MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES

3 4456 0278611 7

ORNL-6408

ornl

OAK RIDGE NATIONAL LABORATORY

MARTIN MARIETTA

Pulsed Eddy-Current Inspection of Thin-Walled Stainless Steel Tubing

C. V. Dodd
D. W. Koerner
W. E. Deeds
C. A. Pickett

OAK RIDGE NATIONAL LABORATORY
CENTRAL RESEARCH LIBRARY
CIRCULATION SECTION
4500N ROOM 175
LIBRARY LOAN COPY
DO NOT TRANSFER TO ANOTHER PERSON
If you wish someone else to see this report, send in name with report and the library will arrange a loan.

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

Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road, Springfield, Virginia 22161
NTIS price codes—Printed Copy: A06 Microfiche A01

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL-6408
Distribution
Categories UC-23,
-25, -38

Metals and Ceramics Division

PULSED EDDY-CURRENT INSPECTION OF THIN-WALLED STAINLESS STEEL TUBING

C. V. Dodd
D. W. Koerner
W. E. Deeds
C. A. Pickett

Date Published - September 1987

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

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES
3 4456 0278611 7

CONTENTS

ABSTRACT 1
INTRODUCTION 1
TEST DESIGN 2
ANALYTICAL DESIGN STUDY 2
IMPEDANCE ANALYZER EXPERIMENTAL MEASUREMENTS 5
PULSED INSTRUMENT REQUIREMENTS 7
INSTRUMENTATION DEVELOPMENT 11
MANUALLY ADJUSTED PULSED INSTRUMENT CLOCK 11
PULSE DRIVER 13
COIL AND ATTENUATOR CIRCUITS 13
WIDEBAND AMPLIFIER 14
VOLTAGE LEVEL COMPARATOR 14
FLIP-FLOP CIRCUIT 16
LOW-PASS FILTER AND BALANCE CIRCUIT 16
COMPUTER-ADJUSTED PULSED INSTRUMENT 17
ANALOG-TO-DIGITAL CONVERTER BOARD 20
NDT COMP9B MICROCOMPUTER 20
DRIFT STUDIES FOR EACH MODULE 20
SCANNER AND MECHANICAL HANDLING SYSTEM 23
DATA PROCESSING 26
STANDARDS 26
PULSE SHAPE AND PRESET VOLTAGE LEVELS 27
DATA DISPLAY AND STORAGE 29
DATA EVALUATION AID PROGRAMS 31
SUMMARY 31
REFERENCES 32
Appendix 33
PROGRAM LISTINGS 33
PULRDG2 33
PULFIT3 42
SSTPUF3 58
POSPUL 63
FLAWLOC3 67

DEFTST 73
PULG2 79
PULG 83
PORT I INTERFACE OF NDT COMP 9B 99
PORT 2 INTERFACE OF NDT COMP 9B 100
INSTRUMENT WIRING DIAGRAMS 101

PULSED EDDY-CURRENT INSPECTION OF THIN-WALLED STAINLESS STEEL TUBING

C. V. Dodd, D. W. Koerner,* W. E. Deeds,** and C. A. Pickett***

ABSTRACT

A pulsed eddy-current system has been developed for production inspection of small, thin-wall, non-ferromagnetic tubing. To detect and accurately size both outer and bore-side flaws required an operating frequency higher than available from present commercial equipment. A pulsed eddy-current instrument was designed and constructed that used 3.2 MHz square waves, with a bandwidth of 20 MHz. The system is able to reliably detect flaws as small as 0.015 mm (0.6 mils) on either the inner or outer surface of the tube. A computer controls the scanning of the tube, as well as recording, analyzing, and plotting the data. The computer programs and instrument details are given in the report.

INTRODUCTION

A bellows is fabricated from a 0.3 m (12 in.) section of thin-walled stainless steel tubing. Because of failures in the bellows, methods of improving the quality of the tubing through nondestructive testing were investigated. The only existing inspection for screening the tubing was an ultrasonic shear-wave test. Unfortunately, this inspection could not be calibrated to detect interior defects, and required all defects to be oriented perpendicular to the sonic field. A feasibility study was conducted to determine if an eddy-current technique could be developed to adequately inspect the tubing.

The problem consisted of inspecting a large quantity of thin-walled, stainless steel tubing to a sensitivity level of 0.015 mm (0.0006 in.) for cracklike flaws on either the inner or outer surface of the tube. The tubes were nominally 3.66 ± 0.05 mm (0.144 ± 0.002 in.) in diameter and

*Product Certification Division, Oak Ridge Y-12 Plant, Oak Ridge, Tenn.
**Adjunct research participant, University of Tennessee, Knoxville.
***Development Division, Oak Ridge Y-12 Plant, Oak Ridge, Tenn.

0.127 ± 0.013 mm (0.005 ± 0.0005 in.) thick, made of stainless steel (type 321 and type 316 were both tested). The flaws could be either longitudinal or transverse.

TEST DESIGN

ANALYTICAL DESIGN STUDY

Although a full-scale analytical design study, starting with Maxwell's equations and using our computer design programs[1] for both wall thickness variations and flaws, would have probably given better results, there was not enough time due to project schedules. Instead, we estimated the optimum coil size and operating frequency from earlier work.[2] The general coil and conductor system used is a reflection-type probe, as shown in Fig. 1. This coil system has been well analyzed, and we have successfully used it on a large number of inspection problems in the past. We also had a number of coils with associated instrumentation on hand.

An eddy-current test can easily detect and measure near-side flaws, so most of our design study is concentrated on the detection and sizing of far side flaws (i.e., inner surface flaws with an external coil). From the previously derived curves shown in Fig. 2, we can determine first the optimum coil size to get the maximum phase shift from a far-side flaw. A ratio of thickness-to-coil mean radius of 0.6 gives the best sensitivity for flaws about 12% of the wall thickness. The coil mean radius is calculated to be 0.20 mm (0.008 in.) for the nominal wall thickness. The product of frequency, permeability, conductivity, and mean coil radius squared for maximum sensitivity is 7, read from the curve in Fig. 2, which gives a value of 16 MHz for the operating frequency. However, it has been impractical to construct a reflection coil smaller than 0.51 mm (0.020 in.), and this size coil would be very sensitive to lift-off. A 0.51 mm (0.020 in.) mean radius reflection coil has a thickness-to-mean radius ratio of 0.25, and the sensitivity to flaws is about half that of the optimum size coil. The product of frequency, permeability,

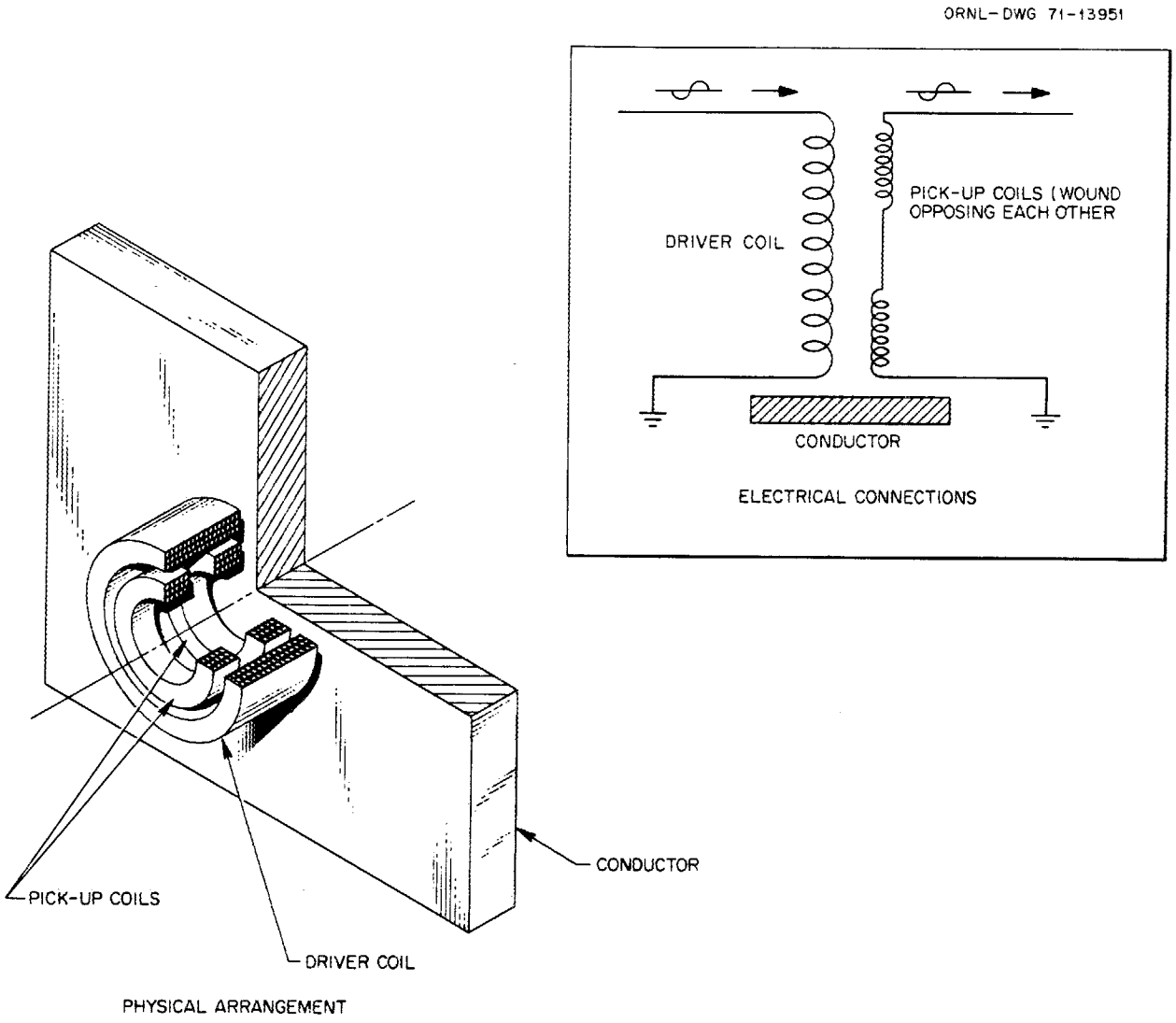


Fig. 1. General coil and conductor system for a reflection probe.

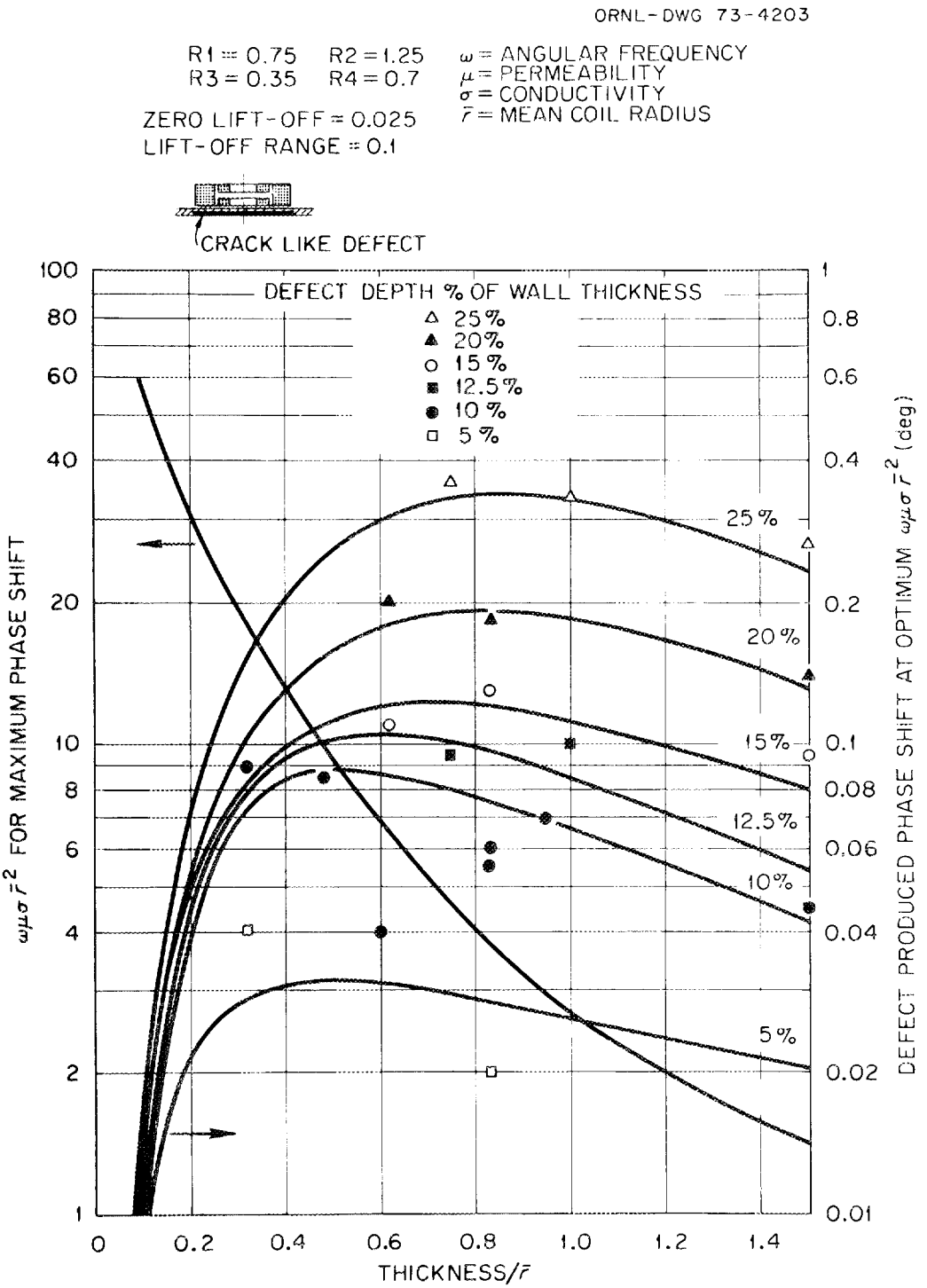


Fig. 2. Optimum frequency and phase shift at optimum frequency for different ratios of mean coil radius to conductor thickness.

conductivity, and mean coil radius squared for maximum sensitivity is 25 [for the 0.51 mm (0.020 in.) coil], which gives a value of 9 MHz for the frequency. The optimum frequency for detecting near-side flaws would be much higher, as would the optimum frequency for measuring the lift-off effects. The optimum frequency for maximum sensitivity to thickness changes is 5.5 MHz, as determined from an earlier report.[3] Using these approximate values as guidelines and a starting point, we then moved to the experimental tests.

IMPEDANCE ANALYZER EXPERIMENTAL MEASUREMENTS

An experimental design study was performed, using the Hewlett-Packard 4192A Low Frequency Impedance Analyzer. A report[4] has been written describing the use of this instrument for multiple-property tests. The Impedance Analyzer and a mechanical scanner system are both controlled by the IBM System 9000 Computer over the IEEE-488 bus. For the initial design study, measurements were made on a single tube with three inner surface and three outer surface flaws with depths of 0.025, 0.051, and 0.076 mm (0.001, 0.002, and 0.003 in.). The frequencies used were 3.25, 6.5, and 13.0 MHz. The readings were made using the program RDGANA, and least-squares fits of the readings to the flaw sizes were made using the program FITANA. (All programs mentioned in this report are listed in the appendix.) The best results were obtained using the magnitude and phase of only the 13.0 MHz frequency. The best fit was obtained using a constant, linear term, and square term in both the magnitude and phase and a cross term between the magnitude and phase. The rms fit error was 0.008 mm, (0.0003 in.) and the drift error was 0.0004 mm (0.000015 in.) for a change in magnitude of 0.001 dB and a change in phase of 0.01°.

Two programs were written to take these magnitude and phase readings on the tubing and compute the flaw size. The program SSTTUB plots a three-dimensional view of the tube with flaws on the CRT of the IBM System 9000, and the program SSTGRA plots a similar view on the printer for the IBM System 9000 Computer. In Fig. 3, we show the output of a plot of SSTTUB from a scan of the standard tube, containing the 0.025, 0.051, and 0.076 mm (0.001, 0.002, and 0.003 in.) flaws on both the inner surface and

ORNL-DWG 87-7032

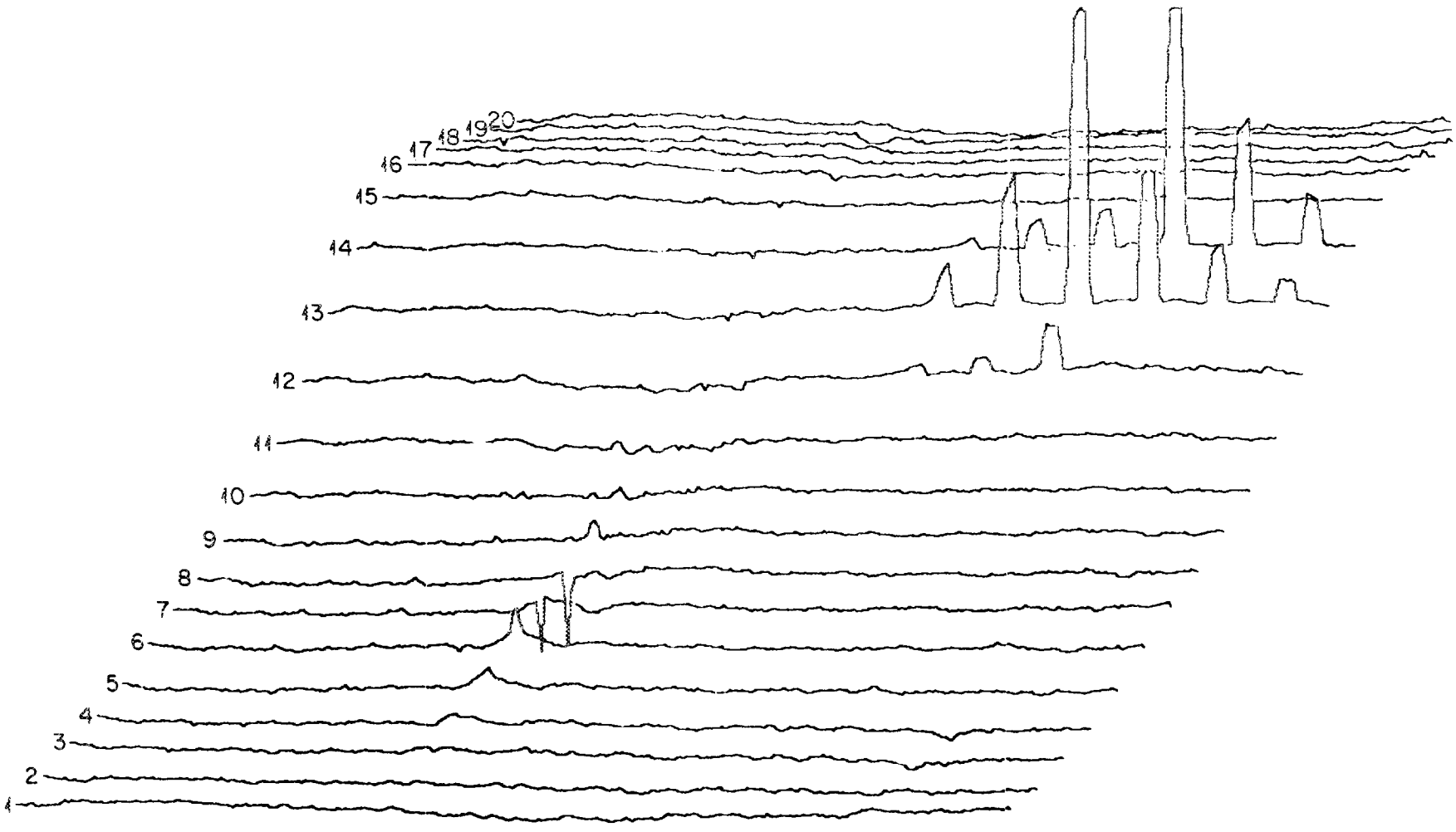


Fig. 3. Plot of defect size of a standard tube using the Impedance Analyzer. From left to right, the scans show calculated flaw sizes for 0.025, 0.051 mm flaws on the outer surface and then 0.076, 0.051, and 0.025 mm flaws on the inner surface of the tube.

outer surface of the tube. Because of the indexing of the flaw locations, the outer surface flaws show up more clearly on the thirteenth scan line, and the inner surface flaws show up better on the fourteenth scan line. It is more important that the computer-calculated flaw sizes are very nearly the same for a given size of flaw, regardless of whether the flaw is located on the inner or outer surface, and are also directly proportional to the actual depth of the flaw. The system demonstrated the ability to reliably detect both inner surface and outer surface, longitudinal and transverse, flaws as small as 0.015 mm (0.0006 in.) in relatively clean tubing. This sensitivity was judged to be adequate for tubes with no wall thickness variations, but the scan time for even this single frequency measurement was excessive. The time for a low resolution hard-copy scan was about 41.3 s/mm (17.5 min/in.) with the main delay time caused by the slow speed of the Impedance Analyzer. A new instrument, the Hewlett Packard 4194A Impedance/Gain-Phase Analyzer, was tested as a possible replacement. The new instrument could perform the readings at a rate of 60 readings per second, compared to a rate of 1 reading per second for the old instrument. However, the noise was excessive at this high speed. The noise could be reduced by averaging, but when enough readings were taken to reduce the noise level to that of the old instrument, the speed was then less than that of the old instrument. Because of the high volume of tubing to be inspected it was decided to proceed with the development of a high-speed system that would be capable of multiple-property measurements. These are measurements where several different test properties such as lift-off, wall thickness, and flow size can vary at the same time. This type of measurement requires that the interrogating signal contain enough information that the effects of the different property variations on the instrument readings can be separated. The pulsed system described in this report scans the tubes at about 3.5 cm (1.5 in.) per minute, 26 times faster that the Impedance Analyzer fixed-frequency system.

