     ; MASK TO CLEAR PARITY BIT FROM CONSOLE CHAR
RBR    EQU     2        ; MASK TO TEST RECEIVER STATUS FOR A 1
REGS   EQU     DATA+255-18 ; START OF REGISTER SAVE AREA
RSTU   EQU     38H      ; TRANSFER LOC FOR RST7 INSTRUCTION
RTCCTL EQU     0E7H     ; REAL-TIME CLOCK CONTROL PORT
RTCDAT EQU     0E6H     ; REAL-TIME CLOCK DATA (PORT#2)
RTCMD  EQU     0BEH     ; MODE WORD FOR REAL-TIME CLOCK
;RTABS  --  SIZE OF ENTRY IN RTAB TABLE - DEFINED LATER
TERM   EQU     1BH      ; ASCII CODE FOR ICMD TERMINATING CHAR (ESCAPE)
TRDY   EQU     1        ; MASK TO TEST TRANSMITTER STATUS
UPPER  EQU     0FFH     ; DENOTES UPPER HALF OF BYTE IN ICMD
USINT  EQU     0800H    ; INTERRUPT BRANCH LOCATION
;
;
;
;
```

```
;*****************************************************************
;
;
;           MONITOR RAM ADDRESS DEFINITIONS
;
;
;*****************************************************************
;
;
MSTAK  EQU     3FEDH   ; START OF MONITOR STACK
ASAVE  EQU     3FF2H   ; REGISTER SAVE LOCS
BSAVE  EQU     3FF0H
CSAVE  EQU     3FEFH
DSAVE  EQU     3FEEH
ESAVE  EQU     3FEDH
FSAVE  EQU     3FF1H
HSAVE  EQU     3FF4H
LSAVE  EQU     3FF3H
PSAVE  EQU     3FF5H   ; PC SAVE LOC
SSAVE  EQU     3FF7H   ; USER SP SAVE LOC
TEMP   EQU     3FF9H   ; TEMPORARY MONITOR CELL
;
;
;*****************************************************************
;
;
;           MONITOR MACROS
;
;
;*****************************************************************
;
;
TRUE   MACRO   WHERE   ; BRANCH IF FUNCTION RETURNS TRUE (SUCCESS)
       JC      WHERE
       ENDM
;
;
FALSE  MACRO   WHERE   ; BRANCH IF FUNCTION RETURNS FALSE (FAILURE)
       JNC     WHERE
       ENDM
;
;
;
;
;*****************************************************************
;
;
;           RESTART ENTRY POINT
;
;
;*****************************************************************
;
;
```

```
GO:     SHLD    LSAVE   ; SAVE HL REGISTERS
        POP     H       ; GET TOP OF STACK ENTRY
        SHLD    PSAVE   ; ASSUME THIS IS LAST PC
        LXI     H,0     ; CLEAR HL
        DAD     SP      ; GET STACK POINTER VALUE
        SHLD    SSAVE   ; SAVE USER'S STACK POINTER
        LXI     H,ASAVE+1 ; NEW VALUE FOR STACK POINTER
        SPHL            ; SET MONITOR STACK POINTER FOR REG SAVE
        PUSH    PSW     ; SAVE A AND FLAGS
        PUSH    B       ; SAVE B AND C
        PUSH    D       ; SAVE D AND E
;
;
;*****************************************************************
;
;
;          8253 REAL-TIME CLOCK INITIALIZATION
;
;
;          THE REAL-TIME CLOCK MUST BE INITIALIZED FIRST, SINCE IT
;          DETERMINES THE CLOCK FREQUENCY AND HENCE THE BAUD RATE
;          FOR THE USART.  A MODE WORD IS SENT FIRST SPECIFYING WHICH
;          COUNTER AND MODE, AND THEN A 16-BIT INTEGER N TO PROVIDE A
;          DIVIDE-BY-N FUNCTION.  THE TABLE BELOW SPECIFIES WHICH
INTEGER
;          TO USE FOR A GIVEN TIME-BASE:
;
;         BAUD RATE     N WITH 18.432 XTAX      N WITH 18.000 XTAL
;           9600              000D                    000D
;           4800              001B                    001A
;           2400              0035                    0034
;           1200              006B                    0068
;            300              01AA                    01A1
;
;          THIS ASSUMES A DIVIDE-BY-16 FACTOR FOR THE USART.
;
;          THE MODE WORD SPECIFIES:
;
;          SELECT COUNTER #2
;          LOAD LSB FIRST, THEN MSB OF N
;          RUN IN MODE 3 (SQUARE WAVE OUTPUT)
;          LOAD IN BINARY
;
;
;*****************************************************************
;
;
```

```
START: MVI     A,RTCMD
       OUT     RTCCTL  ; SEND MODE WORD TO RTC
       MVI     A,NLSB
       OUT     RTCDAT  ; SEND LSB OF DIVIDE COUNT
       MVI     A,NMSB
       OUT     RTCDAT  ; SEND MSB OF DIVIDE COUNT
;
;
;
;
;*****************    ****************************************
;
;
;          USART INITIALIZATION CODE
;
;
;          THE USART IS ASSUMED TO COME UP IN THE RESET POSITION (THIS
;          FUNCTION IS TAKEN CARE OF BY THE HARDWARE).  THE USART WILL
;          BE INITIALIZED FOR A CRT INTERFACE.  THE PARAMETERS BELOW
;          APPLY:
;
;          MODE INSTRUCTION
;          ----------------
;
;          2 STOP BITS
;          PARITY DISABLED
;          8 BIT CHARS
;          INTERNAL FREQUENCY DIVIDE-BY-16
;
;          COMMAND INSTRUCTION
;          ------ -----------
;
;          NO HUNT MODE
;          NOT(RTS) FORCED TO ZERO
;          RECEIVE ENABLED
;          DATA TERMINAL READY
;          TRANSMIT ENABLED
;
;
;**************************************************************
;
;
       MVI     A,MODE  ;,16X,2 STOP BITS,NO PARITY,8 BIT CHARACTER.
       OUT     CNCTL   ; OUTPUT MODE SET TO USART
       MVI     A,CMD
       OUT     CNCTL   ; OUTPUT COMMAND WORD TO USART
;
;
;**************************************************************
;
;
;          PRINT LOGON MESSAGE
;
```

```
;
;**********************, ****************************************
;
;
      CALL      PRTSU   ;SET UP MICROCOMPUTER PORTS
       MVI      B,LSGNON ; COUNTER FOR CHARACTERS IN MESSAGE
MSGL:  MOV      C,M      ; FETCH NEXT CHAR TO C REG
       CALL     CO       ; SEND IT TO THE CONSOLE
       INX      H        ; POINT TO NEXT CHARACTER
       DCR      B        ; DECREMENT BYTE COUNTER
       JMP      PRSGN     ;JUMP AROUND SUBROUTINE,INTERRUPT BRANCH
;
       ORG      RSTU     ; ORG TO RST TRANSFER LOC
       JMP      USINT    ; JUMP TO USER INTERRUPT LOC
;
PRSGN:JNZ     MSGL     ; RETURN FOR NEXT CHARACTER
;
;
;***************************************************************
;
;
;           ROUTINE GETCM
;
;           THIS ROUTINE RECOGNIZES COMMAND CHAR'S INPUT BY THE USER.
;           UPON RECEIPT OF A CHAR FROM THE TERMINAL, IT CHECKS IF
;           IT IS IN THE COMMAND CHARACTER TABLE.  IF THE CHAR IS FOUND,
;           THE CORRECT ROUTINE IS SELECTED FROM A TABLE OF COMMAND
ROUTINE
;           ADDRESSES AND CONTROL IS TRANSFERRED TO THAT ROUTINE.  IF THE
;           ORIGINAL CHAR IS NOT A VALID COMMAND, AN ERROR CONDITION IS
;           CREATED.THIS IS THE NORMAL RETURN POINT TO MONITOR FOR TEST
;           ROUTINES.A PROMPT "." IS PRINTED WHEN MONITOR IS READY.
;
;           EXTERNAL REFERENCES:
;           --------------------
;
;           ECHO      ERROR     GETCH
;
;           REGISTERS AFFECTED: A,B,C,H,L,FLAGS
;
;
;***************************************************************
;
;
GETCM: LXI      H,MSTAK ; ALWAYS WANT TO RESET STACK PTR TO MONITOR
       SPHL              ; /STARTING VALUE SO ROUTINES NEEDN'T CLEAN UP
       MVI      C,'.'    ; PROMPT CHARACTER TO C
       CALL     ECHO     ; SEND PROMPT CHAR TO USER TERMINAL
GETC   EI                ; ENABLE INTERRUPTS
       CALL     GETCH    ; GET COMMAND CHARACTER TO A
       CALL     ECHO     ; ECHO CHARACTER TO USER
```

```
        MOV     A,C      ; PUT COMMAND CHAR INTO ACCUMULATOR
        LXI     B,NCMDS  ; C CONTAINS LOOP AND INDEX COUNT
        LXI     H,CTAB   ; HL POINTS INTO COMMAND TABLE
GTC05:  CMP     M        ; COMPARE TABLE ENTRY AND CHAR
        JZ      GTC10    ; BRANCH IF EQUAL - COMMAND RECOGNIZED
        INX     H        ; ELSE, INCREMENT TABLE POINTER
        DCR     C        ; DECREMENT LOOP COUNT
        JNZ     GTC05    ; BRANCH IF NOT AT TABLE END
        JMP     ERROR    ; ELSE, COMMAND CHAR IS ILLEGAL
GTC10:  LXI     H,CADR   ; IF GOOD COMMAND, LOAD ADDRESS OF TABLE
        DAD     B        ; ADD WHAT IS LEFT OF LOOP COUNT
        DAD     B        ; ADD AGAIN - EACH ENTRY IN CADR IS 2 BYTES LONG
        MOV     A,M      ; GET LSP OF ADDRESS OF TABLE ENTRY TO A
        INX     H        ; POINT TO NEXT BYTE IN TABLE
        MOV     H,M      ; GET MSP OF ADDRESS OF TABLE ENTRY TO H
        MOV     L,A      ; PUT LSP OF ADDRESS OF TABLE ENTRY INTO L
        PCHL             ; NEXT INSTRUCTION COMES FROM COMMAND ROUTINE
;
;
;**************************************************************
;**************************************************************
;
;
;           COMMAND IMPLEMENTING ROUTINES
;
;
;**************************************************************
;**************************************************************
;
;
;           ROUTINE DCMD
;
;           THIS ROUTINE IMPLEMENTS THE DISPLAY MEMORY (D) COMMAND.
;
;           EXTERNAL REFERENCES:
;           --------------------
;
;           CROUT     ECHO      GETCM     GETNM     HILO
;           NMOUT
;
;           REGISTERS AFFECTED:  ALL
;
;
;**************************************************************
;
;
DCMD:   MVI     C,2      ; GET 2 NUMBERS FROM INPUT STREAM
        CALL    GETNM
        POP     D        ; ENDING ADDRESS TO DE
        POP     H        ; STARTING ADDRESS TO HL
DCM05:  CALL    CROUT    ; ECHO CARRIAGE RETURN/LINE FEED
        MOV     A,H      ; DISPLAY ADDRESS OF FIRST LOCATION IN LINE
```

```
        CALL    NMOUT
        MOV     A,L      ; ADDRESS IS 2 BYTES LONG
        CALL    NMOUT
DCM10:  MVI     C,' '
        CALL    ECHO     ; USE BLANK AS SEPARATOR
        MOV     A,M      ; GET CONTENTS OF NEXT MEMORY LOC
        CALL    NMOUT    ; DISPLAY CONTENTS
        CALL    BREAK    ; SEE IF USER WANTS OUT
        TRUE    DCM12    ; IF SO, BRANCH
        CALL    HILO     ; SEE IF ADDRESS OF DISPLAYED LOCATION IS
                         ; /GREATER THAN OR EQUAL TO ENDING ADDRESS
        FALSE   DCM15    ; IF NOT, MORE TO DISPLAY
DCM12:  CALL    CROUT    ; CARRIAGE RETURN/LINE FEED TO END LINE
        JMP     GETCM    ; ALL DONE
DCM15:  INX     H        ; IF MORE TO GO, POINT TO NEXT LOC TO DISPLAY
        MOV     A,L      ; GET LOW ORDER BITS OF NEW ADDRESS
        ANI     NEWLN    ; SEE IF LAST HEX DIGIT OF ADDRESS DENOTES
                         ; /START OF NEW LINE
        JNZ     DCM10    ; NO - NOT AT END OF LINE
        JMP     DCM05    ; YES - START NEW LINE WITH ADDRESS
;
;
;*************************************************************
;
;
;          ROUTINE GCMD
;
;          THIS ROUTINE IMPLEMENTS THE BEGIN EXECUTION (G) COMMAND.
;
;          EXTERNAL REFERENCES:
;          --------------------
;
;          ERROR     GETHX     RSTTF
;
;          REGISTERS AFFECTED:  ALL
;
;
;*************************************************************
;
;
GCMD:   CALL    GETHX    ; GET ADDRESS (IF PRESENT) FROM INPUT STREAM
        FALSE   GCM05    ; BRANCH IF NO NUMBER PRESENT
        MOV     A,D      ; ELSE, GET TERMINATOR
        CPI     CR       ; SEE IF CARRIAGE RETURN
        JNZ     ERROR    ; ERROR IF NOT PROPERLY TERMINATED
        LXI     H,PSAVE  ; WANT NUMBER TO REPLACE SAVE PGM COUNTER
        MOV     M,C
        INX     H
        MOV     M,B
        JMP     GCM10
GCM05:  MOV     A,D      ; IF NO STARTING ADDRESS, MAKE SURE THAT
        CPI     CR       ; /CARRIAGE RETURN TERMINATED COMMAND
```

```
        JNZ     ERROR   ; ERROR IF NOT
GCM10:  JMP     RSTTF   ; RESTORE REGISTERS AND BEGIN EXECUTION
;
;
;***************************************************************
;
;
;            ROUTINE  ICMD
;
;            THIS ROUTINE IMPLEMENTS THE INSERT INTO MEMORY (I) COMMAND.
;
;            EXTERNAL REFERENCES:
;            --------------------------
;
;            CNVBN   CROUT   ECHO    ERROR   GETCH
;            GETNM   STHLF   VALDG   VALDL
;
;            REGISTERS AFFECTED:  ALL
;
;
;***************************************************************
;
;
ICMD:   MVI     C,1
        CALL    GETNM   ; GET SINGLE NUMBER FROM INPUT STREAM
        MVI     A,UPPER
        STA     TEMP    ; TEMP WILL HOLD THE UPPER/LOWER HALF BYTE FLAG
        POP     D       ; DE GETS ADDR OF START
ICM05:  CALL    GETCH   ; GET ONE CHAR FROM INPUT STREAM
        MOV     C,A
        CALL    ECHO    ; ECHO THE CHAR
        MOV     A,C     ; RESTORE IT INTO A
        CPI     TERM    ; IS IT A TERMINATING CHAR?
        JZ      ICM25   ; YES - NO MORE CHARS COMING IN
        CALL    VALDL   ; NO - IS IT A VALID DELIMITER?
        TRUE    ICM05   ; YES - IGNORE IT
        CALL    VALDG   ; NO - WELL, IS IT A VALID HEX DIGIT?
        FALSE   ICM20   ; NO - MUST BE AN ERROR
        CALL    CNVBN   ; YES - AT LAST A GOOD DIGIT.  CONVERT IT TO
BINARY
        MOV     C,A     ; MOVE RESULT TO C
        CALL    STHLF   ; STORE IN APPROPRIATE HALF WORD
        LDA     TEMP    ; GET HALF BYTE FLAG
        ORA     A       ; SET FLAGS
        JNZ     ICM10   ; BRANCH IF FLAG SET FOR UPPER
        INX     D       ; IF LOWER, INC ADDR OF DESTINATION BYTE
ICM10:  XRI     INVRT   ; TOGGLE FLAG
        STA     TEMP    ; PUT NEW VALUE OF FLAG BACK
        JMP     ICM05   ; LOOP AND DO ANOTHER CHAR
ICM20:  CALL    STHFO   ; ILLEGAL CHAR
        JMP     ERROR   ; FILL BYTE AND ERROR
ICM25:  CALL    STHFO   ; BRANCH TO HERE WHEN INPUT FINISHED
```

```
        CALL    CROUT   ; PRINT A CR,LF
        JMP     GETCM   ; RETURN TO COMMAND INTERPRETER
;
;
;****************************************************************
;
;
;       ROUTINE MCMD
;
;       THIS ROUTINE IMPLEMENTS THE MOVE DATA WITHIN MEMORY (M)
;       COMMAND.
;
;       EXTERNAL REFERENCES:
;       --------------------
;
;       GETCM     GETNM
;
;       REGISTERS AFFECTED:  ALL
;
;
;****************************************************************
;
;
MCMD:   MVI     C,3
        CALL    GETNM   ; GET 3 INPUT VALUES
        POP     D       ; DE=DESTINATION ADDRESS OF FIRST BYTE
        POP     B       ; BC=END ADDRESS OF BLOCK TO BE TRANSFERRED
        POP     H       ; HL=STARTING ADDRESS OF BLOCK TO BE XFERED
       CALL     COPLP   ;CALL COPY DATA LOOP
       JMP      GETCM   ;JUMP BACK TO GET COMMAND PROGRAM
COPLP:  MOV     A,M     ; MOVE BYTE IN MEM TO A
        STAX    D       ; STORE SAME BYTE IN LOC ADDRESSED BY DE
        INX     H
        INX     D
COPDT:  MOV     A,C     ; SET UP TEST TO DETERMINE IF HL>BC
        SUB     L       ; SUBTRACT HL FROM BC - GET A "BORROW"
        MOV     A,B     ; /WHEN HL>BC (CARRY IS SET)
        SBB     H       ; THEN RETURN TO CALL SUB
        RC      GETCM   ; RETURN IF DONE (HL>BC)
        JMP     COPLP   ; OTHERWISE DO ANOTHER BYTE
;
;
;****************************************************************
;
;
;       ROUTINE SCMD
;
;       THIS ROUTINE IMPLEMENTS THE SUBSTITUTE IN MEMORY (S) COMMAND.
;
;       EXTERNAL REFERENCES:
;       --------------------
;
;
```

```
;            ECHO       GETCM      GETHX      NMOUT
;
;            REGISTERS AFFECTED:  ALL
;
;
;*****************************************************************
;
;
SCMD:  CALL    GETHX    ; GET A NUMBER, IF PRESENT, FROM INPUT
       PUSH    B
       POP     H        ; PUT IT INTO HL
SCM05: MOV     A,D      ; GET TERMINATOR
       CPI     ' '      ; SEE IF SPACE
       JZ      SCM10    ; YES - CONTINUE
       CPI     ','      ; NO - THEN CHECK IF COMMA
       JNZ     GETCM    ; NO - TERMINATE COMMAND
SCM10: MOV     A,M      ; GET CONTENTS OF SPECIFIED LOC
       CALL    NMOUT    ; DISPLAY THEM
       MVI     C,'-'
       CALL    ECHO     ; USE DASH FOR SEPARATOR
       CALL    GETHX    ; GET NEW VALUE FOR MEM LOC, IF ANY
       FALSE   SCM15    ; IF NO VALUE THERE, THEN ADVANCE
       MOV     M,C      ; OTHERWISE STORE LSB OF VALUE ENTERED
SCM15: INX     H        ; INCREMENT ADDR OF MEM LOC TO EXAMINE
       JMP     SCM05
;
;
;*****************************************************************
;
;
;            ROUTINE XCMD
;
;            THIS ROUTINE IMPLEMENTS THE REGISTER EXAMINE AND CHANGE (X)
;            COMMAND.
;
;            EXTERNAL REFERENCES:
;            -------------------
;
;            CROUT      ECHO       ERROR      GETCH      GETCM
;            GETHX      NMOUT      RGADR      REGDS
;
;            REGISTERS AFFECTED:  ALL
;
;
;*****************************************************************
;
;
XCMD:  CALL    GETCH    ; GET REGISTER IDENTIFIER
       MOV     C,A
       CALL    ECHO     ; ECHO IT
       MOV     A,C
       CPI     CR
```

```
        JNZ     XCM05   ; BRANCH IF NOT A CR
        CALL    REGDS   ; OTHERWISE, SHOW REG CONTENTS
        JMP     GETCM   ; AND RETURN TO COMMAND INTERPRETER
XCM05:  MOV     C,A     ; GET REGISTER IDENTIFIER TO C
        CALL    RGADR   ; CONVERT INTO RTAB ADDR
        PUSH    B
        POP     H       ; PUT POINTER TO REGISTER ENTRY INTO HL
        MVI     C,' '
        CALL    ECHO    ; ECHO SPACE TO USER
        MOV     A,C
        STA     TEMP    ; PUT SPACE INTO TEMP AS DELIMITER
XCM10:  LDA     TEMP    ; GET TERMINATOR
        CPI     ' '     ; IS IT A SPACE?
        JZ      XCM15   ; YES - GO CHECK POINTER INTO TABLE
        CPI     ','     ; NO - WELL, IS IT A COMMA?
        JNZ     GETCM   ; NO - THEN IT MUST BE A CR
XCM15:  MOV     A,M
        ORA     A       ; SET FLAGS
        JNZ     XCM18   ; BRANCH IF NOT AT END OF TABLE
        CALL    CROUT   ; NOT AT END - PRINT CR,LF
        JMP     GETCM   ; AND EXIT
XCM18:  PUSH    H       ; PLACE POINTER ON STACK
        MOV     E,M
        MVI     D,.HIGH.DATA ; FETCH ADDR OF SAVE LOC FROM TABLE
        INX     H
        MOV     B,M     ; FETCH LENGTH FLAG FROM TABLE
        PUSH    D       ; SAVE ADDR OF SAVE LOC
        PUSH    D
        POP     H       ; MOVE ADDR TO HL
        PUSH    B       ; KEEP LENGTH FLAG
        MOV     A,M     ; GET 8 BITS OF REG FROM SAVE LOC
        CALL    NMOUT   ; SHOW IT TO THE USER
        POP     PSW     ; GET BACK THE LENGTH FLAG
        PUSH    PSW     ; SAVE IT AGAIN
        ORA     A       ; SET FLAGS
        JZ      XCM20   ; IF AN 8 BIT REG. THEN WE ARE DONE
        DCX     H       ; OTHERWISE GET 8 MORE BITS
        MOV     A,M
        CALL    NMOUT   ; NOW SHOW THEM
XCM20:  MVI     C,'-'
        CALL    ECHO    ; USE DASH AS SEPARATOR
        CALL    GETHX   ; IS THERE A VALUE THERE?
        FALSE   XCM30   ; NO - GO CHECK NEXT REGISTER
        MOV     A,D
        STA     TEMP    ; YES - SAVE THE TERMINATOR FOR A WHILE
        POP     PSW     ; GET BACK LENGTH FLAG
        POP     H       ; STORE SAVE LOC ADDR IN HL
        ORA     A       ; SET FLAGS
        JZ      XCM25   ; IF AN 8 BIT REG, THEN BRANCH
        MOV     M,B     ; SAVE UPPER 8 BITS
        DCX     H       ; POINT TO SAVE LOC FOR LOWER 8 BITS
XCM25:  MOV     M,C     ; STORE LSB OF REG
```

```
XCM27: LXI     D,RTABS ; SIZE OF ENTRY IN RTAB TABLE
       POP     H       ; GET POINTER INTO RTAB
       DAD     D       ; ADD ENTRY SIZE TO POINTER
       JMP     XCM10   ; DO NEXT REG
XCM30: MOV     A,D     ; GET TERMINATOR
       STA     TEMP    ; SAVE IN MEM
       POP     D       ; CLEAR STACK OF LENGTH FLAG AND ADDR
       POP     D       ; /OF SAVE LOC
       JMP     XCM27   ; INC RTAB POINTER
;
;
;*****************************************************************
;
;          ZCMD
;
;          COMMAND TO GET 3 NUMBERS FROM THE INPUT STREAM
;          AND COPY THE THIRD NUMBER INTO RAM FROM THE 1ST
;          ADDRESS TO THE 2ND ADDRESS.
;
;*****************************************************************
;
ZCMD:  MVI     C,3     ;LOAD C FOR 3 CHARACTERS
       CALL    GETNM   ;GET 3 INPUT VALUES
       POP     D       ;DE=CHARACTER TO BE COPIED IN E,D FILLER
       POP     B       ;END ADDRESS TO BE COPIED THROUGH
       POP     H       ;HL=STARTING ADDRESS OF RAM WHERE E IS WRITTEN
       CALL    ZEROM   ;CALL ZERO ROUTINE
       JMP     GETCM   ;JUMP BACK TO GET NEXT COMMAND WHEN DONE
ZEROM: MOV     A,E     ;DATA TO BE WRITTEN MOVED INTO RAM
       MOV     M,A     ;WRITE DATA IN ACC INTO RAM ADDR BY HL
       INX     H       ;INCREMENT HL FOR NEXT ADDRESS
       MOV     A,C     ;SET UP TEST TO DETERMINE IF HL.GT.BC
       SUB     L       ;SUBTRACT HL FROM BC IF NEG,GET A BORROW
       MOV     A,B     ;WHEN HL .GT. BC ,CARRY IS SET
       SBB     H       ;THEN RETURN TO CALLING PROGRAM
       RC              ;RETURN WHEN DONE
       JMP     ZEROM   ;JUMP BACK,WE WERE NOT DONE
;
;
;
;*******************************************************************
;
;          ROUTINE CCMD
;          THIS ROUTINE COMPARES DATA FROM ADDR1 THROUGH ADDR2 TO
;          DATA STARTING AT ADDR3.ANY DIFFERENCES ARE PRINTED ON
;          THE TERMINAL
;
;*******************************************************************
;
CCMD:  MVI     C,3     ;LOAD C REGISTER TO GET 3 NUMBERS FROM INPUT
       CALL    GETNM   ;3 NUMBERS WILL BE ON STACK
       POP     D       ;DE CONTAINS THE VALUES TO BE COMPARED TO
```

```
        POP     B       ;VALUES FROM ADDR HL TO BC WILL BE COMPARED
        POP     H       ;TO VALUES STARTING AT ADDR DE.
        CALL    COMPAR  ;COMPARE ROUTINE IS CALLED
        JMP     GETCM   ;JUMP BACK TO GET NEXT COMMAND WHEN DONE
COMPAR: LDAX    D       ;LOAD ACCUMULATOR WITH DATA ADDR BY DE
        CMP     M       ;COMPARE TO DATA ADDR BY HL
        JZ      DATCK   ;JUMP TO DATA CHECKS IF DE DATA=HL DATA
        PUSH    B       ;DATA DID NOT AGREE,SAVE BC,PRINT DATA
        MOV     A,H     ;LOAD H IN ACC
        CALL    NMOUT   ;PRINT H REG VAL
        MOV     A,L     ;LOAD L IN ACC
        CALL    NMOUT   ;PRINT L VALUE
        MVI     C,03DH  ;LOAD ASCII "=" INTO C
        CALL    CO      ;PRINT IT
        MVI     C,020H  ;LOAD BLANK INTO C
        CALL    CO      ;PRINT A SPACE
        MOV     A,M     ;LOAD DATA AT HL ADDR INTO A
        CALL    NMOUT   ;PRINT DATA AT HL ADDR
        MVI     C,020H  ;PRINT 2 SPACES
        CALL    CO
        MVI     C,020H
        CALL    CO
        MOV     A,D     ;LOAD D ADDR INTO A
        CALL    NMOUT
        MOV     A,E     ;PRINT DE ADDRESS
        CALL    NMOUT
        MVI     C,03DH  ;LOAD ASCII REP OF "=" INTO C
        CALL    CO      ;PRINT IT
        MVI     C,020H  ;LOAD IN SPACE
        CALL    CO      ;PRINT IT
        LDAX    D       ;LOAD DATA ADDRESSED BY DE INTO ACC
        CALL    NMOUT   ;PRINT DATA ADDRESSED BY DE
        CALL    CROUT   ;CARRIAGE RETURN,LINEFEED
        POP     B       ;BC RESTORED
DATCK:  INX     H       ;INCREMENT HL TO NEXT DATA BYTE
        INX     D       ;INCREMENT DE TO NEXT DATA BYTE
        MOV     A,C     ;SET UP TEST TO DETERMINE IF HL.GT.BC
        SUB     L       ;SUBTRACT HL FROM BC, WE WILL GET A
        MOV     A,B     ;A "BORROW" WHEN HL.GT.BC(CARRY IS SET)
        SBB     H       ;
        RC              ;RETURN TO CALLING SUB WHEN DONE
        JMP     COMPAR  ;JUMP BACK IF NOT DONE
;
;*************************************************************
;*************************************************************
;
;
;       UTILITY ROUTINES
;
;
;*************************************************************
;*************************************************************
```