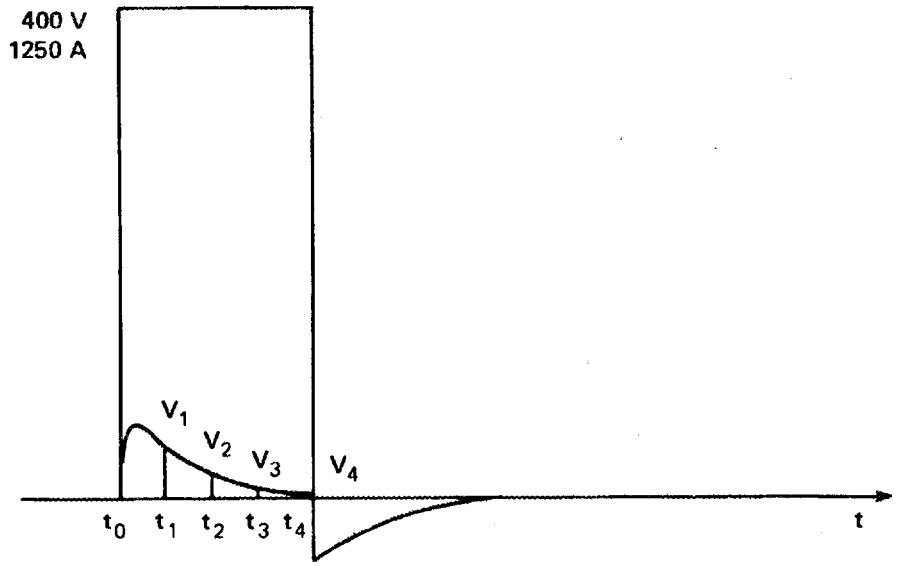
PULSED INSTRUMENT REQUIREMENTS

The traditional multiple-frequency instruments that had been developed at ORNL[5] were unable to operate reliably above 2 MHz, and instruments that

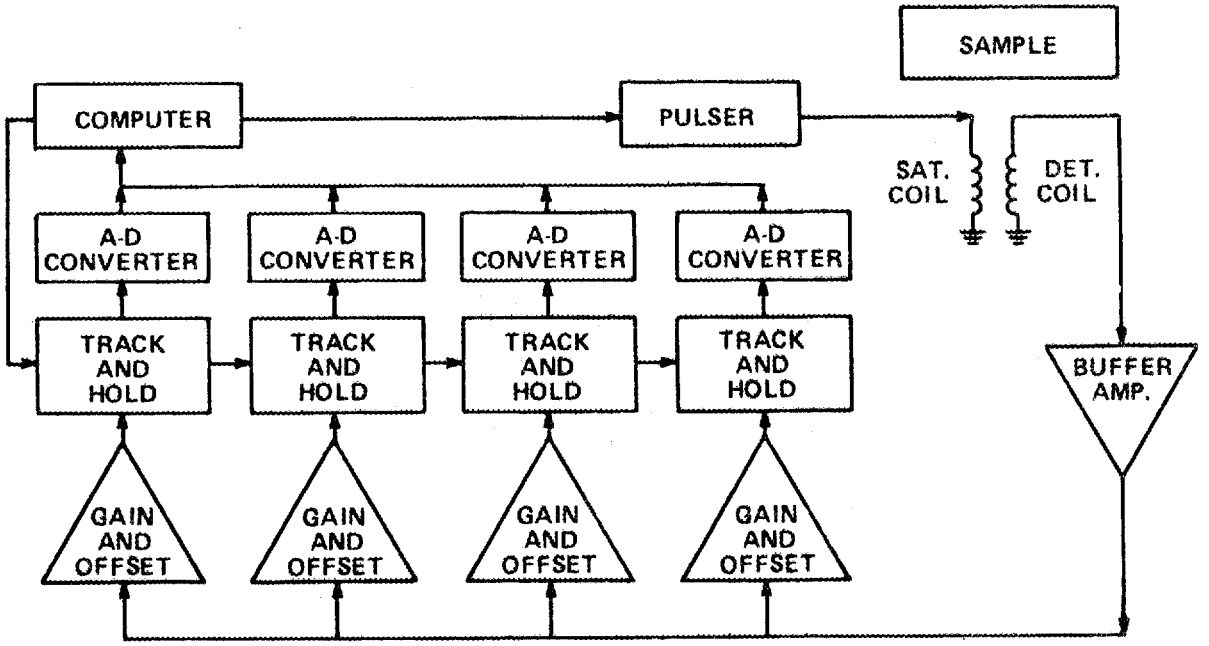
multiplex the frequencies, such as the Impedance Analyzer, tend to be slow and noisy. Therefore we decided to develop a pulsed instrument, with an effective bandwidth of 20 MHz.

A block diagram of the first type of pulsed instrument considered and a simulated pulse are shown in Fig. 4. This instrument transmits a pulse to the driver coil; the resulting electromagnetic field interacts with the tube specimen and then is detected by the receiver coil. The detected pulse is amplified and fed to sample-and-hold (or track-and-hold) modules. A timer, synchronized to the transmitted pulse, "holds" the voltage amplitude at given times so that the slower analog to digital converters can measure the voltage and send this digital value to the computer. These values of the voltage at given times can be used to compute the different specimen properties of a multiple-property eddy-current test just as the magnitudes and phases of sinusoidal eddy currents can be used in multiple-frequency instruments. This type of instrument was developed for low-frequency pulsed-saturation measurements[6]; and, in the particular sampling mode used, the frequency response was limited by the slew rate of the sample-and-hold modules. An alternative to this type of pulsed instrument is one that measures the times taken for the pulse to reach given voltage levels, as shown in Fig. 5. This instrument also transmits a pulse to the driver coil, and the pulse, modified by the sample, is detected by the receiver coil. The pulse is amplified and then fed to voltage height detectors that put out a pulse when the voltage passes through their preset voltage level. Each detector can be set to give the pulse when the voltage is increasing or when it is decreasing through the preset level. The time between the transmitted pulse and the time that the voltage passes through the set level is measured. This value is then converted to a digital value and can be used in the multiple-property calculations in the same manner as the readings from the other types of instruments. Since the process to measure these time intervals accurately at these frequencies had already been successfully proven with phase-sensitive instruments, it was decided to develop an instrument of this type. One drawback of this type of eddy-current instrument is that, for large values of lift-off, the preset voltage level is never reached and the

ORNL-DWG 85-13153



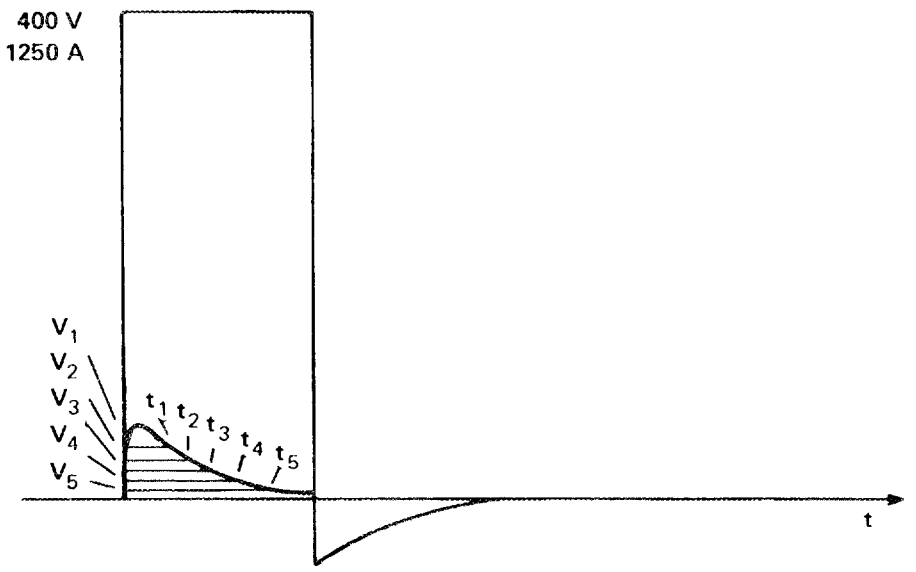
(a) **PULSE TO SATURATING COIL AND DETECTOR RESPONSE**



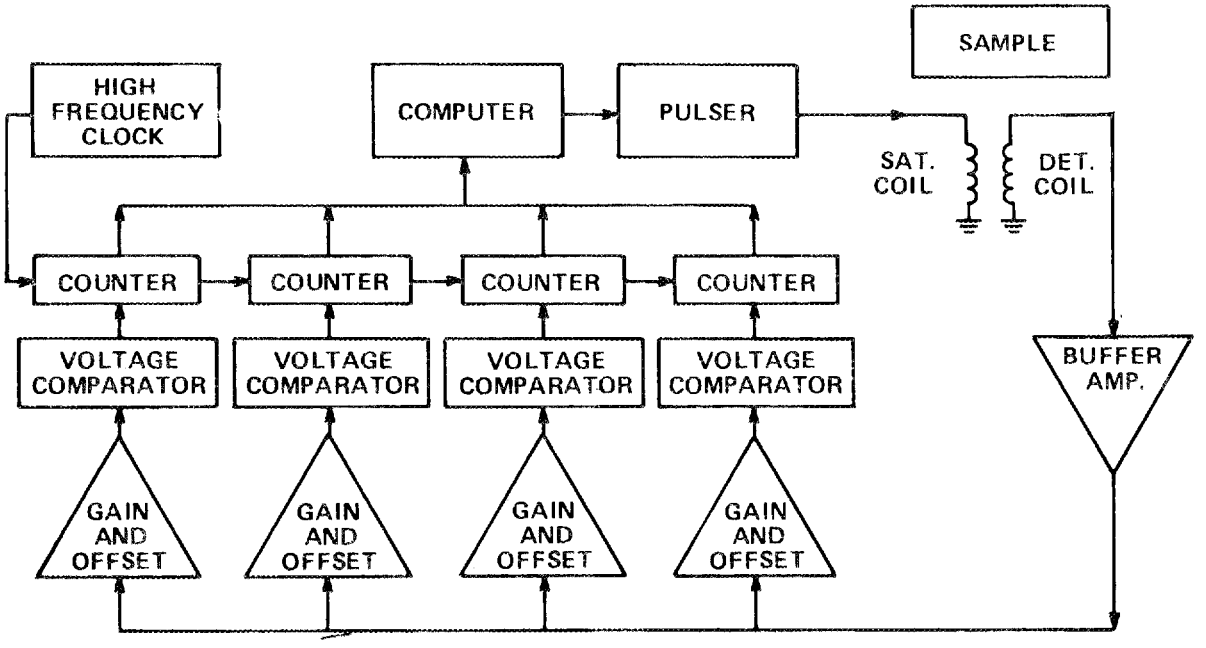
(b) **A PULSE AMPLITUDE INSTRUMENT**

Fig. 4. Pulsed eddy-current instrument to measure magnitudes at given times.

ORNL–DWG 85-13154



(a) **PULSE TO SATURATING COIL AND DETECTOR RESPONSE**



(b) **A PULSE TIME INTERVAL INSTRUMENT**

Fig. 5. Pulsed eddy-current instrument to measure time required to reach given voltage levels.

measured time interval becomes very large. The amplitude measuring pulse instrument will only lose resolution and fail in a more graceful manner as the lift-off becomes large.

INSTRUMENTATION DEVELOPMENT

A simplified block diagram of the pulsed eddy-current inspection system is shown in Fig. 6. The IBM System 9000 computer is used to control the mechanical scanner, control the pulsed eddy-current instrument, collect the data, compute the properties from the data, plot the results on paper, and store the data for later reference. The clock operates at a frequency of 3.2 MHz, and the pulser feeds square waves at that repetition frequency to a reflection probe through an RLC network. The reflected signal, modified by the tube, is amplified by a wideband amplifier, buffered, and fed to eight voltage height detectors. The voltage height detector produces a signal when the pulse passes a preset level. A signal from the pulser turns on a flip-flop circuit, which is then turned off by the signal from the voltage height detector. This flip-flop circuit drives a higher voltage circuit that swings between two reference voltages as the flip-flop turns on and off. The average voltage from this circuit is proportional to the time between the pulser voltage turning on and the amplified reflected pulse passing a certain voltage level. This voltage is averaged by a low-pass filter, balanced against an offset voltage, and amplified. It is then converted to a digital signal by the analog-to-digital converter (ADC) board, which consists of eight integrating ADCs. The NDT-COMP9B[7] Computer reads each of the ADCs when they have completed a measurement and sends the data to the IBM System 9000 Computer over the IEEE-488 bus. Two versions of the pulsed instrument were constructed, one with manual adjustments and one with adjustments made by the IBM System 9000. We shall first discuss the individual circuits of the manually adjusted instrument. Drawings for both instruments are included in the appendix.

MANUALLY ADJUSTED PULSED INSTRUMENT CLOCK

The clock is constructed using two sections of a 74F04 Hex Invertor, with a 3.2 MHz crystal to control the frequency. Another stage is used to buffer the output, and another used to buffer a test point.

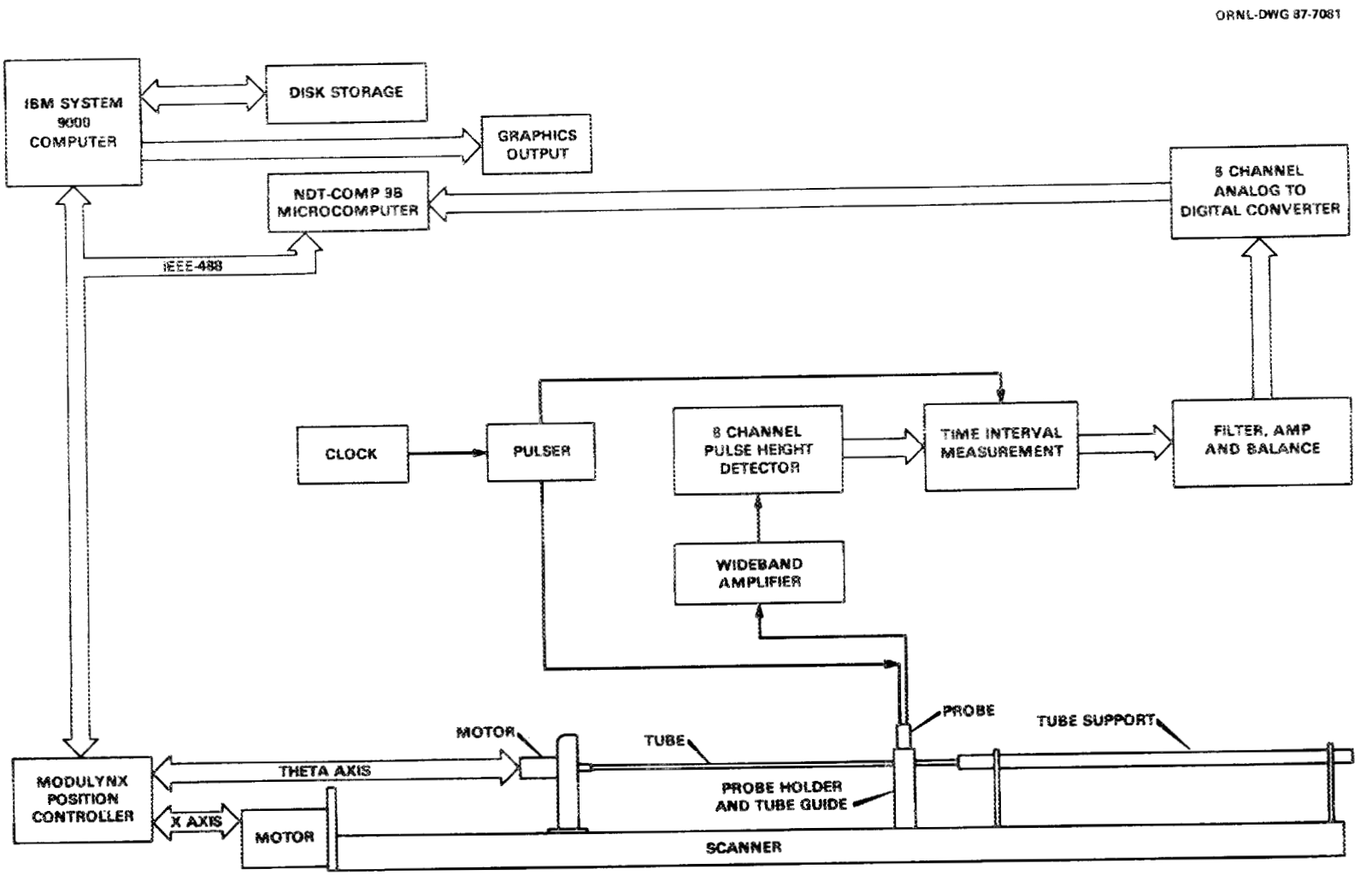


Fig. 6. Simplified block diagram computer setup.

PULSE DRIVER

The pulse output driver consists of two integrated circuits, the first a dual TTL-to-MOS driver, the Fairchild 9643. This integrated circuit takes the low-level TTL signal, inverts it, and translates it into a signal slightly less than the power supply voltage, with about 9 ns delay. This signal is ideal for driving the next stage, a Signetics VQ7254. The VQ7254 is an integrated circuit consisting of two N-Channel and two P-Channel power field-effect transistors. One P-channel transistor is used as a low-resistance switch to a very stable 10 V supply and is turned on when the output from the 9643 is low. One N-Channel transistor is used as a low-resistance switch to ground and is turned on when the output from the 9643 is high. This circuit can furnish up to an ampere of drive current, switching between two stable voltage levels, with rise times on the order of 30 ns. There is very little change in either voltage levels or delay times for this circuit as the temperature is varied. The outputs of the two N- and P-Channel switches are tied together through two small resistors. This helps reduce the current consumption when switching is occurring and both switches are conducting. A resistor divider network is also included to drive the reference timing circuit.

COIL AND ATTENUATOR CIRCUITS

The pulse driver has to drive an eddy-current reflection-type coil through a shielded cable. This represents a RLC circuit, and it must be properly terminated to decrease the ringing on the pulse. The effective driving impedance is 50 ohms, to match the cable. A 50 ohm resistance is also placed in parallel with the driver coil and 50 ohm resistors are both in parallel and in series with the pickup coil. The impedance matching of this network allows several feet of cable to be driven without too much signal deterioration, but in the actual test we only needed 0.6 mm (2 ft) to go from the instrument to the probe.

WIDEBAND AMPLIFIER

The reflected signal from the pickup coil is amplified about 200 times by a wideband amplifier. The amplifier gain can be adjusted from 11 to 1250 by turning two potentiometers.

A plot of the magnitude and phase of the gain versus frequency is shown in Fig. 7. The gain is down by 6 dB at 13 MHz, which helps to compensate somewhat for the increase in mutual coupling between the driver and pickup coils with frequency. The increase in mutual coupling with frequency would cause more high frequency components of the pulse to be passed if the amplifier bandwidth and circuit capacitance did not combine to attenuate it. Drifts in the phase shifts of the wideband amplifier cause changes in the measured time intervals, but in general the entire pulse tends to drift together, and these drifts cause only an offset in the base line of the data. This effect can be greatly diminished by the method of data processing, to be described later. This is necessary because this module, as we shall see in the table of temperature drifts, is by far the major contributor to these drifts.

VOLTAGE LEVEL COMPARATOR

This circuit produces a change in dc level on the output when the input voltage passes through a preset level, and consists of an LT1016 integrated circuit. The circuit has a low-temperature drift of both the input offset voltage and the delay time. The circuit output switches from ground to a positive voltage as the input goes from less than the preset voltage to greater than the preset voltage. It also has an inverted output so that either the positive or negative going portion of the reflected pulse can be used as the trigger. A small amount of positive feedback is used to give the circuit some hysteresis. This eliminates multiple triggers that can occur due to noise on the input and also increases the circuit switching speed. There are eight separate comparators that are driven by the wideband amplifier, with a small isolating resistor on the input of each comparator. A switch is provided so that either positive-going or negative-going triggering can be chosen for each comparator, and the voltage trigger level for each can be set using a

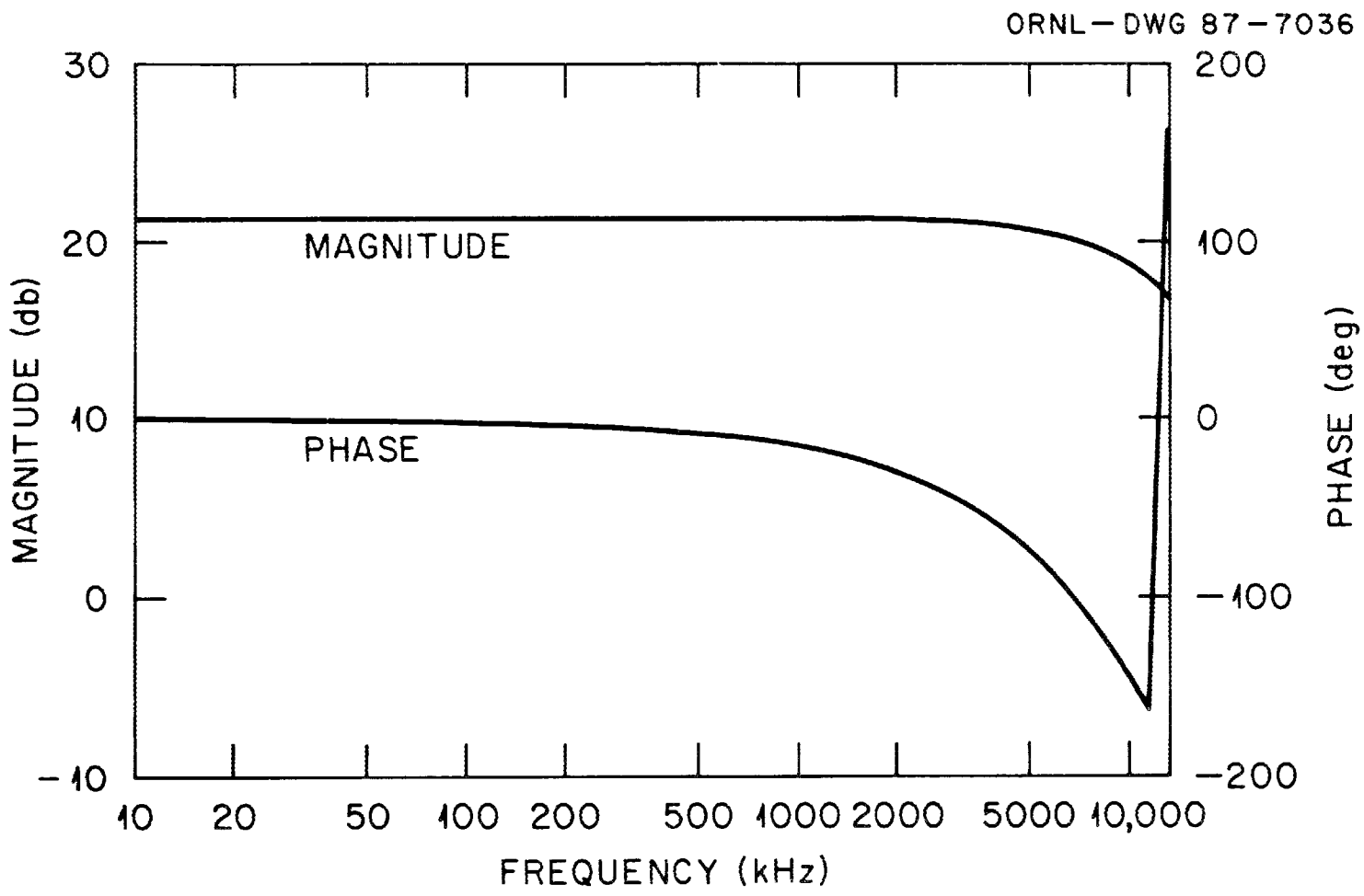


Fig. 7. Bode plot of wideband amplifier response.

potentiometer. There is a ninth comparator circuit, identical to these, triggered from the pulse transmitted to the driver coil.

FLIP-FLOP CIRCUIT

This circuit is used to produce a voltage proportional to the time between the switching of the transmitted-signal comparator and the switching of each of the eight received-signal comparators. The first integrated circuit is a 74F74 dual D-type positive-edge-triggered flip-flop. It is connected in a manner such that the output switches to the high state when the transmitted signal comparator goes positive and switches to the low state when the received signal comparator goes positive. The transitions to the low state by either comparator have no effect on the output of the flip-flop. The flip-flop drives the same type of TTL-to-MOS driver integrated circuit, the 9643, that is used in the pulse driver circuit. This circuit in turn drives the same type of integrated circuit used in the pulse driver, the VQ7254, which has a pair of complementary N-Channel and P-Channel power field-effect transistors. One P-Channel transistor is used as a low-resistance switch to a very stable 10 V supply and is turned on when the output from the 9643 is low. One N-Channel transistor is used as a low-resistance switch to ground, and is turned on when the output from the 9643 is high. The output voltage from this circuit switches between these two stable voltages at the start of the transmitted pulse and when the reflected signal passes through a given level. The time that it is on is proportional to the time between the two events. This signal can be averaged to furnish a voltage proportional to the time it takes the reflected signal to reach the set voltage level.