```
;
;
;           ROUTINE BREAK
;
;           THIS ROUTINE CHECKS FOR A BREAK CHAR FROM THE CONSOLE.
;           IF THERE IS NO PENDING CHAR OR IF IT IS NOT THE BREAK
;           CHAR (USUALLY AN ESC), A FAILURE RETURN IS TAKEN (CARRY=0).
;           THE CHAR IS LOST.  IF IT IS THE BREAK CHAR, A SUCCESS RETURN
;           IS TAKEN (CARRY=1).
;
;           OUTPUT:                 CARRY
;                                   BIT
;                                   -----
;                      SUCCESS        1
;
;                      FAILURE        0
;
;           DESTROYS:  A,FLAGS
;
;
;*******************************************************************
;
;
BREAK: IN      CONST   ; GET TERMINAL STATUS
       ANI     RBR     ; IS A CHAR THERE?
       JZ      FRET    ; NO - FAILURE RETURN
       IN      CNIN    ; YES - GET THE CHAR
       ANI     PRTYO   ; GET RID OF THE PARITY BIT
       CPI     BRCHR   ; IS IT A BREAK CHAR?
       JZ      SRET    ; YES - TAKE A SUCCESS RET
       JMP     FRET    ; NO - TAKE A FAILURE RET
;
;
;*******************************************************************
;
;
;           ROUTINE CI
;
;           THIS ROUTINE WAITS FOR A CHAR TO BE ENTERED AT THE
;           TERMINAL AND THEN PLACES THE CHAR INTO THE A REGISTER,
;           ALLOWING THE CALLING ROUTINE ACCESS TO IT.
;
;           OUTPUT:     THE RECEIVED CHAR IN REG A
;
;           REGISTERS AFFECTED:  A,FLAGS
;
;
;*******************************************************************
;
;
CI:    IN      CONST   ; GET STATUS OF CONSOLE
       ANI     RBR     ; CHECK FOR RECEIVER READY
```

```
        JZ      CI      ; NOT YET, SO WAIT
        IN      CNIN    ; OK - NOW GET CHAR
        RET             ; RETURN TO CALLER
;
;
;*************************************************************
;
;
;       ROUTINE CNVBN
;
;       THIS ROUTINE CONVERTS THE ASCII CODE OF A HEX DIGIT TO
;       THE CORRESPONDING BINARY VALUE.  NO CHECK FOR VALID INPUT
;       IS PERFORMED.
;
;       INPUT:      REGISTER C - ASCII CODE FOR ONE HEX DIGIT
;
;       OUTPUT:     REGISTER A - BINARY VALUE OF THE INPUT DIGIT
;
;       REGISTERS AFFECTED:  A,FLAGS
;
;
;*************************************************************
;
;
CNVBN:  MOV     A,C
        SUI     '0'     ; SUBTRACT A HEX 30H FROM ASCII INPUT CODE
        CPI     10      ; BETWEEN 0 AND 9?
        RM              ; YES - ALL DONE
        SUI     7       ; NO - BETWEEN 17 AND 23 DECIMAL
        RET             ; SUBTRACT OFFSET OF 7 (DECIMAL) AND RETURN
;
;
;*************************************************************
;
;
;       ROUTINE CO
;
;       THIS ROUTINE WAITS UNTIL THE TERMINAL IS READY AND THEN
;       SENDS THE INPUT ASCII CODE TO IT.
;
;       INPUT:      REGISTER C - ASCII CODE OF DESIRED CHAR TO BE
PRINTED
;
;       OUTPUT:     REGISTER C - CODE PRINTED AT TERMINAL
;
;       REGISTERS AFFECTED:  A,FLAGS
;
;
;*************************************************************
;
;
CO:     IN      CONST   ; GET TERMINAL'S STATUS
```

```
        ANI     TRDY    ; SEE IF TRANSMITTER IS READY
        JZ      CO      ; NO - WAIT
        MOV     A,C     ; YES - MOVE CODE TO THE ACCUMULATOR FOR OUTPUT
        OUT     CNOUT   ; SEND IT TO THE CONSOLE
        RET
;
;
;*****************************************************************
;
;
;       ROUTINE CROUT
;
;       THIS ROUTINE PRINTS A <CR>,<LF> ON THE TERMINAL
;
;       EXTERNAL REFERENCES:  ECHO
;
;       REGISTERS AFFECTED:  A,B,C,FLAGS
;
;
;*****************************************************************
;
;
CROUT:  MVI     C,CR
        CALL    ECHO
        RET
;
;
;*****************************************************************
;
;
;       ROUTINE ECHO
;
;
;       THIS ROUTINE TAKES A SINGLE CHAR AS INPUT AND PRINTS IT ON
THE
;       USER TERMINAL.  A <CR> IS ECHOED AS A <CR>,<LF> AND AN <ESC>
;       IS ECHOED AS $.
;
;       INPUT:  REGISTER C - ASCII CODE OF CHAR TO ECHO
;
;       OUTPUT:  REGISTER C - CODE ECHOED TO TERMINAL
;
;       EXTERNAL REFERENCES:  CO
;
;       REGISTERS AFFECTED:  A,B,FLAGS
;
;
;*****************************************************************
;
;
ECHO:   MOV     B,C     ; SAVE ARG
        MVI     A,ESC   ; LOAD ASCII CODE OF <ESC>
```

```
        CMP     B           ; SEE IF ARG IS AN <ESC>
        JNZ     ECH05       ; NO - THEN BRANCH
        MVI     C,'$'       ; YES - THEN ECHO A $
ECH05:  CALL    CO          ; DO OUTPUT THROUGH MONITOR
        MVI     A,CR
        CMP     B           ; WAS CHAR A <CR>?
        JNZ     ECH10      ; NO - NO SPECIAL ACTION NEEDED
        MVI     C,LF        ; YES - ADD A <LF>
        CALL    CO
ECH10:  MOV     C,B         ; RESTORE ARG
        RET                 ; RETURN TO CALLER
;
;
;*****************************************************************
;
;
;           ROUTINE ERROR
;
;           THIS ROUTINE PRINTS THE ERROR CHAR ON THE TERMINAL, THEN A
;           <CR>,<LF> SEQUENCE, AND RETURNS TO THE COMMAND RECOGNIZER.
;
;
;           REGISTERS AFFECTED:  A,B,C,FLAGS
;
;
;****************************************************************
;
;
ERROR:  MVI     C,'*'
        CALL    ECHO        ; PRINT A '*' ON THE TERMINAL
        CALL    CROUT       ; GO TO THE NEXT LINE
        JMP     GETCM       ; RETURN TO COMMAND INTERPRETER
;
;
;****************************************************************
;
;
;           ROUTINE FRET
;
;           THIS ROUTINE IS CALLED BY ANOTHER ROUTINE THAT NEEDS TO
;           REPORT A FAILURE ON RETURN.  THIS IS DONE BY SETTING THE
;           CARRY FALSE (0).
;
;           OUTPUT:  CARRY=0
;
;           REGISTERS AFFECTED:  FLAGS - CARRY ONLY
;
;
;*****************************************************************
;
;
FRET:   STC                 ; FIRST SET CARRY TRUE
```

```
       CMC              ; COMPLEMENT AND MAKE IT FALSE
       RET              ; RETURN TO CALLER
;
;
;*****************************************************************
;
;
;           ROUTINE GETCH
;
;           THIS ROUTINE RETURNS THE NEXT CHAR IN THE INPUT STREAM TO
;           THE CALLING ROUTINE.
;
;           OUTPUT:  REGISTER C - THE NEXT CHAR IN INPUT STREAM
;
;           EXTERNAL REFERENCES:  CI
;
;           REGISTERS AFFECTED:  A,C,FLAGS
;
;
;*****************************************************************
;
;
GETCH: CALL    CI       ; GET CHAR FORM TERMINAL
       ANI     PRTY0    ; TURN OFF PARITY BIT IF SET
       MOV     C,A      ; PUT VALUE IN C REG FOR RETURN
       RET              ; RETURN TO CALLER
;
;
;*****************************************************************
;
;
;           ROUTINE GETHX
;
;           THIS ROUTINE TAKES THE LAST FOUR HEX DIGITS FROM THE INPUT
;           STREAM AND CONVERTS THEIR VALUE TO A 16-BIT BINARY VALUE.
;           A VALID DELIMITER TERMINATES THE INPUT NUMBER AND IS RETURNED
;           WITH THE BINARY VALUE AS AN OUTPUT.  ANY ILLEGAL CHARS
;           CREATE AN ERROR CONDITION.  IF THE FIRST (VALID) CHAR IS NOT
;           A DELIMITER, THE ROUTINE SETS THE CARRY TO 1 AND RETURNS;
;           OTHERWISE, THE CARRY IS SET TO 0 AND THE CONTENTS OF BC ARE
;           UNDEFINED.
;
;           OUTPUT:
;           -------
;
;                     REGISTERS B,C - 16 BIT BINARY INTEGER
;
;                     REGISTER D - CHAR THAT TERMINATED THE INTEGER INPUT
;
;                     FLAGS (CARRY) - 1 IF FIRST CHAR NOT A DELIMITER
;                                     0 IF IT IS (THEN B,C ARE UNDEFINED)
;
```

```
;         EXTERNAL REFERENCES:
;         --------------------
;
;         CNVBN     ECHO      ERROR     GETCH     VALDG
;         VALDL
;
;         REGISTERS AFFECTED:  A,B,C,D,E,FLAGS
;
;
;*****************************************************************
;
;
GETHX:  PUSH    H        ; SAVE HL
        LXI     H,0      ; INIT RETURN VALUE
        MVI     E,0      ; INIT DIGIT FLAG TO FALSE
GHX05:  CALL    GETCH    ; GET ONE CHAR
        MOV     C,A
        CALL    ECHO     ; ECHO IT
        CALL    VALDL    ; SEE IF IT IS A DELIMITER
        FALSE   GHX10    ; NO - BRANCH
        MOV     D,C      ; YES - FINISHED, MUST RETURN DELIMITER
        PUSH    H
        POP     B        ; STORE VALUE IN BC
        POP     H        ; RESTORE HL
        MOV     A,E      ; GET FLAG
        ORA     A        ; SET FLAGS
        JNZ     SRET     ; NONZERO FLAG - NUMBER FOUND
        JZ      FRET     ; ZERO FLAG - DELIMITER WAS 1ST CHAR
GHX10:  CALL    VALDG    ; IF NOT A DELIMITER, SEE IF A HEX DIGIT
        FALSE   ERROR    ; NO - ERROR
        CALL    CNVBN    ; CONVERT TO ITS BINARY VALUE
        MVI     E,0FFH   ; SET DIGIT FLAG TO TRUE
        DAD     H        ; *2
        DAD     H        ; *4
        DAD     H        ; *8
        DAD     H        ; *16
        MVI     B,0      ; CLEAR THE HIGH 8 BITS OF BC
        MOV     C,A      ; GET BINARY VALUE OF CHAR INTO C
        DAD     B        ; ADD THIS VALUE TO PARTIAL RESULT
        JMP     GHX05    ; NOW GET THE NEXT CHAR
;
;
;*****************************************************************
;
;
;         ROUTINE GETNM
;
;         THIS ROUTINE FINDS A SPECIFIED NUMBER OF INPUT NUMBERS
;         (BETWEEN 1 AND 3) AND RETURNS THEIR VALUES ON THE STACK.
;
;         IF TWO OR MORE ARE REQUESTED, THEN THE FIRST MUST BE LESS
;         THAN OR EQUAL TO THE SECOND.  OTHERWISE THE FIRST AND SECOND
```

```
;           NUMBERS WILL BE SET EQUAL.  THE LAST NUMBER MUST BE FOLLOWED
;           BY A <CR> OR AN ERROR CONDITION WILL OCCUR.
;
;           INPUT:  REG C - NUMBER OF INPUT VALUES TO READ
;
;           OUTPUT:  TOP OF STACK - VALUES OF INPUTS (LAST ON TOP)
;
;           EXTERNAL REFERENCES:
;           --------------------
;
;           ERROR     HILO      GETHX
;
;           REGISTERS AFFECTED:  ALL
;
;
;*****************************************************************
;
GETNM: MVI     L,3       ; PUT MAX ARGUMENT COUNT INTO L
       MOV     A,C       ; GET THE ACTUAL ARG COUNT
       ANI     3         ; FORCE TO MAX OF 3
       RZ                ; IF 0, DON'T DO ANYTHING
       MOV     H,A       ; ELSE, PUT THE ACTUAL COUNT IN H
GNM05: CALL    GETHX     ; GET A NUMBER FROM THE INPUT STREAM
       FALSE   ERROR     ; ERROR IF NOT THERE (TOO FEW INPUTS)
       PUSH    B         ; ELSE, SAVE NUMBER ON THE STACK
       DCR     L         ; DECREMENT ACTUAL ARG COUNT
       DCR     H         ; DECREMENT ACTUAL ARG COUNT
       JZ      GNM10     ; BRANCH IF NO MORE INPUTS ARE NEEDED
       MOV     A,D       ; ELSE, GET NUMBER TERMINATOR TO A
       CPI     CR        ; IS IT A <CR>?
       JZ      ERROR     ; YES - ERROR (TOO FEW NUMBERS)
       JMP     GNM05     ; NO - DO ANOTHER NUMBER
GNM10: MOV     A,D       ; WHEN DOWN TO ZERO, CHECK LAST CHAR
       CPI     CR
       JNZ     ERROR     ; ERROR IF IT ISN'T A <CR>
       LXI     B,0FFFFH  ; HL GETS THE LARGEST VALUE
       MOV     A,L       ; GET WHAT'S LEFT OF THE MAX ARG COUNT
       ORA     A         ; CHECK IF 0
       JZ      GNM20     ; YES - 3 NUMBERS WERE INPUT
GNM15: PUSH    B         ; NO - FILL UNUSED ARGS WITH 0FFFFH
       DCR     L
       JNZ     GNM15
GNM20: POP     B         ; NOW GET THE THREE ARGS OUT
       POP     D
       POP     H
       CALL    HILO      ; SEE IF FIRST VAL >= THE SECOND
       FALSE   GNM25     ; NO - TAKE A BRANCH
       MOV     D,H
       MOV     E,L       ; YES - SET SECOND EQUAL TO FIRST
GNM25: XTHL              ; PUT THE FIRST ON THE STACK AND GET RETURN ADDR
       PUSH    D         ; PUT THE SECOND ON STACK
       PUSH    B         ; PUT THIRD ON THE STACK
```

```
       PUSH    H         ; PUT RET ADDR ONTO STACK
GNM30: DCR     A         ; DECREMENT RESIDUAL COUNT
       RM                ; IF IT IS NEGATIVE, THEN CORRECT
                         ; NUMBERS ARE ON THE STACK
       POP     H         ; OTHERWISE, GET RETURN ADDR
       XTHL              ; REPLACE TOP RESULT WITH RET ADDR
       JMP     GNM30     ; TRY AGAIN
;
;
;****************************************************************
;
;
;          ROUTINE HILO
;
;          THIS ROUTINE COMPARES TWO UNSIGNED 16-BIT INTEGERS IN HL
;          AND DE.  THE CARRY FLAG IS SET ACCORDING TO THE COMPARISON.
;
;          INPUT:      DE - 16-BIT INTEGER
;                      HL - 16-BIT INTEGER
;
;          OUTPUT:     CASE     CARRY
;                      ----     -----
;
;                      HL<DE      0
;
;                      HL>=DE     1
;
;          REGISTERS AFFECTED:  FLAGS
;
;
;****************************************************************
;
;
HILO:  PUSH    B         ; SAVE BC
       MOV     B,A       ; SAVE A IN B REG
       DCX     D         ;DE DECREMENTED SO THAT WE WILL GET CARRY
       MOV     A,E       ;WHEN HL.GE.DE
       SUB     L         ;SUBTRACT HL FROM DE-GET A BORROW WHEN
       MOV     A,D       ;WHEN HL.GT.DE-1(CARRY IS SET)
       SBB     H
       INX     D         ;DOESN'T WORK IF DE=0,RESTORE DE
       MOV     A,B       ;RESTORE ACCUMULATOR
       POP     B         ;RESTORE BC
       RET               ;RETURN TO CALLING ROUTINE
;
;
;****************************************************************
;
;
;          ROUTINE NMOUT
;
;          THIS ROUTINE PRINTS THE CONTENTS OF REGISTER A IN HEX
```

```
;           ON THE TERMINAL.
;
;           INPUT:   REGISTER A - 8-BIT INTEGER
;
;           EXTERNAL REFERENCES:
;           --------------------
;
;           ECHO       PRVAL
;
;           REGISTERS AFFECTED:  A,B,C,FLAGS
;
;
;****************************************************************
;
;
NMOUT:  PUSH    H          ; SAVE HL - DESTROYED BY PRVAL
        PUSH    PSW        ; SAVE ARG
        RRC
        RRC
        RRC
        RRC                ; MOVE HI 4 BITS TO LO 4 BITS
        ANI     HCHAR      ; MASK OUT HI 4 - ONLY WANT ONE HEX CHAR
        MOV     C,A
        CALL    PRVAL      ; CONVERT LOWER 4 BITS TO ASCII
        CALL    ECHO       ; SEND TO TERMINAL
        POP     PSW        ; GET BACK ARG
        ANI     HCHAR      ; MASK OUT UPPER 4 BITS - JUST WANT 1 HEX CHAR
        MOV     C,A
        CALL    PRVAL
        CALL    ECHO
        POP     H          ; RESTORE ORIGINAL HL
        RET                ; RETURN TO CALLER
;
;
;****************************************************************
;
;
;           ROUTINE PRVAL
;
;           THIS ROUTINE CONVERTS A BINARY NUMBER BETWEEN 0H AND 0FH,
;           INCLUSIVE, INTO THE CORRESPONDING ASCII CHAR '0' - '9' OR
;           'A' - 'F'.  NO CHECK IS MADE OF THE VALIDITY OF THE INPUT.
;
;           INPUT:  REGISTER C - INTEGER N, 0<=N<=0FH
;
;           OUTPUT: REGISTER C - CORRESPONDING ASCII CODE
;
;           REGISTERS AFFECTED:  B,C,H,L,FLAGS
;
;
;****************************************************************
;
```

```
;
PRVAL: LXI     H,DIGTB ; ADDR OF TABLE
       MVI     B,0     ; CLEAR HI 8 BITS OF BC
       DAD     B       ; ADD DIGIT VAL TO HL ADDR
       MOV     C,M     ; FETCH ASCII CODE FROM MEM
       RET             ; RETURN TO CALLER
;
;
;************************************************************
;
;
;          ROUTINE REGDS
;
;          THIS ROUTINE SHOWS THE REGISTER SAVE LOC'S ON THE TERMINAL IN
;          FORMATTED FORM.  THIS ROUTINE IS DRIVEN FROM A TABLE, RTAB,
;          WHICH CONTAINS THE REG'S PRINT SYMBOL, SAVE LOC ADDR, AND
;          LENGTH (8 OR 16 BITS).
;
;          EXTERNAL REFERENCES:
;          --------------------
;
;          CROUT     ECHO      ERROR     NMOUT
;
;          REGISTERS AFFECTED:  ALL
;
;
;*****************************************************************
;
;
REGDS: LXI     H,RTAB  ; LOAD HL WITH ADDR OF START OF TABLE
REG05: MOV     C,M     ; GET PRINT SYMBOL OF REG
       MOV     A,C
       ORA     A         TEST FOR 0 - END OF TABLE
       JNZ     REG10   ; NO - THEN BRANCH
       CALL    CROUT   ; YES - SEND <CR>,<LF>
       RET             ; RETURN TO CALLER
REG10: CALL    ECHO    ; ECHO THE CHAR
       MVI     C,'='
       CALL    ECHO    ; OUTPUT EQUALS SIGN
       INX     H       ; POINT TO START OF SAVE LOC ADDR
       MOV     E,M     ; GET LSB OF SAVE LOC ADDR TO E
       MVI     D,.HIGH.DATA ; P[illegible] MSB OF SAVE LOC ADDR INTO D
       INX     H       ; POINT TO LENGTH FLAG
       [illegible]     D       ; GET CONTENTS OF SAVE ADDR
       [illegible]     NMOU[illegible]   ; PRINT THEM ON THE TERMINAL
       [illegible]     A,M     ; [illegible]ET LENGTH FLAG
       [illegible]             ; [illegible]T SIGN FLAG
       [illegible]             ; [illegible]F 0, REGISTER IS 8 BITS
       [illegible]             ; [illegible]THERWISE, 16-BIT REG SO MORE TO DISPLAY
       [illegible]             ; [illegible]ET LO 8 BITS
       [illegible]             ; PRINT THEM
REG15: MVI     [illegible]
```

```
        CALL    ECHO
        INX     H       ; POINT TO START OF NEXT TABLE ENTRY
        JMP     REG05   ; DO NEXT REG
;
;
;******************************************************************
;
;
;               ROUTINE RGADR
;
;               THIS ROUTINE TAKES ONE CHAR, DENOTING A REGISTER, AND
;               SEARCHES THE RTAB TABLE FOR A MATCH.  IF FOUND, THE ROUTINE
;               RETURNS THE ADDR OF THE ADDR OF THE SAVE LOC OF THE REG.
;               THIS ADDR POINTS INTO RTAB.  IF THERE IS NO MATCH, THEN THE
;               REGISTER IDENTIFIER IS ILLEGAL AND AN ERROR CONDITION OCCURS.
;
;               INPUT:      REGISTER C - ASCII CODE OF REG SYMBOL
;
;               OUTPUT:     REGISTERS B,C - ADDR OF ENTRY IN RTAB FOR REG
DESIRED
;
;               EXTERNAL REFERENCES:
;               --------------------
;
;               ERROR
;
;               REGISTERS AFFECTED:  ALL
;
;
;***********************************************************************
;
;
RGADR:  LXI     H,RTAB  ; HL GETS ADDR OF START OF TABLE
        LXI     D,RTABS ; DE GETS SIZE OF AN ENTRY
RGA05:  MOV     A,M     ; GET REGISTER IDENTIFIER
        ORA     A       ; CHECK FOR TABLE END (IDENTIFIER IS 0)
        JZ      ERROR   ; IF AT END OF TABLE, ARG IS ILLEGAL
        CMP     C       ; ELSE, COMPARE TABLE ENTRY AND ARG
        JZ      RGA10   ; EQUAL - REG IS FOUND
        DAD     D       ; NOT EQUAL - INC POINTER AND TRY AGAIN
        JMP     RGA05
RGA10:  INX     H       ; IF A MATCH, INC TABLE POINTER TO SAVE LOC ADDR
        MOV     B,H
        MOV     C,L     ; RETURN THIS VALUE
        RET             ; RETURN TO CALLER
;
;
;******************************************************************
;
;
;               ROUTINE RSTTF
;
```

```
;          THIS ROUTINE RESTORES ALL REGS, ALL FLIP/FLOPS, THE SP AND
;          THE PC FROM THE SAVE LOCS IN MEMORY.  THE ROUTINE RETURNS
;          CONTROL TO THE NEW PC LOC WITH ALL INTERRUPTS ENABLED.
;
;          REGISTERS AFFECTED:  ALL
;
;
;*****************************************************************
;
;
;
RSTTF: DI               ; DISABLE INTERRUPTS WHILE RESTORING THINGS
       LXI    H,MSTAK   ; SET MONITOR SP TO START OF STACK
       SPHL
       POP    D         ; START ALSO END OF REG SAVE AREA
       POP    B
       POP    PSW
       LHLD   SSAVE     ; RESTORE USER STACK POINTER
       SPHL
       LHLD   PSAVE
       PUSH   H         ; PUT USER RET ADDR ON USER STACK
       LHLD   LSAVE     ; RESTORE HL REGS
       EI               ; ENABLE INTERRUPTS NOW
       RET              ; JUMP TO RESTORED PC LOC
;
;
;*****************************************************************
;
;
;          ROUTINE SRET
;
;          THIS ROUTINE IS CALLED BY OTHER ROUTINES WHICH WANT TO
;          RETURN SUCCESS.  IT SETS THE CARRY TRUE (1).
;
;          OUTPUT:     CARRY = 1
;
;          REGISTERS AFFECTED:  FLAGS (CARRY ONLY)
;
;
;*****************************************************************
;
;
SRET:  STC              ; SET CARRY TRUE
       RET              ; RETURN TO CALLER
;
;
;*****************************************************************
;
;
;          ROUTINE STHFO
;
;          THIS ROUTINE CHECKS THE HALF-BYTE FLAG IN TEMP TO SEE IF
```

```
;           IT IS SET TO LOWER.  IF SO, STHFO STORES A 0 TO PAD OUT
;           THE LOWER HALF OF THE ADDRESSED BYTE; OTHERWISE, IT DOES
;           NOTHING.
;
;           INPUT:      REGISTERS DE - 16-BIT ADDR OF BYTE TO BE STORED
INTO
;
;           EXTERNAL REFERENCES:
;           --------------------
;
;           STHLF
;
;           REGISTERS AFFECTED:  A,B,C,H,L,FLAGS
;
;
;*****************************************************************
;
;
STHFO: LDA      TEMP    ; GET HALF BYTE FLAG
       ORA      A       ; SET FLAGS
       RNZ              ; IF SET TO UPPER, DON'T DO ANYTHING
       MVI      C,0     ; ELSE, WANT TO STORE THE VALUE 0
       CALL     STHLF   ; DO IT
       RET              ; RETURN TO CALLER
;
;
;*****************************************************************
;
;
;           ROUTINE STHLF
;
;           THIS ROUTINE TAKES THE 4-BIT VALUE IN C AND STORES IT IN
;           HALF OF THE BYTE ADDRESSED BY REGS DE.  THE HALF-BYTE USED
;           (UPPER OR LOWER) IS DENOTED BY THE FLAG IN TEMP.  STHLF
;           ASSUMES THAT THIS FLAG HAS ALREADY BEEN SET, USUALLY BY
;           ICMD.
;
;           INPUT:      REGISTER C - 4-BIT VALUE TO STORE IN HALF-BYTE
;                       REGISTERS DE - 16-BIT ADDR OF BYTE TO BE STORED
INTO
;
;           REGISTERS AFFECTED:  A,B,C,H,L,FLAGS
;
;
;*****************************************************************
;
;
STHLF: PUSH     D
       POP      H       ; MOVE ADDR OF BYTE INTO HL
       MOV      A,C     ; GET VAL
       ANI      0FH     ; FORCE TO 4-BIT LENGTH
       MOV      C,A     ; PUT VAL BACK
```

```
        LDA     TEMP    ; GET HALF BYTE FLAG
        ORA     A       ; CHECK FOR LOWER HALF
        JNZ     STH05   ; BRANCH IF NOT
        MOV     A,M     ; ELSE, GET BYTE
        ANI     0F0H    ; CLEAR LOWER 4 BITS
        ORA     C       ; OR IN VAL
        MOV     M,A     ; PUT BYTE BACK
        RET             ; RETURN TO CALLER
STH05:  MOV     A,M     ; IF UPPER HALF, GET BYTE
        ANI     0FH     ; CLEAR UPPER 4 BITS
        MOV     B,A     ; SAVE BYTE IN B
        MOV     A,C     ; GET VALUE
        RRC
        RRC
        RRC
        RRC             ; ALIGN TO UPPER 4 BITS
        ORA     B       ; OR IN ORIGINAL LOWER 4 BITS
        MOV     M,A     ; PUT NEW CONFIGURATION BACK
        RET             ; RETURN TO CALLER
;
;
;*****************************************************************
;
;
;           ROUTINE VALDG
;
;           THIS ROUTINE RETURNS SUCCESS IF ITS INPUT ARG IS AN ASCII
;           CHAR REPRESENTING A VALID HEX DIGIT AND FAILURE OTHERWISE.
;
;           INPUT:  REGISTER C - ASCII CODE
;
;           REGISTERS AFFECTED:  A,FLAGS
;
;
;*****************************************************************
;
;
VALDG:  MOV     A,C
        CPI     '0'     ; TEST CHAR AGAINST '0'
        JM      FRET    ; CANNOT BE A HEX DIGIT IF LESS THAN ASCII FOR
ZERO
        CPI     '9'     ; ELSE, SEE IF IN RANGE '0' - '9'
        JM      SRET    ; LESS THAN '9'??
        JZ      SRET    ; EQUAL TO '9'?
        CPI     'A'     ; NOT A NUMERAL, SO TRY A DIGIT
        JM      FRET    ; NO - RETURN
        CPI     'G'
        JP      FRET    ; NO - CODE GREATER THAN 'F'
        JMP     SRET    ; YES - CODE BETWEEN 'A' - 'F', INCLUSIVE
;
;
;*****************************************************************
```

```
;
;
;         ROUTINE VALDL
;
;         THIS ROUTINE RETURNS SUCCESS IF ITS INPUT ARG IS A
;         VALID DELIMITER CHAR (SPACE, COMMA, OR <CR>) AND FAILURE
;         OTHERWISE.
;
;         INPUT:     REGISTER C - INPUT ASCII CODE
;
;         OUTPUT:    CARRY = 1 IF CHAR IS VALID DELIMITER
;                          = 0 IF IT IS NOT
;
;         REGISTERS AFFECTED:  A,FLAGS
;
;
;*****************************************************************
;
;
VALDL: MOV     A,C
       CPI     ','      ; SEE IF COMMA
       JZ      SRET
       CPI     CR       ; SEE IF <CR>
       JZ      SRET
       CPI     ' '      ; SEE IF SPACE
       JZ      SRET
       JMP     FRET     ; ERROR IF NONE OF THE ABOVE
;
;
;*****************************************************************
;*****************************************************************
;
;
;         MONITOR TABLES
;
;
;*****************************************************************
;*****************************************************************
;
;
SGNON: DB      CR,LF,'NDT-COMP9',CR,LF ; LOGON MESSAGE
LSGNON EQU     $-SGNON ; LENGTH OF SIGN-ON MESSAGE
;
;
CADR:  DW      0        ; TABLE OF COMMAND ROUTINE ADDR'S - 1ST VAL IS
DUMMY
       DW      CCMD
       DW      XCMD
       DW      SCMD
       DW      MCMD
       DW      ICMD
       DW      GCMD
```

```
        DW      DCMD
        DW      CCMD
;
;
CTAB:   DB      'C'      ; TABLE OF VALID COMMAND CHARS
        DB      'D'
        DB      'G'
        DB      'I'
        DB      'M'
        DB      'S'
        DB      'X'
        DB      'Z'
NCMDS   EQU     $-CTAB   ; NUMBER OF VALID COMMANDS
;
;
DIGTB:  DB      '0'      ; TABLE OF ASCII CODE OF HEX DIGITS
        DB      '1'
        DB      '2'
        DB      '3'
        DB      '4'
        DB      '5'
        DB      '6'
        DB      '7'
        DB      '8'
        DB      '9'
        DB      'A'
        DB      'B'
        DB      'C'
        DB      'D'
        DB      'E'
        DB      'F'
;
;
RTAB:   DB      'A'      ; REGISTER IDENTIFIER
        DB      .LOW.ASAVE; ADDR OF REG SAVE LOC
        DB      0        ; LENGTH FLAG - 0=8 BITS, 1=16 BITS
RTABS   EQU     $-RTAB   ; SIZE OF AN ENTRY IN THIS TABLE
        DB      'B'
        DB      .LOW.BSAVE
        DB      0
        DB      'C'
        DB      .LOW.CSAVE
        DB      0
        DB      'D'
        DB      .LOW.DSAVE
        DB      0
        DB      'E'
        DB      .LOW.ESAVE
        DB      0
        DB      'F'
        DB      .LOW.FSAVE
        DB      0
```

```
        DB      'H'
        DB      .LOW.HSAVE
        DB      0
        DB      'L'
        DB      .LOW.LSAVE
        DB      0
        DB      'M'
        DB      .LOW.HSAVE
        DB      1
        DB      'P'
        DB      .LOW.(PSAVE+1)
        DB      1
        DB      'S'
        DB      .LOW.(SSAVE+1)
        DB      1
        DB      0         ; END OF TABLE MARKERS
        DB      0
;
;
        MVI     B,04H     ; PRINT FOUR <LF> CHARS
        MVI     C,0AH     ; LOAD REG C
RPT:    CALL    CO        ; PRINT A <LF>
        DCR     B         ; DECREMENT COUNTER
        RZ                ; RETURN TO CALLER WHEN DONE
        JMP     RPT       ; LOOP ONCE MORE
;
;
;****************************************************************
;****************************************************************
;
;
;           END OF MONITOR
;
;
;****************************************************************
;****************************************************************
;
;
       END      START
```

APPENDIX B

THE GPIF PROGRAM

The program GPIF is used to transfer data into and out of the three-frequency instrument, control the calibration and reading modes of the instrument, and perform readings of different types at the different frequencies. The program sets the I/O ports and receives a two-byte command from the controller over the IEEE-488 bus. The first word is the command for the type program to be run. There are ten command options, and these are listed in the program. They include a program to take a set of readings and then transmit the data, a program to start a set of readings and then transmit the prior readings while the new readings are being made, and a program to jump to an unused ram address so that temporary test programs can be typed in and tried. The second byte determines the state of the calibrate, reading, or multiplexed reading switches. A listing of the program GPIF follows:

```
            TITLE 'GPIF PROGRAM VERSION 18 DECEMBER 84'
            NAME    GPIF
            LIST   B,G,O,T
;           NLIST   I,M,S,T
;
;           PROGRAM TO READ THE AD CONVERTERS ON COMMAND FROM THE
;           PROGRAM TO CONFIGURE THE COMP9B MICROCOMPUTER
;           AS A TALKER/LISTENER ON THE IEEE488/GPIB BUSS
;           USING AN 8291A& 2 8293'S.
;
            ORG 08C0H           START PROGRAM IN SECOND PROM.
; SYMBOL DEFINITIONS
;
;  DEFINE PORT ADDRESSES
;
PORT1       EQU 0CFH            PORT 1 CONTROL WORD ADDRESS
PORT1A      EQU 0CCH            PORT1A ADDRESS
PORT1B      EQU 0CDH            PORT1B ADDRESS
PORT1C      EQU 0CEH            PORT1C ADDRESS
PORT2       EQU 0D7H            PORT 2 CONTROL WORD ADDRESS
PORT2A      EQU 0D4H            PORT2A ADDRESS
PORT2B      EQU 0D5H            PORT2B ADDRESS
PORT2C      EQU 0D6H            PORT2C ADDRESS
APDATA      EQU 0EEH            ARITHMETIC PROCESSOR DATA PORT
;           GPIB ADDRESS AND MASK DEFINITIONS
BUSPRT      EQU 0D8H             BASE ADDR OF GPIB BUSS
BUSIN       EQU 0D8H             DATA IN FROM BUSS ADDR
BUSOUT      EQU BUSIN            DATA OUT TO BUSS ADDR
S1          EQU 0D9H             INTERRUPT STATUS REGISTER 1 ADDR
INTE1       EQU 0D9H             INTERRUPT ENABLE REGISTER 1 ADDR
BOM         EQU 02               BYTE OUT INTR MASK,BYTE
;                                SHOULD BE WRITTEN IN BUSOUT
BIM         EQU 01               BYTE IN INTR MASK,BYTE SHOULD
;                                BE READ FROM BUSIN REGISTER
ENDMK       EQU 10H              END INTERRUPT MASK
CPT         EQU 080H             COMMAND PASS THROUGH MASK
;           REG #2 INTERRUPTS
INTE2      EQU 0DAH             INTERRUPT REGISTER 2 ADDR
;           REG #4 ADDRESS MODE CONSTANTS
```

```
ADRMD       EQU 0DCH            ADDRESS MODE REGISTER ADDR
TON         EQU 080H            TALK ONLY,NOT LISTEN  MODE
LON         EQU 040H            LISTEN ONLY,NOT TALK MODE
TLON        EQU 0C0H            TALK AND LISTEN ONLY MODE
MODE1       EQU 01H             MODE 1 ADDRESSING
;           REG #4 (READ) ADDRESS STATUS REGISTER
ADRST       EQU ADRMD           ADDRESS STATUS REGISTER ADDR
EOIST       EQU 20H             END OR IDENTIFY MASK
TA          EQU 02H             TALKER ADDR OR ACT;SER POLL-TADS TACS SPAS
LA          EQU 04H             LISTENER ADDRESSED OR ACTIVE-LADS OR LACS
MJMN        EQU 01H             MAJOR OR MINOR TALKER/LISTENER,1=MINOR
;           REG #5(WRITE) AUXILIARY MODE REGISTER
AUXMD       EQU 0DDH            AUXILIARY MODE REG ADDR
CLKRT       EQU 022H            CLOCK SET FOR 2MHZ
FNHSK       EQU 03H             FINISH HANDSHAKE AUX COMMAND
SDEOI       EQU 06H             SEND END OR IDENTIFY WITH NEXT BYTE
AXRA        EQU 080H            WRITE DDDDD INTO AUX REGISTER A
HOHSK       EQU 01H             HOLD OFF HANDSHAKE ON ALL BYTES
HOEND       EQU 02H             HOLD OFF HANDSHAKE ON END BYTE
CAHCY       EQU 03H             CONTINUOUS ACCEPTOR HANDSHAKE CYCLING
EDEOS       EQU 04H             END ON EOS RECEIVED DAT REG MATCHES EOS
REG
EOIS        EQU 08H             OUTPUT EOI ON EOS SENT
EOSBC       EQU 40H             EOS FUNCTIONS AS FULL 8-BIT REG
VSCMD       EQU 0FH             VALID COMMAND PASS THROUGH
NVCMD       EQU 07H             INVALID COMMAND PASS THROUGH
AXRB        EQU 0A0H            AUXILIARY REG B PATTERN
CPTEN       EQU 01H             COMMAND PASS THROUGH ENABLE
;           REG #5(READ)
CPTRG       EQU 0DDH            ADDR TO READ COMMAND PASS THROUGH
;           REG #6 (WRITE) ADDRESS 0/1 REG CONSTANTS
ADR01       EQU 0DEH            COMP 9 GPIB ADDRESSES
DTDL1       EQU 060H            DISABLE MAJOR TALKER/LISTENER
DTDL2       EQU 0E0H            DISABLE MINOR TALKER/LISTENER
ADRTL       EQU 05H             TALKER LISTENER ADDRESS SET TO 5
;           REG #7 EOS-END OF SEQUENCE CHARACTER REG
EOSR        EQU 0DFH            FLAGS END OF BLOCK BY CHAR IN REG
;
;           GPIB MESSAGES(COMMANDS)
;
 PUBLIC        PRTSU
;
;
;  MATH SUBROUTINES FOR THE COMP 9 ARE STORED AS PUBLIC.ANY ROUTINE
;  CAN BE CALLED USING AN 'EXTRN' STATEMENT.
;
 EXTRN          ACOS,ASIN,ATAN,ATANA,BIDEC,BIDECF
 EXTRN          CHSD,CHSDA,CHSF,CHSFA,CHSS,CHSSA,COS,COSA
 EXTRN          DADD,DADDA,DADDB,DDIV,DDIVA,DDIVB,DECNO
 EXTRN          DMUL,DMULA,DMULB,DMUU,DMUUA,DMUUB,DSUB,DSUBA,DSUBB
 EXTRN          EXP,EXPA,EXP10,FADD,FADDA,FADDB,FDIV,FDIVA,FDIVB
 EXTRN          FIXD,FIXDA,FIXS,FIXSA,FLTD,FLTDA,FLTS,FLTSA
 EXTRN          FMUL,FMULA,FMULB,FSUB,FSUBA,FSUBB,LN,LNA,LOG,LOGA
```

```
 EXTRN          MDAD,POPD,POPS,PTOD,PTOS,PUPI,PWR,PWRA,PWRB
 EXTRN          SADD,SADDA,SADDB,SDIV,SDIVA,SDIVB,SIN,SINA
 EXTRN          SMUL,SMULA,SMULB,SMUU,SMUUA,SMUUB,SQRT,SQRTA
 EXTRN          SSUB,SSUBA,SSUBB,TAN,TANA,TOS2,TOS4
 EXTRN          WRT2,WRT4,XCHD,XCHS
 EXTRN          CROUT,COPDT,GETCM,PRINT,PRINTF,EPRNT,FPRNT,ZERO
 EXTRN          GETC,GETNM,CO,NMOUT,ECHO,GETHX,SGNON
;
;
;CONSTANTS ARE SET
NRDG        EQU 001H           NRDG=2**N=NUMBER OF RDGS PER CHANNEL=1
;
NCHA        EQU 06H            NCHA=NUMBER OF CHANNELS TO BE READ
NCH2        EQU 02H            NCH2=2 CHANNELS TO BE READ FOR SING FREQ
PRT1O       EQU 088H           PRT1O=OUTPUT MODE,A,B,C0-3 OUTPUT,C4-7 IN
PRT1I       EQU 099H           PRT1I=INPUT MODE,A&C INPUT:B OUTPUT
;
;       RAM ADDRESS DEFINITIONS
;
RAWDA       EQU 03C10H         RAW RDGS,4 BYTES/CHNL,MG1,PH1,MG2.. HI-LO
OLDSW       EQU RAWDA+4*NCHA   CALIBRATION SWITCH POSITION FROM
CONTROLLER
ASCDAT      EQU OLDSW+1        ASCII DATA,8*NCHA+1 BYTES
DIAD        EQU ASCDAT+8*NCHA+1  ;ADDRESS FOR INPUT DATA
NCH         EQU DIAD+2     CHANNEL NUMBER COUNT FOR ADC RDGS
DATIN       EQU NCH+1          START OF DATA READ IN FROM GPIB,32 BYTES
RAMADR      EQU 3800H          ADDRESS OF RAM PROGRAM,CALLED BY CMD4
INTRP       EQU 1000H          ADDRESS OF FRONT PANEL INTERRUPTS
;       ENTRY POINT FOR INTERRUPTS
            POP PSW            POP STACK SO WE WON'T HAVE OLD ADDR
            IN S1              READ INPUT STATUS
            CPI BOM            CHECK FOR BYTE OUT REQUEST
            JZ BOUT            JUMP TO OUTPUT DATA
            CPI BIM            CHECK FOR BYTE IN REQUEST
            JZ BIN             JUMP TO BYTE IN ROUTINE
            CPI ENDM           CHECK FOR END OF TRANSMISSION
            JZ BEND            JUMP TO END OF TRANSMISSION ROUTINE
            CPI 00H            COMPARE TO ZERO
            JZ INTRP           INTERRUPT WAS NOT FROM 8291.
            CALL NMOUT         PRINT A REGISTER
            JMP GETCM          RETURN TO MONITOR
BOUT        MOV A,M            LOAD ASCII BYTE ADDR BY HL INTO A
            OUT BUSPRT         WRITE ONTO GPIB BUSS
            INX H              ADDRESS NEXT BYTE
            DCR C              DECREMENT COUNTER
            JZ WREND           FINISHED TRANSMISSION
            EI                 ENABLE INTERRUPT
            HLT                HALT COMPUTER UNTIL NEXT INTERRUPT
BIN         IN BUSPRT          READ A BYTE IN FROM THE BUSS
            LHLD DIAD          LOAD ADDRESS FOR DATA-IN TO HL
            MOV M,A            STORE DATA READ IN INTO RAM
            INX H              INCREMENT FOR NEXT BYTE
            SHLD DIAD          STORE NEXT ADDRESS
```

```
               EI                  ENABLE INTERRUPT
               HLT                 WAIT FOR NEXT INTERRUPT
BEND           LDA DATIN           LOAD FIRST BYTE WRITTEN OVER GPIB
               ANI OFH             DUMP HI BITS,CONVERT TO ADDRESS
               ADD A               DOUBLE VALUE,ADDRESSES ARE 2 BYTES
               MOV C,A             LOAD INTO C
               MVI B,00            ZERO B,BC CONTAIN ADDR INCREMENT
               LXI H,CADDR         COMMAND ADDRESS LISTS IN HL
               DAD B               OFFSET TO STARTING ADDRESS OF PROGRAMS
               MOV A,M             PUT LOW ORDED CMD ADDR IN A
               INX H               ADDRESS HI BYTE OF CMD ADDR
               MOV H,M             SEND TO H
               MOV L,A             PUT LOW ORDER IN L
               PCHL                TRANSFER COMMAND TO ADDR IN HL
;WREND          MVI A,011H          DISABLE INTERRUPT FOR DATAOUT
;              OUT INTE1           SEND TO INTER ENABLE REG.
WREND          MVI A,06H           SEND EOI WITH NEXT BYTE OUT
               OUT AUXMD           SEND TO AUX MODE REGISTER
               EI                  ENABLE INTERRUPT
               HLT                 HALT COMPUTER UNTIL NEXT INTERRUPT
;
;GPIB PORT SET-UP
PRTSU          MVI A,PRT1I         LOAD PROGRAM WORD FOR A&C INPUT,B OUTPUT
               OUT PORT1          SEND TO PORT 1
               MVI A,090H          LOAD PRG WRD,A=INPUT,B&C=OUTPUT
               OUT PORT2           SEND TO PORT2
               MVI A,07H           LOAD CALIBRATE SWITCHES HI
               OUT PORT2B          SEND TO PORT2B
               MVI A,070H          SET MAG TAPE READ,WRITE,DAC LATCH HI
               OUT PORT2C          LEAVE STROBE LINES HI
               MVI A,02            RESET THE 8291A
               OUT AUXMD           SEND TO THE AUXILIARY MODE REG
               MVI A,00            ZERO A REGISTER
               OUT INTE1           DISABLE INTERRUPTS TEMPORARILY
               OUT INTE2           DISABLE SECONDARY INTERRUPTS
               MVI A,MODE1         ENABLE MODE 1 ADDRESSING
               OUT ADRMD           SEND TO THE ADDRESS MODE REG
               MVI A,CLKRT         SET THE CLOCK RATE FOR 2 MHZ
               OUT AUXMD           SEND TO THE AUX MODE REG
               MVI A,ADRTL         LOAD TALKER/LISTENER ADDRESS
               OUT AUXMD           SEND TO AUXILIARY COMMAND REG
               OUT ADR01           SEND TO COMP 9 GPIB ADDRESSES
               MVI A,ODH           LOAD A CARRIAGE RETURN AS END OF MESS
               OUT EOSR            SEND TO END OF SEQUENCE REGISTER
               MVI A,084H          LOAD END ON EOS RECEIVED DAT
               OUT AUXMD           SEND TO AUXILIARY MODE REGISTER
               MVI A,013H          ENABLE END BYTE,BYTE IN,BYTE OUT INTRPS
               OUT INTE1           ENABLE INTERRUPTS
               MVI A,00            LOAD ZERO INTO A
               OUT AUXMD           PUT 8291A INTO IDLE MODE
               LXI H,DATIN         STORE ADDRESS FOR INPUT DATA INTO HL
               SHLD DIAD           STORE IN RAM
               LXI H,SGNON         LOAD SIGNON MESSAGE ADDRESS
```

```
                RET
;
;               TABLE OF ADDRESSES FOR COMMANDS FOR MICROCOMPUTER
;               THE COMMAND TRANSFER ADDRESSES ARE PICKED UP FROM
;               THIS TABLE.FIRST BYTE XMITTED BY CONTROLLER IS CMD NUMBER
;               SECOND BYTE DETERMINES THE CALIBRATOR/MULTIPLEXER POSITION
;               0-3 OPERATE THE CALIBRATOR.4&6 TOGGLE THE MULTIPLEXER,5
;
RESETS
CADDR           DW CMD0          NORMAL MULTI CHAN,MULTI RDG ADC AND ASCII
CONV
                DW CMD1          SINGLE ADC RDG,FAST CONV TO ASCII REP OF
DEC
                DW CMD2          LIKE CMD1 CEPT OLD DATA IS XMITED WHILE
NEW RD
                DW CMD9          SENDS PROGRAM CONTROL TO RAM ADDR 3800
;
;               BASK             CONVERTS BINARY BYTE ADDR BY HL
;                                TO ASCII NO ADDR BY DE FOR C BYTES
;
BASK            MOV A,M          LOAD BYTE ADDRESSED BY HL
                RAR              CONVERT 4 HI BITS FIRST
                RAR              ROTATE 4 HI BITS TO LOW POS
                RAR
                RAR
                ANI 0FH          LOOK AT ONLY 4 LOW BITS
                DAA              DEC ADJUST ACC
                ADI 0F6H         SET CARRY FOR BYTE GT 9
                ACI 03AH         ADD REST OF VALUE WITH CARRY
                STAX D           STORE IN RAM ADDR BY DE
                INX D            ADDRESS NEXT ASCII MEM LOC
                MOV A,M          READ HEX BYTE BACK IN
                ANI 0FH          LOOK AT 4 LOW BITS
                DAA              DEC ADJUST ACC
                ADI 0F6H         SET CARRY FOR BYTE GT 9
                ACI 03AH         ADD REST OF NUMBER WITH CARRY
                STAX D           STORE IN RAM ADDR BY DE
                INX D            ADDRESS NEXT ASCII MEM LOC
                INX H            ADDRESS NEXT HEX BYTE
                DCR C            DECREMENT COUNTER
                JNZ BASK         GO BACK FOR NEXT BYTE IF NOT DONE
                MVI A,0DH        ADD A CARRIAGE RETURN AS THE LAST CHAR
                STAX D           STORE IT IN RAM AFTER THE DATA
                RET              RETURN TO CALLER IF DONE
;
;               CMD0             LOOP TO TAKE READINGS AND SEND THEM OUT
;                                MAGNET CONTROL IS B3,CALIB BITS B2,B1,B0
;                                WILL BE ON STACK,FIRST WORD.PROGRAM IS
;                                CALLED WITH CTRLWD,CMD ADDR,CR BY CALLER.
;                                CALL WITH ADDR OF 6 OR 4 TO CLOCK
;                                CALL WITH ADDR OF 5 TO RESET
;                                CALL WITH CTRLWD OF 0,1,2 OR 3 FOR CALIB.
;
CMD0            LDA DATIN+1      LOAD VALUE OF SECOND BYTE TRANSMITTED
```

```
          ANI 0FH          DUMP HI BITS,LOOK AT B2,B1,B0&SAT MAG(B3)
          OUT PORT2B       SET CALIBRATOR SWITCHES,SAT MAG
          CPI 05H          COMPARE TO 05-TEST FOR STROBE OR RESET
          CZ STROBE        PUT OUT A PULSE FOR RESET
          CPI 04H          COMPARE TO SEE IF RDG OR CALIB
          JNC ROUND1       JUMP AROUND DELAY FOR STROBE READINGS
          LHLD OLDSW       LOAD OLD SWITCH POSITION INTO HL
          STA OLDSW        STORE PRESENT SWITCH POS IN RAM
          CMP L            SEE IF SWITCH POS WAS CHANGED FROM LAST
TIME
          JZ ROUND1        JUMP AROUND IF NO CHANGE
          LXI B,0008       LOAD DELAY INTO BC
          CALL DELAY       DELAY UNTIL READINGS SETTLE DOWN
          CALL DELAY
ROUND1    MVI A,PRT1I
          OUT PORT1        SET PORT1 FOR INPUT
          MVI A,0B0H       STROBE ADC 1011/0000,BIT B5 HI
          OUT PORT1B
          MVI A,090H       THEN STROBE 1001/0000,BIT B5 LO
          OUT PORT1B       A START COMMAND HAS BEEN SENT TO ADC'S
          CALL ADBUS       HANG IN LOOP UNTIL ADC READY
          MVI B,NRDG       LOAD THE NUMBER OF READINGS INTO B
          MVI C,NCHA       LOAD THE NUMBER OF CHANNELS INTO C
          CALL SMDAT       SUM THE READINGS
          MVI A,PRT1O      SET PORT1 FOR OUTPUT
          OUT PORT1
          CALL HEXDEC      CONVERT HEX TO DECIMAL
          LXI H,RAWDA      LOAD STARTING ADDRESS OF RAW DAT
          LXI D,ASCDAT     LOAD START ADDR OF ASCII DAT
          MVI C,4*NCHA     16 BYTES OF DATA TO BE CONVERTED
          CALL BASK        HEX DAT TO ASCII DAT
          LXI H,DATIN      LOAD ADDRESS OF FIRST BYTE OF INPUT DATA
          SHLD DIAD        STORE IN RAM,RESET COUNTER FOR NEXT READ
          LXI H,ASCDAT     LOAD ADDRESS OF DATA TO BE SENT
          MVI C,8*NCHA       NUMBER OF CHARACTERS TO BE SENT
          MVI A,013H       ENABLE READ,WRITE OR EOI
          OUT INTE1        SEND TO 8291 INT ENA
          EI               ENABLE INTERRUPTS ON 8080
          HLT              WAIT FOR INTERRUPT FROM 8291 OVER GPIB
;
STROBE    ANI 0CH          DUMP B0,B1 BITS
          OUT PORT2B       SEND STROBE,RESET BACK LOW
          LDA DATIN+1      LOAD BYTE BACK INTO ACC
          ANI 0FH          DUMP HI BITS AGAIN
          RET              GO BACK
;
;SMDAT - DATA SUMMATION ROUTINE,STARTING WITH CHANNEL 1,READS NO OF
        ;CHANNELS IN C,WITH NO OF READINGS PER CHANNEL IN B (UP TO
        ;255).SUM OF THE DATA WILL BE IN RAWDA -4 + (4*CH. NO.)
        ;RAW DATA IS STORED AS MAG1,PH1,MAG2,PH2,MAG3,PH3;4 BYTES EACH
SMDAT     LXI H,RAWDA      HL LOADED WITH LAST RAW DATA ADDRESS
          PUSH B           COPY OF REGS B,C STORED ON STACK
          MOV A,C          NUMBER OF CHANNELS LOADED INTO A
```

```
            ADD A
            ADD A               4*NO OF CHANNELS IN A
            MOV C,A             4*NO OF CHANNELS IN REG C
            XRA A               A REGISTER ZEROED
            CALL ZERO           ZERO RAM FROM HL TO HL+C
            POP B               NO OF CHANNELS (REG C) & TIMES/CH (B) RES
            LXI D,0004H         LOAD +4 INTO DE
SMLOP       LXI H,RAWDA+03H     HL LOADED WITH FIRST RAW DATA ADDRESS
            PUSH B              AND COPY STORED ON STACK AGAIN
CHLOP       MVI A,NCHA          TOTAL NUMBER OF CHANNELS IN A
            SUB C               NCH = TOTAL NO. - CHA COUNT IN REG C
            CALL ADRDG          MAKE AD CON RDG,ANSWER ON AP TOP OF STACK
            CALL DADDA          32 BIT FIXED ADD TO DATA AT HL,HL=HL-3
            CALL TOS4           AP TOP OF STACK STORED IN RAM AT HL
            DAD D               DE (+4) ADDED TO HL,POINTS TO NEXT ADDR
            DCR C               CHANNEL COUNT DECREMENTED
            JNZ CHLOP           JUMP BACK TO CHANNEL LOOP IF NOT FINISHED
            POP B               RESTORE BC FROM STACK
            DCR B               DECREMENT THE NUMBER OF RDGS/CHANNEL
            JNZ SMLOP           GO BACK TO THE SUM LOOP IF NOT FINISHED
            RET
;ADRDG      SELECTS THE AD CONVERTER NUMBER THAT IS IN A,READS THE
;           12 BITS FROM THE CONVERTER,ADDS 20 ZEROS AND RETURNS WITH
;           A POSITIVE 32 BIT FIXED NUMBER ON THE TOP OF THE APSTACK.
;           FOR DATEL ADC-ET12B ANALOG TO DIGITAL CONVERTER.
ADRDG       ADI 098H            CH SEL NO +CD'S ON 8304,SEL ON ADC
            OUT PORT1B          SET CH. SWITCH TO CH. NO.
            IN PORT1A           LOW ORDER BYTE IS BROUGHT IN AND
            OUT APDATA          STORE ON ARITHMETIC PROCESSOR STACK.
            IN PORT1C           HIGH ORDER BYTE IS BROUGHT IN.
            ANI 0FH             4 HIGHEST ORDER BITS ARE DUMPED
            OUT APDATA          STORE ON ARITHMETIC PROCESSOR STACK.
            XRA A               ZERO ACCUMULATOR
            OUT APDATA          STORE ON ARITHMETIC PROCESSOR STACK.
            OUT APDATA          STORE ON ARITHMETIC PROCESSOR STACK.
            MVI A,090H          LEAVE START LO,DESELECT AD CONVERTERS
            OUT PORT1B          SEND TO PORT1B
            RET                 RETURNS WITH POS. ANSWER ON AP STACK.
;
;           ADBUS          HANG HERE UNTIL ALL BUSY LINES OF THE DATEL
;                          ADC-ET12B ANALOG TO DIGITAL CONVERTERS GOES LOW
;                          DATA FROM THE LAST CONVERSION CAN BE READ FOR
;                          THE NEXT 9 MILLISECONDS.
;
ADBUS       IN PORT1C           CHECK PORT1C,BIT4 FOR BUSY (HI)
            ANI 010H            DUMP ALL BITS EXCEPT 1C4,/0001/0000/
            JNZ ADBUS           STAY IN LOOP UNTIL BUSY LINE LOW
            RET                 START BIT WILL GO LO WHEN DATA CH READ
;DELAY 2 FREQ. - GENERATES A TIME DELAY,DEPENDING ON THE CONSTANT LOADED
;              INTO BC, # CYCLES = 20*FFFF/(BC)
;           WITH 18.432 MHZ CLOCK.TIME=.640/BC SEC OR BC=.640/TIME(SEC)
DELAY       PUSH H              SAVE THE CONTENTS OF HL AND
            LXI H,0H            ZERO HL
```