LOW-PASS FILTER AND BALANCE CIRCUIT

The low-pass filter and balance circuit consists of one integrated circuit that contains four operational amplifiers. The first three are connected to form a state-variable type of filter. It has a Butterworth response (maximum amplitude attenuation with no ripple), with a break frequency at 1 KHz. A plot of the magnitude and phase of this filter

versus frequency is shown in Fig. 8. The filter circuit has a gain of 5, and a fourth operational amplifier is used to set the dc level. This level can be adjusted using the balance potentiometer.

COMPUTER-ADJUSTED PULSED INSTRUMENT

An advanced model of the pulsed instrument that included computer adjustment of most of the controls was designed, constructed, and tested. Since earlier work with the impedance analyzer had demonstrated the usefulness of an instrument that could be adjusted and set up by a computer program without operator intervention, we constructed an instrument to perform these functions. A block diagram of this instrument is shown in Fig. 9. The individual circuits will be discussed here only where they differ from the manually adjusted instrument. The basic components added were two programmable counters and two different types of digital-to-analog converters (DAC). The programmable counters were 74F525, and one counter controls the on-time of the pulse and the other controls the off-time of the pulse. The clock rate has been increased to 10 MHz, to allow greater resolution. The counters can count to 65,535 so that base frequencies from about 70 Hz to 2.5 MHz can be obtained. The on-time and the off-time of the pulse can be varied independently. The DACs are of two different types, the bipolar AD394 and the unipolar AD395. The height of the output pulse can be set from 0 to 10 V, and the voltage level at which any of the nine comparators triggers can be set anywhere between -5 and 5 V. The balance control can be set anywhere between 0 and 10 V, and the setup computer program will automatically determine the voltage to set the balance voltage so that the output of the filter circuits is in the middle of the input range for the DACs. Both the AD394 and AD395 have four independent 12-bit ADCs in each integrated circuit, so only seven more integrated circuits are needed to replace the manually adjusted potentiometers, and the overall board size is the same. The rest of the instrument is common to both the manually adjusted and computer-adjusted instruments.

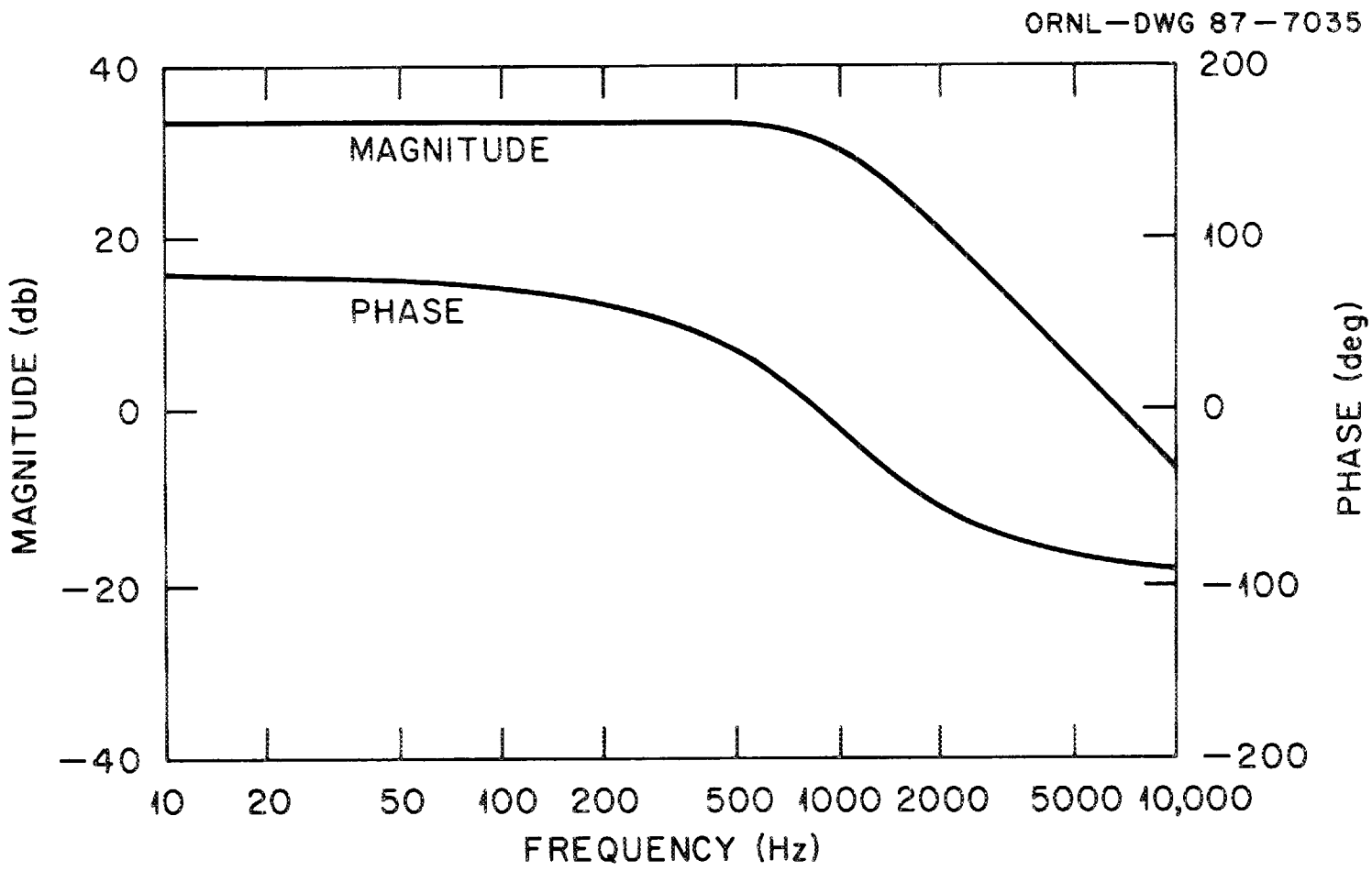


Fig. 8. Bode plot of filter circuit response.

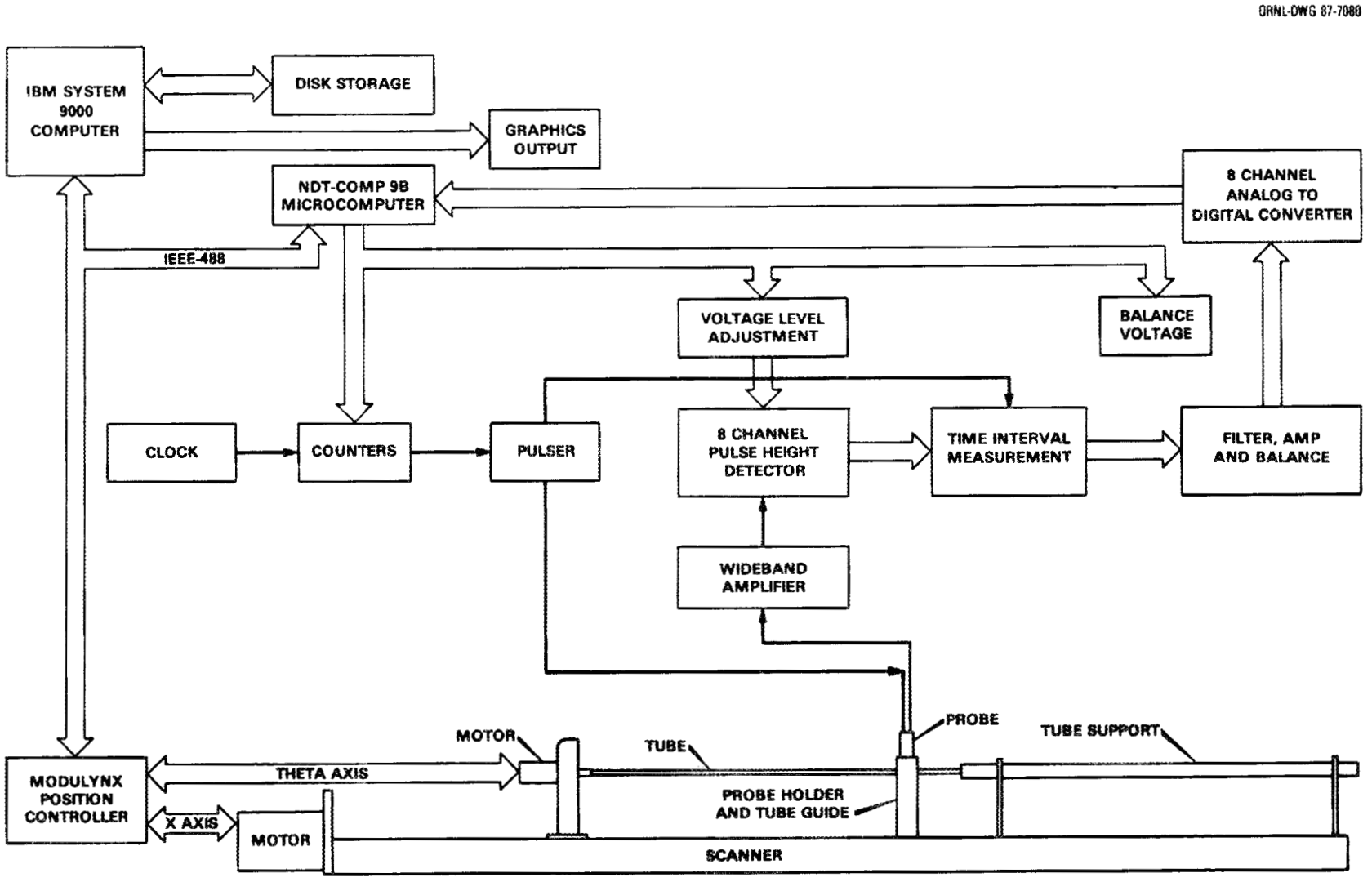


Fig. 9. Simplified block diagram computer setup.

ANALOG-TO-DIGITAL CONVERTER BOARD

This board consists of eight 12-bit integrating analog-to-digital converters (ADCs) that digitize the voltage from each of the eight pulse data channels. They are driven by the outputs of each of the filter circuits and work over a 0 to 5 V range. The conversion speed of each of the ADCs is set by an external resistor to approximately 10 ms. The busy lines of the ADCs are all tied together in an eight-input OR gate. The computer can start a conversion for all the ADCs, then sense when they have all completed their conversions. The decoding circuitry is on the board to allow the computer to address and read each individual ADC. This board was designed and constructed for the three-frequency eddy-current instrument, and more details are included in the report[5] of that instrument.

NDT COMP9B MICROCOMPUTER

The instrument is controlled by the COMP9B microcomputer. This microcomputer was designed to go in different types of eddy-current instruments and is also described more fully in another report.[7] The COMP9B is the same as the COMP9, except that an IEEE-488 bus has been added to the COMP9B. This allows it to transmit to and receive data from a large number of different types of computers and other peripheral devices.

DRIFT STUDIES FOR EACH MODULE

In order to insure that the instrument would give reliable readings under a variety of different temperature conditions and to pinpoint which components were more susceptible to drift, a series of temperature drift tests was performed. The temperature was varied on each of the integrated circuits in the instrument on an individual basis, and the resultant drift was recorded. The components were cooled by spraying them with Freon-12 until they were completely frosted and showed no more drift. It is estimated that the components were cooled from approximately 25°C to -45°C, or about a 70°C drop. It is recognized that this test is only approximate, but it gives a quick indication of the temperature stability of the circuits and helps pinpoint problem areas. Some of the components tended to run hotter than 25°C and probably experienced a greater temperature

drop. Also, the moisture condensing on the circuit during the test probably caused additional drift in some instances; and sometimes the spraycooling doused the nearby circuits, causing their drift to add to the drift of the circuit we intended to measure. The drifts of the various circuits are recorded in Table 1 in units of change of each of the ADCs. The final column is the change in the calculated flaw size in millimeters that would be encountered for this drastic temperature change. The major contributor to the drift, as we can see from the table, is the wideband amplifier; its drift is about an order of magnitude greater than the other drifts. The drift in the filter circuits is due to the large input pulse that is being averaged, and a new integrated circuit filter has reduced this drift. The drift in the 9643TC pulse drivers is small, but the drivers used were selected for low drift. The reference-voltage integrated circuits all gave very low drifts. For the most part their drift was due to an adjacent component with lower stability being cooled. Both the ADCs and the DACs used in the computer-controlled instrument gave very little contribution to the drift. The information on the drifts is helpful in circuit development and in evaluating the susceptibility of various possible computer algorithms used for calculating the tube properties to errors caused by instrumental drifts.

The flaw detection ability of the system is based on the changes between the computed flaw size at one point and the computed flaw size at nearby locations, rather than the absolute value of the computed flaw size. The computer program POSPUL was used to measure the stability of the data channels under various conditions and print the results in the table. It will plot the computer flaw amplitude on the CRT so that the effects of cooling on the different components can be observed. It also can be used to test the accuracy of the mechanical positioner. A copy of this utility program is included in the appendix.

Table 1. Temperature drift tests of data channels and calculated flaw size change for various components of the pulsed eddy-current instrument

Component tested	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7	Channel 8	Flaw size (mm)
Pulse generator board	92	92	76	67	80	76	23	57	0.04
Wideband amp, first stage	811	791	385	352	884	857	213	261	0.49
Wideband amp, second stage	1213	1214	991	914	2735	3509	354	435	0.13
Wideband amp, third stage	762	747	504	471	673	633	150	179	0.36
Voltage comparators	208	215	364	179	172	261	95	159	0.20
Type D flip flops (74F74)	92	112	120	106	101	151	66	100	0.05
Pulse driver 9643TC	68	79	52	46	45	46	18	26	0.03
Hi-Level flip-flop VQ7254	34	20	61	49	27	33	9	56	0.02
Filters LF347BN	297	172	239	204	161	227	101	554	0.09
Voltage references	33	23	31	23	23	24	8	80	0.02

SCANNER AND MECHANICAL HANDLING SYSTEM

The system used to scan the tubing was based on an earlier design used to develop eddy-current inspections for steam generator tubing. A Modulynx controller, driven by the IBM System 9000 Computer over the IEEE-488 bus, is used to control the position of a two-axis Velmex scanner, as shown in Fig. 10. The scanner has a linear motion of a slide,

ORNL-PHOTO 4148-87

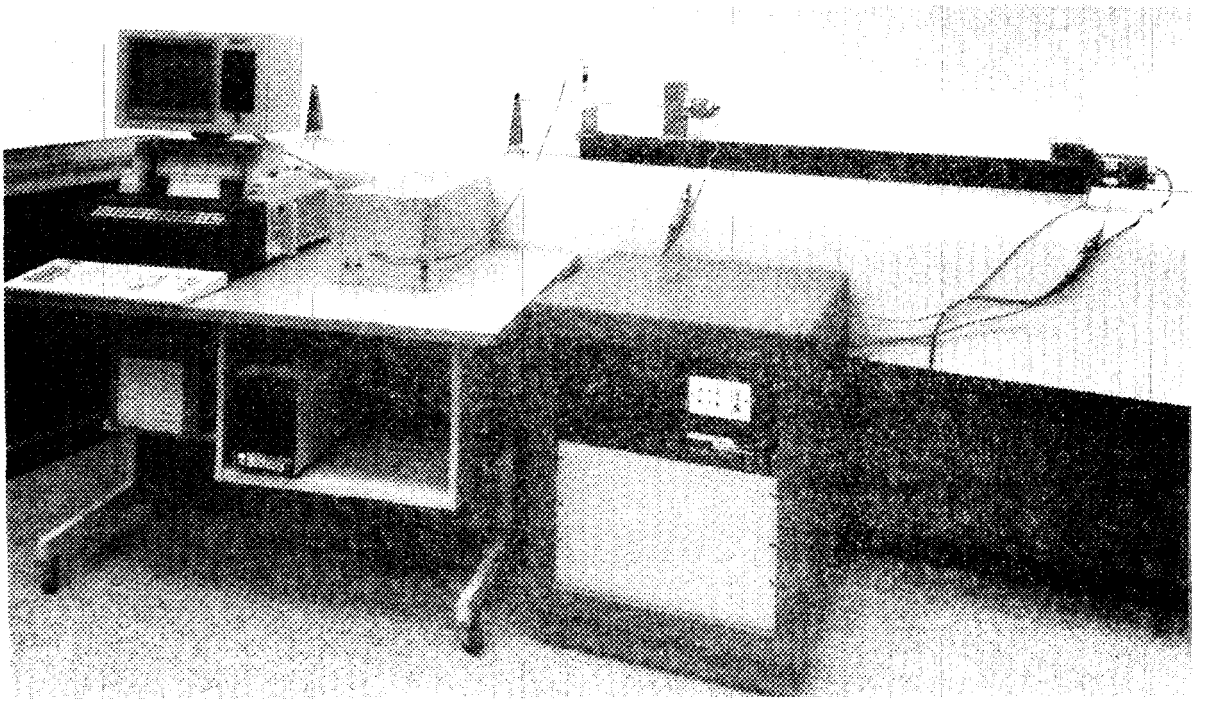


Fig. 10. Modulynx controller and two-axis Velmex scanner.

which is designated the X-axis, which can be as much as 1.086 m (42.75 in.) long. The slide moves 0.006 mm (0.00025 in.) for each step of the X-axis. A second motor is mounted on the slide, and this motor gives the rotational or Theta-axis. The motor makes 200 steps per revolution, giving an angular resolution of 1.8°. Both axes of the scanner have encoders that use a separate board in the Modulynx controller to keep track of the position of the tube, independent of the IBM System 9000.

The IBM can issue a command to the Modulynx to correct for any difference between the number of counts that have been sent to the controller and the number of counts recorded by the encoders. In actual scanning this feature is seldom used because of the time it takes and the fact that, if the correction command is issued too soon after the move command, it causes an erroneous correction to be issued. A mandrel is inserted into the tube to be examined, and the tube is clamped in a collet mounted on the motor. This gives a firm grip on the tube without marking it. Other types of mounting tended to work loose due to the stop-and-go rotation of the motor. The tube passes through a teflon probe holder with teflon inserts to keep the tube from getting scratched. An aluminum bushing is fitted around the probe tip and around the tube to provide uniform cooling to the probe. The probe is 0.05 mm (0.002 in.) above the tube, with only air between the probe and the tube. In Fig. 11, we show the location of the points that are read as the probe is rotated. There are 20 readings made around the tube for each axial translation of the tube. The circumferential spacing between readings is 0.57 mm (0.023 in.), which is adequate to insure that no flaw is missed, or even has a lower value than would be read if the probe were more favorably positioned. However, the position of a flaw can be more precisely located if we increase the number of readings around the circumference of the tube to 40. The readings are made every 0.60 mm (0.024 in.) in the X direction, with one X-axis translation of the tube made for every complete rotation of the tube. The scanning speed is dependent on the rest of the control program, including the data transfer over the bus, the plotting, and data storage. The base speed, the top speed, and the acceleration of each motor can be set by the IBM System 9000 over the bus. A series of scans was run to determine the best settings for each axis, with the computer varying the motor parameters and determining the optimum values for maximum speed and minimum number of positioning errors. These values, which may change with changes in the controlling programs, are transmitted to the Modulynx controller at the start of each scan.

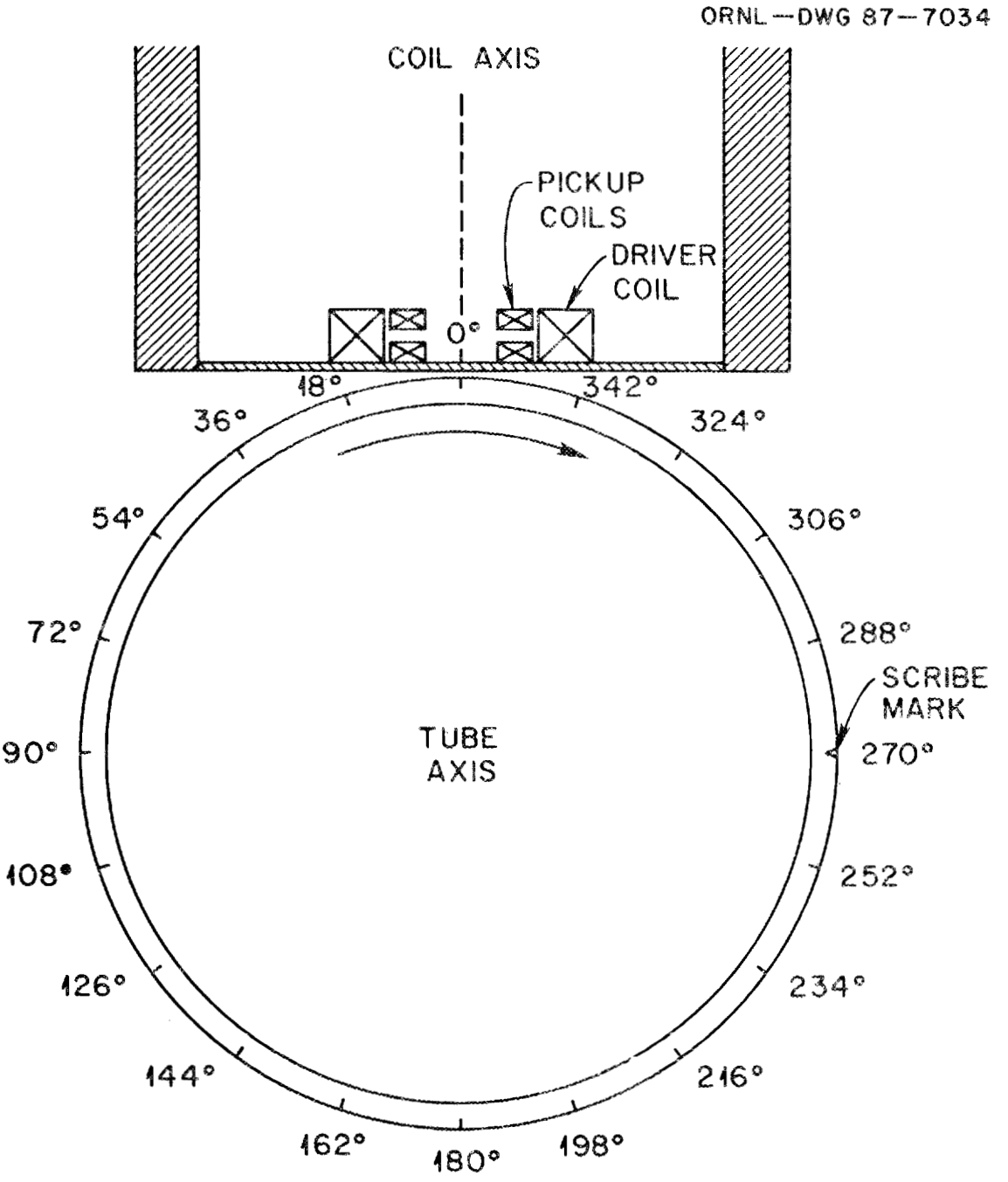


Fig. 11. Location of reading points as tube is rotated.

DATA PROCESSING

The methods of analysis and handling the data are quite important to the success of any inspection; this is particularly true in this case. These include the generation and utilization of standards, the selection of the bandwidth of and the voltage levels on the pulse, the processing of the data, plotting and storage of data, programs to aid the operator in more detailed analysis of the data, and programs to automatically analyze the data.