```
LOPSM       DAD B               ADD BC TO THE CONTENTS OF HL
            JNC LOPSM           STAY IN LOOP UNTIL HL OVERFLOWS.
            POP H               THEIR ORIGINAL VALUES
            RET                 AND RETURN.
;
;           HEXDEC              PROGRAM CONVERTS FIXED HEX TO DECIMAL,12
BITS
;           THIS IS NOT NEEDED IF THE HOST COMPUTER CAN READ BINARY
;           THE OLD HEX DATA IS OVER WRITTEN WITH BCD DATA
;
HEXDEC      LXI H,RAWDA+3       START WITH LO BYTE OF 1ST RAW DATA WRD
            MVI C,NCHA          LOAD THE NUMBER OF CHANNELS TO BE
CONVERTED
CONLOP      PUSH B              SAVE A COPY OF THE CHANNEL COUNTER
            PUSH H              SAVE A COPY OF HL
            CALL WRT4           WRITE THE NUMBER TO BE CONVERTED ON THE
APSTACK
            MVI D,04            LOAD NUMBER OF BYTES TO BE CONVERTED
            CALL BIDEC          CONVERT THE BINARY NO TO DECIMAL
            POP H               GET THE ADDRESS OF THE CONVERTED NO BACK
            LXI D,3C0AH         ADDRESS OF THE DECIMAL NUMBER
            LDAX D              LOAD LO BYTE OF DECIMAL NUMBER
            MOV M,A             STORE IT OVER THE HEX NUMBER
            DCX H               NEXT BYTE OF HEX NO
            INX D               NEXT BYTE OF DEC NO
            LDAX D              LOAD NEXT BYTE OF DEC NO INTO A
            MOV M,A             SEND IT TO MEMORY
;           REPEAT TWICE MORE FOR LARGER NUMBERS
            LXI D,05            LOAD OFFSET ADDRESS FOR HL INTO DE
            DAD D               HL ADDRESSES LO ORDER BYTE FOR NEXT
CHANNEL
            POP B               GET COUNTER BACK FROM STACK
            DCR C               DECREMENT THE COUNTER
            JNZ CONLOP          LOOP BACK IF NOT DONE
            RET                 RETURN IF FINISHED
;
;           CMD1                LOOP FOR FAST READINGS WITH NO
;                               STORAGE IN THE COMP9B MICROCOMPUTER
;           CMD2                SAME AS CMD1 EXCEPT CMD2 TAKES NEXT
;                               READING WHILE THE PREVIOUS ONE IS
;                               BEING PROCESSED.
;
CMD1        NOP                 BOTH PROGS ARE SAME EXCEPT FOR ADBUSY
CMD2        CALL ADBUS          DON'T SWITCH COILS IF WE ARE READING THEM
            LDA DATIN+1         LOAD VALUE OF SECOND BYTE TRANSMITTED
            ANI 0FH             DUMP HI BITS,LOOK AT B2,B1,B0&SAT MAG(B3)
            OUT PORT2B          SET CALIBRATOR SWITCHES,SAT MAG
            CPI 05H             COMPARE TO 05-TEST FOR STROBE OR RESET
            CZ STROBE           PUT OUT A PULSE FOR RESET
            CPI 04H             COMPARE TO SEE IF RDG OR CALIB
            JNC ROUND2          JUMP AROUND DELAY FOR PROBE READINGS
            LHLD OLDSW          LOAD OLD SWITCH POSITION INTO HL
            STA OLDSW           STORE PRESENT SWITCH POS IN RAM
```

```
            CMP L              SEE IF SWITCH POS WAS CHANGED FROM LAST
TIME
            JZ ROUND2          JUMP AROUND IF NO CHANGE
            LXI B,0008         LOAD DELAY INTO BC
            CALL DELAY         DELAY UNTIL READINGS SETTLE DOWN
            CALL DELAY
ROUND2      CPI 07H            JUMP AROUND 4MS DELAY IF CIR COIL
            JZ CIRCOL
            LXI B,0A0H         LOAD 4 MS DELAY IN BC FOR NOW
            CALL DELAY         TEMPORARY DELAY TO ALLOW DATA TO SETTLE
;           THIS CAN PROBABLY BE REMOVED AND REPLACED WITH COMPUTATIONS
CIRCOL      MVI A,PRT1I
            OUT PORT1          SET PORT1 FOR INPUT
            MVI A,0B0H         STROBE ADC 1011/0000,BIT B5 HI
            OUT PORT1B
            MVI A,090H         THEN STROBE 1001/0000,BIT B5 LO
            OUT PORT1B         A START COMMAND HAS BEEN SENT TO ADC'S
            LXI H,ASCDAT+4*NCHA;LOAD HL FOR DATA COUNTER VALUE
            MVI M,0DH          LOAD CARRIAGE RETURN INTO MEMORY AT END OF
DAT
            DCX H              DECREMENT FOR NEXT DATA BYTE
            SHLD DIAD          STORE DATA COUNT ADDRESS IN RAM
            LDA DATIN          TEST FOR CMD1 OR CMD2
            CPI 031H           SEE IF CMD1 WAS TRANSMITTED
            CZ ADBUS           WAIT FOR RDG TO END BEFORE DATA IS READ
;                              READ EACH ADC AND CONVERT TO ASCII REP OF
BCD
            MVI A,NCHA         LOAD CHANNEL COUNT
            STA NCH            STORE IN RAM,CH NO GOES FROM 0 TO 7
ADRDGL      ADI 097H           CH SEL NO +CD'S ON 8304,SEL ON ADC
            OUT PORT1B         SET CH. SWITCH TO CH. NO.
            IN PORT1C          HIGH ORDER BYTE IS BROUGHT IN.
            ANI 0FH            4 HIGHEST ORDER BITS ARE DUMPED
            RLC                ROTATE LEFT,DOUBLE VALUE,ADDR ARE 2 BYTES
            MOV E,A            STORE IN E TEMPORARILY
            IN PORT1A          LOW ORDER BYTE IS BROUGHT IN AND
            ANI 0F0H           DUMP LOW NIBBLE
            RRC                ROTATE RIGHT 3 TIMES
            RRC                RESULT IS 2ND NIBBLE DOUBLED
            RRC                ADDRESSES OF VALUES ARE 2 BYTES WIDE
            MOV C,A            STORE IN C
            MVI B,0            BC CONTAIN OFFSET ADDR FOR 2ND NIBBLE
            LXI H,NIB2B        LOAD BASE ADDRESS FOR 2ND NIBBLE
            DAD B              ADD OFFSET TO HL ADDRESS
            MOV C,E            BC NOW CONTAINS OFFSET FOR 3RD NIBBLE
            IN PORT1A          LOW ORDER BYTE IS ENTERED AGAIN
            ANI 0FH            DUMP 2ND NIBBLE,HI 4 BITS
            DAA                DECIMAL ADJUST ACCUMULATOR
            ADD M              ADD 2ND NIBBLE DECIMAL VALUE FROM MEMORY
            DAA                DECIMAL ADJUST ACCUMULATOR
            MOV E,A            STORE SUM IN DE
            INX H              HI BYTE OF 2ND NIBBLE
            MVI A,0            ZERO A
```

```
ADC M          ADD HI BYTE OF DEC VAL OF 2ND NIB
MOV D,A        DE CONTAINS SUM
LXI H,NIB3B    BASE ADDRESS FOR 3RD NIBBLE
DAD B          ADD OFFSET POINTER FOR DEC VAL OF 3RD NIB
MOV A,E        BRING SUM OF 1ST 2 NIBS BACK IN
ADD M          ADD LO BYTE OF 3RD
DAA            DECIMAL ADJUST
MOV E,A        STORE BACK IN DE
INX H          INCREMENT TO HI BYTE OF DEC VAL OF 3RD NIB
MOV A,D        BRING IN 2ND BYTE OF SUM OF 1ST 2 NIBS
ADC M          ADD HI BYTE OF 3RD NIB
DAA            DECIMAL ADJUST
MOV D,A        STORE SUM IN DE
LHLD DIAD      LOAD ADDRESS FOR ASCII DATA IN HL
MOV A,E        LOAD LOW ORDER BYTE FROM DE
ANI OFH        DUMP 4 HI BYTES OF LOW ORDER BYTE
ADI 30H        CONVERT TO ASCII
MOV M,A        STORE IN RAM
DCX H          ADDRESS NEXT MEMORY BYTE
MOV A,E        BRING LO BYTE BACK
ANI OFOH       DUMP LOW 4 BITS
RAR
RAR            ROTATE TO LOW 4 BIT POSITION
RAR
RAR
ADI 30H        CONVERT TO ASCII
MOV M,A        STORE IN RAM
DCX H          ADDRESS NEXT MEMORY BYTE
MOV A,D        LOAD HI ORDER BYTE FROM DE
ANI OFH        DUMP 4 HI BITS OF HI ORDER BYTE
ADI 30H        CONVERT TO ASCII
MOV M,A        STORE IN RAM
DCX H          ADDRESS NEXT MEMORY BYTE
MOV A,D        BRING HI BYTE BACK
ANI OFOH       DUMP LOW 4 BITS
RAR
RAR            ROTATE TO LOW 4 BIT POSITION
RAR
RAR
ADI 30H        CONVERT TO ASCII
MOV M,A        STORE IN RAM
DCX H          DECREMENT ASCII MEMORY COUNTER
SHLD DIAD      STORE BACK FOR NEXT LOOP
LDA NCH        LOAD CHANNEL COUNT FROM RAM
DCR A          DECREMENT CHANNEL COUNT
STA NCH        STORE IT BACK IN RAM
JNZ ADRDGL     GO BACK TO ADC READINGS IF NOT DONE
LXI H,DATIN    RESET INPUT DATA ADDRESS FOR NEXT WRITE
SHLD DIAD      STORE ADDRESS IN RAM
LXI H,ASCDAT   LOAD ADDRESS OF DATA TO BE SENT
MVI C,4*NCHA     NUMBER OF CHARACTERS TO BE SENT
MVI A,013H     ENABLE READ,WRITE OR EOI
OUT INTE1      SEND TO 8291 INT ENA
```

```
        EI              ENABLE INTERRUPTS ON 8080
        HLT             WAIT FOR INTERRUPT FROM 8291 OVER GPIB
;
;
;       CMD9            PROGRAM WILL JUMP TO RAM ADDRESS 3800
;                       THERE SHOULD BE A PROGRAM THERE
;
;
CMD9    JMP RAMADR      GO TO RAM ADDRESS
;
NIB2B   DB 0,0,16H,0,32H,0,48H,0,64H,0,080H,0,096H,0,12H,1
        DB 28H,1,44H,1,60H,1,76H,1,092H,1,8,2,24H,2,40H,2
NIB3B   DB 0,0,56H,2,12H,5,68H,7,24H,10H,080H,12H
        DB 36H,15H,092H,17H,048H,20H,4,23H,60H,25H
        DB 16H,28H,72H,30H,28H,33H,084H,35H,40H,38H
        END
```

APPENDIX C

THE BIGRDG PROGRAM

The program BIGRDG is used to position a sample in one or more dimensions with respect to an eddy-current probe and then take multiple frequency magnitude and phase readings at these locations. The input data containing the locations and positions of the properties, along with the coil and instrument set-up data are read from a data file. The program reads the data file, takes calibration readings of the three-frequency instrument, positions the samples using commands sent to a Modulynx controller over the IEEE-488 bus, takes three sets of readings, averages the readings, and stores the readings and their associated property values in a data file to be used by the fitting program, BIGFIT. A listing of the program BIGRDG follows:

```
      PROGRAM BIGRDG
C     DATE  November 3, 1987
C
C      THIS PROGRAM READS MAGNITUDE AND PHASE DATA AT DIFFERENT
C     FREQUENCIES,USING THE PHASE SENSITIVE INSTRUMENT AND THE
C     MECHANICAL SCANNER.IT IS DESIGNED FOR LARGE ARRAYS OF READINGS,
C     TOO LARGE FOR THE NORMAL READING AND FITTING ROUTINES.
C     THE MAGNITUDE AND PHASE DATA IS STORED AFTER EACH SET OF
C     PROPERTIES.  THE PROPERTY AND POSITIONING DATA IS READ FROM DEVICE
C     LID.
C
C     LI=LOGICAL INPUT TERMINAL(FROM OPERATOR)
C     LID=LOGICAL INPUT TERMINAL(MAY BE ASSIGNED TO A DISK FILE)
C     LOT=LOGICAL OUTPUT TERMINAL(TO OPERATOR)
C     LPT=LINE PRINTER(OUTPUT)TERMINAL
C     MAG=INDEX TO TURN ON THE SATURATING MAGNET(=0 IF NONE USED)
C     MSET=NO OF SETS OF READINGS THAT WILL BE TAKEN
C     NCH=NUMBER OF CHANNELS TO BE READ,USUALLY 2*NFT
C     NFT=NUMBER OF FREQUENCIES
C     LNF=2*NFT
C     NF=FREQUENCY INDEX
C     NAXIS=NUMBER OF AXIS THAT ARE POSITIONED
C     NSTOP=INDEX TO STOP THE INSTRUMENT READINGS
C     NP=PROPERTY INDEX
C     NPROM=MAX NUMBER OF PROPERTIES THAT MAY VARY
C     NPT=TOTAL NUMBER OF SETS OF PROPERTY VALUES FOR A COMPLETE
C          SET OF READINGS=NO SAMPLES+NO DEFECTS
C     NPHCAL=NUMBER OF PHASE CALIBRATIONS
C     NMGCAL=NUMBER OF MAGNITUDE CALIBRATIONS
C     NCAL=NPHCAL*NMGCAL=TOTAL NUMBER OF CALIBRATION READINGS
C
      INTEGER*4 NSER,NOLD(3)
      CHARACTER*6 NPROBE,NCABLE,INSTNO,POWOSC,CALIB,PICKAM,PHADET,COIL
      CHARACTER INDAT*8,OTDAT*8,FNAME*12,BLANKS*4,PRONAM(8)*4
      CHARACTER PHASW(4)*4,AXISNM(4)*4,MOTON*8,MOTOF*8,HOME*5
C     DIMENSION FREQ(NFT),PICKAM(NFT),PHADET(NFT),PHASW(NFT)
C     DIMENSION TMAG(NPT,NFT),PHASE(NPT,NFT),PROP(NPT,NPROM)
C     DIMENSION SUMCAL(NCH,NCAL),SSCAL(NCH,NCAL),SDVCAL(NCH,NCAL)
C     DIMENSION VOLTS(NCH),VOLSTD(NCH),RDGC(NCH,NCAL),OLRDGC(NCH,NCAL)
C     DIMENSION XNEW(NAXIS),NOLD(NAXIS),POST(NPT,NAXIS),STDPOS(NAXIS)
C
```

```
      DIMENSION TMAG(7695,3),PHASE(7695,3),PROP(7695,7)
      DIMENSION SUMCAL(6,4),SSCAL(6,4),SDVCAL(6,4)
      DIMENSION VOLTS(6),VOLSTD(6),RDGC(6,4),OLRDGC(6,4)
      DIMENSION XNEW(3),POST(7695,3),STDPOS(3)
C
      DATA LI/5/,LOT/0/,LPT/8/,NCH/6/,LID/9/,NCL/28/,LOD/37/,MSET/3/
      DATA MFEC/5/,MBUS/6/,NOLD/3*0/,IBM/21/,LEVEL/0/
      DATA BLANKS/'    '/,INDAT/'NRCTUB5 '/,OTDAT/'NRCRDG '/
      DATA ITIMC/10/,ITIMS/12/,ITIM/16/,MAG/0/,NPHCAL/2/,NMGCAL/2/
      DATA NPT/7695/,NPROM/7/,NAXIS/2/
      DATA MOTON/'XD YD ZD'/,MOTOF/'XE YE ZE'/,HOME/'<CAH>'/
C
C     PRINT TITLE AND DATE
      CALL GETTIM(IHR,IMN,ISE,IFR)
      CALL GETDAT(IYR,IMO,IDA)
      WRITE(LOT,2)IHR,IMN,ISE,IMO,IDA,IYR
    2 FORMAT(' BIGRDG    TIME ',I2,':',I2,':',I2,'  DATE '
     *,I2,'/',I2,'/',I4)
C
   30 FORMAT(1X)
50    FORMAT(' TYPE IN THE FOLLOWING DATA AS REQUESTED.')
C
C     OPEN FILE FOR INST. DESCRIPTION INPUT - ASSUMED ON DEFAULT DISK
   40 FORMAT(A8)
      FNAME=INDAT//'.DAT'
      OPEN(LID,FILE=FNAME,STATUS ='OLD')
      WRITE(0,*)'INPUT FILE OPENED'
C     OPEN FILE FOR OUTPUT DATA STOR.- ASSUMED LOCATION IN FORTRAN DIR.
      FNAME=OTDAT//'.DAT'
      OPEN(LOD,FILE=FNAME,STATUS ='NEW')
      WRITE(0,*)'OUTPUT FILE OPENED'
C     INITIALIZE BUS
      CALL INITIALI(IBM,LEVEL)
C
C     MULTI FREQUENCY EDDY CURRENT INSTRUMENT ADDR=5 ON GPIB BUS
C     MODULYNX AXIS POSITION CONTROLLER ADDR=6 ON GPIB BUS
      CALL SEND(MBUS,MOTON,ISTAT)
      CALL SEND(MBUS,HOME,ISTAT)
      CALL SEND(MBUS,MOTOF,ISTAT)
C
C INPUT DESCRIPTION OF EXPERIMENTAL APPARATUS
C
      WRITE(LOT,60)
60    FORMAT(' PROBE #:  ')
      READ(LID,70)NPROBE
      WRITE(LOT,70)NPROBE
70    FORMAT(A6)
      WRITE(LOT,80)
80    FORMAT(' SERIAL #:  ')
      READ(LID,*)NSER
      WRITE(LOT,*)NSER
      WRITE(LOT,90)
90    FORMAT(' DRIVER SERIES RESISTANCE:  ')
```

```
      READ(LID,*)R0
      WRITE(LOT,*)R0
      WRITE(LOT,100)
100   FORMAT(' DRIVER SHUNT CAP:  ')
      READ(LID,*)CAPDR
      WRITE(LOT,*)CAPDR
      WRITE(LOT,110)
110   FORMAT(' PICK-UP SHUNT RESISTANCE:  ')
      READ(LID,*)R9
      WRITE(LOT,*)R9
      WRITE(LOT,120)
120   FORMAT(' PICK-UP SHUNT CAP:  ')
      READ(LID,*)CAPPU
      WRITE(LOT,*)CAPPU
      WRITE(LOT,130)
130   FORMAT(' CABLE I.D. #:  ')
      READ(LID,70)NCABLE
      WRITE(LOT,70)NCABLE
      WRITE(LOT,140)
140   FORMAT(' LENGTH OF CABLE:  ')
      READ(LID,*)CABLEL
      WRITE(LOT,*)CABLEL
      WRITE(LOT,150)
150   FORMAT(' CAPACITANCE OF CABLE:  ')
      READ(LID,*)CCABLE
      WRITE(LOT,*)CCABLE
      WRITE(LOT,160)
160   FORMAT(' EDDY CURRENT INSTRUMENT #:  ')
      READ(LID,70) INSTNO
      WRITE(LOT,70) INSTNO
      WRITE(LOT,170)
170   FORMAT(' POWER OSC I.D.:  ')
      READ(LID,175)POWOSC,CALIB
      WRITE(LOT,175)POWOSC,CALIB
175   FORMAT(A6,1X,A6)
C
C INPUT FREQUENCY VALUES
C
      WRITE(LOT,190)
190   FORMAT(' NO. OF FREQUENCIES:')
      READ(LID,*)NFT
      WRITE(LOT,*)NFT
      WRITE(LOT,200)
200   FORMAT(' INPUT THE VALUE OF EACH FREQ SEPARATED BY A SPACE:',/)
      READ(LID,*)(FREQ(NF),NF=1,NFT)
      WRITE(LOT,*)(FREQ(NF),NF=1,NFT)
      WRITE(LOT,210)(FREQ(NF),NF=1,NFT)
210   FORMAT(' INPUT THE FOLLOWING DATA,6 CHAR WITH A SPACE BETWEEN',/,
     * ' FREQUENCY:',10X,10(1PE9.2))
      WRITE(LOT,220)
220   FORMAT(' PICK-UP AMP I.D.:     ')
      READ(LID,230)(PICKAM(NF),NF=1,NFT)
      WRITE(LOT,230)(PICKAM(NF),NF=1,NFT)
```

```
230   FORMAT(10(A6,1X))
      WRITE(LOT,240)
240   FORMAT(' PHASE DETECTOR I.D.: ')
      READ(LID,230)(PHADET(NF),NF=1,NFT)
      WRITE(LOT,230)(PHADET(NF),NF=1,NFT)
      WRITE(LOT,250)
250   FORMAT(' 180-DEG SW(OFF/ON):    ')
      READ(LID,260)(PHASW(NF),NF=1,NFT)
      WRITE(LOT,260)(PHASW(NF),NF=1,NFT)
260   FORMAT(10(A3,1X))
      READ(LID,*)NPROM.NAXIS,NPT
      WRITE(LOT,*)NPROM,NAXIS,NPT
      READ(LID,265)(STDPOS(NAX),NAX=1,NAXIS)
265   FORMAT(F6.3,1X,F6.3,1X,F6.3)
      WRITE(LOT,265)(STDPOS(NAX),NAX=1,NAXIS)
      READ(LID,270)(PRONAM(NPR),NPR=1,NPROM),(AXISNM(NAX),NAX=1,NAXIS)
261   WRITE(LOT,280)(PRONAM(NPR),NPR=1,NPROM),(AXISNM(NAX),NAX=1,NAXIS)
CTHIK1DFSZ1DFLC12RDCL12FE12CU12TSPAXS123TAXS123456789
CTHIK1DFSZ1DFLC12LOFF1TS1FE1CU123PAXS123TAXS123THAX89
  270 FORMAT(1X,A4,1X,A4,1X,A4,2X,A4.1X,A2,1X,A2,1X,A2,3(3X,A4))
280   FORMAT(10(3X,A4))
C
C READ PROPERTY VALUES AND LOCATION DATA FOR THE POSITIONERS FROM LID
C
      DO 300 NP=1,NPT
      READ(LID,287)(PROP(NP,NPR),NPR=1,NPROM),(POST(NP,NAX),NAX=1,NAXIS)
C     READ(LID,*)(PROP(NP,NPR),NPR=1,NPROM),(POST(NP,NAX),NAX=1,NAXIS)
C 285 WRITE(LOT,290)NP,(PROP(NP,NPR),NPR=1,NPROM)
C    *,(POST(NP,NAX),NAX=1,NAXIS)
  287 FORMAT(F5.4,1X,F4.3,1X,F4.3,1X,F5.4,1X,F2.0,1X,F2.0,1X,F2.0,1X,
     *F6.3,1X,F6.3,1X,F6.3)
  290 FORMAT(I4,1X,10(F7.3))
  300 CONTINUE
      LNF=2*NFT
      NCAL=NPHCAL*NMGCAL
C
C     THIS SECTION TAKES THE CALIBRATION READINGS.
C
700   WRITE(LOT,*)' CALIBRATION READINGS:TYPE ANY KEY TO CONTINUE '
750   CALL CALMIC(OLRDGC,MFEC,ITIMC,NCH)
      CALL GETKEY(IKY)
      IF(IKY.EQ.0)GO TO 750
C
      CALL SEND(MBUS,MOTON,ISTAT)
C
C END OF SECTION WHICH TAKES CALIBRATION READINGS
C
C TAKE READINGS ON NOMINAL STANDARD FOR VOLSTD(I) READINGS
C
      DO 760 NAX=1,NAXIS
      XNEW(NAX)=STDPOS(NAX)
760   CONTINUE
      CALL POSIT(XNEW,NOLD,NAXIS,MBUS)
```

```
773   WRITE(LOT,775)
775   FORMAT(' TAKE STD RDGS ----- HIT ANY KEY TO STOP ')
      WRITE(LOT,920)(BLANKS,II.II, II=1,NFT)
      WRITE(LOT,30)
780   CALL READ(VOLSTD,MFEC,ITIMS,5,NCH)
      CALL GETKEY(IKY)
      IF(IKY.EQ.0)GO TO 780
      WRITE(LOT,785)
785   FORMAT(' SET UP SCANNER AND CALIB STANDARD,HIT ANY KEY TO GO')
 787  CALL READ(VOLTS,MFEC,ITIM,6,NCH)
      CALL GETKEY(IKY)
      IF(IKY.EQ.0)GO TO 787
C      ZERO ARRAYS THAT WILL CONTAIN SUMS OF THE PHASE & MAG READINGS,
C     SUMS OF CALIBRATION READINGS, AND SUMS OF SQUARES OF EACH
C
790   CONTINUE
      DO 800 NF=1,NFT
      DO 800 NP=1,NPT
      TMAG(NP,NF)=0.
      PHASE(NP,NF)=0.
800   CONTINUE
      DO 810 NF=1,LNF
      DO 810 NC=1,NCAL
      SUMCAL(NF,NC)=OLRDGC(NF,NC)
      SSCAL(NF,NC)=OLRDGC(NF,NC)*OLRDGC(NF,NC)
810   CONTINUE
C
C THIS SECTION TAKES THE ACTUAL PHASE & MAGNITUDE DATA READINGS
C
      DO 1150 MSE=1,MSET
      CALL READ(VOLTS,MFEC,ITIM, ,NCH)
      DO 980 NP=1,NPT
      WRITE(LOT,900)NP,(PROP(NP,NPR),NPR=1,NPROM)
     *,(POST(NP,NAX),NAX=1,NAXIS)
900   FORMAT(I4,10(F7.3))
C
C     POSITION SAMPLES BEFORE TAKING READINGS
C
      DO 910 NAX=1,NAXIS
      XNEW(NAX)=POST(NP,NAX)
910   CONTINUE
      CALL POSIT(XNEW,NOLD,NAXIS,MBUS)
c      MAXCH=2*(NFT-1)+1
      WRITE(LOT,920)(BLANKS,II,II, II=1,NFT)
920   FORMAT(3X,10(A1,'MAG(',I1,')',4X,' PH(',I1,')',5X))
C     WRITE(LOT,30)
930   CALL READ(VOLTS,MFEC,ITIM,6,NCH)
      WRITE(LOT,950)(VOLTS(JJ),JJ=1,LNF)
950   FORMAT(10(F9.3,2X))
      IF(MAG.EQ.1)CALL READ(VOLTS,MFEC,ITIM,6,NCH)
      DO 980 NF=1,NFT
      TMAG(NP,NF)=TMAG(NP,NF)+VOLTS(2*NF-1)
      PHASE(NP,NF)=PHASE(NP,NF)+VOLTS(2*NF)
```