STANDARDS

Several sets of standards were used to develop and test the equations used to calculate the tube properties from the instrument readings. The first type of standard used consisted of a set of six electrodischarge machined (EDM) notches (three inner surface and three outer surface) with nominal depths of 0.025, 0.051, 0.076 mm (0.001, 0.002, and 0.003 in.). This type of standard was used for the majority of the calibration ("training the computer") and most of the verification testing. The next general type of standard consisted of verification standards that had both longitudinal and transverse EDM notches on both the inner surface and outer surfaces that ranged as small as 0.01 mm (0.0004 in.). The problem with these standards is the difficulty of certifying the accuracy of the inner surface notches. The inner surface notches are first replicated with a low shrinkage RTV silicone rubber mixture, and the replica of the notch is then measured with a microscope. This method has given good results in the past, but is always susceptible to the rubber remaining inside the notch or not filling the notch completely. Because of these problems this type of inner surface notch could not be certified. Therefore, it was decided to split the tube, machine and certify the notches, then reassemble the tube on a nonconducting mandrel. Although the material was very fragile and several early attempts resulted in damaged standards, several standards were successfully constructed. In addition to varying the flaw size, we also varied the wall thickness of the reassembled standards so that this variation could be eliminated from

the flaw size computation, and the evaluation of flaw size would be simplified. The reassembled standards had training flaws with depths of 0.025 and 0.051 mm (0.001 and 0.002 in.) and demonstration flaws with depths of 0.008, 0.013, and 0.018 mm (0.0003, 0.0005, and 0.0007 in.). The reassembled standard did not work well in our initial tests because of the high probe power and the difference in thermal conductivity and heat capacity of the normal tubing and standards. The reassembled standard cooled the probe much more than the ordinary standards and tubing, so it did not work well as a training standard. After this problem was discovered, modifications were made in the test, such as reduced power to the probe, heat sinking of the probe tip, and removal of the teflon tape, which is normally placed between the probe and the tube as a wear surface. The probe-cooling problem also showed up in the early scans as a change in the calculated flaw value after each time the measurements were stopped to allow the printer to print. This caused a series of waves on the first two scan lines, which was corrected when we started to collect all of the data before doing any printing. The reassembled standards were still more difficult to use than the regular standards because the splits and joints produced very large signals that the plotting and data analysis programs had to ignore. This required special programs to ignore the sections of tube with these joints and to process the rest of the tubing. Because of these problems and the lack of time, the reassembled standards were used for demonstration only and not for training. This is unfortunate, because they had the potential to reduce the thickness effects and make the other data processing easier.

PULSE SHAPE AND PRESET VOLTAGE LEVELS

Since this type of eddy-current measurement had never been performed before, we had to do a certain amount of experimentation to get the best combination of pulse shape, voltage preset levels, and attenuator values to give the most accurate flaw-size calculations with the least amount of drift. The attenuator values were chosen to match the characteristic cable impedance, and this seemed to produce a stable signal that was not particularly sensitive to cable flexing or cable length. In Fig. 12 we

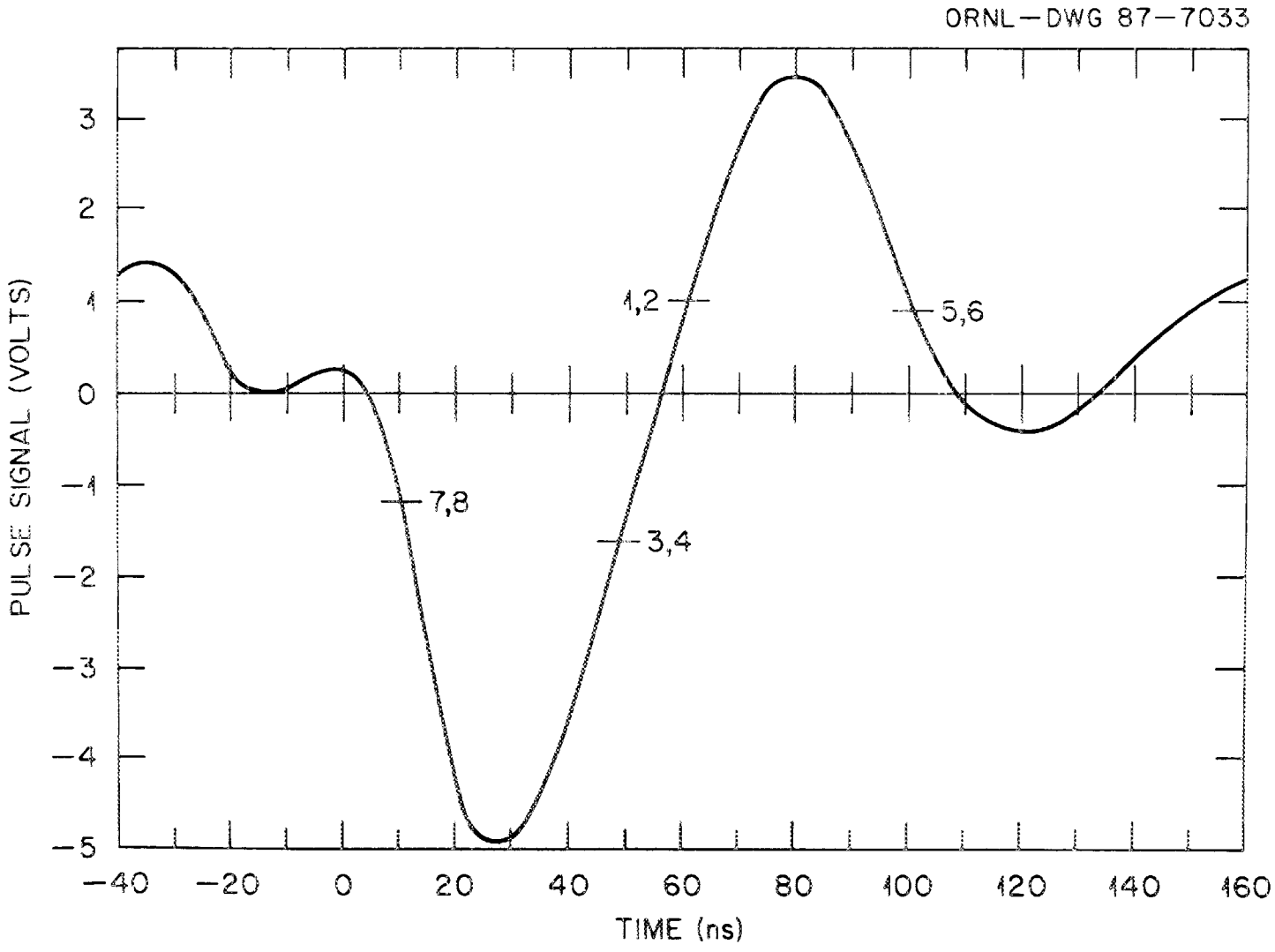


Fig. 12. Amplified voltage from the receiver coil with voltage trigger levels shown.

show the amplified signal from the receiver coil with the preset voltage trigger levels for the different data channels marked. The standard was read using the program PULRDG2, and the fits of the readings to the flaw properties are done by the program PULFIT3. These programs are listed in the appendix. The readings and fitting runs showed that we needed only four data channels to give the desired results, so we triggered two sets of channels together and then took their average. We also determined that by using the 0.076 mm (0.003 in.) flaws we were losing sensitivity and accuracy to the smaller flaws, so we dropped the larger flaws from the fits. They are readily detected but not as accurately sized, which is not important since they are much greater than the rejection limit. Only a few fitting combinations were run due to time limitations, and a more thorough fitting study using our automatic searching routines may give better results. However, the results obtained in this quick study were quite good, and much better than the impedance analyzer results. The optimum combinations gave 3.33 microns (0.131 mils) for the fit and 0.20 microns (0.0078 mils) for each digit of drift. These values were considered to be quite good, with the fit approaching the accuracy of the flaw-size measurements, particularly for inner surface flaws.

DATA DISPLAY AND STORAGE

The entire tube is scanned at a rate of about 3.8 cm (1.5 in.) per minute, with the data being transmitted over the IEEE-488 bus at a rate of 20 KHz. The channels are read by the microcomputer, the data converted to ASCII, and then all eight channels are transmitted to the IBM System 9000 Computer. The tube is moved to a new location and the next point is measured. After the entire tube is scanned, the flaw sizes are computed and plotted. The raw data are then stored on the disk as unformatted binary data. The program SSTPUF3 is used to perform this operation, and a copy is included in the appendix. In Fig. 13 we show a plot of a standard tube made by SSTPUF3. The plots may be made at either a high or a low resolution, with the high resolution taking about 4 times as long as the low resolution. The high-resolution graphics are at 200 dots per inch and

ORNL-DWG 87-1939

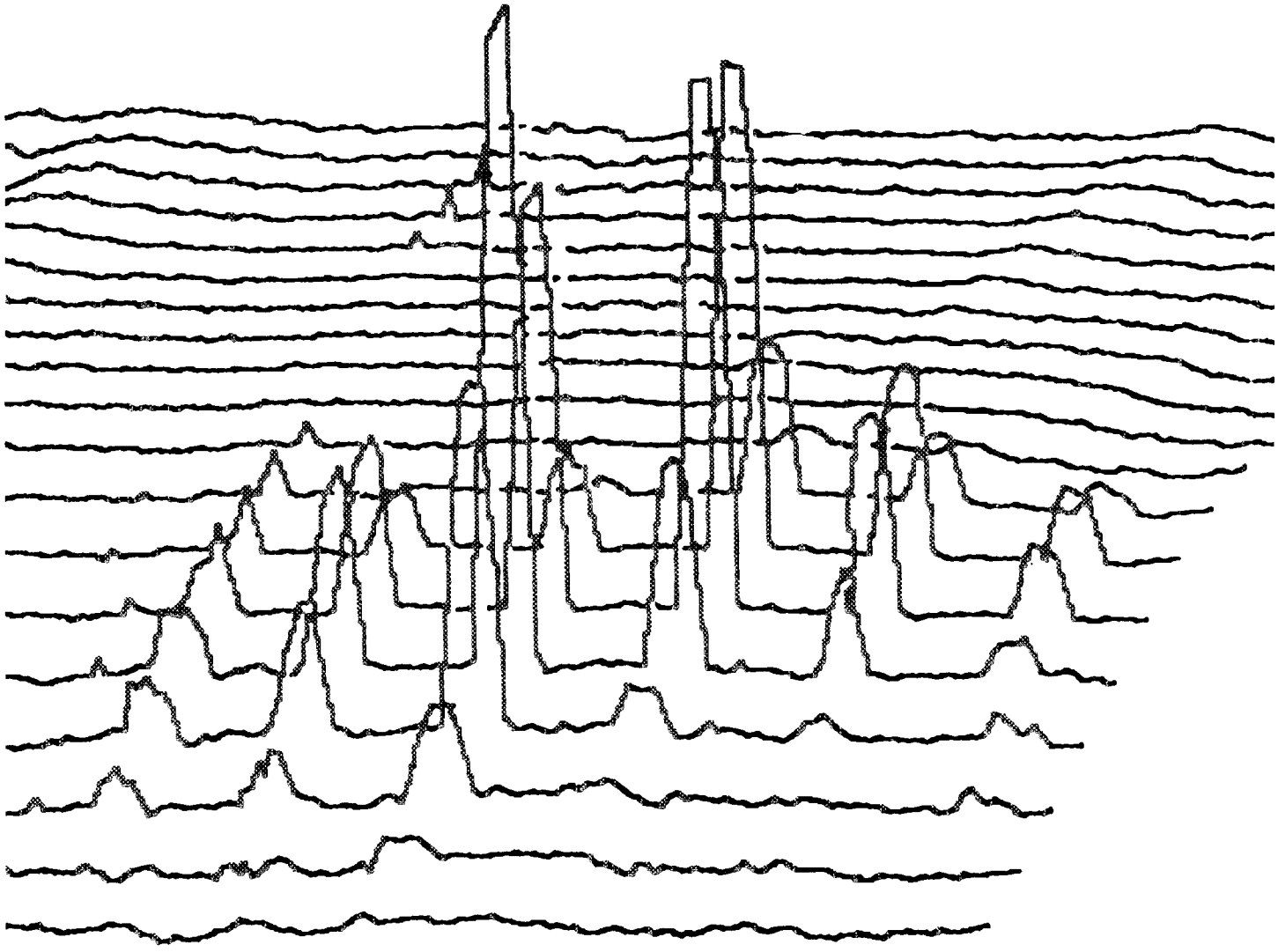


Fig. 13. Plot of defect size of a standard tube using the pulsed instrument.

give approximately a one-to-one correspondence to the tube length. The plotting time for the high-resolution graphics is 25 min for a 1.086 m (42.75 in.) tube, compared to 5.6 min for a low-resolution plot of the same length tube.

DATA EVALUATION AID PROGRAMS

Several programs were written to aid the operator in the evaluation of the data. The most helpful program is FLAWLOC3, which allows the operator to look at an individual reconstructed scan line along the tube, move a cursor along that scan line to any flaw or indication, with a display of the cursor location and the value of the flaw size computed at that location. This program is a great aid in evaluating the flaw size and location, and it uses the stored binary data files. In particular, it is needed to pinpoint the location of the flaws in a standard so that the reading program can locate and measure them. Another program, DEFTST, will analyze the data file without operator intervention. This program performs a scan along each axial reading location of the tube, and when the slope of the calculated flaw signal changes in either direction by a preset value, a flaw is recorded. All the flaw indications with the same axial location, and those with adjacent axial locations, are gathered into a single flaw, with both the location and value of the maximum flaw signal stored. This program works quite well for relatively clean tubing, but not so well for tubing with a large number of flaws. However, the tubing with many flaws will be rejected anyway. Both of these programs are included in the appendix. It would save a considerable amount of inspection time if DEFTST had been incorporated into SSTPUF3 with the graphic output eliminated, but this was not done.

SUMMARY

A high speed pulsed eddy-current inspection system has been designed, developed, and tested on a number of tubing samples. The sensitivity levels to flaws 0.015 mm (0.0006 in.) deep were obtained, along with the data analysis and storage programs. This system and technology will be very useful for future inspections of similar natures.

REFERENCES

1. 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).

2. C. V. Dodd, W. E. Deeds, and W. G. Spoeri, "Optimizing Defect Detection in Eddy Current Testing," Mater. Eval. **29**(3), 59-63 (March 1971).

3. C. V. Dodd and W. A. Simpson, "Thickness Measurements Using Eddy Current Techniques," Mater. Eval. **31**(5), 73-79 (May 1973).

4. C. V. Dodd, L. M. Whitaker, and W. E. Deeds, An Eddy-Current Laboratory Test System Using Commercial Equipment, ORNL-6366 (April 1987).

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

6. C. V. Dodd and W. E. Deeds, Eddy-Current Inspection of Ferromagnetic Tubing Using Pulsed Magnetic Saturation, ORNL-6305 (July 1986).

7. C. V. Dodd and R. F. Cowan, The NDT-COMP9 Microcomputer, NUREG/CR-1548, ORNL/NUREG/TM-390 (September 1980).

Appendix

The appendix contains a listing of the various programs used for this system, a listing of the port assignments for the COMP9B microcomputer when used to drive this instrument, the circuit diagrams for both the manually adjusted and the computer adjusted instruments, and the data on the R20F reflection probe that was used with the instruments.

PROGRAM LISTINGS

The first seven programs are Fortran programs, run on the IBM System 9000 under the 2.0 CSOS operating system. The final program, PULG, is an assembly language program, written in Intel 8080 Assembly language, loaded into a PROM in the COMP9B and on that computer.

PULRDG2

This program is used to take readings at given locations on a standard tube. The reading locations along with the tube properties are stored in a data file, INPUL3.DAT, that is read by the program. Also included in the data file is information on the coil, instrument settings, and attenuator circuit. A description of the data is given in the program PULRDG2, and the use of similar programs is given in earlier references. The readings taken by PULRDG2 are stored in another data file, OTDATA.DAT, for use by the fitting programs. A listing of the program PULRDG2 and the input data file follow.

```
      PROGRAM PULRDG2
C     25 APRIL 1986  PULRDG2 MODIFIED FOR 2.0 SOFTWARE
C
C      THIS PROGRAM READS MAGNITUDE OF THE VOLTAGE AT DIFFERENT TIMES
C     USING THE PULSED EDDY CURRENT INSTRUMENT AND MECHANICAL SCANNER.
C     THE PROGRAM READS THE POSITION DATA FROM A FILE,ALONG WITH THE
C     PROPERTIES.THE AMPLITUDE DATA ARE STORED AFTER EACH SET OF
C     PROPERTIES. THE PROPERTY AND POSITIONING DATA ARE READ FROM
C     DEVICE LID, DISK FILE.
C
C     ICMD=INTEGER COMMAND,DETERMINES TYPE OF PROGRAM THE COMPUTER RUNS
C     LI=LOGICAL INPUT TERMINAL(FROM OPERATOR)
C     LID=LOG.INPUT DEV.(ASSIGNED TO DISK FILE,POSITION,PROPERTY DATA)
```

```
C       LOD=OUTPUT DEVICE TO STORE PULSE DATA
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
C       NCL=COIL DATA FILE
C       NAXIS=NUMBER OF AXES THAT ARE POSITIONED
C       NP=PROPERTY INDEX
C       NPROPM=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+NO LIFT-OFFS
C       NT=TIME INDEX
C       NTCAL=NUMBER OF TIME CALIBRATIONS
C       NCH=TOTAL NUMBER OF TIME INTERVALS
        CHARACTER INDAT*8,OTDAT*8,FNAME*12,NPROBE*6,NCABLE*6,INSTNO*6
        CHARACTER*4 PRONAM(6),AXISNM(3)
        CHARACTER ADUM*83,FF*1,COIL*6
C       DIMENSION PROP(NPT,NPROPM),TIM(NCH)
C       DIMENSION SUMMAG(NPT,NCH),SSMAG(NPT,NCH),SDVMAG(NPT,NCH)
C       DIMENSION SUMCAL(NCH,NCAL),SSCAL(NCH,NCAL),SDVCAL(NCH,NCAL)
C       DIMENSION IVOL(NCH),IN(NCH),RDGC(NCH,NCAL),OLRDGC(NCH,NCAL)
C       DIMENSION PRONAM(NPROPM),POST(NPT,NAXIS),DIFMAG(NPT,NCH)
C
        DIMENSION PROP(72,6),TIM(8)
        DIMENSION SUMMAG(72,8),SSMAG(72,8),SDVMAG(72,8),DIFMAG(72,8)
C       DIMENSION SUMCAL(8,4),SSCAL(8,4),SDVCAL(8,4)
        DIMENSION IVOL(8),VOLSTD(8),RDGC(8,4),OLRDGC(8,4)
        DIMENSION POST(72,3),NOLD(3),XNEW(3),XHOM(3)
C
        DATA LI/0/,LID/9/,LOD/37/,LOT/0/,LPT/6/,NCH/8/,NCL/28/,MSET/3/
        DATA NRDG/100/,NTCAL/4/,INDAT/'INPUL3'/,OTDAT/'OTPUL'/
        DATA NAXIS/2/,NOLD/3*0/,XHOM/3*0./,NPROPM/3/,MPEC/45/,MBUS/46/
        DATA NPT/36/,NLT/3/
        NPTL=NPT/NLT
        FF=CHAR(12)
        OPEN(LPT,FILE='#PR')
        OPEN(NCL,FILE='REF.DAT')
C       MULTI PULSE EDDY CURRENT INSTRUMENT ADDR=5 ON GPIB BUSS
C       ADDR=E? ON IBM BUSS DRIVER, FORTRAN LOGICAL UNIT = 45
        OPEN(MPEC,FILE='#BUSE?')
C       SET FOR DMA WITH MORE THAN 4 CHARACTERS READ OR WRITE
        CALL _SETDMA(MPEC,2)
        CALL _STIMEOUT(MPEC,800)
        WRITE(0,*)'FILE OPENED FOR PULSED I/O OVER IEEE-488 BUS'
   25   CALL SETUP(MPEC,NCH,NRDG,IER)
        IF(IER.NE.0)READ(0,913)ADUM
        IF(IER.NE.0)GO TO 25
C       SET THE TIMEOUT ON PULSE RDGS FOR 300 MILLISECONDS
C       MODULYNX AXIS POSITION CONTROLLER ADDR=6 ON GPIB BUSS,ADDR=F? ON
C       IBM BUSS DRIVER,FORTRAN LOGICAL UNIT = 46
        OPEN(MBUS,FILE='#BUSF?')
        CALL _STIMEOUT(MBUS,800)
```

```
      CALL _SETEOS(MBUS,$00,$0A)
      CALL _SETOPTIONS(MBUS,0,0,0,0,0,1,0)
      WRITE(MBUS,*)'XA6000B340H2600'
      WRITE(MBUS,*)'YA40000B600H3600'
      WRITE(MBUS,*)'XD YD ZE'
      WRITE(MBUS,*)'<CAH>'
C
C     PRINT TITLE AND DATE
      CALL _GETTIME(IYR,IMO,IDA,IHR,IMN,ISE,IFR)
    1 WRITE(LPT,2,ERR=1990)IHR,IMN,ISE,IMO,IDA,IYR
    2 FORMAT(' PULRDG2    TIME ',I2,':',I2,':',I2,'  DATE ',I2,
     *'/',I2,'/',I2)
   30 FORMAT(1X)
C
C     OPEN FILE FOR INSTRUMENT AND PROPERTY DATA - ASSUMED LOCATION ON DEFAULT
        DISK
      FNAME=INDAT//'.DAT'
      OPEN(LID,FILE=FNAME,STATUS ='OLD')
C
C     OPEN FILE FOR OUTPUT TO STORE DATA TAKEN - LOCATE ON DEFAULT DISK
C
      FNAME=OTDAT//'.DAT'
      OPEN(LOD,FILE=FNAME,STATUS ='NEW')
C
C     INPUT DESCRIPTION OF EXPERIMENTAL APPARATUS
C
      WRITE(LOT,60)
60    FORMAT(' PROBE #:  ')
      READ(LID,70)NPROBE
70    FORMAT(A6)
      WRITE(LOT,80)
80    FORMAT(' SERIAL #:  ')
      READ(LID,*)NSER,RDVR
      WRITE(LOT,90)
90    FORMAT(' DRIVER SERIES RESISTANCE:  ')
      READ(LID,*)R0
      WRITE(LOT,100)
100   FORMAT(' DRIVER SHUNT CAP:  ')
      READ(LID,*)CAPDR
      WRITE(LOT,110)
110   FORMAT(' PICK-UP SHUNT RESISTANCE:  ')
      READ(LID,*)R9,RSER
      WRITE(LOT,120)
120   FORMAT(' PICK-UP SHUNT CAP:  ')
      READ(LID,*)CAPPU
      WRITE(LOT,130)
130   FORMAT(' CABLE I.D. #:  ')
      READ(LID,70)NCABLE
      WRITE(LOT,140)
140   FORMAT(' LENGTH OF CABLE:  ')
      READ(LID,*)CABLEL
      WRITE(LOT,150)
```

```
150   FORMAT(' CAPACITANCE OF CABLE:  ')
      READ(LID,*)CCABLE
    WRITE(LOT,160)
160   FORMAT(' EDDY CURRENT INSTRUMENT #:  ')
      READ(LID,70) INSTNO
C
C INPUT TIME VALUES
C
       WRITE(LOT,190)
  190 FORMAT(' NO. OF TIME VALUES,PROP SETS,PULSE TIME,RDG TIM,PULSE ',
     *'VOLTS:')
      READ(LID,*)NCH,NPT,ICMD,PVOLT
C     WRITE(LOT,200)
200   FORMAT(' INPUT EACH VALUE OF TIME SEPARATED BY A SPACE:',/)
C     READ(LID,*)(TIM(NT),NT=1,NCH)
C
      READ(LID,210)(PRONAM(NPR),NPR=1,NPROPM),(AXISNM(NAX),NAX=1,NAXIS)
210   FORMAT(10(A4,1X))
C
C READ PROPERTY VALUES AND LOCATION DATA FOR THE POSITIONERS FROM LID
C
      DO 300 NP=1,NPT
      READ(LID,*)(PROP(NP,NPR),NPR=1,NPROPM),(POST(NP,NAX),NAX=1,NAXIS)
      WRITE(LOT,270)(PROP(NP,NPR),NPR=1,NPROPM),(POST(NP,NAX),NAX=1,
     *NAXIS)
270   FORMAT(10(F7.3))
300   CONTINUE
C
C
C STORE INITIAL INFORMATION IN DISK FILE
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
 1275 FORMAT(A6,1X,I5,4(E13.4),1X,A6,2(E13.4),1X,A6)
      WRITE(LOD,1277) NCH,NPT,NPROPM,NTCAL ,ICMD,PVOUT
 1277 FORMAT(5(1X,I5),1F6.0)
C     WRITE(LOD,1279)(TIM(NT),NT=1,NCH)
 1279 FORMAT(8(1PE10.3))
      WRITE(LOD,1280)(PRONAM(NPR),NPR=1,NPROPM)
C     THIS SECTION TAKES THE CALIBRATION READINGS.
C
C
C END OF SECTION WHICH TAKES CALIBRATION READINGS
C
C TAKE READINGS ON NOMINAL STANDARD FOR VOLSTD(I) READINGS
C
C SET ITIM1 TO CORRECT VALUE BEFORE STARTING TAKING READINGS
C ZERO ARRAYS THAT WILL CONTAIN SUMS OF THE VOLTAGE MAG READINGS,
C     SUMS OF CALIBRATION READINGS, AND SUMS OF SQUARES OF EACH
C
```

```
 790  CONTINUE
      DO 800 NT=1,NCH
      DO 800 NP=1,NPT
      SUMMAG(NP,NT)=0.
      SSMAG(NP,NT)=0.
800   CONTINUE
C      DO 810 NT=1,NCH
C      DO 810 NC=1,NTCAL
C      SUMCAL(NT,NC)=RDGC(NT,NC)
C      SSCAL(NT,NC)=RDGC(NT,NC)*RDGC(NT,NC)
 810   CONTINUE
C
C THIS SECTION TAKES THE ACTUAL MAGNITUDE DATA READINGS
C
      DO 1150 MSE=1,MSET
      DO 980 NP=1,NPT
      WRITE(LOT,900)NP,(PROP(NP,NPR),NPR=1,NPROPM),(POST(NP,NAX),NAX=1,
     *NAXIS)
900   FORMAT(I4,10(F8.4))
C
C     POSITION SAMPLES BEFORE TAKING READINGS
C
      DO 905 NAX=1,NAXIS
      XNEW(NAX)=POST(NP,NAX)
 905  CONTINUE
      CALL POSIT(XNEW,NOLD,NAXIS,MBUS)
      IF(MOD((NP-1),NPTL).NE.0)GO TO 910
 914  WRITE(0,*)'INSERT LIFT-OFF SHIM MANUALLY-HIT ANY KEY WHEN READY'
 915  CALL _GETKEY(IKY,IS,KBF1,KBF2,IRR)
      IF(IRR.NE.0)GO TO 915
 910  CALL PRDAVG(IVOL,MPEC,NCH,ICMD,NRDG,IER)
      IF(IER.NE.0)READ(0,913)ADUM
 913  FORMAT(A2)
      IF(IER.NE.0)GO TO 910
      WRITE(LOT,950)(IVOL(NC),NC=1,NCH)
 950  FORMAT(8I5)
C     TEMPORARY TRANSFER AROUND TO USE WITHOUT AUTO POSITIONER
C     IRR=0
C     CALL GETKEY(IKY,IS,KBF1,KBF2,IRR)
C     IF(IRR.NE.0)GO TO 910
      DO 980 NT=1,NCH
      SUMMAG(NP,NT)=SUMMAG(NP,NT)+IVOL(NT)
      SSMAG(NP,NT)=SSMAG(NP,NT)+IVOL(NT)*IVOL(NT)
980   CONTINUE
1150  CONTINUE
C
C
C     RETURN POSITIONER TO XNEW(II)=0.0 SO THAT POSITIONS CAN BE CHECKED
C     TURN OFF MOTOR CURRENT
C
C     WRITE(MBUS,*)'<CAH>'
C
C     CALL POSIT(XHOM,NOLD,NAXIS,MBUS)
```

```
C     WRITE(MBUS,*)'XE YE'
C
C      BEFORE STOPPING CALCULATE AVERAGES & STANDARD DEVIATIONS
C
C     DO 1250 NT=1,NCH
C     DO 1250 NC=1,NTCAL
C     SUMCAL(NT,NC)=SUMCAL(NT,NC)/(FLOAT(MSET+1))
C     SDVCAL(NT,NC)=SQRT(ABS((SSCAL(NT,NC)-SUMCAL(NT,NC)*SUMCAL(NT,NC)
C    *     *FLOAT(MSET+1))/FLOAT(MSET)))
 1250 CONTINUE
C
C     CALCULATE AVERAGES OF THE READINGS
C
      DO 1260 NT=1,NCH
      DO 1260 NP=1,NPT
      SUMMAG(NP,NT)=SUMMAG(NP,NT)/FLOAT(MSET)
      SDVMAG(NP,NT)=SQRT(ABS((SSMAG(NP,NT)-SUMMAG(NP,NT)*SUMMAG(NP,NT)
     *     *FLOAT(MSET))/FLOAT(MSET-1)))
      IF(NP.GT.1) DIFMAG(NP-1,NT)=SUMMAG(NP,NT)-SUMMAG(NP-1,NT)
1260  CONTINUE
C
C STORE ALL DATA TAKEN IN DISK FILE
C
 1280 FORMAT(7(1X,A4))
C     DO 1295 NC=1,NTCAL
C     WRITE(LOD,1290)(SUMCAL(NT,NC),NT=1,NCH)
 1290 FORMAT(8(F7.4))
1295  CONTINUE
      DO 1300 NP=1,NPT
      WRITE(LOD,1297)(SUMMAG(NP,NT),NT=1,NCH),(PROP(NP,NPR),NPR=1,
     *NPROPM)
1297  FORMAT(8(F7.1),3(F5.2),F5.0)
C1297  FORMAT(5(F7.3),3(F5.3),F5.0)
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,RDVR
1350  FORMAT(' PROBE NO.:',A6,5X,' SERIAL NO.:',I5,' D.C. RES',F8.3 )
      WRITE(LPT,1360) R0,CAPDR
1360  FORMAT(' DRIVER SERIES RESISTANCE:',F10.3,5X,'DRIVER SHUNT CAP.:',
     *      E12.4)
      WRITE(LPT,1370) R9,RSER,CAPPU
1370  FORMAT(' PICK-UP  RES: SHUNT',F9.1,'SERIES',F9.1,'  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,NRDG
1390  FORMAT(' EDDY CURRENT INSTRUMENT NO.:',A6,'  NRDG ',I5)
      PWR=PVOLT*PVOLT/(2.0*R0)
      WRITE(LPT,1395)PVOLT,PWR,ICMD
```

```
1395  FORMAT('PULSE:ONTIME=       OFFTIME=      VOLTS= ',F7.3,
     *' POWER(KW)',F6.1,' COMMAND=',I2)
      WRITE(LPT,30)
C
C     PRINT CALIBRATION READINGS
C
1600  CONTINUE
CC    WRITE(LPT,1650) NTCAL
 1650  FORMAT('AVERAGES & STANDARD DEVIATIONS OF CALIBRATION RDGS:',I3,
     *' MAGS')
CC     WRITE(LPT,1660)(TIM(NT),NT=1,NCH)
 1660  FORMAT('TIMES',8(1PE9.2))
       DO 1700 NC=1,NTCAL
CC     WRITE(LPT,1680)((SUMCAL(NT,NC)),NT=1,NCH)
 1680  FORMAT(5X,8(F9.4))
CC     WRITE(LPT,1690)((SDVCAL(NT,NC)),NT=1,NCH)
 1690  FORMAT(' S.D.',8(F9.4))
 1700  CONTINUE
CC     WRITE(LPT,30)
C
C     READINGS FROM NOMINAL SAMPLE
C
C     WRITE(LPT,1745)
1745  FORMAT(' READINGS FROM NOMINAL TUBE SAMPLE')
C     WRITE(LPT,1680)(VOLSTD(NT),NT=1,NCH)
C     PRINT OUT READINGS AND PROPERTIES
C
      WRITE(LPT,1750)(NT,NT=1,NCH),(PRONAM(NPR),NPR=1,NPROPM)
 1750  FORMAT('PSET',8('  MAG',I1,1X),6(1X,A4))
C1750  FORMAT('PSET',5('  MAG',I1,1X),6(1X,A4))
      DO 1800 NP=1,NPT
      WRITE(LPT,1760)NP,(SUMMAG(NP,NT),NT=1,NCH),(PROP(NP,NPR),NPR=1,
     *NPROPM)
      WRITE(LPT,1765)(SDVMAG(NP,NT),NT=1,NCH)
C     IF(NP.LT.NPT) WRITE(LPT,1766)(DIFMAG(NP,NT),NT=1,NCH)
      WRITE(LPT,30)
1760  FORMAT(I3,8(F7.1),3(F5.2),F5.0)
1765  FORMAT(' SD',8(F7.1))
1766  FORMAT('DIF',8(F7.1))
1800  CONTINUE
      WRITE(LPT,1810)FF
1810  FORMAT(1A)
C     GO TO 2000
C
C     RETURN POSITIONER TO XNEW(II)=0.0 SO THAT POSITIONS CAN BE CHECKED
C     TURN OFF MOTOR CURRENT
C
C     WRITE(MBUS,*)'<CAH>'
C
1900  CALL POSIT(XHOM,NOLD,NAXIS,MBUS)
C     WRITE(MBUS,*)'XE YE ZE'
      GO TO 2000
 1990 WRITE(LOT,*)'ERROR -CHECK FOR PRINTER OFF-LINE HIT CR TO CONTINUE'
```

```
      READ(0,913)ADUM
      GO TO 1
2000  STOP
      END
C
      SUBROUTINE POSIT(XNEW,NOLD,NAXIS,MBUS)
C
C     VERSION 6 JULY 1984
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 PULSES 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.THE EDDY
C     CURRENT READINGS WILL NOT BE MADE UNTIL THE POSITION IS CORRECT.
C
      DIMENSION NOLD(3),XNEW(3),NSTEPS(3)
      CHARACTER*2 AXC(3),GC
      DATA AXC/'XM','YM','ZM'/,GC/'G '/,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(MBUS,30)(AXC(IAXIS),NSTEPS(IAXIS),GC,IAXIS=1,NAXIS)
      WRITE(MBUS,*)';<CFA>'
      WRITE(MBUS,*)';'
   30 FORMAT(4(A2,I7,A2))
      RETURN
      END
      SUBROUTINE PRDG(IN,MPEC,NCH,ICMD,IER)
C     PROGRAM TO READ PULSE DATA OVER THE GPIB BUSS NO AVERAGING
C     CMD0   SET INSTRUMENT UP
C     CMD1   FAST READING, MADE ON PULSE TURN ON,RDG THEN TRANSMIT.
C     CMD2   SAME AS CMD1 EXCEPT TRANSMITS OLD DATA WHILE NEW RDGS MADE
C     CMD3   USED FOR CALIBRATIONS
C     CMD4   SLOWER READINGS,USES THE SUBROUTINES THAT SUPPORT AVERAGING
      CHARACTER*1 CR
      DIMENSION IN(8)
      CR=CHAR(13)
      IER=0
      WRITE(MPEC,35,ERR=400)ICMD,CR
      READ(MPEC,70,ERR=500)(IN(I),I=1,NCH)
   35 FORMAT(I1,A1)
   70 FORMAT(8I4)
      GO TO 600
  400 IER=1
      WRITE(0,*)'BUSS WRITE ERROR TO PULSER;HIT RESET,CR AND CONTINUE'
      GO TO 600
  500 IER=2
      WRITE(0,*)'BUSS WRITE ERROR TO PULSER;HIT RESET,CR AND CONTINUE'
  600 RETURN
      END
      SUBROUTINE PRDAVG(IN,MPEC,NCH,ICMD,NRDG,IER)
```

```
C     SUBROUTINE AVERAGES NRDG PULSE READINGS
      DIMENSION IN(8),INSUM(8)
      CALL PRDG(INSUM,MPEC,NCH,ICMD,IER)
      DO 50 NR=2,NRDG
      CALL PRDG(IN,MPEC,NCH,ICMD,IER)
      DO 40 NC=1,NCH
      INSUM(NC)=INSUM(NC)+IN(NC)
   40 CONTINUE
   50 CONTINUE
      DO 100 NC=1,NCH
      IN(NC)=INSUM(NC)/NRDG
  100 CONTINUE
      RETURN
      END
      SUBROUTINE SETUP(MPEC,NCH,NRDG,IER)
C     PROGRAM TO SET UP PULSE INSTRUMENT OVER THE GPIB BUSS
      CHARACTER*1 CR,C,O,P,D,R,A,L,B,G
      DATA ICMD/0/,C/'C'/,O/'O'/,P/'P'/,D/'D'/,R/'R'/,A/'A'/,L/'L'/,
     *B/'B'/,G/'G'/
      CR=CHAR(13)
      IER=0
      NRH=NRDG/256
      NRM=(NRDG-NRH*256)/16
      NRL=NRDG-NRH*256-16*NRM
      WRITE(0,35,ERR=400)ICMD,C,NCH,CR
      WRITE(MPEC,35,ERR=400)ICMD,C,NCH,CR
C     WRITE(MPEC,40,ERR=400)ICMD,A,NRH,NRM,NRL,CR
C     WRITE(0,40,ERR=400)ICMD,A,NRH,NRM,NRL,CR
   35 FORMAT(I1,A1,I1,A1)
   40 FORMAT(I1,A1,Z1,Z1,Z1,A1)
      GO TO 600
  400 IER=1
      WRITE(0,*)'BUSS WRITE ERROR,COMMAND TO MPEC,HIT RESET & TRY AGAIN'
  600 RETURN
      END
```

LISTING OF INPUL3.DAT

```
R20F
54717 0.6159
20.0
0.
50. 50.
0.0
CAB3
36.
120.E-12
PULG1
8,36,1,10.
```

THIK	DEF	LOFF	XAXS	THAS
5.00	0.00	0.000	4.631	.020
5.00	0.90	0.000	4.631	.0344
5.00	1.90	0.000	5.106	.0344
5.00	0.00	0.000	5.106	.020
5.00	3.10	0.000	5.581	.0344
5.00	0.00	0.000	5.581	.020
5.00	3.20	0.000	6.080	.0363
5.00	0.00	0.000	6.080	.020
5.00	2.10	0.000	6.570	.0363
5.00	0.00	0.000	6.570	.020
5.00	1.10	0.000	7.077	.0363
5.00	0.00	0.000	7.077	.020
5.00	0.00	0.002	4.631	.020
5.00	0.90	0.002	4.631	.0344
5.00	1.90	0.002	5.106	.0344
5.00	0.00	0.002	5.106	.020
5.00	3.10	0.002	5.581	.0344
5.00	0.00	0.002	5.581	.020
5.00	3.20	0.002	6.080	.0363
5.00	0.00	0.002	6.080	.020
5.00	2.10	0.002	6.570	.0363
5.00	0.00	0.002	6.570	.020
5.00	1.10	0.002	7.077	.0363
5.00	0.00	0.002	7.077	.020
5.00	0.00	0.004	4.631	.020
5.00	0.90	0.004	4.631	.0344
5.00	1.90	0.004	5.106	.0344
5.00	0.00	0.004	5.106	.020
5.00	3.10	0.004	5.581	.0344
5.00	0.00	0.004	5.581	.020
5.00	3.20	0.004	6.080	.0363
5.00	0.00	0.004	6.080	.020
5.00	2.10	0.004	6.570	.0363
5.00	0.00	0.004	6.570	.020
5.00	1.10	0.004	7.077	.0363
5.00	0.00	0.004	7.077	.020

PULFIT3

This program is used to fit the readings taken with PULRDG2 to the properties of the standard that produced the properties. Both of these are included in the data file, OTDATA.DAT, produced by PULRDG2 and now read by PULFIT3. The arrays in PULFIT3 must match the data in the data file, as noted in the earlier references on running this type of program. The operator can assume different combinations of the readings and determine