```
980   CONTINUE
1000  CONTINUE
      DO 1130 NC=1,NCAL
      CALL READ(VOLTS,MFEC,ITIMC,NC,NCH)
      DO 1120 NF=1,LNF
      SUMCAL(NF,NC)=SUMCAL(NF,NC)+VOLTS(NF)
      SSCAL(NF,NC)=SSCAL(NF,NC)+VOLTS(NF)*VOLTS(NF)
1120  CONTINUE
1130  CONTINUE
1150  CONTINUE
C
C     RETURN POSITIONER TO XNEW(II)=0.0 SO THAT POSITIONS CAN BE CHECKED
C     THEN TURN MOTOR CURRENT OFF
C
       CALL SEND(MBUS,HOME,ISTAT)
       GO TO 1170
 1160  WRITE(0,*)'ERROR IN FINAL HOME POSITION'
 1170  CALL SEND(MBUS,MOTOF,ISTAT)
       GO TO 1190
 1180  WRITE(0,*)' ERROR IN CURRENT SHUT-DOWN'
C
C BEFORE STOPPING, ADD FINAL SET OF CALIBRATION READINGS TO CUMULATIVE
C     SUM & CALCULATE AVERAGES & STANDARD DEVIATIONS
C
1190  DO 1250 NF=1,LNF
      DO 1250 NC=1,NCAL
      SUMCAL(NF,NC)=SUMCAL(NF,NC)/(FLOAT(MSET+1))
      SDVCAL(NF,NC)=SQRT(ABS((SSCAL(NF,NC)-SUMCAL(NF,NC)*SUMCAL(NF,NC)
     *             *FLOAT(MSET+1))/FLOAT(MSET)))
1250  CONTINUE
C
C CALCULATE AVERAGES OF THE READINGS
C
      DO 1260 NF=1,NFT
      DO 1260 NP=1,NPT
      TMAG(NP,NF)=TMAG(NP,NF)/(FLOAT(MSET))
      PHASE(NP,NF)=PHASE(NP,NF)/(FLOAT(MSET))
1260  CONTINUE
C
C STORE ALL INFORMATION IN DIRECT ACCESS FILE LOD ON DISK
C
      WRITE(LOD,1270)IHR,IMN,ISE,IMO,IDA,IYR
1270  FORMAT(6(1X,I4))
      WRITE(LOD,1275) NPROBE,NSER,R0,CAPDR,R9,CAPPU,NCABLE,
     *             CABLEL,CCABLE,INSTNO,POWOSC,CALIB
1275  FORMAT(A6,1X,I3,4(E13.4),1X,A6,2(E13.4),3(1X,A6))
      WRITE(LOD,1277) NFT,NPT,NPROM,NPHCAL,NMGCAL
1277  FORMAT(5(1X,I5))
      WRITE(LOD,1280)(FREQ(NF),NF=1,NFT)
1280  FORMAT(4(E13.4))
      WRITE(LOD,1282)(PICKAM(NF),PHADET(NF),PHASW(NF),NF=1,NFT)
1282  FORMAT(4(1X,A6,1X,A6,1X,A4))
      WRITE(LOD,1285)(PRONAM(NPR),NPR=1,NPROM)
```

```
1285  FORMAT(7(1X,A4))
      DO 1295 NF=1,LNF
      WRITE(LOD,1290)(SUMCAL(NF,NC),NC=1,NCAL)
1290  FORMAT(6(F7.4))
1295  CONTINUE
      WRITE(LOD,1290)(VOLSTD(NF),NF=1,LNF)
      DO 1300 NP=1,NPT
      WRITE(LOD,1297)(TMAG(NP,NF),PHASE(NP,NF),NF=1,NFT),(PROP(NP,NPR)
     *,NPR=1,NPROM)
1297  FORMAT(14(1X,F8.4))
1300  CONTINUE
C
C END OF SECTION WHICH WRITES DIRECT ACCESS FILE
C PRINT SUMMARY OF JOB STATISTICS ON LPT
C
      WRITE(LPT,1350) NPROBE,NSER
1350  FORMAT(' PROBE NO.:',A6,5X,' SERIAL NO.:',I5)
      WRITE(LPT,1360) R0,CAPDR
1360  FORMAT(' DRIVER SERIES RESISTANCE:',F10.1,5X,'DRIVER SHUNT CAP.:',
     *         E12.4)
      WRITE(LPT,1370) R9,CAPPU
1370  FORMAT(' PICK-UP SHUNT RESISTANCE:',F10.1,5X,'PICK-UP SHUNT CAP.:'
     *          ,E12.4)
      WRITE(LPT,1380) NCABLE,CABLEL,CCABLE
1380  FORMAT(' CABLE I.D. NO.:',A6,5X,'LENGTH:',F10.1,5X,'CAP.:',
     *          E12.4)
      WRITE(LPT,1390) INSTNO
1390  FORMAT(' EDDY CURRENT INSTRUMENT NO.:',A6)
      WRITE(LPT,1400) POWOSC,CALIB
1400  FORMAT(' POWER OSC I.D.',A6,1X,' CALIBRATOR MODULE ',A6)
      WRITE(LPT,1410) (FREQ(NF),NF=1,NFT)
1410  FORMAT(/,10X,'FREQUENCY:',10(1PE12.4,5X))
      WRITE(LPT,1420) (PICKAM(NF),NF=1,NFT)
1420  FORMAT(' PICK-UP AMP I.D.:',7X,10(A6,9X))
      WRITE(LPT,1430) (PHADET(NF),NF=1,NFT)
1430  FORMAT(' PHASE DETECTOR I.D.:',4X,10(A6,9X))
      WRITE(LPT,1440) (PHASW(NF),NF=1,NFT)
1440  FORMAT(' 180  PHASE SWITCH:',8X,10(A3,12X))
C
C     PRINT CALIBRATION READINGS
C
1600  CONTINUE
      WRITE(LPT,1650) NPHCAL,NMGCAL
1650  FORMAT(1H0,' AVERAGES & STANDARD DEVIATIONS OF CALIBRATION ',
     *        'READINGS:',I3,' MAG',I3,' PHA'/)
      WRITE(LPT,1660) (FREQ(NF),NF=1,NFT)
1660  FORMAT(3(10X,1PE12.4))
      WRITE(LPT,1670)(NF,NF,NF=1,NFT)
1670  FORMAT(5X,3(5X,'MAG',I1,6X,'PHA',I1,1X))
      DO 1700 NC=1,NCAL
      WRITE(LPT,30)
      WRITE(LPT,1680)((SUMCAL(NF,NC)),NF=1,LNF)
```

```
1680  FORMAT(5X,6(F10.4))
      WRITE(LPT,1690)((SDVCAL(NF,NC)),NF=1,LNF)
1690  FORMAT(' S.D.',6(F10.4))
1700  CONTINUE
C
C     READINGS FROM NOMINAL SAMPLE
C
      WRITE(LPT,1745)
1745  FORMAT(' READINGS FROM NOMINAL TUBE SAMPLE')
      WRITE(LPT,1680)(VOLSTD(NF),NF=1,LNF)
C
C     PRINT OUT READINGS AND PROPERTIES
C
      WRITE(LPT,1750)(NF,NF,NF=1,NFT),(PRONAM(NPR),NPR=1,NPROM)
1750  FORMAT('PSET',3('  MAG',I1,'  PHA',I1),1X,4(1X,A4,1X),3(1X,A4))
      DO 1800 NP=1,NPT
C     WRITE(LPT,1760)NP,(TMAG(NP,NF),PHASE(NP,NF),NF=1,NFT),(PROP(NP,
C    *NPR),NPR=1,NPROM)
1760  FORMAT(I4,11(F6.3),2(F5.3))
1800  CONTINUE
      STOP
      END
      SUBROUTINE POSIT(XNEW,NOLD,NAXIS,MBUS)
C
C     DETERMINES THE NEW LOCATION FOR 3 DIFFERENT POSITIONERS AND FROM
C     THE OLD LOCATION,THE NUMBER OF STEPS NEEDED TO REACH THE NEW
C     LOCATION.  THEN SENDS THE NUMBER OVER THE IEEE-488 BUSS TO THE
C     CONTROLLER.  AFTER THE NEW POSITION IS REACHED THE PULSE COUNT IS
C     COMPARED TO THE ENCODER COUNT AND CORRECTED IF DIFFERENT.
C
      INTEGER*4 NOLD(3),NSTEPS(3),NSTP
      DIMENSION XNEW(3)
      CHARACTER*2 AXC(3),GC,DOUT*35,HOME*6,SC*1
      DATA AXC/'XM','YM','ZM'/,GC/'G '/,HOME/';<CFA>'/,SC/';'/
      DATA STEPSZ/0.00025/
      DO 20 IAXIS=1,NAXIS
      NSTP=(XNEW(IAXIS)+0.000125)/STEPSZ
      NSTEPS(IAXIS)=NSTP-NOLD(IAXIS)
      NOLD(IAXIS)=NSTP
   20 CONTINUE
       WRITE(DOUT,30)(AXC(IAXIS),NSTEPS(IAXIS),GC,IAXIS=1,NAXIS)
      WRITE(0,*)DOUT
   24 CALL SEND(MBUS,DOUT,ISTAT)
      IF(ISTAT.NE.0)GO TO 40
   25 CALL SEND(MBUS,HOME,ISTAT)
      IF(ISTAT.NE.0)GO TO 50
   26 CALL SEND(MBUS,SC,ISTAT)
      IF(ISTAT.NE.0)GO TO 60
      GO TO 70
   30 FORMAT(4(A2,I7,A2))
   40 WRITE(0,*)'ERROR,UNABLE TO WRITE POSITION DATA'
      GO TO 70
   50 WRITE(0,*)'ERROR,UNABLE TO WRITE HOME COMMAND'
```

```
      GO TO 70
   60 WRITE(0,*)'ERROR,WAIT TOO LONG FOR AXIS READY COMMAND'
      GO TO 26
   70 RETURN
      END
      SUBROUTINE READ(VOLTS,MFEC,NRDGS,NEXT,NCH)
C     VERSION 7 JULY 1987
C     PROGRAM TO WRITE CONTROL INFO. TO AND READ DATA FROM THE COMP9
C
      INTEGER *2 CMD,OUTC,OUT8(8),OUT16(16)
      CHARACTER DOUT*15,DIN*64
      DIMENSION IN(8),LL(8),VOLTS(NCH)
      DATA CMD/0/
      DATA OUT8/6,4,6,4,6,4,6,5/
      DATA OUT16/6,4,6,4,6,4,6,4,6,4,6,4,6,4,6,5/
      VF1=5.0/4096.
      VFA=VF1/FLOAT(NRDGS)
      GO TO (100,100,100,100,200,300,400,500,999),NEXT
   30 FORMAT(I1,I1)
   40 FORMAT(16I4)
   50 FORMAT(1X,6F6.3)
   60 FORMAT(1X,I2,6F6.3)
   90 FORMAT(1X)
  100 CONTINUE
C     CALIBRATION LOOP
      OUTC=NEXT-1
      DO 110 I=1,NCH
  110 VOLTS(I)=0.
      DO 150 JJ=1,NRDGS
      WRITE(DOUT,30)CMD,OUTC
      CALL SEND(MFEC,DOUT,ISTAT)
      CALL ENTER(DIN,LEN,MFEC,ISTAT)
      READ(DIN,40)(LL(I),IN(I),I=1,NCH)
      DO 145 I=1,NCH
      VOLTS(I)=VFA*FLOAT(IN(I))+VOLTS(I)
  145 CONTINUE
  150 CONTINUE
      WRITE(0,60)NEXT,(VOLTS(I),I=1,NCH)
      IF(NEXT.EQ.4)WRITE(0,90)
      GO TO 999
  200 CONTINUE
C     SINGLE READING LOOP
      OUTC=4
      WRITE(DOUT,30)CMD,OUTC
      CALL SEND(MFEC,DOUT,ISTAT)
      CALL ENTER(DIN,LEN,MFEC,ISTAT)
      READ(DIN,40)(LL(I),IN(I),I=1,NCH)
      DO 245 I=1,NCH
      VOLTS(I)=VF1*FLOAT(IN(I))
  245 CONTINUE
C     WRITE(0,50)(VOLTS(I),I=1,NCH)
      GO TO 999
  300 CONTINUE
```

```
C      AVERAGE NRDGS LOOP
       OUTC=4
       DO 310 I=1,NCH
   310 VOLTS(I)=0.
       DO 350 JJ=1,NRDGS
       WRITE(DOUT,30)CMD,OUTC
       CALL SEND(MFEC,DOUT,ISTAT)
       CALL ENTER(DIN,LEN,MFEC,ISTAT)
       READ(DIN,40)(LL(I),IN(I),I=1,NCH)
       DO 345 I=1,NCH
       VOLTS(I)=VFA*FLOAT(IN(I))+VOLTS(I)
   345 CONTINUE
   350 CONTINUE
C      WRITE(0,50)(VOLTS(I),I=1,NCH)
       GO TO 999
   400 CONTINUE
C      MULTIPLEX AND READ 8 COILS LOOP
       DO 430,II=1,8
       WRITE(DOUT,30)CMD,OUT8(II)
       CALL SEND(MFEC,DOUT,ISTAT)
       CALL ENTER(DIN,LEN,MFEC,ISTAT)
       READ(DIN,40)(LL(I),IN(I),I=1,NCH)
       DO 420 I=1,NCH
       VOLTS(I)=VF1*FLOAT(IN(I))
   420 CONTINUE
       WRITE(0,60)II,(VOLTS(I),I=1,NCH)
   430 CONTINUE
       GO TO 999
   500 CONTINUE
C      MULTIPLEX AND READ 16 COILS LOOP
       DO 530,II=1,16
       WRITE(DOUT,30)CMD,OUT16(II)
       CALL SEND(MFEC,DOUT,ISTAT)
       CALL ENTER(DIN,LEN,MFEC,ISTAT)
       READ(DIN,40)(LL(I),IN(I),I=1,NCH)
       DO 520 I=1,NCH
       VOLTS(I)=VF1*FLOAT(IN(I))
   520 CONTINUE
       WRITE(0,60)II,(VOLTS(I),I=1,NCH)
   530 CONTINUE
   999 RETURN
       END
       SUBROUTINE CALMIC(RDGC,MFEC,NRDGS,NCHS)
C
C      PROGRAM TO DRIVE AUTOMATIC CALIBRATOR MODULE
C      READINGS ARE TAKEN THROUGH MICROCOMPUTER
C
       DIMENSION RDGC(6,4),VOLTS(8)
       DO 100 NEXT=1,4
       CALL READ(VOLTS,MFEC,NRDGS,NEXT,NCHS)
       DO 50 NCH=1,NCHS
       RDGC(NCH,NEXT)=VOLTS(NCH)
    50 CONTINUE
```

```
100 CONTINUE
    RETURN
    END
```

The sample data file, NRCTUB5.DAT, contains the input data for the instrument and coil description, the locations for the positioners, and the tube properties for those positions. A positioner, standard, and the probe would be needed along with the three-frequency instrument to make actual measurements. A listing of the first part of NRCTUB5.DAT follows:

```
720-4
1
1.48E3
0.
333333.
0.
4
480.0
6.29E-10
1512
15A-87 15A113
3
2.E4 1.E5 5.E5
15A115 15A116 15A117
15A-79 15A118 15A-78
 ON  ON  ON
7 2 513
32.500   0.750
 THIK DFSZ DFLC  RDCL  TS  FE  CU PAXS   TAXS
.0191 .000 .000 .0025 0. 0. 0.    2.000  3.700
.0191 .000 .000 .0025 0. 0. 0.    2.000  3.600
.0191 .000 .000 .0025 0. 0. 0.    2.000  3.500
.0191 .000 .000 .0025 0. 0. 0.    2.000  3.450
.0191 .000 .000 .0025 0. 0. 0.    2.000  3.400
.0191 .000 .000 .0025 0. 0. 0.    2.000  3.350
.0191 .000 .000 .0025 0. 0. 0.    2.000  3.300
.0191 .000 .000 .0025 0. 0. 0.    2.000  3.250
.0191 .000 .000 .0025 1. 0. 0.    2.000  3.200
.0191 .000 .000 .0025 1. 0. 0.    2.000  3.150
.0191 .000 .000 .0025 1. 0. 0.    2.000  3.100
.0191 .000 .000 .0025 1. 0. 0.    2.000  3.050
.0191 .000 .000 .0025 1. 0. 0.    2.000  3.000
.0191 .000 .000 .0025 1. 0. 0.    2.000  2.950
.0191 .000 .000 .0025 1. 0. 0.    2.000  2.900
```

The complete file contains 513 lines of data which corresponds to the properties and their location in the standard.

APPENDIX D

THE BIGFIT PROGRAM

The program BIGFIT takes the data written on the disk by the program BIGRDG and performs least squares fits of the properties to the instrument readings. The properties to be fitted, the degree of the polynomial, the functions of the readings to be used, and the cross terms are all selected by the operator on an interactive basis. If the operator desires to fit a limited set of points, rather than the entire set, the program can be modified to transfer around the fitting and the drift calculation sections by using the property values or the number of the reading point (or a function constructed from this reading point). The transfer points are at line 224 and line 267. Examples of functions used to transfer ard instructions are given in the comment statements at these locations. The data arrays must be adjusted to the number of data points to be fitted. In particular, the array READNG(NPT+1,IRDPRM+1) must have the correct dimensions and the array PRO(NPT) must be large enough to contain all the data. Since the arrays can be quite large, this program is usually compiled with the /b option to include big arrays. A listing of BIGFIT follows:

```
      PROGRAM BIGFIT
C     VERSION November 10, 1987
C     PROGRAM TO PERFORM A LEAST SQUARES FIT TO DATA READ INTO A DISK
C     FILE BY BIGRDG PROGRAM AND THEN PERFORM CONTINUOUS BACK
C     CALCULATIONS USING INSTRUMENT READINGS THAT ARE MADE BETWEEN EACH
C     DISPLAYED SET.
C
C     IRDPRM=MAXIMUM NUMBER OF COEFFICIENTS IN EXPANSION
C     LITEK=LOGICAL INPUT UNIT
C     LOTEK=OPERATOR OUTPUT UNIT FOR PROMPTING AND DISPLAY
C     LOU=LOGICAL OUTPUT UNIT FOR PERMANENT RECORD
C     NCHS=NUMBER OF DATA CHANNELS
C     NF=FREQUENCY INDEX
C     NFT=NUMBER OF FREQUENCIES
C     LNF=2*NFT
C     NP=PROPERTY INDEX
C     NPROPM=MAXIMUM NUMBER OF PROPERTIES CALCULATED(=6 NOW)
C     NPT=TOTAL NUMBER OF SETS OF PROPERTY VALUES USED FOR THE FIT
C     NPTT=VALUE OF NPT READ FROM FILE 30, TOTAL POINTS FROM BIGRDG
C          SOME SETS OF RDGS FROM FILE 30 MAY NOT BE USED; NPTT.GE.NPT
C     NPHCAL=NUMBER OF PHASE CALIBRATIONS
C     NMGCAL=NUMBER OF MAGNITUDE CALIBRATIONS
C     NCAL=NPHCAL*NMGCAL=TOTAL NUMBER OF CALIBRATION READINGS
C     NPRINT=PRINT AND TRANSFER INDEX.
C     NSTOP=INDEX TO STOP THE INSTRUMENT READINGS
C     MAG=INDEX TO TURN ON THE SATURATING MAGNET(=0 IF NONE USED)
C
      REAL L2,L4,L3,L5,L6
      CHARACTER*6 NPROBE,NCABLE,INSTNO,POWOSC,PICKAM,PHADET,COIL
      CHARACTER COEDAT*8,FNAME*12,BLANKS*4,PRONAM(7)*4,AXISNM(3)*4
      CHARACTER ADUM*2,CONDITIONS*35
      CHARACTER PROPTY(7)*4,STOP*4,INDAT*8,PHASW(3)*4,POLARY(16,5)*4
C  DIMENSIONS THAT ARE NOT CHANGED:
      DIMENSION VOLTS(12),VOLSTD(6),STDV(6),NCONV(4),CGAIN(6),COFSET(6)
```

```
C
C  DIMENSIONS THAT ARE CHANGED:
C **  DIMENSION READNG(NPT+1,IRDPRM+1),PRO(NPT),POLARY(IRDPRM+1,5)
C     DIMENSION PROP(1,NPROPM),TMDFT(NFT),PHDFT(NFT),JPOL(6,NFT,NPROPM)
C     DIMENSION COE(IRDPRM),COEF(IRDPRM,NPROPM)
C     DIMENSION RDG1(IRDPRM),NPOL(6,NFT)
C     DIMENSION JOFSET(NPROPM),JRDPR(NPROPM)
C     DIMENSION PROPTY(NPROPM),PRONAM(NPROPM),FREQ(NFT),GAIN(NFT)
C     DIMENSION TMAG1(NFT),PHASE1(NFT)
C     DIMENSION PICKAM(NFT),PHADET(NFT)
C     DIMENSION SUMCAL(NCHS,NCAL),SDVCAL(NCHS,NCAL)
C     DIMENSION RDGC(NCHS,NCAL),RDGO(NCHS,NCAL)
C     DIMENSION TOFSET(NCHS),TSLOPE(NCHS)
C
C  THE APPROPRIATE NUMBERS SHOULD BE INSERTED IN THE FOLLOWING
C  DIMENSION STATEMENTS; COMMENTED STATEMENTS MARKED ** MANDATORY
C
      DIMENSION READNG(7696,16),PRO(7695),PROP(1,7)
      DIMENSION TMDFT(3),PHDFT(3),JPOL(6,3,7),COE(15),COEF(15,7)
      DIMENSION RDG1(15),NPOL(6,3),JOFSET(7),JRDPR(7)
      DIMENSION FREQ(3),GAIN(3)
      DIMENSION TMAG1(3),PHASE1(3)
      DIMENSION PICKAM(3),PHADET(3)
      DIMENSION RDGC(6,4),RDGO(6,4)
C     DIMENSION TOFSET(6),TSLOPE(6)
C
C  DATA THAT MAY NEED TO BE CHANGED:
      DATA NPT/7695/,NPRINT/0/,LOU/8/,LITEK/0/,LOTEK/0/,NRDG/30/
      DATA NCL/28/,LID/37/,LOD/38/,MFEC/5/,IBM/21/,LEVEL/0/
      DATA IRDPRM/15/,IR/1/,NCHS/6/,NPROPM/7/,NPROPT/1/,ITIMS/32/
      DATA STOP/'STOP'/,BLANKS/'    '/,NUNIT/1/
      DATA NLINES/13/,INDAT/'NRCRDG '/,COEDAT/'NRCCOE  '/
      DATA COFSET/6*0.0/,CGAIN/6*1./
      DATA CONDITIONS/'TEST OF R40FX REFLECTION PROBE     '/
C
C     TIME AND DATE ARE PRINTED
      CALL GETTIM(IHR,IMN,ISE,IFR)
      CALL GETDAT(IYR,IMO,IDA)
      IYR=IYR-1900
    2 FORMAT('BIGFIT  TIME ',I2,':',I2,':',I2,
     *'  DATE ',I2,'/',I2,'/',I2,2X,A35)
      WRITE(LOU,2,ERR=990)IHR,IMN,ISE,IMO,IDA,IYR,CONDITIONS
C
   30 FORMAT(1X)
      CALL INITIALI(IBM,LEVEL)
C
C     OPEN  FILE FOR INPUT DATA FROM BIGRDG,ASSUMED ON      DEFAULT DISK
   40 FORMAT(A8)
      FNAME=INDAT//'.DAT'
   41 FORMAT(A12)
      OPEN(LID,FILE=FNAME,STATUS ='OLD',ERR=991)
C     OPEN FILE FOR OUTPUT COEF DATA STORAGE - ASSUMED ON DEFAULT DIR.
      FNAME=COEDAT//'.DAT'
      OPEN(LOD,FILE=FNAME,STATUS ='NEW',ERR = 992)
```

```
C
C     MULTI FREQUENCY EDDY CURRENT INSTRUMENT ADDR=5 ON GPIB BUSS
C
C READ INITIAL INFORMATION IN DIRECT ACCESS FILE LID ON DISK
C
   50 READ(LID,55)IHR,IMN,ISE,IMO,IDA,IYR
  55  FORMAT(6(1X,I4))
      READ(LID,60) NPROBE,NSER,R0,CAPDR,R9,CAPPU,NCABLE,
     *             CABLEL,CCABLE,INSTNO,POWOSC
  60  FORMAT(A6,1X,I5,4(E13.4),1X,A6,2(E13.4),2(1X,A6))
      READ(LID,65) NFT,NPTT,NPROPM,NPHCAL,NMGCAL
      WRITE(0,65) NFT,NPTT,NPROPM,NPHCAL,NMGCAL
  65  FORMAT(5(1X,I5))
      READ(LID,70)(FREQ(NF),NF=1,NFT)
  70  FORMAT(4(E13.4))
      READ(LID,75)(PICKAM(NF),PHADET(NF),PHASW(NF),NF=1,NFT)
  75  FORMAT(4(1X,A6,1X,A6,1X,A4))
      READ(LID,80)(PRONAM(NPR),NPR=1,NPROPM)
  80  FORMAT(7(1X,A4))
      NCAL=NPHCAL*NMGCAL
      LNF=2*NFT
      DO 100 NF=1,LNF
      READ(LID,85)(RDG0(NF,NC),NC=1,NCAL)
  85  FORMAT(6(F7.4))
  100 CONTINUE
C     READ NOMINAL  STANDARD VLOTAGES
C
      READ(LID,85)(VOLSTD(NF),NF=1,LNF)
C
C END OF SECTION WHICH READS INITIAL DIRECT ACCESS FILE
C
C      THE DATE AND TIME THE DATA WAS TAKEN ARE PRINTED
C
      IF(IYR.GT.1900)IYR=IYR-1900
      WRITE(LOU,140)IHR,IMN,ISE,IMO,IDA,IYR,NPTT,NPT
  140 FORMAT(' CALIBRATION DATA TAKEN ',I2,':',I2,':',I2,
     *'  DATE ',I2,'/',I2,'/',I2,' POINTS:TOTAL=',I5,' USED=',I5)
  160 FORMAT(1X)
      GO TO 630
C
C     LEAST SQUARES DESIGN SECTION.
C
C     SELECT PROPERTY TO BE FITTED AND SET UP PROPERTY ARRAY.
C
  300 NPTT1=NPT+1
      IF(NPROPT.GT.NPROPM) GO TO 860
  310 WRITE(LOTEK,320)(NPR,PRONAM(NPR),NPR=1,NPROPM)
  320 FORMAT(' SELECT NUMBER OF THE PROPERTY TO BE FITTED:',
     */,7(I3,1X,A4),' ? ')
      READ(LITEK,*,ERR=630)NPROP
      IF(NPROP.GT.NPROPM)NPROP=NPROPM
  350 WRITE(LOTEK,360)
  360 FORMAT(' TYPE 1 IF THERE IS OFFSET; 0 IF NO OFFSET:',/)
```

```
      READ(LITEK,*,ERR=630) JOFSET(NPROPT)
      IOFSET=JOFSET(NPROPT)
      IRDPR=IOFSET
  370 WRITE(LOTEK,380)
  380 FORMAT(' SELECT THE NUMBER OF THE FUNCTION TYPE, POLYNOMIAL',
     *' DEGREE, & # OF '/,
     *' CROSS TERMS FOR EACH MAGNITUDE & PHASE')
      WRITE(LOTEK,390)
  390 FORMAT(' FUNCTION TYPE:1=LINEAR 2=LOG 3=EXP 4=INV ')
  400 WRITE(LOTEK,160)
      WRITE(LOTEK,410)
  410 FORMAT(25X,'FCTN POL # CROSS'/,25X,'TYPE DEG TERMS')
      DO 450 NF=1,NFT
      DO 440 NC=1,2
      NCC=NC*3
      NCP=NCC-1
      NCF=NCP-1
      IF(NC.EQ.1) WRITE(LOTEK,420) FREQ(NF)
      IF(NC.EQ.2) WRITE(LOTEK,430) FREQ(NF)
  420 FORMAT(' MAG AT ',1PE12.6,' HZ    ',\)
  430 FORMAT(' PHA AT ',1PE12.6,' HZ    ',\)
      READ(LITEK,*,ERR=630)
     *JPOL(NCF,NF,NPROPT),JPOL(NCP,NF,NPROPT),JPOL(NCC,NF,NPROPT)
C     JPOL(3,NF,NPROPT)=0
      IRDPR=IRDPR+JPOL(NCP,NF,NPROPT)+JPOL(NCC,NF,NPROPT)
      JRDPR(NPROPT)=IRDPR
      NPOL(NCF,NF)=JPOL(NCF,NF,NPROPT)
      NPOL(NCP,NF)=JPOL(NCP,NF,NPROPT)
      NPOL(NCC,NF)=JPOL(NCC,NF,NPROPT)
  440 CONTINUE
  450 CONTINUE
      IRDPR1=IRDPR+1
      IF(IRDPRM.LT.JRDPR(NPROPT))WRITE(LOTEK,460)
      IF (IRDPRM.LT.JRDPR(NPROPT))GO TO 630
  460 FORMAT(' ERROR: # OF TERMS IN POLARY EXCEEDS DIMENSION')
      JROW=IRDPRM+1
      CALL POLTYP(POLARY,JROW,IRDPR,NPOL,6,NFT,2,IOFSET,LOTEK)
C
C      EXPAND THE RAW READINGS INTO IRDPR READINGS.
C
  470 DO 480 NF=1,NFT
      TMDFT(NF)=0.
      PHDFT(NF)=0.
  480 CONTINUE
      REWIND(LID)
      DO 481 IREC=1,NLINES
      READ(LID,483)ADUM
  483 FORMAT(A2)
  481 CONTINUE
      NR=1
      DO 490 NP=1,NPTT
      READ(LID,*)(TMAG1(NF),PHASE1(NF),NF=1,NFT),(PROP(1,NPR)
     *,NPR=1,NPROPM)
```

```
 482  FORMAT(14(1X,F8.4))
C
C     THE PROPERTIES CAN BE SET AND MODIFIED IN THIS SECTION
C     IF THIS PROPERTY IS NOT TO BE USED TRANSFER TO 490
C
C     IF(PROP(1,7).GT.0.5)GO TO 490
C     IF(ABS(PROP(1,6)+1.000).GT.0.0001)GO TO 490
C     IF(ABS(PROP(1,6)).GT.0.155)GO TO 490
C     IF(PROP(1,2).GT.0.001)GO TO 490
C     IF(MOD(NP,57).GT.45.OR.MOD(NP,57).EQ.0)GO TO 490
C     IF(NP.GT.7650)GO TO 490
      CALL RDGEXP(RDG1,TMAG1,PHASE1,NPOL,IOFSET,TMDFT,PHDFT,IRDPRM,NFT)
      PRO(NR)=PROP(1,NPROP)
      DO 485 IRD=1,IRDPRM
      READNG(NR,IRD)=RDG1(IRD)
 485  CONTINUE
      NR=NR+1
  490 CONTINUE
      NRF=NR-1
C
C     DO THE LEAST SQUARES FIT OF THE READINGS TO THE PROPERTIES.
C
      CALL ALSQS(READNG,PRO,COE,RSOS,NPT,IRDPR,NPTT1,IRDPR1)
C
C     CALCULATE THE DIFFERENCES IN THE FIT AND THE MAXIMUM DRIFTS.
C
  500 SSDRIF=0.
      SSDIFF=0.
      IF(NPRINT.EQ.2)WRITE(LOTEK,510)PRONAM(NPROP)
      IF(NPRINT.EQ.2)WRITE(LOU,510)PRONAM(NPROP)
  510 FORMAT(' PSET',8X,A4,9X,'CAL',8X,'DIFF',7X,'DRIFT')
      REWIND(LID)
      DO 515 IREC=1,NLINES
      READ(LID,483)ADUM
  515 CONTINUE
      NR=1
      DO 570 NP=1,NPTT
      DRIFT=0.
      READ(LID,*)(TMAG1(NF),PHASE1(NF),NF=1,NFT),(PROP(1,NPR)
     *,NPR=1,NPROPM)
C
C      TRANSFER TO 570 IF PROPERTY IS NOT THE ONE WE WANT
C
C     IF(PROP(1,7).GT.0.5)GO TO 570
C     IF(PROP(1,2).GT.0.001)GO TO 570
C     IF(MOD(NP,57).GT.45.OR.MOD(NP,57).EQ.0)GO TO 570
C     IF(NP.GT.7650)GO TO 570
      DO 540 NF=1,NFT
      DO 530 NC=1,2
C
C     ONE MAGNITUDE OR PHASE DRIFT IS SET ON AT A TIME.
C     TYPICAL ERROR IS 1 DIGIT OR 1.2 MILLIVOLTS
C
```

```
      IF(NC.EQ.1)TMDFT(NF)=.0012
      IF(NC.EQ.2)PHDFT(NF)=.0012
      CALL RDGEXP(RDG1,TMAG1,PHASE1,NPOL,IOFSET,TMDFT,PHDFT,IRDPRM,NFT)
C
C     THE POLYNOMIAL IS CALCULATED
C
      SUM=0.
      DO 520 IR=1,IRDPR
      SUM=SUM+COE(IR)*RDG1(IR)
  520 CONTINUE
      DRIFT=DRIFT+ABS(READNG(NR,IRDPR1)-SUM)
      TMDFT(NF)=0.
      PHDFT(NF)=0.
  530 CONTINUE
  540 CONTINUE
      DIFF=PRO(NR)-READNG(NR,IRDPR1)
      SSDIFF=SSDIFF+DIFF*DIFF
      SSDRIF=SSDRIF+DRIFT*DRIFT
      IF(NPRINT.NE.2)GO TO 565
C
C      THE ENTIRE FIT IS PRINTED OUT
C
      WRITE(LOU,560)NP,PRO(NR),READNG(NR,IRDPR1),DIFF,DRIFT
      WRITE(LOTEK,560)NP,PRO(NR),READNG(NR,IRDPR1),DIFF,DRIFT
  560 FORMAT(1X,I4,4F12.5)
  565 NR=NR+1
  570 CONTINUE
      NRD=NR-1
      IF(NPT.NE.NRF.OR.NPT.NE.NRD)WRITE(0,*)'READING ERROR:
     *ASSUMED=',NPT,'  FIT =',NRF,'  DRIFT=',NRD
      SDRIF=SQRT(ABS(SSDRIF/FLOAT(NPT)))
      SDIFF=SQRT(ABS(SSDIFF/FLOAT(NPT)))
      WRITE(LOU,580)PRONAM(NPROP),SDIFF,SDRIF,(NPOL(1,NF),NPOL(4,NF)
     *,NF=1,NFT)
      WRITE(LOU,585)IOFSET,(NPOL(2,NF),NPOL(5,NF),NF=1,NFT)
      WRITE(LOU,587)(NPOL(3,NF),NPOL(6,NF),NF=1,NFT)
      WRITE(LOU,30)
      WRITE(LOTEK,575)PRONAM(NPROP),SDIFF,SDRIF
  575 FORMAT(' RMS DIF IN ',A4,'=',F10.5,2X,'DRIFT=',F10.5)
  580 FORMAT(' RMS DIF IN ',A4,'=',F10.5,2X,
     *'DRIFT=',F10.5,3X,'FCTN',12I2)
  585 FORMAT(I2,' CONSTANT',37X,'POL ',12I2)
  587 FORMAT(48X,'XTRM',12I2)
  590 IF (NPRINT.NE.3)GO TO 630
      IF(NPROPT.NE.1)GO TO 597
      DO 595 NC=1,NCAL
      WRITE(LOD,*)(RDGO(JJ,NC),JJ=1,LNF)
      DO 595 NF=1,LNF
  595 CONTINUE
      WRITE(LOD,*)(VOLSTD(NF),NF=1,LNF)
      DO 596 NF=1,LNF
  596 CONTINUE
  597 WRITE(LOU,160)
```

```
      WRITE(LOTEK,160)
      NCOED1=NCOED+4*IRDPRM+1
      WRITE(LOD,*)IRDPR
      DO 610 I=1,IRDPR
C     WRITE(LOU,600)I,COE(I),(POLARY(I,J),J=1,5)
      WRITE(LOTEK,600)I,COE(I),(POLARY(I,J),J=1,5)
  600 FORMAT(' COEF(',I2,')=',1PE15.7,4X,5A4)
      COEF(I,NPROPT)=COE(I)
      DO 610 NCO=1,4
C     WRITE(LOD,*)NCONV(NCO)
  610 CONTINUE
      NCOED=NCOED1
C
C     THE COEFFICIENT,OFFSET,NPOL AND IRDPR ARE WRITTEN ON THE DISC
C
      WRITE(LOD,615)PRONAM(NPROP),IRDPR,JOFSET(NPROPT)
  615 FORMAT(1X,A4,2I4)
      WRITE(LOD,*)(COEF(IR,NPROPT),IR=1,IRDPR)
      DO 620 NF=1,NFT
      WRITE(LOD,*)(NPOL(I,NF),I=1,6)
  620 CONTINUE
      PROPTY(NPROPT)=PRONAM(NPROP)
      ICOEF=ICOEF-1
      NPROPT=NPROPT+1
C
  630 WRITE(LOTEK,640)
  640 FORMAT(' 1 FIT PROP 2 PRT ENTIRE FIT 3 PRT/SV COEF 4 CHG FCTN/POL'
     *,'TYP 5 RUN TEST',/)
      READ(LITEK,*)NPRINT
      GO TO(300,500,590,350,650),NPRINT
  650 NPROPT=NPROPT-1
      WRITE(LOU,160)
      GO TO 880
C     CALCULATES PROPERTIES FROM MAGNITUDES AND PHASES AND CONTINUOUSLY
C     DISPLAYS THE VALUES ON THE CRT TERMINAL.
C
  700 CONTINUE
      IF(NPRINT.EQ.2)WRITE(LOTEK,710)(PROPTY(NPRO),NPRO=1,NPROPT)
  710 FORMAT(1X,6(8X,A4,1X))
      IF(NPRINT.EQ.3)WRITE(LOTEK,715)(II,II,II=1,NFT)
  715 FORMAT(1X,3('      MAG(',I1,')       PHA(',I1,')'))
      WRITE(LOTEK,160)
C
C     EXPANSION OF TMAG1(NF) AND PHASE1(NF) INTO READNG(1,IRDPRM)
C
  720  DO 800 NPRO=1,NPROPT
      DO 790 NF=1,NFT
      DO 780 I=1,6
      NPOL(I,NF)=JPOL(I,NF,NPRO)
  780 CONTINUE
C     WRITE(0,721)(NPOL(I,NF),I=1,6)
C 721 FORMAT(6I3)
      TMDFT(NF)=0.
```

```
      PHDFT(NF)=0.
  790 CONTINUE
      IOFSET=JOFSET(NPRO)
      IRDPR=JRDPR(NPRO)
      IRDPR1=IRDPR+1
      CALL RDGEXP(RDG1,TMAG1,PHASE1,NPOL,IOFSET,TMDFT,PHDFT,IRDPRM,NFT)
      PRO(NPRO)=0.
      DO 800 IR=1,IRDPR
      PRO(NPRO)=PRO(NPRO)+COEF(IR,NPRO)*RDG1(IR)
  800 CONTINUE
C     NSTART=1
      IF(NPRINT.EQ.2)WRITE(LOTEK,805)(PRO(NPRO),NPRO=1,NPROPT)
  805 FORMAT(1X,7(F13.5))
C
C     NEW READINGS ARE MADE FROM THE EDDY CURRENT INSTRUMENT.
C
  825 CALL READ(VOLTS,MFEC,ITIMS,6,NCHS)
C     IF(MAG.EQ.1) CALL SATMAG(0)
C     VOLTS(8)=25.*VOLTS(7)/VOLTS(8)
C     DO 830 NC=1,NCHS
C     VOLTS(NC)=VOLTS(NC)+TOFSET(NC)+TSLOPE(NC)*VOLTS(8)
  830 CONTINUE
      IF(NPRINT.EQ.3)WRITE(LOTEK,820)(VOLTS(2*NF).VOLTS(2*NF-1)
     *,NF=2,LNF,2)
      CALL CORRDG(VOLTS,COFSET,CGAIN,NCHS)
  840 DO 845 NF=1,NFT
      TMAG1(NF)=VOLTS(2*NF-1)
      PHASE1(NF)=VOLTS(2*NF)
  845 CONTINUE
  820 FORMAT(1X,6(F12.4))
      CALL GETKEY(IRR)
      IF(IRR.EQ.0)GO TO 720
C
C     PROGRAM WILL STAY IN THIS LOOP UNTIL ANY KEY IS STRUCK.
C
      IF(NLABL+NPRINT.EQ.3)WRITE(LOU,710)(NPRO,NPRO=1,NPROPT)
      NLABL=NPRINT
      IF(NPRINT.EQ.2)WRITE(LOU,850)(PRO(NPRO),NPRO=1,NPROPT)
  850 FORMAT(1X,6(F12.5))
      IF(NPRINT.EQ.3)WRITE(LOU,850)(VOLTS(2*NF-1),VOLTS(2*NF),
     *NF=1,NFT)
      GO TO 880
  855 WRITE(LOTEK,857)
  857 FORMAT(' LIMIT OF FILE 37 IS EXCEEDED.')
      GO TO 880
  860 WRITE(LOTEK,870)
  870 FORMAT(' PROP ARRAY IS FILLED.')
  880 WRITE(LOTEK,890)
  890 FORMAT(' PRINT :1.MEAS VOLT & PROPERTIES 2.CAL&DISPLAY PROPS'
     *,'3.RAW RDGS.4.RE-CALIB 5.STOP',/)
      READ(LITEK,*)NPRINT
      GO TO (1300,700,700,895,900),NPRINT
  895 CALL CALMIC(RDGC,MFEC,ITIMS,NCHS)
```

```
      CALL RESET(RDGC,RDGO,COFSET,CGAIN,STDV,NCHS)
      GO TO 880
C
C     PRINT SUMMARY OF DATA ON FILE 37
C
1300  WRITE(LOU,1350) NPROBE,NSER
1350  FORMAT(' PROBE NO.:',A6,5X,' SERIAL NO.:',I5)
      WRITE(LOU,1360) RO,CAPDR
1360  FORMAT(' DRIVER SERIES RESISTANCE:',F10.1,5X,'DRIVER SHUNT CAP.:',
     *         E12.4)
      WRITE(LOU,1370) R9,CAPPU
1370  FORMAT(' PICK-UP SHUNT RESISTANCE:',F10.1,5X,'PICK-UP SHUNT CAP.:'
     *          ,E12.4)
      WRITE(LOU,1380) NCABLE,CABLEL,CCABLE
1380  FORMAT(' CABLE I.D. NO.:',A6,5X,'LENGTH:',F10.1,5X,'CAP.:',
     *          E12.4)
      WRITE(LOU,1390) INSTNO
1390  FORMAT(' EDDY CURRENT INSTRUMENT NO.:',A6)
      WRITE(LOU,1400) POWOSC
1400  FORMAT(' POWER OSC I.D.',A6)
      WRITE(LOU,1410) (FREQ(NF),NF=1,NFT)
1410  FORMAT(/,10X,'FREQUENCY:',10(1PE12.4,5X))
      WRITE(LOU,1420) (PICKAM(NF),NF=1,NFT)
1420  FORMAT(' PICK-UP AMP I.D.:',7X,10(A6,9X))
      WRITE(LOU,1430) (PHADET(NF),NF=1,NFT)
1430  FORMAT(' PHASE DETECTOR I.D.:',4X,10(A6,9X))
      WRITE(LOU,1440) (PHASW(NF),NF=1,NFT)
1440  FORMAT(' 180  PHASE SWITCH:',8X,10(A3,12X))
C
C     PRINT CALIBRATION READINGS
C
1600  CONTINUE
      WRITE(LOU,1650) NPHCAL,NMGCAL
1650  FORMAT(1H0,' AVERAGES & STANDARD DEVIATIONS OF CALIBRATION ',
     *        'READINGS:',I3,' MAG',I3,' PHA'/)
      WRITE(LOU,1660) (FREQ(NF),NF=1,NFT)
1660  FORMAT(3(10X,1PE12.4))
      WRITE(LOU,1670)(NF,NF,NF=1,NFT)
1670  FORMAT(5X,3(5X,'MAG',I1,6X,'PHA',I1,1X))
      DO 1700 NC=1,NCAL
      WRITE(LOU,160)
      WRITE(LOU,1680)((RDGO(NF,NC)),NF=1,LNF)
1680  FORMAT(5X,6(F10.4))
1700  CONTINUE
C
C     PRINT OUT READINGS AND PROPERTIES
C
      WRITE(LOU,160)
      WRITE(LOU,1750)(NF,NF,NF=1,NFT),(PRONAM(NPR),NPR=1,NPROPM)
1750  FORMAT(1X,'PSET',3('     MAG',I1,'      PHA',I1),7(4X,A4,1X))
      REWIND(LID)
      DO 1755 IREC=1,NLINES
      READ(LID,483)ADUM
```

```
 1755 CONTINUE
      DO 1800 NP=1,NPTT
      READ(LID,482)(TMAG1(NF),PHASE1(NF),NF=1,NFT),(PROP(1,NPR)
     *,NPR=1,NPROPM)
      WRITE(LOU,1760)NP,(TMAG1(NF),PHASE1(NF),NF=1,NFT),(PROP(1,
     *NPR),NPR=1,NPROPM)
1760  FORMAT(1X,I4,13(F9.4))
1800  CONTINUE
C
      GO TO 880
  990 WRITE(LOTEK,*)' ERROR IN OPENING PRINTER FILE-CHECK FOR OFF-LINE'
      GO TO 900
  991 WRITE(LOTEK,*)'ERROR IN OPENING INPUT DATA FILE'
      GO TO 900
  992 WRITE(LOTEK,*)'ERROR IN OPENING OUTPUT COEF DATA FILE'
      GO TO 50
  993 WRITE(LOTEK,*)'ERROR IN OPENING INSTRUMENT DATA FILE'
      GO TO 900
  900 STOP
      END
      SUBROUTINE RDGEXP(READNG,TMAG,PHASE,NPOL,IOFSET,TMDFT,PHDFT
     1,IRDPRM,NFT)
C     REAL*8 READNG,RDG
      DIMENSION READNG(IRDPRM),NPOL(6,NFT),TMDFT(NFT),PHDFT(NFT)
      DIMENSION TMAG(NFT),PHASE(NFT)
C
C     NPOL CONTAINS A NUMBER FOR THE FUNCTION TYPE, THE POLYNOMIAL
C     DEGREE, AND THE NUMBER OF CROSS TERMS
C     FOR THE MAGNITUDE AND PHASE AT EACH FREQUENCY,STORED AS NPOL
C     (NF;1-MAG FUN, 2-MAG POL,3-MAG #CROSS TERMS,4-PH FUN,5-PH POL,
C     6-PH # CROSS TERMS).  IF IOFSET=0, NO OFF-SET
C     WILL BE INCLUDED,=1 OFF-SET IS INCLUDED.THE VALUES OF TMDFT(NF)&
C     PHDFT(NF) GIVE THE AMOUNT OF DRIFT IN THE MAGNITUDE AND PHASE.IF
C      NPOL(NCP,NF) =0,THAT PARTICULAR MAGNITUDE AND PHASE FOR THAT
C     FREQUENCY WILL BE SKIPPED.
C
      READNG(1)=1.
      N=1
      IF(IOFSET.EQ.1)N=2
      DO 210 NF=1,NFT
      DO 200 NC=1,2
      NCC=NC*3
      NCP=NCC-1
      NCF=NCP-1
      ROLD=RDG
      IF(NPOL(NCP,NF).EQ.0) GO TO 200
      IF(NC.EQ.1) RDG=TMAG(NF)+TMDFT(NF)
      IF(NC.EQ.2) RDG=PHASE(NF)+PHDFT(NF)
C
C     THE TYPE OF FUNCTION IS SELECTED
C
      IF(NPOL(NCF,NF).EQ.1)RDG=RDG
      IF(NPOL(NCF,NF).EQ.2)RDG=ALOG(RDG)
```

```
      IF(NPOL(NCF,NF).EQ.3)RDG=EXP(RDG)
      IF(NPOL(NCF,NF).EQ.4)RDG=1./RDG
C
C     THE TYPE OF POLYNOMIAL IS SELECTED
C     AND THE POLYNOMIAL VALUES ARE CONSTRUCTED.
C
      READNG(N)=RDG
      N=N+1
      NDEG=NPOL(NCP,NF)-1
      IF(NDEG.LT.1) GO TO 15
      DO 10 I=1,NDEG
      READNG(N)=RDG*READNG(N-1)
      N=N+1
   10 CONTINUE
C
C     CROSS TERMS ARE CONSTRUCTED
C
15    IF(NPOL(NCC,NF).EQ.0) GO TO 200
      RDY=ROLD
      NCTERM=NPOL(NCC,NF)
      DO 20 I=1,NCTERM
      RDY=ROLD*RDY
   20 CONTINUE
      IF(RDY.NE.0) RINV=RDG/ROLD
      IF(RDY.EQ.0) RINV=0.
      DO 30 I=1,NCTERM
      RDY=RDY*RINV
      READNG(N)=RDY
      N=N+1
   30 CONTINUE
C
  200 CONTINUE
  210 CONTINUE
      RETURN
      END
C
      SUBROUTINE POLTYP(POLARY,JROW,MROW,NPOL,IROW,NFT,NC,IOFSET,IDEV)
C
C THIS SUBROUTINE CONSTRUCTS AN ARRAY FOR PRINTING
C
      CHARACTER*2 RDGTYP(2),INUM(9),IDEG(12)
      CHARACTER*4 FUNTYP(4),BLANKS,CONSTA(2),POLARY(JROW,5),ROLD(4)
      DIMENSION NPOL(IROW,NFT)
      DATA  RDGTYP/' M',' P'/,CONSTA/'CONS','TANT'/
      DATA FUNTYP/'(LIN','(LOG','(EXP','(INV'/
      DATA INUM/'1)','2)','3)','4)','5)','6)','7)','8)','9)'/
      DATA IDEG/'1 ','2 ','3 ','4 ','5 ','6 ','7 ','8 ',
     *'9 ','10','11','12'/
      DATA BLANKS/'    '/
C
C     BLANK OUT POLARY ARRAY
C
      DO 71 J=1,5
```

```
      DO 71 I=1,JROW
71    POLARY(I,J)=BLANKS
C
      NROW=1
      IF(IOFSET.NE.1) GO TO 70
      POLARY(NROW,1)=CONSTA(1)
      POLARY(NROW,2)=CONSTA(2)
      NROW=NROW+1
C
70    DO 200 NF=1,NFT
      DO 300 NTYP=1,NC
      NCC=NTYP*3
      NCP=NCC-1
      NCF=NCP-1
      ROLD(1)=POLARY(NROW-1,1)
      ROLD(2)=POLARY(NROW-1,2)
      NDEG=NPOL(NCP,NF)
      IF(NDEG.EQ.0) GO TO 300
      DO 400 I=1,NDEG
      POLARY(NROW,1)=FUNTYP(NPOL(NCF,NF))
      POLARY(NROW,2)=RDGTYP(NTYP)//INUM(NF)
      POLARY(NROW,3)=IDEG(I)
      NROW=NROW+1
400   CONTINUE
C
C    CREATE CROSS TERMS
C
      NCTERM=NPOL(NCC,NF)
      IF(NCTERM.EQ.0) GO TO 300
      IF(NF.EQ.1.AND.NTYP.EQ.1) GO TO 99
      J=NCTERM
      DO 500 I=1,NCTERM
      POLARY(NROW,1)=POLARY(NROW-1,1)
      POLARY(NROW,2)=POLARY(NROW-1,2)
      POLARY(NROW,3)=ROLD(1)
      POLARY(NROW,4)=ROLD(2)
      POLARY(NROW,5)=IDEG(J)
      J=J-1
      NROW=NROW+1
500   CONTINUE
300   CONTINUE
200   CONTINUE
C
C  PRINT RESULTS
C
      WRITE(IDEV,21)((POLARY(I,J),J=1,5),I=1,MROW)
21    FORMAT(' POLARY='/(1X,5A4))
      GO TO 1000
C
C  PRINT ERROR MESSAGES
C
99    WRITE(IDEV,31)
31    FORMAT(' ERROR: CANNOT HAVE A CROSS TERM ON 1ST ITERATION')
```

```
C
1000  RETURN
      END
      SUBROUTINE ALSQS(A,Y,B,R2,NN,MM,NA,NB)
C
C     ALSQS IS A FORTRAN IV SUBROUTINE TO SOLVE THE LINEAR LEAST
C     SQUARES PROBLEM  NORM(AB - Y) = MIN.   CALLING SEQUENCE IS
C            CALL ALSQS(A,Y,B,R2,N,M,NA)
C     WHERE
C      A   IS AN ARRAY CONTAINING THE LEAST SQUARES MATRIX.
C          UPON RETURN THE (M+1)-TH COLUMN CONTAINS THE
C          APPROXIMATING VECTOR AB.
C      Y   IS THE VECTOR TO BE FIT
C      B   CONTAINS UPON RETURN THE COEFFICIENTS OF THE FIT.
C      R2  CONTAINS UPON RETURN THE RESIDUAL SUM OF SQUARES.
C      N   IS THE NUMBER OF ROWS IN THE LEAST SQUARES MATRIX.
C      M   IS THE NUMBER OF COLUMNS IN THE LEAST SQUARES
C          MATRIX.
C      NA  IS THE FIRST DIMENSION OF THE ARRAY A.
C
C     CALL ALSQS(READNG,PRO,COE,RSOS,NPT,IRDPR,NPTT1,IRDPR1)
C     CALL ALSQS(     A,  Y,  B,  R2, NN,   MM,   NA,    NB)
C
      DIMENSION A(NA,NB),Y(NN),B(MM)
      N = NN
      N1 = N+1
      M = MM
      M1 = M+1
      MM1 = M-1
C
C     REDUCE THE LEAST SQUARES MATRIX TO UPPER TRIANGULAR FORM
C
      DO 60 L=1,M
       SS = 0.
       DO 10 I=L,N
   10     SS = SS + A(I,L)**2
       S2 = SS
       S =SQRT(S2)
       IF (A(L,L).LT.0.) S=-S
       D = S2 + S*A(L,L)
       A(L,L) = A(L,L) + S
       IF (L.EQ.M) GO TO 50
       L1 = L+1
       DO 30 J=L1,M
          PP = 0.
          DO 20 I=L,N
   20        PP = PP + A(I,L)*A(I,J)
   30     A(N1,J) = PP/D
       DO 40 J=L1,M
          DO 40 I=L,N
   40        A(I,J) = A(I,J) - A(I,L)*A(N1,J)
   50  A(N1,L) = -S
   60  CONTINUE
```

```
C
C     REDUCE THE VECTOR Y
C
      DO 80 I=1,N
   80 A(I,M1) = Y(I)
      DO 100 L=1,M
       PP = 0.
       DO 90 I=L,N
   90     PP = PP + A(I,L)*A(I,M1)
       D = PP/(-A(L,L)*A(N1,L))
       DO 100 I=L,N
  100     A(I,M1) = A(I,M1) - D*A(I,L)
C
C     CALCULATE THE COEFFICIENT VECTOR B
C
      B(M) = A(M,M1)/A(N1,M)
      IF (M.EQ.1) GO TO 130
      DO 120 LL=1,MM1
       L = M-LL
       L1 = L+1
       PP = A(L,M1)
       DO 110 I=L1,M
  110     PP = PP - A(L,I)*B(I)
  120 B(L) = PP/A(N1,L)
C
C     CALCULATE R2
C
  130 SS = 0.
      MP1 = M+1
      DO 140 I=MP1,N
       SS = SS + A(I,M1)**2
  140 A(I,M1) = 0.
      R2 = SS
C
C     PERFORM THE BACK CALCULATIONS
C
      DO 170 LL=1,M
       L = M-LL+1
       PP = 0.
       DO 150 I=L,N
  150     PP = PP + A(I,L)*A(I,M1)
       D = PP/(-A(L,L)*A(N1,L))
       DO 160 I=L,N
  160     A(I,M1) = A(I,M1) - D*A(I,L)
  170 CONTINUE
      RETURN
      END
C
      SUBROUTINE READ(VOLTS,MFEC,NRDGS,NEXT,NCH)
C     VERSION 7 JULY 1987
C     PROGRAM TO WRITE CONTROL INFO. TO AND READ DATA FROM THE COMP9
C
      INTEGER *2 CMD,OUTC,OUT8(8),OUT16(16)
```