how well these combinations of readings will fit the test properties (mostly flaw size in our particular case). The coefficients to convert the readings into the properties are saved on another data file, PULCOE.DAT. When the best fit has been determined, the coefficients can be incorporated into the programs that do the property calculation from the actual test data. A listing of the program PULFIT3 follows.

```
      PROGRAM PULFIT3
C     VERSION 17 MARCH 1986 PULFIT2.FOR MODIFIED FOR 2.0 SOFTWARE
C     PROGRAM TO PERFORM A LEAST SQUARES FIT TO DATA READ INTO A DISK
C     FILE BY PULRDG2 AND THEN PERFORM CONTINUOUS BACK CALCULATIONS
C     USING INSTRUMENT READINGS MADE BETWEEN EACH DISPLAYED SET.
C
C     IRDPRM=MAXIMUM NUMBER OF COEFFICIENTS IN EXPANSION
C     LITEK=LOGICAL INPUT UNIT
C     LOTEK=LSI OUTPUT UNIT FOR PROMPTING AND DISPLAY
C     LOU=LOGICAL OUTPUT UNIT FOR PERMANENT RECORD (PRINTER)
C     NCHS=NUMBER OF DATA CHANNELS (TIME INTERVALS)
C     NT=TIME INDEX
C     NTT=NUMBER OF TIMES
C     NP=PROPERTY INDEX
C     NPROPM=MAXIMUM NUMBER OF PROPERTIES CALCULATED(=6 NOW)
C     NPT=TOTAL NUMBER OF SETS OF PROPERTY  VALUES THAT WILL BE
C     CONSIDERED. EACH SET OF PROPERTY VALUES IS ON ONE LINE
C     IN THE DATA FILE
C     NPTT=NUMBER OF DATA LINES IN THE INPUT DATA FILE
C     NTCAL=NUMBER OF TIME CALIBRATIONS
C     NPRINT=PRINT AND TRANSFER INDEX.
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,COIL
      CHARACTER COEDAT*8,FNAME*12,BLANKS*4,PRONAM(7)*4,AXISNM(3)*4
      CHARACTER PROPTY(7)*4,STOP*4,INDAT*8,PHASW(3)*4,POLARY(16,5)*4
      CHARACTER ADUM*82,CR*1
C
C  DIMENSIONS THAT ARE CHANGED:
C **  DIMENSION READNG(NPT+1,IRDPRM+1),PRO(NPT)
C     DIMENSION PROP(1,NPROPM),TMDFT(NTT),VLDFT(NTT),JPOL(2,NTT,NPROPM)
C     DIMENSION COE(IRDPRM),COEF(IRDPRM,NPROPM),TIM(NTT)
C     DIMENSION RDG1(1,IRDPRM),NPOL(2,NTT)
C     DIMENSION JOFSET(NPROPM),JRDPR(NPROPM)
C     DIMENSION PROPTY(NPROPM),PRONAM(NPROPM)
C     DIMENSION TMAG1(1,NTT)
C     DIMENSION SUMCAL(NTT,NCAL),SDVCAL(NTT,NCAL)
C     DIMENSION RDGC(NTT,NCAL),RDGO(NTT,NCAL),COFSET(NTT),CGAIN(NTT)
C     DIMENSION VOLTS(NTT),VOLSTD(NTT),STDV(NTT),NCONV(NTT)
C     DIMENSION TOFSET(NTT),TSLOPE(NTT)
C
```

```
C  THE APPROPRIATE NUMBERS SHOULD BE INSERTED IN THE FOLLOWING
C  DIMENSION STATEMENTS; COMMENTED STATEMENTS MARKED ** MANDATORY
C
      DIMENSION TMDFT(8),VLDFT(8),JPOL(2,8,7),COE(15),COEF(15,7),TIM(8)
      DIMENSION RDG1(1,15),NPOL(2,8),JOFSET(7),JRDPR(7)
      DIMENSION TMAG1(1,8),IN(8)
      DIMENSION RDGC(8,4),RDGO(8,4),COFSET(8),CGAIN(8)
      DIMENSION VOLTS(8),VOLSTD(8),STDV(8),NCONV(8)
      DIMENSION READNG(37,16),PRO(72),PROP(1,7)
C  DATA THAT MAY NEED TO BE CHANGED:
      DATA NPT/36/,NPRINT/0/,LOU/6/,LITEK/0/,LOTEK/0/
      DATA NCL/28/,LID/37/,LOD/38/,MPEC/45/,MBUS/46/,MAG/0/
      DATA IRDPRM/15/,IR/1/,NCH/8/,NRDG/32/,NPROPM/7/,NPROPT/1/
      DATA STOP/'STOP'/,BLANKS/'    '/,NUNIT/1/,ICMD/1/
      DATA NLINES/4/,INDAT/'OTPUL'/,COEDAT/'PULCOE  '/
      DATA COFSET/8*0.0/,CGAIN/8*1./
C
C PRINT TITLE AND DATE
      OPEN(LOU,FILE='#PR',ERR=1990)
      CALL _GETTIME(IYR,IMO,IDA,IHR,IMN,ISE,IFR)
    1 WRITE(LOU,2,ERR=1990)IHR,IMN,ISE,IMO,IDA,IYR,INDAT
    2 FORMAT('PULFIT2 TIME ',I2,':',I2,':',I2,'  DATE ',I2,'/',I2,'/'
     *I2,4X,A8)
C
   30 FORMAT(1X)
C
C     OPEN FILE FOR INPUT DATA FROM PULRDG, ASSUMED ON DEFAULT DISK
   40 FORMAT(A8)
      FNAME=INDAT//'.DAT'
   41 FORMAT(A12)
      OPEN(LID,FILE=FNAME,STATUS ='OLD',ERR=991)
C
C     MULTI PULSED EDDY CURRENT INSTRUMENT ADDR=5 ON GPIB BUSS
C     ADDR=E? ON IBM BUSS DRIVER, FORTRAN LOGICAL UNIT = 45
      OPEN(MPEC,FILE='#BUSE?',ERR=993)
C     SET FOR DMA WITH MORE THAN 4 CHARACTERS READ OR WRITE
      CALL _SETDMA(MPEC,2)
      CALL _STIMEOUT(MPEC,10)
      WRITE(0,*)'FILE OPENED FOR PULSED I/O OVER IEEE-488 BUS'
   25 CALL SETUP(MPEC,NCH,NRDG,IER)
      IF(IER.NE.0)READ(0,1991)ADUM
      IF(IER.NE.0)GO TO 25
C
C READ INITIAL INFORMATION IN DIRECT ACCESS FILE LID ON DISK
C
       READ(LID,55)IHR,IMN,ISE,IMO,IDA,IYR
   55 FORMAT(6(1X,I4))
      WRITE(LOTEK,55)IHR,IMN,ISE,IMO,IDA,IYR
      READ(LID,60) NPROBE,NSER,RO,CAPDR,R9,CAPPU,NCABLE,
     *              CABLEL,CCABLE,INSTNO
   60 FORMAT(A6,1X,I5,4(E13.4),1X,A6,2(E13.4),1X,A6)
      WRITE(LOTEK,60) NPROBE,NSER,RO,CAPDR,R9,CAPPU,NCABLE,
```

```
     *                CABLEL,CCABLE,INSTNO
      READ(LID,65)     NTT,NPTT,NPROPM,NTCAL,ICMD,PVOLT
  65  FORMAT(5(1X,I5),1F6.0)
      IF(NTT.NE.NCH)WRITE(0,*)'ERROR IN CHANNEL(NCH) AND TIME INTERVALS'
     *,'(NTT)')
      WRITE(LOTEK,65) NCH,NPT,NPROPM,NTCAL,ICMD,PVOLT
C     READ(LID,70)(TIM(NT),NT=1,NTT)
C     WRITE(LOTEK,70)(TIM(NT),NT=1,NTT)
  70  FORMAT(8(1PE10.3))
      READ(LID,80)(PRONAM(NPR),NPR=1,NPROPM)
      WRITE(LOTEK,80)(PRONAM(NPR),NPR=1,NPROPM)
  80  FORMAT(7(1X,A4))
C     DO 100 NC=1,NTCAL
C     READ(LID,85)(RDCO(NT,NC),NT=1,NTT)
  85  FORMAT(8(F7.4))
  100 CONTINUE
C     READ NOMINAL  STANDARD VOLTAGES
C
C     READ(LID,85)(VOLSTD(NT),NT=1,NTT)
C     WRITE(LOTEK,85)(VOLSTD(NT),NT=1,NTT)
C
C END OF SECTION WHICH READS INITIAL DIRECT ACCESS FILE
C
C      THE DATE AND TIME THE DATA WERE TAKEN ARE PRINTED
C
      WRITE(LOU,140)IHR,IMN,ISE,IMO,IDA,IYR
  140 FORMAT(' CALIBRATION DATA TAKEN ',I2,':',I2,':',I2,'  DATE ',I2,
     *'/',I2,'/',I2)
      WRITE(LOU,*)'DATA FILE ',INDAT,' COIL ',NPROBE,'  SERIAL NO ',NSER
  160 FORMAT(1X)
C
C     COIL DATA ARE READ FROM REF.DAT OR CIR.DAT FILE
C
C     FILE TO INSERT INTO REFLECTION OR CIRCULAR PROGRAMS FOR COIL
C     DATABASE REFERENCES
C
C    THE INPUT DATA FOR THE REFLECTION COIL ARE READ FROM FILE 28
C    SEE FIG.2,P.7 AND FIG.4, P.7, ORNL-TM-4107, FOR DEFINITIONS.
C
C     OPEN(28,FILE='REF.DAT',STATUS ='OLD',ERR=994)
C  10 READ(28,11,ERR=996)COIL,RBAR,R1,R2,L3,R3,R4,L4,L5,L6
C    *,RDCDR,RDCPU,TNDR,TNPU
   11 FORMAT(A6,9F8.4,F10.4,F11.4,2F8.1)
C     IF(COIL.EQ.'END   ')WRITE(0,*)'REF COIL NOT FOUND'
C     IF(COIL.EQ.'END   ')GO TO 15
C     IF(COIL.NE.NPROBE)GO TO 10
C     GO TO 630
C
C     THE INPUT DATA FOR CIRCULAR COILS ARE READ FROM FILE 29
C     SEE TABLE 3,PAGE 10 OF ORNL/NUREG/TM-335 FOR DEFINITIONS
C
C  15 OPEN(29,FILE='CIR.DAT',STATUS ='OLD',ERR=995)
```

```
C     READ(29,20,ERR=997)COIL,RBAR,R1,R2,XL,ZL,RLIM,RDCDR
C    *,RDCPU,TNDR,TNPU,ZLDR,ZLPU,X
   20 FORMAT(A6,6F8.4,2F10.4,2F8.1,2E11.3,F8.1)
C     READ(29,20,ERR=996)COIL,RBAR,R1,R2,L3,R3,R4
C     IF(COIL.EQ.'END   ')WRITE(0,*)' CIR COIL NOT FOUND'
C     IF(COIL.EQ.'END   ')GO TO 300
C     IF(COIL.NE.NPROBE)GO TO 15
      GO TO 630
C
C     LEAST SQUARES DESIGN SECTION.
C
C     SELECT PROPERTY TO BE FITTED AND SET UP PROPERTY ARRAY.
C
  300 MSET=NPT
      MSET1=MSET+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,*)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,*) JOFSET(NPROPT)
      IOFSET=JOFSET(NPROPT)
      IRDPR=IOFSET
  370 WRITE(LOTEK,380)
  380 FORMAT(' SELECT THE NUMBER OF THE FUNCTION TYPE & POLYNOMIAL',
     *' DEGREE, FOR EACH TIME VALUE')
      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 '/,25X'TYPE DEG ')
      DO 450 NT=1,NTT
      WRITE(LOTEK,420) NT
  420 FORMAT(' TIM =  ',I12,' SEC  ',\)
      READ(LITEK,*)JPOL(1,NT,NPROPT),JPOL(2,NT,NPROPT)
      IRDPR=IRDPR+JPOL(2,NT,NPROPT)
      JRDPR(NPROPT)=IRDPR
      NPOL(1,NT)=JPOL(1,NT,NPROPT)
      NPOL(2,NT)=JPOL(2,NT,NPROPT)
  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 POLPUL(POLARY,JROW,IRDPR,NPOL,NTT,IOFSET,LOTEK)
C
C      EXPAND THE RAW READINGS INTO IRDPR READINGS.
C
```

```
  470 DO 480 NT=1,NTT
C     SET THE TIME AND VOLTAGE DRIFTS TO ZERO INITIALLY
      TMDFT(NT)=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,482)(TMAG1(1,NT),NT=1,NTT),(PROP(1,NPR),NPR=1,NPROPM)
      TMAG1(1,1)=TMAG1(1,1)+TMAG1(1,2)
      TMAG1(1,3)=TMAG1(1,3)+TMAG1(1,4)
      TMAG1(1,5)=TMAG1(1,5)+TMAG1(1,6)
      TMAG1(1,7)=TMAG1(1,7)+TMAG1(1,8)
 482  FORMAT(8(F7.1),3(F5.2),F5.0)
C482  FORMAT(14(1X,F9.5))
C
C     THE PROPERTIES CAN BE SET AND MODIFIED IN THIS SECTION
C
C
C     IF THIS PROPERTY IS NOT TO BE USED TRANSFER TO 490
C
C     IF(PROP(1,1).NE.5.1)GO TO 490
C     IF(PROP(1,2).GT.2.200)GO TO 490
      PRO(NR)=PROP(1,NPROP)
      CALL RDGPUL(RDG1,TMAG1,NPOL,IOFSET,TMDFT,1,IRDPR1,1,NTT,1,1,1,1)
      DO 485 IRD=1,IRDPRM
      READNG(NR,IRD)=RDG1(1,IRD)
 485  CONTINUE
      NR=NR+1
  490 CONTINUE
C
C     DO THE LEAST SQUARES FIT OF THE READINGS TO THE PROPERTIES.
C
      CALL ALSQS(READNG,PRO,COE,RSOS,MSET,IRDPR,MSET1)
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,482)(TMAG1(1,NT),NT=1,NTT),(PROP(1,NPR),NPR=1,NPROPM)
      TMAG1(1,1)=TMAG1(1,1)+TMAG1(1,2)
      TMAG1(1,3)=TMAG1(1,3)+TMAG1(1,4)
      TMAG1(1,5)=TMAG1(1,5)+TMAG1(1,6)
      TMAG1(1,7)=TMAG1(1,7)+TMAG1(1,8)
C
C     THE PROPERTIES CAN BE SET AND MODIFIED IN THIS SECTION
C
C
C      TRANSFER TO 570 IF PROPERTY IS NOT THE ONE WE WANT
C
C     IF(PROP(1,1).NE.5.1)GO TO 570
C     IF(PROP(1,3).GT.0.001)GO TO 570
C     IF(PROP(1,2).GT.2.200)GO TO 570
      DO 540 NT=1,NTT
C
C     ONE MAGNITUDE OR PHASE DRIFT IS SET ON AT A TIME.
C
      TMDFT(NT)=1.0
      CALL RDGPUL(RDG1,TMAG1,NPOL,IOFSET,TMDFT,1,IRDPR1,1,NTT,1,1,1,1)
C
C     THE POLYNOMIAL IS CALCULATED
C
      SUM=0.
      DO 520 IR=1,IRDPR
      SUM=SUM+COE(IR)*RDG1(1,IR)
  520 CONTINUE
      DRIFT=DRIFT+ABS(READNG(NR,IRDPR1)-SUM)
      TMDFT(NT)=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
      SDRIF=SQRT(SSDRIF/FLOAT(MSET))
      SDIFF=SQRT(SSDIFF/FLOAT(MSET))
      WRITE(LOU,580)PRONAM(NPROP),SDIFF,SDRIF,(NPOL(1,NT),NT=1,NTT)
      WRITE(LOU,585)IOFSET,(NPOL(2,NT),NT=1,NTT)
      WRITE(LOU,30)
      WRITE(LOTEK,580)PRONAM(NPROP),SDIFF,SDRIF
  580 FORMAT(' RMS DIF IN ',A4,'=',F10.5,2X,'DRIFT=',F10.5,3X,
     *'FCTN'10I2)
  585 FORMAT(I2,' CONSTANT',37X,'POL ',10I2)
```

```
  590 IF (NPRINT.NE.3)GO TO 630
      IF(NPROPT.NE.1)GO TO 597
C   OPEN FILE FOR OUTPUT COEFFICIENT DATA STORAGE,ASSUMED ON DEFAULT DISK
      FNAME=COEDAT//'.DAT'
      OPEN(LOD,FILE=FNAME,STATUS ='NEW',ERR = 992)
      DO 595 NC=1,NTCAL
      WRITE(LOD,*)(RDCO(JJ,NC),JJ=1,NTT)
  595 CONTINUE
      WRITE(LOD,*)(VOLSTD(NT),NT=1,NTT)
  596 CONTINUE
  597 WRITE(LOU,160)
      WRITE(LOTEK,160)
      NCOED1=NCOED+4*IRDPRM+1
      WRITE(LOD,*)IRDPR
      DO 610 I=1,IRDPR
      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
C
      WRITE(LOD,615)PRONAM(NPROP),IRDPR,JOFSET(NPROPT)
  615 FORMAT(1X,A4,2I4)
C     WRITE(LOD,*)(COEF(IR,NPROPT),IR=1,IRDPR)
      WRITE(LOD,616)(COEF(IR,NPROPT),IR=1,IRDPR)
  616 FORMAT('       DATA C  /',1PE13.6,14('/,C  /',1PE13.6))
      DO 620 NT=1,NTT
      WRITE(LOD,*)(NPOL(I,NT),I=1,2)
  620 CONTINUE
      PROPTY(NPROPT)=PRONAM(NPROP)
      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=1,NTT)
  715 FORMAT(8('  MAG(',I1,')'))
      WRITE(LOTEK,160)
C
C     EXPANSION OF TMAG1(1,NT)      INTO READNG(1,IRDPRM)
C
  720  DO 800 NPRO=1,NPROPT
      DO 790 NT=1,NTT
      DO 780 I=1,2
      NPOL(I,NT)=JPOL(I,NT,NPRO)
  780 CONTINUE
C     WRITE(0,721)(NPOL(I,NT),I=1,6)
C 721 FORMAT(6I3)
      TMDFT(NT)=0.
  790 CONTINUE
      IOFSET=JOFSET(NPRO)
      IRDPR=JRDPR(NPRO)
      IRDPR1=IRDPR+1
      CALL RDGPUL(RDG1,TMAG1,NPOL,IOFSET,TMDFT,1,IRDPR1,1,NTT,1,1,1,1)
      PRO(NPRO)=0.
      DO 800 IR=1,IRDPR
      PRO(NPRO)=PRO(NPRO)+COEF(IR,NPRO)*RDG1(1,IR)
  800 CONTINUE
      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 PRDG(IN,MPEC,NCH,ICMD,IER)
C     DO 830 NC=1,NCH
C     VOLTS(NC)=VOLTS(NC)+TOFSET(NC)+TSLOPE(NC)*VOLTS(8)
  830 CONTINUE
      IF(NPRINT.EQ.3)WRITE(LOTEK,820)(IN(NT),NT=1,NTT)
C     CALL CORRDG(VOLTS,COFSET,CGAIN,NTT)
  840 DO 845 NT=1,NTT
      TMAG1(1,NT)=IN(NT)
  845 CONTINUE
      TMAG1(1,1)=TMAG1(1,1)+TMAG1(1,2)
      TMAG1(1,3)=TMAG1(1,3)+TMAG1(1,4)
      TMAG1(1,5)=TMAG1(1,5)+TMAG1(1,6)
      TMAG1(1,7)=TMAG1(1,7)+TMAG1(1,8)
  820 FORMAT(8(I5))
      CALL _GETKEY(IKY,IS,KBF1,KBF2,IRR)
      IF(IRR.NE.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(8(F9.5))
      IF(NPRINT.EQ.3)WRITE(LOU,850)(VOLTS(NT),NT=1,NTT)
```

```
      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 CONTINUE
      CALL RESET(RDGC,RDGO,COFSET,CGAIN,STDV,NTT)
      GO TO 880
C
C     PRINT SUMMARY OF DATA FROM DISK FILE
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,1410) (TIM(NT),NT=1,NTT)
1410  FORMAT('TIME:',8(1PE10.3))
C
C     PRINT CALIBRATION READINGS
C
1600  CONTINUE
      WRITE(LOU,1650) NTCAL
1650  FORMAT(' AVERAGES & STD.DEVIATIONS OF CALIBRATION READINGS:',I3,
     *' MAGS')
      WRITE(LOU,160)
      WRITE(LOU,1660) (TIM(NT),NT=1,NTT)
1660  FORMAT(8(1PE10.3))
      DO 1700 NC=1,NTCAL
      WRITE(LOU,1680)((RDGO(NT,NC)),NT=1,NTT)
1680  FORMAT(8(F6.1))
1700  CONTINUE
C
C     PRINT OUT READINGS AND PROPERTIES
C
      WRITE(LOU,160)
      WRITE(LOU,1750)(NT,NT=1,NTT),(PRONAM(NPR),NPR=1,NPROPM)
1750  FORMAT(1X,'PSET',8('  MAG',I1),4(2X,A4,1X))
      REWIND(LID)
      DO 1755 IREC=1,NLINES
      READ(LID,483)ADUM
```

```
 1755 CONTINUE
      DO 1800 NP=1,NPTT
      READ(LID,482)(TMAG1(1,NT),NT=1,NTT),(PROP(1,NPR),NPR=1,NPROPM)
      TMAG1(1,1)=TMAG1(1,1)+TMAG1(1,2)
      TMAG1(1,3)=TMAG1(1,3)+TMAG1(1,4)
      TMAG1(1,5)=TMAG1(1,5)+TMAG1(1,6)
      TMAG1(1,7)=TMAG1(1,7)+TMAG1(1,8)
      WRITE(LOU,1760)NP,(TMAG1(1,NT),NT=1,NTT),(PROP(1,NPR),NPR=1,
     *NPROPM)
1760  FORMAT(1X,I4,8(F6.0),4F7.4)
1800  CONTINUE
C
      GO TO 880
 1990 WRITE(LOTEK,*)'ERROR-CHECK IF PRINTER OFF-LINE,HIT CR TO CONTINUE'
      READ(0,1991)ADUM
 1991 FORMAT(A2)
      GO TO 1
  991 WRITE(LOTEK,*)'ERROR IN OPENING INPUT DATA FILE'
      GO TO 900
  992 WRITE(LOTEK,*)'ERROR IN OPENING OUTPUT COEF DATA FILE'
      GO TO 900
  993 WRITE(LOTEK,*)'ERROR IN OPENING INSTRUMENT DATA FILE'
      GO TO 900
  994 WRITE(LOTEK,*)'ERROR IN OPENING REF DATA FILE'
      GO TO 900
  995 WRITE(LOTEK,*)'ERROR IN OPENING CIR DATA FILE'
      GO TO 900
  996 WRITE(LOTEK,*)'ERROR IN READING  COIL;',COIL
      WRITE(LOTEK,11,ERR=996)COIL,RBAR,R1,R2,L3,R3,R4,L4,L5,L6
     *,RDCDR,RDCPU,TNDR,TNPU
c     GO TO 10
  997 WRITE(LOTEK,*)'ERROR IN READING  COIL;',COIL
c     GO TO 15
  900 STOP
      END
      SUBROUTINE ALSQS(A,Y,B,R2,NN,MM,NA)
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
```

```
      DIMENSION A(NA,1),Y(1),B(1)
      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 RESET(RDGC,RDGO,COFSET,GAIN,STDV,NCH)
C
C     SUBROUTINE DOES NCH LEAST SQUARES FITS OF RDGO=COFSET+GAIN*RDGC
C     LEAST SQUARES SUBROUTINE IS STANDARD FOR Y(J)=COFSET+GAIN*X(J)
C
      DIMENSION RDGC(NCH,4),RDGO(NCH,4),COFSET(NCH),GAIN(NCH),STDV(NCH)
      DATA NRD/4/
      DO 100 NC=1,NCH
      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(NC,NR)
      SUMY=SUMY+RDGO(NC,NR)
      SUMXY=SUMXY+RDGC(NC,NR)*RDGO(NC,NR)
      SUMXX=SUMXX+RDGC(NC,NR)*RDGC(NC,NR)
      SUMYY=SUMYY+RDGO(NC,NR)*RDGO(NC,NR)
   50 CONTINUE
      XBAR=SUMX/NRD
      YBAR=SUMY/NRD
      GAIN(NC)=(SUMXY-SUMX*YBAR)/(SUMXX-SUMX*XBAR)
```

```
      COFSET(NC)=YBAR-GAIN(NC)*XBAR
      STDV(NC)=SQRT(ABS((SUMYY-SUMY*YBAR-GAIN(NC)*(SUMXY-SUMX*YBAR))/
     *FLOAT(NRD-2)))
  100 CONTINUE
      WRITE(0,210)(I,I=1,NCH)
      WRITE(0,220)(COFSET(NC),NC=1,NCH)
      WRITE(0,230)(GAIN(NC),NC=1,NCH)
      WRITE(0,240)(STDV(NC),NC=1,NCH)
  210 FORMAT(5X,8('  VLT(',I1,')'))
  220 FORMAT('OFSET',8(F8.4))
  230 FORMAT('GAIN ',8(F8.4))
  240 FORMAT('STDV ',8(F8.4))
      RETURN
      END
C
      SUBROUTINE CORRDG(VOLTS,COFSET,GAIN,NCH)
      DIMENSION VOLTS(NCH),COFSET(NCH),GAIN(NCH)
      DO 100 NC=1,NCH
      VOLTS(NC)=COFSET(NC)+GAIN(NC)*VOLTS(NC)
  100 CONTINUE
      RETURN
      END
C
      SUBROUTINE POLPUL(POLARY,JROW,IRDPR,NPOL,NTIM,IOFSET,LOU)
C
C THIS SUBROUTINE CONSTRUCTS AN ARRAY FOR PRINTING
C
      CHARACTER*2 RDGTYP(2),IDEG(12),INUM(9)
      CHARACTER*4 FUNTYP(4),BLANKS,CONSTA(2),POLARY(JROW,5),ROLD(4)
      DIMENSION NPOL(2,NTIM)
      DATA  RDGTYP/' M',' T'/,CONSTA/'CONS','TANT'/
      DATA FUNTYP/' LIN',' LOG',' EXP',' INV'/
      DATA IDEG/'1 ','2 ','3 ','4 ','5 ','6 ','7 ','8 ','9 ','10','11',
     *'12'/
      DATA INUM/'1)','2)','3)','4)','5)','6)','7)','8)','9)'/
      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)
20    FORMAT(1X,3A4)
C
70    DO 200 NT=1,NTIM
      NDEG=NPOL(2,NT)
      IF(NDEG.EQ.0) GO TO 300
```

```
      DO 400 I=1,NDEG
      NROW=NROW+1
      POLARY(NROW,1)=FUNTYP(NPOL(1,NT))
      POLARY(NROW,2)='VLT('
      POLARY(NROW,3)=INUM(NT)//IDEG(I)
400   CONTINUE
21    FORMAT(/(1X,A4,'VLT(',I2,')',A4))
300   CONTINUE
200   CONTINUE
C
C  PRINT RESULTS
C
      DO 1000 NR=1,NROW
      WRITE(LOU,20)(POLARY(NR,II),II=1,3)
1000  CONTINUE
C
      RETURN
      END
      SUBROUTINE RDGPUL(READNG,SUMT,NPOL,IOFSET,TMDFT,MSET1
     1,IRDPR1,M,NTIM,NL,NLT,NP,NPTT)
C     REAL*8 READNG,RDG
      DIMENSION READNG(MSET1,IRDPR1),NPOL(2,NTIM),TMDFT(NTIM)
      DIMENSION SUMT(NLT,NPTT,NTIM)
C
C     NPOL CONTAINS A NUMBER FOR THE FUNCTION TYPE & THE POLYNOMIAL
C     DEGREE. IF IOFSET=0,NO OFFSET IS INCLUDED,=1 OFFSET IS INCLUDED
C     NPOL(2,NT) =0,THAT PARTICULAR TIME VALUE WILL NOT BE INCLUDED
C     IN THE PROPERTY CALCULATIONS.
C
      READNG(M,1)=1.
      N=1
      IF(IOFSET.EQ.1)N=2
      DO 210 NT=1,NTIM
      IF(NPOL(2,NT).EQ.0) GO TO 200
      RDG=SUMT(NL,NP,NT)+TMDFT(NT)
C
C     THE TYPE OF FUNCTION IS SELECTED
C
      IF(NPOL(1,NT).EQ.1)RDG=RDG
      IF(NPOL(1,NT).EQ.2)RDG=ALOG(RDG)
      IF(NPOL(1,NT).EQ.3)RDG=EXP(RDG)
      IF(NPOL(1,NT).EQ.4)RDG=1./RDG
C
C     THE TYPE OF POLYNOMIAL IS SELECTED
C     AND THE POLYNOMIAL VALUES ARE CONSTRUCTED.
C
      READNG(M,N)=RDG
      N=N+1
      NDEG=NPOL(2,NT)-1
      IF(NDEG.LT.1) GO TO 15
      DO 10 I=1,NDEG
      READNG(M,N)=RDG*READNG(M,N-1)
```

```
      N=N+1
   10 CONTINUE
15    CONTINUE
  200 CONTINUE
  210 CONTINUE
      RETURN
      END
      SUBROUTINE PRDG(IN,MPEC,NCH,ICMD,IER)
C     PROGRAM TO READ PULSE DATA OVER THE GPIB BUSS NO AVERAGING
C     CMD0   SET INSTRUMENT UP
C     CMD1   FAST READING, MADE ON PULSE TURN ON,RDG THEN TRANSMIT.
C     CMD2   SAME AS CMD1 EXCEPT TRANSMITS OLD DATA WHILE NEW RDGS MADE
C     CMD3   USED FOR CALIBRATIONS
C     CMD4   SLOWER READINGS,USES SUBROUTINES THAT SUPPORT AVERAGING
      CHARACTER*1 CR
      DIMENSION IN(8)
      CR=CHAR(13)
      IER=0
      WRITE(MPEC,35,ERR=400)ICMD,CR
      READ(MPEC,70,ERR=500)(IN(I),I=1,NCH)
   35 FORMAT(I1,A1)
   70 FORMAT(8I4)
      GO TO 600
  400 IER=1
      WRITE(0,*)'BUSS WRITE ERROR TO PULSER;HIT RESET,CR AND CONTINUE'
      GO TO 600
  500 IER=2
      WRITE(0,*)'BUSS WRITE ERROR TO PULSER;HIT RESET,CR AND CONTINUE'
  600 RETURN
      END
      SUBROUTINE SETUP(MPEC,NCH,NRDG,IER)
C     PROGRAM TO SET UP PULSE INSTRUMENT OVER THE GPIB BUSS
      CHARACTER*1 CR,C,O,P,D,R,A,L,B,G
      DATA ICMD/0/,C/'C'/,O/'O'/,P/'P'/,D/'D'/,R/'R'/,A/'A'/,L/'L'/,
     *B/'B'/,G/'G'/
      CR=CHAR(13)
      IER=0
      NRH=NRDG/256
      NRM=(NRDG-NRH*256)/16
      NRL=NRDG-NRH*256-16*NRM
      WRITE(0,35,ERR=400)ICMD,C,NCH,CR
      WRITE(MPEC,35,ERR=400)ICMD,C,NCH,CR
C     WRITE(MPEC,40,ERR=400)ICMD,A,NRH,NRM,NRL,CR
C     WRITE(0,40,ERR=400)ICMD,A,NRH,NRM,NRL,CR
   35 FORMAT(I1,A1,I1,A1)
   40 FORMAT(I1,A1,Z1,Z1,Z1,A1)
      GO TO 600
```

```
  400 IER=1
      WRITE(0,*)'BUSS WRITE ERROR,COMMAND TO MPEC,HIT RESET & TRY AGAIN'
  600 RETURN
      END
```