```
      CHARACTER DOUT*15,DIN*64
      DIMENSION IN(8),LL(8),VOLTS(NCH)
      DATA CMD/0/
      DATA OUT8/6,4,6,4 6,4,6,5/
      DATA OUT16/6,4,6,4,6,4,6,4,6,4,6.4,6,4,6,5/
      VF1=5.0/4096.
      VFA=VF1/FLOAT(NRDGS)
      GO TO (100,100,100,100,200,300,400,500,999),NEXT
   30 FORMAT(I1,I1)
   40 FORMAT(16I4)
   50 FORMAT(1X,6F6.3)
   60 FORMAT(1X,I2,6F6.3)
   90 FORMAT(1X)
  100 CONTINUE
C     CALIBRATION LOOP
      OUTC=NEXT-1
      DO 110 I=1,NCH
  110 VOLTS(I)=0.
      DO 150 JJ=1,NRDGS
      WRITE(DOUT,30)CMD,OUTC
      CALL SEND(MFEC,DOUT,ISTAT)
      CALL ENTER(DIN,LEN,MFEC,ISTAT)
      READ(DIN,40)(LL(I),IN(I),I=1,NCH)
      DO 145 I=1,NCH
      VOLTS(I)=VFA*FLOAT(IN(I))+VOLTS(I)
  145 CONTINUE
  150 CONTINUE
      WRITE(0,60)NEXT,(VOLTS(I),I=1,NCH)
      IF(NEXT.EQ.4)WRITE(0,90)
      GO TO 999
  200 CONTINUE
C     SINGLE READING LOOP
      OUTC=4
      WRITE(DOUT,30)CMD,OUTC
      CALL SEND(MFEC,DOUT,ISTAT)
      CALL ENTER(DIN,LEN,MFEC,ISTAT)
      READ(DIN,40)(LL(I),IN(I),I=1,NCH)
      DO 245 I=1,NCH
      VOLTS(I)=VF1*FLOAT(IN(I))
  245 CONTINUE
C     WRITE(0,50)(VOLTS(I),I=1,NCH)
      GO TO 999
  300 CONTINUE
C     AVERAGE NRDGS LOOP
      OUTC=4
      DO 310 I=1,NCH
  310 VOLTS(I)=0.
      DO 350 JJ=1,NRDGS
      WRITE(DOUT,30)CMD,OUTC
      CALL SEND(MFEC,DOUT,ISTAT)
      CALL ENTER(DIN,LEN,MFEC,ISTAT)
      READ(DIN,40)(LL(I),IN(I),I=1,NCH)
      DO 345 I=1,NCH
```

```
      VOLTS(I)=VFA*FLOAT(IN(I))+VOLTS(I)
  345 CONTINUE
  350 CONTINUE
C     WRITE(0,50)(VOLTS(I),I=1,NCH)
      GO TO 999
  400 CONTINUE
C     MULTIPLEX AND READ 8 COILS LOOP
      DO 430,II=1,8
      WRITE(DOUT,30)CMD,OUT8(II)
      CALL SEND(MFEC,DOUT,ISTAT)
      CALL ENTER(DIN,LEN,MFEC,ISTAT)
      READ(DIN,40)(LL(I),IN(I),I=1,NCH)
      DO 420 I=1,NCH
      VOLTS(I)=VF1*FLOAT(IN(I))
  420 CONTINUE
      WRITE(0,60)II,(VOLTS(I),I=1,NCH)
  430 CONTINUE
      GO TO 999
  500 CONTINUE
C     MULTIPLEX AND READ 16 COILS LOOP
      DO 530,II=1,16
      WRITE(DOUT,30)CMD,OUT16(II)
      CALL SEND(MFEC,DOUT,ISTAT)
      CALL ENTER(DIN,LEN,MFEC,ISTAT)
      READ(DIN,40)(LL(I),IN(I),I=1,NCH)
      DO 520 I=1,NCH
      VOLTS(I)=VF1*FLOAT(IN(I))
  520 CONTINUE
      WRITE(0,60)II,(VOLTS(I),I=1,NCH)
  530 CONTINUE
  999 RETURN
      END
      SUBROUTINE CALMIC(RDGC,MFEC,NRDGS,NCHS)
C
C     PROGRAM TO DRIVE AUTOMATIC CALIBRATOR MODULE
C     READINGS ARE TAKEN THROUGH MICROCOMPUTER
C
      DIMENSION RDGC(NCHS,4),VOLTS(6)
      DO 100 NEXT=1,4
      CALL READ(VOLTS,MFEC,NRDGS,NEXT,NCHS)
      DO 50 NCH=1,NCHS
      RDGC(NCH,NEXT)=VOLTS(NCH)
   50 CONTINUE
  100 CONTINUE
      RETURN
      END
C
      SUBROUTINE RESET(RDGC,RDGO,COFSET,GAIN,STDV,NCHS)
C
C     SUBROUTINE DOES NCHS LEAST SQUARES FITS OF RDGO=COFSET+GAIN*RDGC
C     LEAST SQUARES SUBROUTINE IS STANDARD FOR Y(J)=COFSET+GAIN*X(J)
C
      DIMENSION RDGC(NCHS,4),RDGO(NCHS,4),COFSET(NCHS),GAIN(NCHS)
```

```
      DIMENSION STDV(NCHS)
      DATA NRD/4/
      DO 100 NCH=1,NCHS
      SUMX=0.0
      SUMY=0.0
      SUMXY=0.0
      SUMXX=0.0
      SUMYY=0.0
C
C     PERFORM LEAST SQUARES FIT OVER NRD READINGS
C
      DO 50 NR=1,NRD
      SUMX=SUMX+RDGC(NCH,NR)
      SUMY=SUMY+RDGO(NCH,NR)
      SUMXY=SUMXY+RDGC(NCH,NR)*RDGO(NCH,NR)
      SUMXX=SUMXX+RDGC(NCH,NR)*RDGC(NCH,NR)
      SUMYY=SUMYY+RDGO(NCH,NR)*RDGO(NCH,NR)
   50 CONTINUE
      XBAR=SUMX/NRD
      YBAR=SUMY/NRD
      GAIN(NCH)=(SUMXY-SUMX*YBAR)/(SUMXX-SUMX*XBAR)
      COFSET(NCH)=YBAR-GAIN(NCH)*XBAR
      STDV(NCH)=SQRT(ABS((SUMYY-SUMY*YBAR-GAIN(NCH)*(SUMXY-SUMX*YBAR))/
     *FLOAT(NRD-2)))
  100 CONTINUE
      WRITE(0,210)(I,I,I=1,3)
      WRITE(0,220)(COFSET(NCH),NCH=1,NCHS)
      WRITE(0,230)(GAIN(NCH),NCH=1,NCHS)
      WRITE(0,240)(STDV(NCH),NCH=1,NCHS)
  210 FORMAT(6X,3('     MAG(',I1,')     PHA(',I1,')'))
  220 FORMAT(' OFSET',6(F11.5))
  230 FORMAT(' GAIN ',6(F11.5))
  240 FORMAT(' STDV ',6(F11.5))
      RETURN
      END
C
      SUBROUTINE CORRDG(VOLTS,COFSET,GAIN,NCHS)
      DIMENSION VOLTS(NCHS),COFSET(NCHS),GAIN(NCHS)
      DO 100 NCH=1,NCHS
      VOLTS(NCH)=COFSET(NCH)+GAIN(NCH)*VOLTS(NCH)
  100 CONTINUE
      RETURN
      END
C
```

APPENDIX E

THE PLTRDG PROGRAM

The program, PLTRDG, is used to make scans of the tubing using the scanners. The program will plot either raw readings or calculated readings as a function of a motion as the x or y axes are moved. The z axis is not normally used. The start of the scan and the scan increment are requested for both the x and y axes. The program will also plot the raw or calculated readings with no scanner motion to show the effect of instrument noise on the readings. The coefficients will be read from a data file created by BIGFIT and named by the character variable COEDAT. If no data file is present, the program will still run and raw reading scans can be made. The instrument can also be recalibrated, and then subsequent readings will be corrected for any change in the calibration that has occurred since the BIGRDG calibration readings were made. A listing of PLTRDG follows:

```
 PROGRAM PLTRDG
C     October 26, 1987
C
C     PROGRAM WILL PLOT RAW READINGS OR CALCULATED PROPERTIES
C     FROM READINGS MADE BY THE EDDY CURRENT INSTRUMENT ON THE
C     LQ-800.  THE CALCULATE PROPERTIES SECTION MUST BE SELECTED
C     BEFORE THE RECALIBRATE SECTION.  THE COEFFICIENT DATA WILL BE
C     READ FROM THE FILE COEDAT.
C
      INTEGER*4 NSER,NOLD(3)
      CHARACTER PRONAM(7)*4,COEDAT*8,FNAME*13,AXC(3)*2,GC*2,MAPDAT*6
      CHARACTER MOTON*8,MOTOF*8,HOME*5,CHEK*6,XSET*18
      CHARACTER*1 FF,CR,ESC,LPO,SC
      DIMENSION JPOL(6,3,6),JRDPR(6),JOFSET(6)
      DIMENSION RDG1(16),NPOL(6,3),TMAGM(3),PHASE(3),TMDFT(3),PHDFT(3)
      DIMENSION COEF(16,6),PROP(6)
      DIMENSION XLIM(2),YRLIM(2),YPLIM(2),XARG(2)
      DIMENSION VOLSTD(6),VOLTS(6),Y1(2),Y2(2),YVAL(2,12)
      DIMENSION OFSET(6),GAIN(6),STDV(6),RDGC(5,4),RDGO(6,4)
      DIMENSION DELTA(3),XNEW(3),XMAX(3)
      DIMENSION NPR(10),NDACH(6),ROFSET(6),RGAIN(6),POFSET(6),PGAIN(6)
      DATA LOU/8/,LITEK/0/,LOTEK/0/,NCHS/6/,NRDGS/1/,NRDGC/32/,NFT/[illegible]/
      DATA NCAL/4/,NTT/9/,NTT1/8/,NS/2/,NPR/10*700/,NAXIS/3/,NOLD/3*0/
      DATA NCUR/6/,XFAC/0.00555/,NTIM/0/,NAUTO/0/
      DATA INTER/10/,ICOEF/1/,NPROPM/1/,NOLD/3*0/,IBM/21/,LEVEL/0/
      DATA MFEC/5/,MBUS/6/,IND/37/,LOD/38/,COEDAT/'NRCCOE '/
      DATA MAPDAT/'RAWDAT '/,XMAX/33.15,12.5,0.5/
      DATA TMDFT/3*0./,PHDFT/3*0./
      DATA MOTON/'XD YD ZD'/,MOTOF/'XE YE ZE'/,HOME/'<CAH>'/
      DATA CHEK/';<CFA>'/,SC/';'/,XSET/'XA20000XB640XH4000'/
C     **VALUES OF OFFSET AND GAIN FOR READINGS FOR CIRC COILS**
       DATA ROFSET/-1.38,-1.57,+1.53,+0.50,+2.50,+2.50/,RGAIN/6*200./
C     VALUES OF GAIN AND OFFSET FOR PROPERTIES FOR CIRC COILS
      DATA POFSET/-0.01,0.05,0.065,3*0./,PGAIN/1.8E4,1.8E4,1.8E4,3*1./
C     VALUES OF GAIN AND OFFSET FOR CALIBRATIONS ARE SET
      DATA DELTA/3*0.0/,XNEW/3*0.0/,OFSET/6*0.0/,GAIN/6*1.0/
      LNF=2*NFT
C     TIME AND DATE ARE PRINTED
      CALL GETTIM(IHR,IMN,ISE,IFR)
```

```
      CALL GETDAT(IYR,IMO.IDA)
      IYR=IYR-1900
      WRITE(LOU,2,ERR=990)IHR,IMN,ISE,IMO,IDA,IYR
    2 FORMAT(' PLTRDG    TIME ',I2,':',I2,':',I2,
     *'  DATE ',I2,'/',I2,'/',I2)
      CALL INITIALI(IBM,LEVEL)
C     MULTI FREQUENCY EDDY CURRENT INSTRUMENT ADDR=5 ON GPIB BUSS
C     MODULYNX AXIS POSITION CONTROLLER ADDR=6 ON GPIB BUSS
      CALL SEND(MBUS,MOTON,ISTAT)
      CALL SEND(MBUS,HOME,ISTAT)
      CALL SEND(MBUS,MOTOF,ISTAT)
C     CALL SEND(MBUS,XSET,ISTAT)
      FNAME=COEDAT//'.DAT'
      OPEN(LOD,FILE=FNAME,STATUS ='OLD',ERR = [illegible]
C     READ COEFFICIENT DATA WRITTEN BY BIGFIT [illegible] TRANSFER IF NO FILE)
      NPRO=0
      DO 20 NC=1,NCAL
      READ(LOD,*,ERR=992)(RDG0(JJ,NC),JJ=1,LNF)
      WRITE(0,*)(RDG0(JJ,NC),JJ=1,LNF)
C     WRITE(LOU,*)(RDG0(JJ,NC),JJ=1,LNF)
   20 CONTINUE
      READ(LOD,*)(VOLSTD(NT),NT=1,LNF)
      WRITE(0,*)(VOLSTD(NT),NT=1,LNF)
C     WRITE(LOU,*)'VOLSTD'
C     WRITE(LOU,*)(VOLSTD(NT),NT=1,LNF)
   30 NPRO=NPRO+1
      READ(LOD,*,END=60)IRDPR
      WRITE(0,*)IRDPR
C     WRITE(LOU,*)'IRDPR'
C     WRITE(LOU,*)IRDPR
      READ(LOD,40)PRONAM(NPRO),JRDPR(NPRO),JOFSET(NPRO)
   40 FORMAT(1X,A4,2I4)
      WRITE(0,40)PRONAM(NPRO),JRDPR(NPRO),JOFSET(NPRO)
      READ(LOD,*)(COEF(IR,NPRO),IR=1,IRDPR)
      WRITE(0,*)(COEF(IR,NPRO),IR=1,IRDPR)
      DO 50 NF=1,NFT
      READ(LOD,*)(JPOL(L,NF,NPRO),L=1,6)
      WRITE(0,*)(JPOL(L,NF,NPRO),L=1,6)
   50 CONTINUE
      GO TO 30
   60 CONTINUE
      NPROPM=NPRO-1
C     OPEN DATA FILE FOR PLOT COMMANDS
   75 OPEN(IND,FILE='PLTTBS.DAT',STATUS ='OLD',ERR = 995)
   80 WRITE(LOTEK,90)
      JSET=0
      IXX=0
   90 FORMAT(' WHAT NEXT 1.RAW RDG 2.CAL PROPS 3.RECAL'
     *,' 4.SCAN/RAW 5.SCAN/CAL 6.RST 7.READ CMD FILE 8.STOP? ')
      READ(LITEK,*)NPRINT
      IF(NPRINT.NE.3)NEXT=5
      IF(NPRINT.EQ.4.OR.NPRINT.EQ.5)WRITE(LOTEK,100)
```

```
 100 FORMAT(' TYPE XNEW(1),XNEW(2),DELTA(1),DELTA(2) FOR POSITIONER ')
     IF(NPRINT.EQ.4.OR.NPRINT.EQ.5)READ(0,*)XNEW(1),XNEW(2)
    *,DELTA(1),DELTA(2)
 103 IF(NPRINT.EQ.4.OR.NPRINT.EQ.5)
    * WRITE(LOU,102)XNEW(1),XNEW(2),DELTA(1),DELTA(2)
 102 FORMAT('X0=',F7.3,'   Y0=',F7.3,'   DX=',F7.3,'   DY=',F7.3)
     GO TO (105,200,300,105,200,400,500,600),NPRINT
 105 CONTINUE
C    OPEN FILE FOR OUTPUT STORAGE OF RAW DATA POINTS TO FILE 'RAWDAT'
     FNAME=MAPDAT//'.DAT'
C     OPEN(LOD,FILE=FNAME,STATUS ='NEW',ERR = 994)
     WRITE(LOTEK,110)(NF,NF,NF=1,NFT)
C     WRITE(LOD,110)(NF,NF,NF=1,NFT)
     WRITE(LOTEK,120)
C     WRITE(LOD,120)
 110 FORMAT(' X POSIT Y POSIT  ',3('MAG(',I1,') PHA(',I1,') '))
 120 FORMAT(1X)
 130 IF(NPRINT.EQ.1)GO TO 140
     XNEW(1)=XNEW(1)+DELTA(1)
     XNEW(2)=XNEW(2)+DELTA(2)
     IF(XNEW(1).LT.0.0.OR.XNEW(1).GT.XMAX(1))GO TO 163
     IF(XNEW(2).LT.0.0.OR.XNEW(2).GT.XMAX(2))GO TO 163
     CALL SEND(MBUS,MOTON,ISTAT)
     CALL POSIT(XNEW,NOLD,NAXIS,MBUS)
     IF(MOD(IXX,180).EQ.0)CALL SEND(MBUS,CHEK,ISTAT)
     IF(MOD(IXX,180).EQ.0)CALL SEND(MBUS,SC,ISTAT)
 140 CALL READ(VOLTS,MFEC,NRDGS,NEXT,NCHS)
 142 FORMAT(6(F11.3))
     CALL CORRDG(VOLTS,OFSET,GAIN,NCHS)
C     WRITE(LOU,142)(VOLTS(NF),NF=1,LNF)
     IF(MOD(IXX,INTER).EQ.0)
    *WRITE(LOTEK,150)XNEW(1),XNEW(2),(VOLTS(NF),NF=1,LNF)
 150 FORMAT(2(F8.3),1X,6(F7.3),I6)
     DO 160 NF=1,LNF
     NDACH(NF)=RGAIN(NF)*(ROFSET(NF)+VOLTS(NF))
 160 CONTINUE
     CALL GRAPF(NDACH,IXX,NCHS,LOU)
     CALL GETKEY(IRR)
     IF(IRR.EQ.83)GO TO 163
     GO TO 130
 163 WRITE(LOU,185)ESC,LP0
 185 FORMAT(A1,A1)
     WRITE(LOU,*)' '
     NTIM=0
     IXX=0
     IF(NPRINT.EQ.4)WRITE(LOU,180)(VOLTS(NF),NF=1,LNF)
     DO 165 NF=1,LNF
     VOLTS(NF)=VOLTS(NF)+ROFSET(NF)
 165 CONTINUE
     IF(NPRINT.EQ.4)WRITE(LOU,180)(VOLTS(NF),NF=1,LNF)
     IF(NPRINT.EQ.4)WRITE(LOU,180)(ROFSET(NF),NF=1,LNF)
     IF(NPRINT.EQ.4)WRITE(LOU,190)(RGAIN(NF),NF=1,LNF)
```

```
      IF(NPRINT.EQ.4)WRITE(LOU,192)(NDACH(NF),NF=1,LNF)
      IF(XNEW(1).GT.XMAX(1).OR.XNEW(2).GT.XMAX(2))WRITE(LOU,195)FF
  170 FORMAT(6(1PE9.2,'KHZ',1X))
  180 FORMAT(6(F7.3))
  190 FORMAT(6(F7.0))
  192 FORMAT(6(I7))
  195 FORMAT(A1)
      IF(NAUTO.NE.1)GO TO 80
      GO TO 500
C
C     PROPERTY CALCULATION AND DISPLAY SECTION
C
  200 WRITE(LOTEK,217)(PRONAM(NPRO),NPRO=1,NPROPM)
  217 FORMAT(1X,' TUB LOC  T S LOC',6(6X,A4,1X))
      WRITE(LOTEK,120)
  220 IF(NPRINT.EQ.2)GO TO 221
      XNEW(1)=XNEW(1)+DELTA(1)
      XNEW(2)=XNEW(2)+DELTA(2)
      IF(XNEW(1).LT.0.0.OR.XNEW(1).GT.XMAX(1))GO TO 275
      IF(XNEW(2).LT.0.0.OR.XNEW(2).GT.XMAX(2))GO TO 275
      CALL SEND(MBUS,MOTON,ISTAT)
      CALL POSIT(XNEW,NOLD,NAXIS,MBUS)
      IF(MOD(IXX,180).EQ.0)CALL SEND(MBUS,CHEK,ISTAT)
      IF(MOD(IXX,180).EQ.0)CALL SEND(MBUS,SC,ISTAT)
  221 CALL READ(VOLTS,MFEC,NRDGS,NEXT,NCHS)
      CALL CORRDG(VOLTS,OFSET,GAIN,NCHS)
      DO 260 NPRO=1,NPROPM
      IRDPR1=JRDPR(NPRO)+1
      IOFSET=JOFSET(NPRO)
      DO 230 NF=1,NFT
      PHASE(NF)=VOLTS(2*NF)
      TMAGM(NF)=VOLTS(2*NF-1)
      DO 225 I=1,6
      NPOL(I,NF)=JPOL(I,NF,NPRO)
  225 CONTINUE
  230 CONTINUE
      CALL RDGEXP(RDG1,TMAGM,PHASE,NPOL,IOFSET,TMDFT,PHDFT,IRDPRM,NFT)
      PROP(NPRO)=0.
      IRDP=JRDPR(NPRO)
      DO 240 IR=1,IRDP
      PROP(NPRO)=PROP(NPRO)+RDG1(IR)*COEF(IR,NPRO)
  240 CONTINUE
C      WRITE(0,245)(RDG1(IR),IR=1,IRDP)
C 245 FORMAT(10F8.4,/,5F8.4)
      NDACH(NPRO)=PGAIN(NPRO)*(POFSET(NPRO)+PROP(NPRO))
  260 CONTINUE
C     IF(JSET.EQ.0)POFSET(3)=0.140-PROP(3)
      JSET=1
C
C     SECTION TO SMOOTH THE DEFECT INDICATION BY AVERAGING SUCCESSIVE
C     DEFECT CALCULATIONS . CHANNEL NS IS SMOOTHED
C
      DO 265 NT=1,NTT1
      NPR(NT)=NPR(NT+1)
```

```
  265 CONTINUE
      NPR(NTT)=NDACH(NS)
      NDACH(NS)=0
      DO 167 NT=1,NTT
      NDACH(NS)=NDACH(NS)+NPR(NT)
  167 CONTINUE
      NDACH(NS)=NDACH(NS)/NTT
C
C     END OF SECTION
C
      K=MOD(IXX,INTER)
      IF(K.EQ.0)WRITE(LOTEK,270)XNEW(1),XNEW(2),(PROP(NPRO)
     *,NPRO=1,NPROPM)
  270 FORMAT(2(F8.3),1X,6(F11.4))
      CALL GRAPF(NDACH,IXX,NPROPM,LOU)
      CALL GETKEY(IRR)
      IF(IRR.EQ.83)GO TO 275
      GO TO 220
  275 WRITE(LOU,185)ESC,LPO
      WRITE(LOU,*)' '
      WRITE(LOU,290)(PRONAM(NPRO),NPRO=1,NPROPM)
      WRITE(LOU,293)(PROP(NPRO),NPRO=1,NPROPM)
      WRITE(LOU,295)(POFSET(NPRO),NPRO=1,NPROPM)
      WRITE(LOU,297)(PGAIN(NPRO),NPRO=1,NPROPM)
      WRITE(LOU,305)(NDACH(NPRO),NPRO=1,NPROPM)
      IF(XNEW(1).GT.XMAX(1).OR.XNEW(2).GT.XMAX(2))WRITE(LOU,195)FF
  290 FORMAT('PROPERTY',7(6X,A4))
  293 FORMAT(8X,7(F10.3))
  295 FORMAT('OFFSET  ',7(F10.3))
  297 FORMAT('GAIN    ',7(1PE10.2))
  305 FORMAT('PLOT INT',7(I10))
      NTIM=0
      IXX=0
      IF(NAUTO.NE.1)GO TO 80
      GO TO 500
  300 CONTINUE
      CALL CALMIC(RDGC,MFEC,NRDGC,NCHS)
      CALL RESET(RDGC,RDG0,OFSET,GAIN,STDV,NCHS)
      IF(NAUTO.NE.1)GO TO 80
      GO TO 500
  400 WRITE(LOTEK,401)
      CALL READ(VOLTS,MFEC,NRDGS,NEXT,NCHS)
  401 FORMAT(' TYPE XNEW(1),XNEW(2),FOR STANDARD READING ')
      READ(0,*)XNEW(1),XNEW(2)
      CALL POSIT(XNEW,NOLD,NAXIS,MBUS)
      CALL SEND(MBUS,CHEK,ISTAT)
  403 CALL SEND(MBUS,SC,ISTAT)
      IF(ISTAT.NE.0)GO TO 403
      DO 405 NF=1,LNF
      VOLTS(NF)=0.
  405 CONTINUE
      CALL READ(VOLTS,MFEC,NRDGS,NEXT,NCHS)
      D^ 450 NOF=1,NCHS
```

```
      OFSET(NOF)=VOLSTD(NOF)-GAIN(NOF)*VOLTS(NOF)
  450 CONTINUE
      DO 460 NF=1,LNF
      ROFSET(NF)=(FLOAT(800*NF/(LNF+1)))/RGAIN(NF)-VOLTS(NF)
  460 CONTINUE
      WRITE(0,180)(VOLTS(NF),NF=1,LNF)
      WRITE(0,180)(OFSET(NF),NF=1,LNF)
      WRITE(0,180)(ROFSET(NF),NF=1.LNF)
      IF(NAUTO.NE.1)GO TO 80
      GO TO 500
  500 CONTINUE
C     READ COMMANDS AND DATA FROM AN INPUT DATA FILE, CONTINUE UNTIL
C      PROGRAM END
C      1.RAW RDG 2.CAL PROPS 3.RECAL 4.SCAN/RAW 5.SCAN/CAL
C      *6.RST 7.READ CMD FILE 8.STOP? ')
      READ(IND,*,END=80)NPRINT
      IF(NPRINT.EQ.4.OR.NPRINT.EQ.5)READ(IND,*)XNEW(1),XNEW(2),DELTA(1)
     *,DELTA(2),XMAX(1),XMAX(2)
      IF(NAUTO.EQ.0.AND.NPRINT.EQ.5)
     *WRITE(LOU,290)(PRONAM(NPRO),NPRO=1,NPROPM)
      IF(NAUTO.EQ.0.AND.NPRINT.EQ.5)
     *WRITE(LOU,295)(POFSET(NPRO),NPRO=1,NPROPM)
      IF(NAUTO.EQ.0.AND.NPRINT.EQ.5)
     *WRITE(LOU,297)(PGAIN(NPRO),NPRO=1,NPROPM)
      NAUTO=1
      IF(NPRINT.NE.?)NEXT=5
      GO TO 103
  600 CONTINUE
      GOTO 1000
  990 WRITE(LOTEK,*)' ERROR IN OPENING PRINTER FILE-CHECK FOR OFF-LINE'
      GO TO 1000
  991 WRITE(LOTEK,*)'ERROR IN OPENING MOTOR DATA BUSS'
      GO TO 1000
  992 WRITE(LOTEK,*)'ERROR IN OPENING INPUT COEF DATA FILE'
      GO TO 75
  993 WRITE(LOTEK,*)'ERROR IN OPENING INSTRUMENT DATA BUSS'
      GO TO 1000
  994 WRITE(LOTEK,*)'ERROR IN OPENING MOTOR POSITIONER FILE'
      GO TO 1000
  995 WRITE(LOTEK,*)'ERROR IN OPENING INPUT COMMAND FILE'
      GO TO 80
 1000 CONTINUE
      CALL SEND(MBUS,HOME,ISTAT)
 1005 CALL SEND(MBUS,SC,ISTAT)
      IF(ISTAT.NE.0)GO TO 1005
      CALL SEND(MBUS,MOTOF,ISTAT)
      STOP
      END
      SUBROUTINE RDGEXP(READNG,TMAG,PHASE,NPOL,IOFSET,TMDFT,PHDFT
     1,IRDPRM,NFT)
C     REAL*8 READNG,RDG
      DIMENSION READNG(IRDPRM),NPOL(6,NFT),TMDFT(NFT),PHDFT(NFT)
      DIMENSION TMAG(NFT),PHASE(NFT)
```

```
C
C     NPOL CONTAINS A NUMBER FOR THE FUNCTION TYPE, THE POLYNOMIAL
C     DEGREE, AND THE NUMBER OF CROSS TERMS
C     FOR THE MAGNITUDE AND PHASE AT EACH FREQUENCY,STORED AS NPOL
C     (NF;1-MAG FUN, 2-MAG POL,3-MAG #CROSS TERMS,4-PH FUN,5-PH POL,
C     6-PH # CROSS TERMS).  IF IOFSET=0, NO OFF-SET
C     WILL BE INCLUDED,=1 OFF-SET IS INCLUDED.THE VALUES OF TMDFT(NF)&
C     PHDFT(NF) GIVE THE AMOUNT OF DRIFT IN THE MAGNITUDE AND PHASE.IF
C      NPOL(NCP,NF) =0,THAT PARTICULAR MAGNITUDE AND PHASE FOR THAT
C     FREQUENCY WILL BE SKIPPED.
C
      READNG(1)=1
      N=1
      IF(IOFSET.EQ.1)N=2
      DO 210 NF=1,NFT
      DO 200 NC=1,2
      NCC=NC*3
      NCP=NCC-1
      NCF=NCP-1
      ROLD=RDG
      IF(NPOL(NCP,NF).EQ.0) GO TO 200
      IF(NC.EQ.1) RDG=TMAG(NF)+TMDFT(NF)
      IF(NC.EQ.2) RDG=PHASE(NF)+PHDFT(NF)
C
C     THE TYPE OF FUNCTION IS SELECTED
C
      IF(NPOL(NCF,NF).EQ.1)RDG=RDG
      IF(NPOL(NCF,NF).EQ.2)RDG=ALOG(RDG)
      IF(NPOL(NCF,NF).EQ.3)RDG=EXP(RDG)
      IF(NPOL(NCF,NF).EQ.4)RDG=1./RDG
C
C     THE TYPE OF POLYNOMIAL IS SELECTED
C     AND THE POLYNOMIAL VALUES ARE CONSTRUCTED.
C
      READNG(N)=RDG
      N=N+1
      NDEG=NPOL(NCP,NF)-1
      IF(NDEG.LT.1) GO TO 15
      DO 10 I=1,NDEG
      READNG(N)=RDG*READNG(N-1)
      N=N+1
   10 CONTINUE
C
C     CROSS TERMS ARE CONSTRUCTED
C
15    IF(NPOL(NCC,NF).EQ.0) GO TO 200
      RDY=ROLD
      NCTERM=NPOL(NCC,NF)
      DO 20 I=1,NCTERM
      RDY=ROLD*RDY
   20 CONTINUE
      IF(RDY.NE.0) RINV=RDG/ROLD
      IF(RDY.EQ.0) RINV=0.
```