SSTPUF3

This program drives the scanner to position the tube at different locations under the probe, takes the readings, computes the flaw size, plots the readings on the printer, and then stores the raw data. The entire tube is scanned before the computations and plotting are started. The present version is for a high-resolution plot that gives a one-to-one representation of the tube length. A listing of SSTPUF3 follows.

```
      PROGRAM SSTPUF3
C     VERSION MAY 9 1986
C     SCANS THE TUBE BY ROTATING THEN TRANSLATING  THE AVERAGE
C     VOLTAGE LEVEL IS READ OVER THE BUS FROM THE PULSE INSTRUMENT.
C     THE PROPERTY IS CALCULATED FROM THE VOLTAGE LEVEL AND
C     PLOTTED ON THE PRINTER. THE RAW READINGS ARE STORED ON THE DISK.
C     MODIFIED FOR 2.0 CSOS OPERATING SYSTEM & FORTRAN
C     IAXM=NUMBER OF STEPS ALONG THE X AXIS
C     IATHM=NUMBER OF STEPS IN THE THETA AXIS
C
      INTEGER*2 IVOL(40,1801,8)
      DIMENSION IV(8),IOY(40),IYA(40),IY(40),NSTEPS(3)
      CHARACTER*2 AXC(3)*2,GC*2,CR*1,FF*1,TUBE*10,TFILE*14
      DATA IXX/0/
      DATA AXC/'XM','YM','ZM'/,GC/'G '/,SCAN/42.75/,IATHM/20/
      DATA NCH/8/,NRDG/1/,ICMD/2/
      DATA MPEC/45/,MBUS/46/,NAXIS/2/,LOU/6/,NDAT/9/,PA/0./
      DATA CO / 1.255409E+02/,C11/ 2.387605E-02/,C12/-2.864017E-06/,
     *C21/-2.718268E-02/,C22/ 1.552432E-06/,C31/ 1.120601E-03/,
     *C41/-1.671774E-02/,C42/ 2.906185E-07/
C     MULTI PULSED EDDY CURRENT INSTRUMENT ADDR=5 ON GPIB BUSS
C     ADDR=E? ON IBM BUSS DRIVER, FORTRAN LOGICAL UNIT = 45
      OPEN(MPEC,FILE='#BUSE?')
C     SET FOR DMA WITH MORE THAN 4 CHARACTERS READ OR WRITE
      CALL _SETDMA(MPEC,2)
      CALL _STIMEOUT(MPEC,10)
C     MODULYNX AXIS POSITION CONTROLLER ADDR=6 ON GPIB BUSS,ADDR=F? ON
C     IBM BUSS DRIVER,FORTRAN LOGICAL UNIT = 46
      OPEN(MBUS,FILE='#BUSF?')
      CALL _STIMOUT(MBUS,400)
      CALL _SETEOS(MBUS,$00,$0A)
```

```
      CALL _SETOPTIONS(MBUS,0,0,0,0,0,1,0)
      WRITE(MBUS,*)'XA15000B410H2400'
      WRITE(MBUS,*)'YA50000B900H3600'
C     PRINTER GRAPHICS FILE OPENED,LOU=6
      OPEN(LOU,FILE='#PR')
      NSTEPS(1)=95
      NSTEPS(2)=200/IATHM
      NAOF=1000/IATHM
      NADX=160/IATHM
   25 WRITE(0,*)'WHAT IS LENGTH IN INCHES OF SCAN TO BE MADE ON TUBE ?'
      READ(0,*)SCX
      IF (SCX.EQ.0.0)SCX=SCAN
      SCAN=SCX
      IAXM=(SCAN+0.00001)/(0.00025*FLOAT(NSTEPS(1)))+1
      CALL SETUP(MPEC,NCH,NRDG,IER)
      IF(IER.NE.0)GO TO 25
      NCH2=NCH/2
C
C     IDENTIFY TUBE AND GENERATE FILE NAME FOR TUBE DATA DISKETTE
      NSTOR=(NCH2*2+1)*IAXM*IATHM+53
      WRITE(0,*)NSTOR,' BYTES OF DATA STORAGE WILL BE NEEDED'
      WRITE(0,*)' INSERT NEW DISKETTE IN DRIVE IF NEEDED.  ENTER TUBE
     1 NUMBER(8 ALPHA NUMERICS)'
      READ(0,20) TUBE
   20 FORMAT(A10)
      TFILE=TUBE//'.DAT'
      OPEN(NDAT,FILE=TFILE,STATUS='NEW',FORM='UNFORMATTED')
      CALL _GETTIME(IYR,IMO,IDA,IHR,IMN,ISE,IFR)
      TIM1=(FLOAT(IHR)*60.+FLOAT(IMN))*60.+FLOAT(ISE)+FLOAT(IFR)/32768.
   15 WRITE(LOU,10)TFILE,IHR,IMN,ISE,IMO,IDA,IYR,SCAN
      WRITE(0,10)TFILE,IHR,IMN,ISE,IMO,IDA,IYR,SCAN
   2  FORMAT(A8,'  TIME ',I2,':',I2,':',I2,'  DATE ',I2,'/',I2,'/',I2)
   10 FORMAT(6X,A14,'    TIME ',I2,':',I2,':',I2,'     DATE '
     *,I2,'/',I2,'/',I2,'   SCAN LENGTH ',F7.3)
      WRITE(NDAT)TUBE,IHR,IMN,ISE,IMO,IDA,IYR
   27 CONTINUE
      WRITE(NDAT)IAXM,IATHM,NCH2,ICMD
      WRITE(MBUS,*)'XD YD'
      WRITE(MBUS,*)'<CAH>'
      FF=CHAR(12)
      CR=CHAR(13)
C     SCANNING LOOPS FOR ENTIRE TUBE SCAN
      DO 99 IAX=1,IAXM
      DO 90  IATH=1,IATHM
      WRITE(MBUS,30) AXC(2),NSTEPS(2),GC
      CALL PRDG(IV,MPEC,NCH,ICMD,IER)
      DO 65 NT=1,NCH,2
      NT2=NT+1
      NT3=NT2/2
      IVOL(IATH,IAX,NT3)=(IV(NT)+IV(NT2))
   65 CONTINUE
   90 CONTINUE
```

```
      WRITE(MBUS,30) AXC(1),NSTEPS(1),GC
      WRITE(MBUS,*)'<CFA>'
   99 CONTINUE
C     END OF SCANNING,START OF PLOTTING
      CALL _GETTIME(IYR,IMO,IDA,IHR,IMN,ISE,IFR)
      TIM2=(FLOAT(IHR)*60.+FLOAT(IMN))*60.+FLOAT(ISE)+FLOAT(IFR)/32768.
      TIMT=(TIM2-TIM1)
      WRITE(LOU,*)'TOTAL SCANNING TIME =',TIMT,'SEC'
      DO 100 IATH=1,IATHM
      DO 95 NT=1,NCH2
      IV(NT)=IVOL(IATH,1,NT)
   95 CONTINUE
      YD=C0+IV(1)*(C11+IV(1)*C12)
     *+IV(2)*(C21+IV(2)*C22)
     *+IV(3)*C31+IV(4)*(C41+IV(4)*C42)
      IOY(IATH)=YD*75+250+NAOF*IATH
  100 CONTINUE
C     AXIAL LOOP-START OF PLOTTING AFTER TUBE HAS BEEN SCANNED
   30 FORMAT(A2,I7,A2)
      IXPLT=IAXM+NADX*IATHM
      DO 200 IAX=2,IXPLT
C     ROTATIONAL MOTION LOOP
      DO 190  IATH=1,IATHM
      IX1=1*IAX-NADX*IATH
      IY(IATH)=-1000
      IYA(IATH)=-1000
      IF(IX1.LE.0.OR.IX1.GT.IAXM)GO TO 190
      DO 195 NT=1,NCH2
      IV(NT)=IVOL(IATH,IX1,NT)
  195 CONTINUE
      YD=C0+IV(1)*(C11+IV(1)*C12)
     *+IV(2)*(C21+IV(2)*C22)
     *+IV(3)*C31+IV(4)*(C41+IV(4)*C42)
      IY(IATH)=YD*75+250+NAOF*IATH
      IYA(IATH)=(IY(IATH)+IOY(IATH))/2
      IOY(IATH)=IY(IATH)
  190 CONTINUE
      CALL GRAPRT(IYA,IXX,IATHM,LOU)
      CALL GRAPRT(IY,IXX,IATHM,LOU)
  200 CONTINUE
      CALL _SETPRMODE(LOU,0)
      CALL _GETTIME(IYR,IMO,IDA,IHR,IMN,ISE,IFR)
      TIM3=FLOAT(IHR)*3600.+FLOAT(IMN)*60.+FLOAT(ISE)+FLOAT(IFR)/32768.
      TIMT=(TIM3-TIM2)
      WRITE(LOU,*)'TOTAL PLOTTING TIME',TIMT,'SEC'
      WRITE(LOU,2)TUBE,IHR,IMN,ISE,IMO,IDA,IYR
  400 FORMAT(A1)
c     IZSTP=-(IAXM-1)*NSTEPS(1)
c     WRITE(MBUS,30) AXC(1),IZSTP,GC
      DO 500 IAX=1,IAXM
      DO 490 IATH=1,IATHM
      WRITE(NDAT)(IVOL(IATH,IAX,NT),NT=1,NCH2)
```

```
  490 CONTINUE
  500 CONTINUE
      CALL _GETTIME(IYR,IMO,IDA,IHR,IMN,ISE,IFR)
      TIM4=(FLOAT(IHR)*60.+FLOAT(IMN))*60.+FLOAT(ISE)+FLOAT(IFR)/32768.
      TIMT=(TIM3-TIM4)
      WRITE(LOU,*)'TOTAL DATA STORAGE TIME =',TIMT ,'SEC'
      WRITE(LOU,400)FF
C     PRINTER FILE CLOSED,LOU=6
      CLOSE(LOU)
      WRITE(MBUS,*)'<CAS>'
      WRITE(MBUS,*)'XE YE'
      STOP
      END
      SUBROUTINE GRAPRT(IY,IXX,NCUR,LOU)
C     PLOT A GRAPH ON PRINTER,EACH CALL WILL ADVANCE 1/84 OF INCH
C     IXX IS A COUNT OF THE NUMBER OF TIMES THAT SUBROUTINE IS CALLED
C     PRINTING IS DONE EVERY 7TH TIME SUBROUTINE IS CALLED.
C     Y VALUES OF THE CURVES MUST BE SCALED FROM 1 TO 1600
      INTEGER*1 GRAPH(1600),GRD1(1600),GRD2(1600)
      INTEGER*2 BIT(7)
      DIMENSION IBUF(10),IY(NCUR),IOY(40)
      DATA IOY/40*-1000/
      DATA GRD1/1600*0/,GRD2/1600*1/
      DATA BIT/1,2,4,8,16,32,64/,IMAX/1600/,IMIN/1600/,IX/0/,IBUF/10*0/,
     *NA/8/
      IF(IXX.GT.0)GO TO 20
      CALL _SETPRMODE(LOU,2)
C     SET COLOR  CALL;0=RED,1=GREEN,2=BLUE,3=BLACK(DEFAULT)
C     CALL _SETCOLOR(LOU,2)
C     SET UP GRID
      GRD1(1)=127
      GRD2(1)=85
      DO 10 I=1,IMAX
      GRAPH(I)=GRD2(I)
   10 CONTINUE
   20 CONTINUE
      IX=IX+1
      DO 30 NC=1,NCUR
      IF(IY(NC).EQ.-1000)GO TO 30
      IF(IOY(NC).EQ.-1000)IOY(NC)=IY(NC)
      IF(IY(NC).GT.1599)IY(NC)=1599
      IF(IY(NC).LT.1)IY(NC)=1
      IF(IOY(NC).GT.1599)IOY(NC)=1599
      IF(IOY(NC).LT.1)IOY(NC)=1
      I=IY(NC)
      J=IOY(NC)
      IF(J.GT.I)GO TO 22
      I=IOY(NC)
      J=IY(NC)
   22 DO 25 K=I,J
      GRAPH(K)=GRAPH(K)+BIT(IX)
   25 CONTINUE
```

```
      IOY(NC)=IY(NC)
   30 CONTINUE
      IF(IX.LT.7)GO TO 200
      DO 40 II=1,10
   40 IBUF(II)=0
      IOFS=0
      CALL _SWRITE(LOU,GRAPH(1),IOFS,IMAX,IMIN,IMAX,0,IRR)
      IF(IRR.NE.0)WRITE(0,*)'ERROR= ',IRR
      IX=0
      DO 70 I=1,IMAX
      GRAPH(I)=GRD1(I)
   70 CONTINUE
  200 IXX=IXX+1
      RETURN
      END
      SUBROUTINE PRDG(IN,MPEC,NCH,ICMD,IER)
C     PROGRAM TO READ PULSE DATA OVER THE GPIB BUSS NO AVERAGING
C     CMD0   SET INSTRUMENT UP
C     CMD1   FAST READING, MADE ON PULSE TURN ON,RDG THEN TRANSMIT.
C     CMD2   SAME AS CMD1 EXCEPT TRANSMITS OLD DATA WHILE NEW RDGS MADE
C     CMD3   USED FOR CALIBRATIONS
C     CMD4   SLOWER READINGS,USES THE SUBROUTINES THAT SUPPORT AVERAGING
      CHARACTER*1 CR
      DIMENSION IN(8)
      CR=CHAR(13)
      IER=0
      WRITE(MPEC,35,ERR=400)ICMD,CR
      READ(MPEC,70,ERR=500)(IN(I),I=1,NCH)
   35 FORMAT(I1,A1)
   70 FORMAT(8I4)
      GO TO 600
  400 IER=1
      WRITE(0,*)'BUSS WRITE ERROR,HIT RESET AND CONTINUE'
      GO TO 600
  500 IER=2
      WRITE(0,*)'BUSS READ ERROR,HIT RESET AND CONTINUE'
  600 RETURN
      END
      SUBROUTINE SETUP(MPEC,NCH,NRDG,IER)
C     PROGRAM TO SET UP PULSE INSTRUMENT OVER THE GPIB BUSS
      CHARACTER*1 CR,C,O,P,D,R,A,L,B,G
      DATA ICMD/0/,C/'C'/,O/'O'/,P/'P'/,D/'D'/,R/'R'/,A/'A'/,L/'L'/,
     *B/'B'/,G/'G'/
      CR=CHAR(13)
      IER=0
      NRH=NRDG/256
      NRM=(NRDG-NRH*256)/16
      NRL=NRDG-NRH*256-16*NRM
      WRITE(0,35,ERR=400)ICMD,C,NCH,CR
      WRITE(MPEC,35,ERR=400)ICMD,C,NCH,CR
C     WRITE(MPEC,40,ERR=400)ICMD,A,NRH,NRM,NRL,CR
C     WRITE(0,40,ERR=400)ICMD,A,NRH,NRM,NRL,CR
```

```
   35 FORMAT(I1,A1,I1,A1)
   40 FORMAT(I1,A1,Z1,Z1,Z1,A1)
      GO TO 600
  400 IER=1
      WRITE(0,*)'BUSS WRITE ERROR,COMMAND TO MPEC,HIT RESET & TRY AGAIN'
  600 RETURN
      END
```

POSPUL

This program is a utility used for instrument design and check-out. It is also useful in testing the scanner and in making hand scans on tubes without a scanner. The first mode of operation by the program is the scanner check-out mode where it asks for new positions. If a valid number is put in here, it will position the axis, but if a T (or any other character) is typed in, it goes to the instrument test loop. It then asks for a D for numerical data, P for plotted data, or Q for quit. The D and P options are both helpful in checking the instrument drifts. Both will keep track of the maximum amount of change, and the D function will print out a summary of this change for each data channel. In addition, it asks for a heading and types this out to aid in keeping track of what components are being tested. A listing of POSPUL follows.

```
      PROGRAM POSPUL
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 CALLS POST TO MOVE THE MOTORS TO THE NEW LOCATION
C     READS NCH CHANNELS OF PULSE INFORMATION AT NEW LOCATION
C     MODIFIED AND TESTED FOR 2.0 SOFTWARE
      DIMENSION NOLD(3),XNEW(3),NSTEPS(3),IN(8),IV(8),IVOL(1,1,4)
      DIMENSION IMAX(8),IMIN(8),IDIF(8)
      CHARACTER*2 AXC(3),GC,MBA,ADUM
      CHARACTER CMD*1,HEADING*80
      DATA AXC/'XM','YM','ZM'/,GC/'G '/,NCH/8/,ICMD/1/
      DATA MPEC/45/,MBUS/46/,NOLD/3*0/,NAXIS/2/,NGR/39/,LOU/6/
      DATA C0 / 1.046583E+02/,C11/-1.396685E-02/,C12/-1.248535E-06/,
     *C21/-1.918212E-02/,C22/ 6.718219E-06/,C31/ 1.401480E-02/,
     *C32/-4.730374E-06/,C41/-5.574330E-02/,C42/ 1.065772E-05/
C     DATA C0 / 2.709547E+01/,C11/-3.839086E-03/,C12/ 2.322576E-06/,
C    *C21/-1.493036E-02/,C22/-1.189059E-06/,C31/ 1.921997E-03/,
C    *C32/ 2.665246E-06/,C41/-4.514905E-03/,C42/-1.646022E-07/
C     MULTI PULSE EDDY CURRENT INSTRUMENT ADDR=5 ON GPIB BUSS
C     ADDR=E? ON IBM BUSS DRIVER, FORTRAN LOGICAL UNIT = 45
```

```
      OPEN(MPEC,FILE='#BUSE?')
C     SET FOR DMA WITH MORE THAN 4 CHARACTERS READ OR WRITE
      CALL _SETDMA(MPEC,2)
      CALL _STIMEOUT(MPEC,800)
      WRITE(0,*)'FILE OPENED FOR PULSED I/O OVER IEEE-488 BUS'
   25 CALL SETUP(MPEC,NCH,NRDG,IER)
      IF(IER.NE.0)READ(0,*)ADUM
      IF(IER.NE.0)GO TO 25
C     SET UP GRAPHICS FILE
      OPEN(NGR,FILE='#GR')
C     PRINTER FILE OPENED,LOU=6
      OPEN(LOU,FILE='#PR')
C     MODULYNX AXIS POSITION CONTROLLER ADDR=6 ON GPIB BUSS,ADDR=F? ON
C     IBM BUSS DRIVER,FORTRAN LOGICAL UNIT = 46
      OPEN(MBUS,FILE='#BUSF?')
      MBA='F?'
      CALL _STIMEOUT(MBUS,400)
      CALL _SETEOS(MBUS,$00,$0A)
      CALL _SETOPTIONS(MBUS,0,0,0,0,0,1,0)
C     CALL _SPOLL(MBUS,MBA,ISTAT)
      WRITE(MBUS,*)'XD YD ZE'
      WRITE(MBUS,*)'<CAH>'
      NCH2=NCH/2
    1 WRITE(0,*)NAXIS,' NEW POSITIONS '
      READ(0,*,ERR=500)(XNEW(IAXIS),IAXIS=1,NAXIS)
      IF(XNEW(1).GT.100.)GO TO 999
   80 CALL POST(XNEW,NOLD,NAXIS,MBUS)
  100 CALL PRDG(IN,MPEC,NCH,ICMD,IER)
      DO 165 NT=1,NCH,2
      NT2=NT+1
      NT3=NT2/2
      IVOL(1,1,NT3)=(IN(NT)+IN(NT2))/2
  165 CONTINUE
      DO 195 NT=1,NCH2
      IV(NT)=IVOL(1,1,NT)
  195 CONTINUE
      YD=C0+IV(1)*(C11+IV(1)*C12)+IV(2)*(C21+IV(2)*C22)
     *+IV(3)*(C31+IV(3)*C32)+IV(4)*(C41+IV(4)*C42)
      IY=YD*50+250
      WRITE(0,60)(IN(NT),NT=1,8),(IV(NT),NT=1,NCH2),IY,YD
      CALL _GETKEY(IKY,IS,KBF1,KBF2,IRR)
      IF(IRR.NE.0)GO TO 100
   60 FORMAT(8I5,1X,4I5,1I6,1F8.2)
      GO TO 1
      GO TO 999
  200 CONTINUE
      IX1=50
      IY1=250
      CALL PRDG(IMAX,MPEC,NCH,ICMD,IER)
      CALL PRDG(IMIN,MPEC,NCH,ICMD,IER)
      DMAX=YD
      DMIN=YD
```

```
  205 CONTINUE
  210 CALL PRDG(IN,MPEC,NCH,ICMD,IER)
      DO 220 NC=1,NCH
      IF(IN(NC).GT.IMAX(NC))IMAX(NC)=IN(NC)
      IF(IN(NC).LT.IMIN(NC))IMIN(NC)=IN(NC)
      IDIF(NC)=IMAX(NC)-IMIN(NC)
  220 CONTINUE
      DO 265 NT=1,NCH,2
      NT2=NT+1
      NT3=NT2/2
      IVOL(1,1,NT3)=(IN(NT)+IN(NT2))/2
  265 CONTINUE
      DO 295 NT=1,NCH2
      IV(NT)=IVOL(1,1,NT)
  295 CONTINUE
      IF(IX1.EQ.750)IX1=50
      IF(IX1.EQ.50)CALL _CLRWIN(NGR)
      IF(IX1.EQ.50)CALL GRID(NGR)
      IX1=IX1+1
      IX2=IX1+1
      YD=C0+IV(1)*(C11+IV(1)*C12)+IV(2)*(C21+IV(2)*C22)
     *+IV(3)*(C31+IV(3)*C32)+IV(4)*(C41+IV(4)*C42)
      IY2=YD*50+250
      CALL GRAPH(IX1,IY1,IX2,IY2,NGR)
      IF(YD.GT.DMAX)DMAX=YD
      IF(YD.LT.DMIN)DMIN=YD
      DDEF=DMAX-DMIN
      CALL _GOTOXY(0,0)
      WRITE(0,260)(IDIF(NT),NT=1,8),DDEF,YD
  260 FORMAT(8I5,2X,2F8.2)
      IY1=IY2
      CALL _GETKEY(IKY,IS,KBF1,KBF2,IRR)
      IF(IKY.EQ.87.AND.IRR.EQ.0) WRITE(0,*)'TYPE HEADING '
      IF(IKY.EQ.87.AND.IRR.EQ.0) READ(0,270)HEADING
      IF(IKY.EQ.87.AND.IRR.EQ.0) WRITE(LOU,270)HEADING
      IF(IKY.EQ.87.AND.IRR.EQ.0) WRITE(LOU,260)(IDIF(NT),NT=1,8),DDEF,YD
      IF(IRR.NE.0)GO TO 210
  270 FORMAT(A80)
  500 CONTINUE
      WRITE(0,*)' TYPE D FOR NUMERICAL DATA,P FOR PLOT,Q FOR QUIT'
      READ(0,510)CMD
  510 FORMAT(A1)
      IF(CMD.EQ.'D')GO TO 80
      IF(CMD.EQ.'P')GO TO 200
      IF(CMD.EQ.'Q')GO TO 999
      GO TO 999
  999 WRITE(MBUS,*)'XE YE ZE'
      STOP
      END
      SUBROUTINE POST(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 PULSES OVER THE IEEE-488 BUSS TO THE
C     CONTROLLER. AFTER THE NEW POSITION IS REACHED THE PULSE COUNT
C     IS COMPARED TO THE ENCODER COUNT AND CORRECTED IF DIFFERENT.
C
      DIMENSION NOLD(3),XNEW(3),NSTEPS(3)
      CHARACTER*2 AXC(3),GC
      DATA AXC/'XM','YM','ZM'/,GC/'G '/,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(MBUS,30)(AXC(IAXIS),NSTEPS(IAXIS),GC,IAXIS=1,NAXIS)
C     WRITE(0,30)(AXC(IAXIS),NSTEPS(IAXIS),GC,IAXIS=1,NAXIS)
      WRITE(MBUS,*)';<CFA>'
      WRITE(MBUS,*)';'
   30 FORMAT(4(A2,I7,A2))
      RETURN
      END
      SUBROUTINE SETUP(MFEC,NCH,NRDG,IER)
C     PROGRAM TO SET UP PULSE INSTRUMENT OVER THE GPIB BUSS
      CHARACTER*1 CR,C,O,P,D,R,A,L,B,G
      DATA ICMD/O/,C/'C'/,O/'O'/,P/'P'/,D/'D'/,R/'R'/,A/'A'/,L/'L'/,
     *B/'B'/,G/'G'/
      CR=CHAR(13)
      IER=0
      NRH=NRDG/256
      NRM=(NRDG-NRH*256)/16
      NRL=NRDG-NRH*256-16*NRM
      WRITE(MFEC,35,ERR=400)ICMD,C,NCH,CR
      WRITE(MFEC,40,ERR=400)ICMD,A,NRH,NRM,NRL,CR
      WRITE(0,40,ERR=400)ICMD,A,NRH,NRM,NRL,CR
   35 FORMAT(I1,A1,I1,A1)
   40 FORMAT(I1,A1,Z1,Z1,Z1,A1)
      GO TO 600
  400 IER=1
      WRITE(0,*)'BUSS WRITE ERROR,HIT RESET AND CONTINUE'
  600 RETURN
      END
      SUBROUTINE PRDG(IN,MFEC,NCH,ICMD,IER)
C     PROGRAM TO READ PULSE DATA OVER THE GPIB BUSS NO AVERAGING
      CHARACTER*1 CR
      DIMENSION IN(8)
      CR=CHAR(13)
      IER=0
      WRITE(MFEC,35,ERR=400)ICMD,CR
      READ(MFEC,70,ERR=500)(IN(I),I=1,NCH)
   35 FORMAT(I1,A1)
   70 FORMAT(8I4)
      GO TO 600
  400 IER=1
```

```
      WRITE(0,*)'BUSS WRITE ERROR,HIT RESET AND CONTINUE'
      GO TO 600
  500 IER=2
      WRITE(0,*)'BUSS READ ERROR,HIT RESET AND CONTINUE'
  600 RETURN
      END
      SUBROUTINE GRAPH(IX1,IY1,IX2,IY2,NGR)
C     PROGRAM TO DRAW A LINE IN FORTRAN
      CALL _SETCOP(NGR,IX1,IY1)
      CALL _VECTOR(NGR,IX2,IY2)
      RETURN
      END
      SUBROUTINE GRID(NGR)
      CALL GRAPH(50,50,750,50,NGR)
      CALL _SETCOP(NGR,5,50)
      CALL _CHARSTR(NGR,'-4.')
      CALL GRAPH(50,100,750,100,NGR)
      CALL GRAPH(50,150,750,150,NGR)
      CALL GRAPH(50,200,750,200,NGR)
      CALL GRAPH(50,250,750,250,NGR)
      CALL _SETCOP(NGR,5,250)
      CALL _CHARSTR(NGR,' 0.')
      CALL GRAPH(50,300,750,300,NGR)
      CALL GRAPH(50,350,750,350,NGR)
      CALL GRAPH(50,400,750,400,NGR)
      CALL GRAPH(50,449,750,449,NGR)
      CALL _SETCOP(NGR,5,449)
      CALL _CHARSTR(NGR,' 4.')
      CALL GRAPH(50,50,50,449,NGR)
      CALL GRAPH(750,50,750,449,NGR)
      RETURN
      END
```