```
      DO 30 I=1,NCTERM
      RDY=RDY*RINV
      READNG(N)=RDY
      N=N+1
   30 CONTINUE
C
  200 CONTINUE
  210 CONTINUE
      RETURN
      END
C
      SUBROUTINE RESET(RDGC,RDGO,COFSET,GAIN,STDV,NCHS)
C
C     SUBROUTINE DOES NCHS LEAST SQUARES FITS OF RDGO=COFSET+GAIN*RDGC
C     LEAST SQUARES SUBROUTINE IS STANDARD FOR Y(J)=COFSET+GAIN*X(J)
C
      DIMENSION RDGC(NCHS,4),RDGO(NCHS,4),COFSET(NCHS),GAIN(NCHS)
      DIMENSION STDV(NCHS)
      DATA NRD/4/
      DO 100 NCH=1,NCHS
      SUMX=0.0
      SUMY=0.0
      SUMXY=0.0
      SUMXX=0.0
      SUMYY=0.0
C
C     PERFORM LEAST SQUARES FIT OVER NRD READINGS
C
      DO 50 NR=1,NRD
      SUMX=SUMX+RDGC(NCH,NR)
      SUMY=SUMY+RDGO(NCH,NR)
      SUMXY=SUMXY+RDGC(NCH,NR)*RDGO(NCH,NR)
      SUMXX=SUMXX+RDGC(NCH,NR)*RDGC(NCH,NR)
      SUMYY=SUMYY+RDGO(NCH,NR)*RDGO(NCH,NR)
   50 CONTINUE
      XBAR=SUMX/NRD
      YBAR=SUMY/NRD
      GAIN(NCH)=(SUMXY-SUMX*YBAR)/(SUMXX-SUMX*XBAR)
      COFSET(NCH)=YBAR-GAIN(NCH)*XBAR
      STDV(NCH)=SQRT(ABS((SUMYY-SUMY*YBAR-GAIN(NCH)*(SUMXY-SUMX*YBAR))/
     *FLOAT(NRD-2)))
  100 CONTINUE
      WRITE(0,210)(I,I,I=1,3)
      WRITE(0,220)(COFSET(NCH),NCH=1,NCHS)
      WRITE(0,230)(GAIN(NCH),NCH=1,NCHS)
      WRITE(0,240)(STDV(NCH),NCH=1,NCHS)
  210 FORMAT(6X,3('     MAG(',I1,')      PHA(',I1,')'))
  220 FORMAT(' OFSET',6(F11.5))
  230 FORMAT(' GAIN ',6(F11.5))
  240 FORMAT(' STDV ',6(F11.5))
      RETURN
      END
C
```

```
      SUBROUTINE CORRDG(VOLTS,COFSET,GAIN,NCHS)
      DIMENSION VOLTS(NCHS),COFSET(NCHS),GAIN(NCHS)
      DO 100 NCH=1,NCHS
      VOLTS(NCH)=COFSET(NCH)+GAIN(NCH)*VOLTS(NCH)
  100 CONTINUE
      RETURN
      END
C
      SUBROUTINE GRAPF(IY,IXX,NCUR,LOU)
C     VERSION October 20, 1987
C     SUBROUTINE TO RUN GRAPHICS ON LQ800 USING 180 DPI DENSITY
C     PLOT A GRAPH ON PRINTER, EACH CALL WILL ADVANCE 1/180 OF INCH
C     IXX IS A COUNT OF THE NUMBER OF TIMES THAT SUBROUTINE IS CALLED
C     PRINTING IS DONE EVERY 24TH TIME SUBROUTINE IS CALLED.
C     UP TO 20 CURVES CAN BE PLOTTED, IY(NC) SHOULD BE BETWEEN 1 AND
C     AND 1440.
      DIMENSION IBT(24),IRY(1440,3),IY(20),IGRD1(1440,3)
      CHARACTER*1 ESC,TRD,N1,N2,LPI,LP2,GRAC(4320)
      DATA IBT/128,64,32,16,8,4,2,1,128,64,32,16,8,4,2,1,
     *128,64,32,16,8,4,2,1/
C     SET UP GRID, LINE SPACING, CHAR DATA FOR TRIPLE DENSITY
      IF(IXX.GT.0)GO TO 50
      DO 20 I=1,3
      IGRD1(1,I)=255
      IGRD1(1440,I)=255
      DO 10 J=180,1260,180
      IGRD1(J,I)=170
   10 CONTINUE
   20 CONTINUE
      IX=0
      ESC=CHAR(27)
      TRD=CHAR(39)
      N1=CHAR(160)
      N2=CHAR(5)
      LPI=CHAR(51)
      LP2=CHAR(24)
      WRITE(LOU,40)ESC,LPI,LP2
   40 FORMAT(A1,A1,A1)
   50 CONTINUE
      I=IX/8+1
      IX=IX+1
      DO 60 NC=1,NCUR
      IF(IY(NC).GT.1439)IY(NC)=1439
      IF(IY(NC).LT.1)IY(NC)=1
      IRY(IY(NC),I)=IRY(IY(NC),I)+IBT(IX)
      IRY(IY(NC)+1,I)=IRY(IY(NC)+1,I)+IBT(IX)
   60 CONTINUE
      IF(MOD(IXX,180).NE.0)GO TO 90
      DO 80 J=2,1438,2
      IRY(J,I)=IRY(J,I)+IBT(IX)
   80 CONTINUE
   90 IF(IX.LT.24)GO TO 300
      WRITE(LOU,100)ESC,TRD,N1,N2
```

```
  100 FORMAT(A1,'*',A1,A1,A1,\)
      DO 200 J=1,1440
      DO 150 I=1,3
      K=3*(J-1)+I
      KAR=IRY(J,I)+IGRD1(J,I)
      IF(KAR.EQ.26)KAR=25
      GRAC(K)=CHAR(KAR)
      IRY(J,I)=0
  150 CONTINUE
  110 FORMAT(144A1,\)
  200 CONTINUE
      WRITE(LOU,110)(GRAC(K),K=1,4320)
      IX=0
  300 CONTINUE
      IXX=IXX+1
      RETURN
      END
      SUBROUTINE POSIT(XNEW,NOLD,NAXIS,MBUS)
C
C     DETERMINES THE NEW LOCATION FOR 3 DIFFERENT POSITIONERS AND FROM
C     THE OLD LOCATION,THE NUMBER OF STEPS NEEDED TO REACH THE NEW
C     LOCATION.  THEN SENDS THE NUMBER OVER THE IEEE-488 BUSS TO THE
C     CONTROLLER.  AFTER THE NEW POSITION IS REACHED THE PULSE COUNT IS
C     COMPARED TO THE ENCODER COUNT AND CORRECTED IF DIFFERENT.
C
      INTEGER*4 NOLD(3),NSTEPS(3),NSTP
      DIMENSION XNEW(3)
      CHARACTER*2 AXC(3),GC,DOUT*35,CHEK*6,SC*1
      DATA AXC/'XM','YM','ZM'/,GC/'G '/,CHEK/';<CFA>'/,SC/';'/
      DATA STEPSZ/0.00025/
      DO 20 IAXIS=1,NAXIS
      NSTP=(XNEW(IAXIS)+0.000125)/STEPSZ
      NSTEPS(IAXIS)=NSTP-NOLD(IAXIS)
      NOLD(IAXIS)=NSTP
   20 CONTINUE
       WRITE(DOUT,30)(AXC(IAXIS),NSTEPS(IAXIS),GC,IAXIS=1,NAXIS)
C      WRITE(0,*)DOUT
      CALL SEND(MBUS,DOUT ISTAT)
      IF(ISTAT.NE.0)GO TO 40
C      CALL SEND(MBUS,CHEK,ISTAT)
C      CALL SEND(MBUS,SC,ISTAT)
      GO TO 70
   30 FORMAT(4(A2,I7,A2))
   40 WRITE(0,*)'ERROR,UNABLE TO WRITE POSITION DATA'
      GO TO 70
   50 WRITE(0,*)'ERROR,UNABLE TO WRITE HOME COMMAND'
      GO TO 70
   60 WRITE(0,*)'ERROR,WAIT TOO LONG FOR AXIS READY COMMAND'
   70 RETURN
      END
      SUBROUTINE READ(VOLTS,MFEC,NRDGS,NEXT,NCH)
C     VERSION 7 JULY 1987
C     PROGRAM TO WRITE CONTROL INFO. TO AND READ DATA FROM THE COMP9
```

```
C
      INTEGER *2 CMD,OUTC,OUT8(8),OUT16(16)
      CHARACTER DOUT*15,DIN*64
      DIMENSION IN(8),LL(8),VOLTS(NCH)
      DATA CMD/0/
      DATA OUT8/6,4,6,4,6,4,6,5/
      DATA OUT16/6,4,6,4,6,4,6,4,6,4,6,4,6,4,6,5/
      VF1=5.0/4096.
      VFA=VF1/FLOAT(NRDGS)
      GO TO (100,100,100,100,200,300,400,500,999),NEXT
   30 FORMAT(I1,I1)
   40 FORMAT(16I4)
   50 FORMAT(1X,6F6.3)
   60 FORMAT(1X,I2,6F6.3)
   90 FORMAT(1X)
  100 CONTINUE
C     CALIBRATION LOOP
      OUTC=NEXT-1
      DO 110 I=1,NCH
  110 VOLTS(I)=0.
      DO 150 JJ=1,NRDGS
      WRITE(DOUT,30)CMD,OUTC
      CALL SEND(MFEC,DOUT,ISTAT)
      CALL ENTER(DIN,LEN,MFEC,ISTAT)
      READ(DIN,40)(LL(I),IN(I),I=1,NCH)
      DO 145 I=1,NCH
      VOLTS(I)=VFA*FLOAT(IN(I))+VOLTS(I)
  145 CONTINUE
  150 CONTINUE
      WRITE(0,60)NEXT,(VOLTS(I),I=1,NCH)
      IF(NEXT.EQ.4)WRITE(0,90)
      GO TO 999
  200 CONTINUE
C     SINGLE READING LOOP
      OUTC=4
      WRITE(DOUT,30)CMD,OUTC
      CALL SEND(MFEC,DOUT,ISTAT)
      CALL ENTER(DIN,LEN,MFEC,ISTAT)
      READ(DIN,40)(LL(I),IN(I),I=1,NCH)
      DO 245 I=1,NCH
      VOLTS(I)=VF1*FLOAT(IN(I))
  245 CONTINUE
C      WRITE(0,50)(VOLTS(I),I=1,NCH)
      GO TO 999
  300 CONTINUE
C     AVERAGE NRDGS LOOP
      OUTC=4
      DO 310 I=1,NCH
  310 VOLTS(I)=0.
      DO 350 JJ=1,NRDGS
      WRITE(DOUT,30)CMD,OUTC
      CALL SEND(MFEC,DOUT,ISTAT)
      CALL ENTER(DIN,LEN,MFEC,ISTAT)
```

```
      READ(DIN,40)(LL(I),IN(I),I=1,NCH)
      DO 345 I=1,NCH
      VOLTS(I)=VFA*FLOAT(IN(I))+VOLTS(I)
  345 CONTINUE
  350 CONTINUE
      WRITE(0,50)(VOLTS(I),I=1,NCH)
      GO TO 999
  400 CONTINUE
C     MULTIPLEX AND READ 8 COILS LOOP
      DO 430,II=1,8
      WRITE(DOUT,30)CMD,OUT8(II)
      CALL SEND(MFEC,DOUT,ISTAT)
      CALL ENTER(DIN,LEN,MFEC,ISTAT)
      READ(DIN,40)(LL(I),IN(I),I=1,NCH)
      DO 420 I=1,NCH
      VOLTS(I)=VF1*FLOAT(IN(I))
  420 CONTINUE
      WRITE(0,60)II,(VOLTS(I),I=1,NCH)
  430 CONTINUE
      GO TO 999
  500 CONTINUE
C     MULTIPLEX AND READ 16 COILS LOOP
      DO 530,II=1,16
      WRITE(DOUT,30)CMD,OUT16(II)
      CALL SEND(MFEC,DOUT,ISTAT)
      CALL ENTER(DIN,LEN,MFEC,ISTAT)
      READ(DIN,40)(LL(I),IN(I),I=1,NCH)
      DO 520 I=1,NCH
      VOLTS(I)=VF1*FLOAT(IN(I))
  520 CONTINUE
      WRITE(0,60)II,(VOLTS(I),I=1,NCH)
  530 CONTINUE
  999 RETURN
      END
      SUBROUTINE CALMIC(RDGC,MFEC,NRDGS,NCHS)
C
C     PROGRAM TO DRIVE AUTOMATIC CALIBRATOR MODULE
C     READINGS ARE TAKEN THROUGH MICROCOMPUTER
C
      DIMENSION RDGC(6,4),VOLTS(8)
      DO 100 NEXT=1,4
      CALL READ(VOLTS,MFEC,NRDGS,NEXT,NCHS)
      DO 50 NCH=1,NCHS
      RDGC(NCH,NEXT)=VOLTS(NCH)
   50 CONTINUE
  100 CONTINUE
      RETURN
      END
```

APPENDIX F

THE DIG3 PROGRAM

The program, DIG3, is a utility program that is useful in the set-up and calibration of the three-frequency instrument. It will make single readings, average 10 readings, make calibration readings, run the multiplexer and make both 8-and 16-coil multiplexed readings. The program will stay in each loop until a key is struck, then it will go back to the option menu. A listing of the program DIG3 follows:

```
      PROGRAM DIG3
C     VERSION August 14, 1987
C     PROGRAM TO WRITE CONTROL INFORMATION TO AND READ DATA FORM THE
C     COMP9B.  THE CMD VALUE DETERMINES WHICH PROGRAM WILL BE RUN BY THE
C     MICROCOMPUTER.  THE PROM GPIF MUST BE IN THE 800 LOCATION OF THE
C     COMP9B.  MODIFIED FOR IBM-AT COMPUTER, RM FORTRAN
C
      INTEGER CMD,OTC,OT8(8),OT16(16)
      CHARACTER*1 DOUT*15,DIN*64
      DIMENSION IN(8),LL(8),VOLTS(8)
      DATA MFEC/5/,NCH/6/,CMD/1/,NRDGS/64/
      DATA LANA/3/,MBUS/6/,IBM/21/,LEVEL/0/
      DATA OT8/6,4,6,4,6,4,6,5/,OT16/6,4,6,4,6,4,6,4,6,4,6,4,6,4,6,5/
C     MULTI FREQUENCY EDDY CURRENT INSTRUMENT ADDR=5 ON GPIB BUSS
      CALL INITIALI(IBM,LEVEL)
      VF1=5.0/4096.
      VFA=VF1/FLOAT(NRDGS)
      WRITE(0,*)' HIT A KEY TO EXIT FROM LOOPS'
   20 WRITE(0,*)'1.SING RDGS 2.AVG 10 3.CALIB 4.MUX8 5.MUX16 6.STOP ?'
      READ(5,*)NEXT
      GO TO (100,200,300,400,500,999),NEXT
   30 FORMAT(I1,I1)
   40 FORMAT(16I4)
   50 FORMAT(3X,6F8.3)
   60 FORMAT(1X,I2,6F8.3)
   90 FORMAT(1X)
  100 CONTINUE
C     SINGLE READING LOOP
      OTC=4
      WRITE(DOUT,30)CMD,OTC
      CALL SEND(MFEC,DOUT,ISTAT)
      CALL ENTER(DIN,LEN,MFEC,ISTAT)
      READ(DIN,40)(IN(I),I=1,NCH)
      DO 145 I=1,NCH
      VOLTS(I)=VF1*FLOAT(IN(I))
  145 CONTINUE
      WRITE(0,50)(VOLTS(I),I=1,NCH)
      CALL GETKEY(IRR)
      IF(IRR.EQ.0)GO TO 100
      GO TO 20
  200 CONTINUE
C     AVERAGE NRDGS LOOP
      OTC=4
      DO 210 I=1,NCH
  210 VOLTS(I)=0.
      DO 250 JJ=1,NRDGS
```

```
      WRITE(DOUT,30)CMD,OTC
      CALL SEND(MFEC,DOUT,ISTAT)
      CALL ENTER(DIN,LEN,MFEC,ISTAT)
      READ(DIN,40)(IN(I),I=1,NCH)
      DO 245 I=1,NCH
      VOLTS(I)=VFA*FLOAT(IN(I))+VOLTS(I)
  245 CONTINUE
  250 CONTINUE
      WRITE(0,50)(VOLTS(I),I=1,NCH)
      CALL GETKEY(IRR)
      IF(IRR.EQ.0)GO TO 200
      GO TO 20
  300 CONTINUE
C     CALIBRATION LOOP
  305 DO 360 II=1,4
      OTC=II-1
      DO 310 I=1,NCH
  310 VOLTS(I)=0.
      DO 350 JJ=1,NRDGS
      WRITE(DOUT,30)CMD,OTC
      CALL SEND(MFEC,DOUT,ISTAT)
      CALL ENTER(DIN,LEN,MFEC,ISTAT)
      READ(DIN,40)(IN(I),I=1,NCH)
      DO 345 I=1,NCH
      VOLTS(I)=VFA*FLOAT(IN(I))+VOLTS(I)
  345 CONTINUE
  350 CONTINUE
      WRITE(0,60)II,(VOLTS(I),I=1,NCH)
  360 CONTINUE
      WRITE(0,90)
      CALL GETKEY(IRR)
      IF(IRR.EQ.0)GO TO 300
      GO TO 20
  400 CONTINUE
C     MULTIPLEX AND READ 8 COILS LOOP
      DO 430,II=1,8
      DO 425,JJ=1,3
      WRITE(DOUT,30)CMD,OT8(II)
      CALL SEND(MFEC,DOUT,ISTAT)
      CALL ENTER(DIN,LEN,MFEC,ISTAT)
      READ(DIN,40)(IN(I),I=1,NCH)
      DO 420 I=1,NCH
      VOLTS(I)=VF1*FLOAT(IN(I))
  420 CONTINUE
      WRITE(0,60)II,(VOLTS(I),I=1,NCH)
  425 CONTINUE
  430 CONTINUE
      CALL GETKEY(IRR)
      IF(IRR.EQ.0)GO TO 400
      GO TO 20
  500 CONTINUE
C     MULTIPLEX AND READ 16 COILS LOOP
      DO 530,II=1,16
```

```
C     DO 525,JJ=1,3
      WRITE(DOUT,30)CMD,OT16(II)
      CALL SEND(MFEC,DOUT,ISTAT)
      CALL ENTER(DIN,LEN,MFEC,ISTAT)
      READ(DIN,40)(IN(I),I=1,NCH)
      DO 520 I=1,NCH
      VOLTS(I)=VF1*FLOAT(IN(I))
  520 CONTINUE
      WRITE(0,60)II,(VOLTS(I),I=1,NCH)
  525 CONTINUE
  530 CONTINUE
      CALL GETKEY(IRR)
      IF(IRR.EQ.0)GO TO 500
      GO TO 20
  999 STOP
      END
```

APPENDIX G

THE PHNWRK PROGRAM

The program, PHNWRK, is used to design the phase calibrator networks that are used in the three-frequency instrument. The value of the coil inductance, XL1, is selected depending on the frequency, and its exact value is measured using an impedance bridge. The value of capacitance needed to match the inductor is computed and printed out. It must be constructed by paralleling different capacitances until the proper value is reached. The value of tuning resistances is then computed to give the corresponding phase. A listing of the program PHNWRK follows:

```
      PROGRAM PHNWRK
C
C     PHNWRK    October 26, 1987
C     PROGRAM CALCULATES THE RESISTANCE AND CAPACITANCE VALUES FOR A
C     PHASE SHIFT NETWORK.THE PHASE VALUES DESIRED ARE STORED IN AN
C     ARRAY AS PHASE(N).THE PROGRAM ALSO CALCULATES THE ERROR DUE TO
C     DRIFTS OR VARIATIONS IN THE PARAMETER VALUES.
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION PHASE(34)
      DATA FREQ/4.913801D5/,XL1/0.2029D-3/,R3/1.579/,R1/2.00D4/
      DATA NPHT/34/,CAP2/1.0D-6/,CAP3/1.D-10/,V0/1.0D1/,R4/1.00D6/
      DATA LOU/8/,LI/0/
      W=2*3.1415927*FREQ
      CAP1=2./(W*W*XL1)
      PHASE(1)=-85.0D0
      DO 10 NPH=2,NPHT
      PHASE(NPH)=PHASE(NPH-1)+5.0D0
   10 CONTINUE
      WRITE(LOU,20)FREQ,V0,R1,R3,R4
   20 FORMAT(' FREQ ',1PE10.4,' DR VLT',0PF7.3,' SER R',
     11PE10.3,' IND R',0PF8.3,' SHNT R',1PE10.3)
      WRITE(LOU,30)XL1,CAP1,CAP2,CAP3
   30 FORMAT(' INDUC',1PE12.4,' TUN CAP',1PE12.4,' CUP CAP'
     1,1PE11.3,' SHNT CAP',1PE11.3)
      WRITE(LOU,40)
   40 FORMAT(1X)
      WRITE(LOU,45)
   45 FORMAT('    PHASE       TUNING RES  MAGNITUDE')
      DO 100 NPH =1,NPHT
      ARG1=PHASE(NPH)/57.2957795
      R2=(SIN(ARG1)+1.)/(W*CAP1*COS(ARG1))
   50 CALL NETWRK(R1,R2,R3,R4,XL1,CAP1,CAP2,CAP3,W,V0,TMAG,PHA)
      TEST =PHASE(NPH)-PHA
      IF(ABS(TEST).LT.1.0E-6)GO TO 60
      R2=R2*(1+.01*TEST)+TEST
      GO TO 50
   60 CONTINUE
      WRITE(LOU,70)PHA,R2,TMAG
   70 FORMAT(0PF9.3,0PF16.4,0PF11.4)
  100 CONTINUE
      STOP 'JOB '
      END
      SUBROUTINE NETWRK(R1,R2,R3,R4,XL1,CAP1,CAP2,CAP3,W,V0,TMAG,PHA)
```

```
C
C     CALCULATES MAGNITUDES AND PHASES FOR THE PHASE SHIFT NETWORK.
C
      IMPLICIT REAL*8 (A-H,O-Z)
      A1=R4*V0*(W*XL1*R2-R3/(W*CAP1))
      B1=-R4*V0*(R2*R3+XL1/CAP1)
      Q1=R1*R3
      Q2=W*XL1*R1
      Q3=W*CAP3*R2*R4-1./W*(1./CAP1+1./CAP2)
      Q4=-R2-R4*(CAP3/CAP1+CAP3/CAP2+1.)
      Q5=(R1+R3)*R2+XL1/CAP1
      Q6=W*XL1*R2-(R1+R3)/(W*CAP1)
      Q7=-1./(W*CAP2)
      Q8=-R4*(CAP3/CAP2+1.)
      A2=Q1*Q3-Q2*Q4+Q5*Q7-Q6*Q8
      B2=Q1*Q4+Q2*Q3+Q5*Q8+Q6*Q7
      D5=A2*A2+B2*B2
      X=(A1*A2+B1*B2)/D5
      Y=(A2*B1-A1*B2)/D5
      TMAG=SQRT(X*X+Y*Y)
      PHA=57.2957795*ATAN2(Y,X)
      RETURN
      END
```

NUREG/CR-5061
ORNL/TM-10663
Distribution
Category RF

INTERNAL DISTRIBUTION

1-2. Central Research Library
3. Document Reference Section
4-5. Laboratory Records Department
6. Laboratory Records, ORNL RC
7. ORNL Patent Section
8. Nuclear Safety Information Center
9-12. L. D. Chitwood
13-16. W. E. Deeds
17-20. C. V. Dodd
21. W. P. Eatherly
22. H. T. Kerr
23. E. H. Krieg, Jr
24. A. P. Malinauskas
25. R. W. McClung
26. D. L. Selby
27. J. G. Pruett
28. C. E. Pugh
29. W. H. Simpson
30. G. M. Slaughter
31. J. H. Smith
32-33. P. T. Thornton
34. V. C. Vaughen
35. J. R. Weir
36. H. D. Brody (Consultant)
37. G. Y. Chin (Consultant)
38. F. F. Lange (Consultant)
39. W. D. Nix (Consultant)
40. D. P. Pope (Consultant)
41. E. R. Thompson (Consultant)

EXTERNAL DISTRIBUTION

42. BATTELLE PACIFIC NORTHWEST LABORATORIES, P.O. Box 999, Richland, WA 99352

    B. Ferris

43. S. D. Brown, 23 Greensburg Street, Delmont Borough, PA 15626

44-45. ELECTRIC POWER RESEARCH INSTITUTE, P.O. Box 217097, Charlotte, NC 28221

    M. Elmo
    K. Kryzwosz

46-48. NRC, OFFICE OF NUCLEAR REGULATORY RESEARCH, Washington, DC 20555

    G. A. Arlotto
    J. Murcara
    C. Z. Serpan

49. ROCKY FLATS PLANT, P.O. Box 464, Golden, CO 80402

    Bruce Allen

50. DOE, Oak Ridge Operations Office, Oak Ridge, TN 37831
Assistance Manager, Energy Research and Development

51-52. DOE, Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831

53-302. Given RF Category Distribution.

NRC FORM 335 (2 84) NRCM 1102, 3201, 3202

U.S. NUCLEAR REGULATORY COMMISSION

BIBLIOGRAPHIC DATA SHEET

SEE INSTRUCTIONS ON THE REVERSE

1. REPORT NUMBER (Assigned by TIDC add Vol No., if any)
NUREG/CR-5061
ORNL/TM-10663

2. TITLE AND SUBTITLE
Three-Frequency Eddy-Current Instrument

3. LEAVE BLANK

4. DATE REPORT COMPLETED

MONTH	YEAR
January	1988

5. AUTHOR(S)
C. V. Dodd and L. D. Chitwood

6. DATE REPORT ISSUED

MONTH	YEAR

7. PERFORMING ORGANIZATION NAME AND MAILING ADDRESS (Include Zip Code)
Oak Ridge National Laboratory
P. O. Box 2008
Oak Ridge, Tennessee 37831

8. PROJECT/TASK WORK UNIT NUMBER

9. FIN OR GRANT NUMBER
B0417

10. SPONSORING ORGANIZATION NAME AND MAILING ADDRESS (Include Zip Code)

11a. TYPE OF REPORT
Topical

b. PERIOD COVERED (Inclusive dates)

12. SUPPLEMENTARY NOTES

13. ABSTRACT (200 words or less)

A three-frequency eddy-current instrument has been constructed for general multiple property applications, with particular emphasis on light water reactor steam generator tubing examination. A description is given of the overall operating principles of the complete instrument and of the operation of the different modules in the instrument. Also included are wiring and printed circuit diagrams and the necessary computer programs to run the instrument.

14. DOCUMENT ANALYSIS - a. KEYWORDS/DESCRIPTORS

b. IDENTIFIERS/OPEN-ENDED TERMS

15. AVAILABILITY STATEMENT
Unlimited

16. SECURITY CLASSIFICATION
(This page)
Unclassified
(This report)
Unclassified

17. NUMBER OF PAGES

18. PRICE

☆U.S. GOVERNMENT PRINTING OFFICE: 1988-548-11980043

120555078877 1 1AN1RF
US NRC-OARM-ADM
DIV OF PUB SVCS
POLICY & PUB MGT BR-PDR NUREG
P-210
WASHINGTON DC 20555

120555078877 1 1AN1RF
US NRC-OARM-ADM
DIV OF PUB SVCS
POLICY & PUB MGT BR-PDR NUREG
P-210
WASHINGTON DC 20555