FLAWLOC3

This program is used to aid the operator in the location and evaluation of flaws. The data from a scan are read into the program from the disk. The operator can look at the computed and plotted flaw size for each of the different scan lines (circumferential locations) and can move a cross-hair across the screen by use of the function keys. The axial location of the cross and the computed value of the flaw size at that location are displayed on the screen as the cross-hair is moved. The difference in the computed flaw value when the cross-hair is in a good part of the tube and the computed value when the cross-hair is on the peak of the flaw signal part is equal to the flaw size. This takes into account

the changes in computed flaw value due to wall, thickness, and conductivity changes. The tube is usually too long for a single screen on the IBM 9000 so the data plot is broken into segments with each segment fitting on a screen. A listing of FLAWLOC3 follows.

```
      PROGRAM FLAWLOC3
C     VERSION JUNE 4 1986
C     PLOTS DATA TAKEN BY SSTPUL ON THE CRT,MODIFIED FOR 2.0,BINARY
C     DATA FILE
C     IAXM=NUMBER OF STEPS ALONG THE X AXIS
C     IATHM=NUMBER OF STEPS IN THE THETA AXIS
C     GRAPHIC COMMANDS
C     C        CLEAR PLOT
C     F        FRAME INCREMENT +1
C     SHFT-F   FRAME INCREMENT -1
C     I        INVERT AXIS
C     N        NEXT TUBE
C     O        CHANGE OFFSET,GAIN AND CHANNEL NO
C     P        OPEN OR CLOSE HARD PRINTER FILE '#PR'
C     R        1 ROTATIONAL STEP
C     SHFT-R   -1 ROTATIONAL STEP
C     S        STOP (ALSO STOPS FOR NEGATIVE INPUT OF THETA MIN)
C     T        CHANGE ROTATIONAL LIMITS
C     W        WRITE IVOL(IATH,IAX,NF) FOR LAST IATH,IAX
C     F1,F2    MOVE CURSOR
C     F3,F4    MOVE CURSOR-FASTER
C     F5,F6    MOVE CURSOR-FASTEST
C     F7,F9    MOVE H-BARS UP-DOWN
C     F8,F10   MOVE H-BARS UP-DOWN FASTER
C
      integer*2 IVOL(1:9,1:50,1:1801),IV(8),IDIFF(1:3),HOR_BAR_DIFF,
     *HOR_BAR_MIN,HOR_BAR_LEN
      INTEGER*4 CXMIN,CYMIN,CXMAX,CYMAX,IKY,IS,KBF1,KBF2,IRR,IX,IY,
     *ICURSE,IOY,IOX
      CHARACTER TUBE*10,TFILE*14,TNAME*10
      LOGICAL CURSOR,PRINT_ON
      DATA CURSOR/.FALSE./
      DATA NCH/8/,NGR/39/,LOU/6/,INV/1/,NOFF/0/,NCHN/9/,GAIN/.2/,
     *IFRAME/0/,ICURSE/50/,GAGE_OFFSET/1.00/,HOR_BAR_DIFF/40/,
     *HOR_BAR_MIN_/10/,HOR_BAR_LEN/15/
C     --------COEFFICIENTS OF LINEARIZING EQUATION-------------
      DATA D0 /-3.176336E+02/,D11/-2.255382E-02/,D21/-2.219561E-03/,
     *D31/ 4.673161E-03/,D32/ 6.612573E-07/,D33/0.0/,
     *D41/ 2.304575E-01/,D42/-4.317338E-05/,D43/ 2.623878E-09/
C     ----------------------
C     PRINTER FILE OPENED,LOU=6
      OPEN(LOU,FILE='#PR')
      PRINT_ON=.TRUE.

C     GRAPHICS FILE OPENED,NGR=39
```

```
      OPEN(NGR,FILE='#GR')              !screen graphics

C     IDENTIFY TUBE AND GENERATE FILE NAME FOR INPUT DATA
   10 PRINT *,' ENTER TUBE NUMBER (8 ALPHA NUMERICS) -INSERT DISK IF ',
     *'NEEDED'
      READ(0,20) TUBE
      NDAT=9
   20 FORMAT(A10)
      TFILE=TUBE//'.DAT'
      OPEN(NDAT,FILE=TFILE,STATUS='OLD',FORM='UNFORMATTED',ERR=10)
   2  FORMAT(A10,'  TIME ',I2,':',I2,':',I2,'  DATE ',I2,'/',I2,'/',I2)
      READ(NDAT)TNAME,IHR,IMN,ISE,IMO,IDA,IYR
      INQUIRE(FILE='#PR',EXIST=PRINT_ON)
      IF(.NOT.PRINT_ON)OPEN(LOU,FILE='#PR')
      WRITE(LOU,2)TNAME,IHR,IMN,ISE,IMO,IDA,IYR
      WRITE(*,2)TNAME,IHR,IMN,ISE,IMO,IDA,IYR
      READ(NDAT)IAXM,IATHM,NCH,ICMD
      PRINT *,'POINTS;AXIAL ',IAXM,' ;CIRCUMFERENTIAL ',IATHM,' ;NCH ',
     *NCH,'  ICMD ',ICMD
      DELX=95*0.00025
      DELTH=0.00025*(200/IATHM)
      NAOF=200/IATHM
      NADX=80/IATHM

C     AXIAL MOTION LOOP- START OF DATA INPUT
      YDMIN=1E8
      DO 100 IAX=1,IAXM

C     ROTATIONAL MOTION LOOP
      DO 90  IATH=1,IATHM
      READ(NDAT)(IVOL(NT,IATH,IAX),NT=1,NCH)
      DO 95 NT=1,NCH
      IV(NT)=IVOL(NT,IATH,IAX)
   95 CONTINUE
c     ------------------------------------
      YD=D0+IV(1)*D11+IV(2)*D21
     *+IV(3)*(D31+IV(3)*(D32+IV(3)*D33))
     *+IV(4)*(D41+IV(4)*(D42+IV(4)*D43))
c     ----------------------------------
      IF(YD.LT.YDMIN)YDMIN=YD
      IVOL(9,IATH,IAX)=YD*100
   90 CONTINUE
  100 CONTINUE
      NOFF=100*YDMIN-50
      CALL _GETWIN(NGR,CXMIN,CYMIN,CXMAX,CYMAX)
  210 CONTINUE
      PRINT *,'WHAT ARE THETA MIN-MAX?'
      READ(0,*)ITHMIN,ITHMAX
      IF (ITHMIN.NE.ITHMAX) CURSOR=.FALSE.
      IF (ITHMIN.EQ.ITHMAX) CURSOR=.TRUE.
  218 CONTINUE
C     AXIAL MOTION LOOP- START OF PLOTTING
```

```
  219 FORMAT('FRAME->',I2)
      IF (CURSOR) IFT=0
      IF (.NOT.CURSOR) IFT=1
  225     call _spixmod(ngr,1)
      CALL _CLRWIN(NGR)
      CALL _FRAME(NGR)
  230 continue
      IF(CURSOR)THEN
         THLOC=ITHMAX*NAOF*1.8
         XLOC=DELX*(ICURSE/2+300*IFRAME)+GAGE_OFFSET
         CALL _GOTOXY(0,0)
         WRITE(*,405)'FRAME->',IFRAME+1, ' SCAN->',ITHMAX,THLOC,
     *'POS-->',ICURSE/2+300*IFRAME,XLOC,' IVOL-->',IVOL(NCHN,ITHMAX,
     *1+ICURSE/2+300*IFRAME)
      ENDIF
      CALL _GOTOXY(0,0)
      WRITE(*,219)IFRAME+1
      call _spixmod(ngr,1)
C     ROTATIONAL MOTION LOOP
      DO 300  IATH=ITHMIN,ITHMAX
      IOX=NAOF*(IATH-1)*IFT*.5
C     IOY=(IVOL(NCHN,IATH,1+300*IFRAME)-NOFF)*GAIN+NAOF*(IATH-1)*IFT*1.5
      IOY=(IVOL(NCHN,IATH,1+300*IFRAME)-NOFF)*GAIN+NAOF*(IATH-1)*1.7
      CALL _SETCOP(NGR,IOX,IOY)
C     PLOT ENTIRE AXIAL SCAN PRIOR TO INDEXING THETA
      DO 290 IAX=2+300*IFRAME,383+300*IFRAME
      IF (IAX.GT.IAXM) GOTO 300
      IX=2*(IAX-1-300*IFRAME)+NAOF*(IATH-1)*IFT*.5
C     IY=(IVOL(NCHN,IATH,IAX)-NOFF)*GAIN+NAOF*(IATH-1)*IFT*1.5
      IY=(IVOL(NCHN,IATH,IAX)-NOFF)*GAIN+NAOF*(IATH-1)*1.7
      CALL _VECTOR(NGR,IX,IY)
  290 CONTINUE
  300 CONTINUE
C     PROGRAM WILL WAIT HERE UNTIL A KEY IS HIT. PRINT SCREEN CAN
C     NOW BE DONE WITHOUT ANY EXTRA CHARACTERS ON THE SCREEN.
      IF (CURSOR) THEN
         CALL _SETCOP(NGR,ICURSE-HOR_BAR_LEN,HOR_BAR_MIN+CYMIN)
         CALL _VECTOR(NGR,ICURSE+HOR_BAR_LEN,HOR_BAR_MIN+CYMIN)
         CALL _SETCOP(NGR,ICURSE-HOR_BAR_LEN,HOR_BAR_MIN+CYMIN+NINT
     *        (HOR_BAR_DIFF*GAIN))
         CALL _VECTOR(NGR,ICURSE+HOR_BAR_LEN,HOR_BAR_MIN+CYMIN+NINT
     *        (HOR_BAR_DIFF*GAIN))
         CALL _SETCOP(NGR,ICURSE,CYMIN)
         CALL _VECTOR(NGR,ICURSE,CYMAX)    !WRITES FIRST CURSOR
       CALL _SPIXMOD(NGR,2)          !SET PIXELS TO  XOR
      ENDIF
  400 CALL _GETKEY(IKY,IS,KBF1,KBF2,IRR)
      IF (IRR.EQ.0) THEN
      IF (CURSOR) THEN
         if    (is.ge.59.and.is.le.68) then
         CALL _SETCOP(NGR,ICURSE-HOR_BAR_LEN,HOR_BAR_MIN+CYMIN)
         CALL _VECTOR(NGR,ICURSE+HOR_BAR_LEN,HOR_BAR_MIN+CYMIN)
```

```
      CALL _SETCOP(NGR,ICURSE-HOR_BAR_LEN,HOR_BAR_MIN+CYMIN+NINT
     *     (HOR_BAR_DIFF*GAIN))
      CALL _VECTOR(NGR,ICURSE+HOR_BAR_LEN,HOR_BAR_MIN+CYMIN+NINT
     *     (HOR_BAR_DIFF*GAIN))
      CALL _SETCOP(NGR,ICURSE,CYMIN)
      CALL _VECTOR(NGR,ICURSE,CYMAX)     !ERASES CURSOR
      IF (IS.EQ.59) ICURSE=ICURSE-2
      IF (IS.EQ.60) ICURSE=ICURSE+2
      IF (IS.EQ.61) ICURSE=ICURSE-10
      IF (IS.EQ.62) ICURSE=ICURSE+10
      IF (IS.EQ.63) ICURSE=ICURSE-50
      IF (IS.EQ.64) ICURSE=ICURSE+50
      IF (IS.EQ.65) HOR_BAR_MIN=HOR_BAR_MIN+1
      IF (IS.EQ.67) HOR_BAR_MIN=HOR_BAR_MIN-1
      IF (IS.EQ.66) HOR_BAR_MIN=HOR_BAR_MIN+10
      IF (IS.EQ.68) HOR_BAR_MIN=HOR_BAR_MIN-10
      if (ICURSE.lt.0) ICURSE=0
      if (ICURSE.gt.766) ICURSE=766
      IF (ICURSE/2+300*IFRAME.GT.IAXM)ICURSE=2*(IAXM-300*IFRAME)
      CALL _SETCOP(NGR,ICURSE-HOR_BAR_LEN,HOR_BAR_MIN+CYMIN)
      CALL _VECTOR(NGR,ICURSE+HOR_BAR_LEN,HOR_BAR_MIN+CYMIN)
      CALL _SETCOP(NGR,ICURSE-HOR_BAR_LEN,HOR_BAR_MIN+CYMIN+NINT
     *     (HOR_BAR_DIFF*GAIN))
      CALL _VECTOR(NGR,ICURSE+HOR_BAR_LEN,HOR_BAR_MIN+CYMIN+NINT
     *     (HOR_BAR_DIFF*GAIN))
      CALL _SETCOP(NGR,ICURSE,CYMIN)
      CALL _VECTOR(NGR,ICURSE,CYMAX)    !WRITES CURSOR AFTER MOVING
      XLOC=DELX*(ICURSE/2+300*IFRAME)+GAGE_OFFSET
      CALL _GOTOXY(0,0)
      WRITE(*,405)'FRAME->',IFRAME+1, ' SCAN->',ITHMAX,THLOC,
     *'POS-->',ICURSE/2+300*IFRAME,XLOC,' IVOL-->',IVOL(NCHN,ITHMAX,
     *1+ICURSE/2+300*IFRAME)
      endif
    ENDIF
405 FORMAT (A7,I2,A7,I3,F6.0,2X,A6,I4,F7.3,A8,I5)
    IF(IKY.EQ.67) THEN     ! 67 is "C" key
      call _spixmod(ngr,1)
      CALL _CLRWIN(NGR)
    ENDIF
    IF(IKY.EQ.87)THEN            ! 87 is "W" key
      INQUIRE(FILE='#PR',EXIST=PRINT_ON)
      IF(.NOT.PRINT_ON)OPEN(LOU,FILE='#PR')
      WRITE(LOU,410)ITHMAX,ICURSE/2+300*IFRAME,XLOC,THLOC
     *,(IVOL(NT,ITHMAX,1+ICURSE/2+300*IFRAME),NT=1,9)
    ENDIF
410 FORMAT(2I4,1X,2F8.4,1X,9(I4,1X))
    IF(IKY.EQ.82.AND.(KBF1.EQ.97.OR.KBF1.EQ.98))THEN
C   82 IS "R" KEY; 97 IS SHIFT KEY
    ITHMIN=ITHMIN-1
    ITHMAX=ITHMAX-1
C   -----------------ADJUSTMENT FOR "R" AROUND SCAN 1 TO SCAN IATHM
    IF (ITHMAX.LT.1) THEN       !
```

```
         ITHMAX=IATHM          !
         ITHMIN=IATHM          !
      ENDIF                    !
C     ------------------------------!
      GO TO 230
      ENDIF
      IF(IKY.EQ.82.AND.KBF1.EQ.96) THEN  !82 is "R" key;96 IS NOT-SHIFT
      ITHMIN=ITHMIN+1
      ITHMAX=ITHMAX+1
C     -------------------ADJUSTMENT FOR "R" AROUND SCAN IATHM TO SCAN 1
      IF (ITHMAX.GT.IATHM) THEN  !
         ITHMAX=1              !
         ITHMIN=1              !
      ENDIF                    !
C     ------------------------------!
      GO TO 230
      ENDIF
      IF(IKY.EQ.80)THEN        ! 80 IS "P" KEY
         INQUIRE(FILE='#PR',EXIST=PRINT_ON)
         IF(PRINT_ON)CLOSE(LOU)
      ENDIF
      IF(IKY.EQ.83) GO TO 1000           ! 83 is "S" key
      if(iky.eq.84) go to 210        ! 84 is "T" key
      IF(IKY.EQ.79) then      ! 79 is "O" key
 411        format (I8,F5.1,I2,A35)
        write(*,411) noff,gain,nchn,'ENTER NEW OFFSET,GAIN & CH NO'
 413        READ(0,*,ERR=412)NOFF,GAIN,NCHN
        GO TO 218
 412         PRINT *,'U DUMMY!!! ENTER IT RIGHT'
         GOTO 413
      ENDIF
      IF(IKY.EQ.70)THEN              !70 IS F KEY
         IF(KBF1.EQ.96)IFRAME=IFRAME+1     !96 IS NOT SHIFT
         IF(KBF1.EQ.97.OR.KBF1.EQ.98)IFRAME=IFRAME-1 !97,98=SHIFT KEY
         IF (IFRAME.LT.0)IFRAME=0
         IF(IFRAME.GT.5)IFRAME=5
         GOTO 218
      ENDIF
      IF(IKY.EQ.78)THEN              !78 IS N KEY
         CLOSE(NDAT)
         GOTO 10
      ENDIF
      ENDIF
      GO TO 400
 1000 CLOSE(LOU)
      call _spixmod(ngr,1)
      CALL _CLRWIN(NGR)
      STOP
      end
```

DEFTST

This program processes the data in a manner similar to the previous program, except that it is completely automatic. The raw data are read into the program, and the flaw size is computed for each scan line along the axis. The computed size is plotted, and if the positive or negative slope is exceeded, the program records a flaw. The plot is also underlined on the screen at the flaw locations, and a summary of the flaw sizes, locations and lengths is printed on the screen at the end of each scan line. The tube is then "rotated" and the process repeated for the next scan line. Then the flaws with axial dimensions that overlap are merged together into a common flaw, and all the flaws for a given tube are collected together in a table. This summary table is then printed for flaws larger than a given amount. A listing of DEFTST follows.

```
      PROGRAM DEFTST
C     VERSION JUNE 23 1986
C     SUMMARIZES DEFECT SIZE AND LOCATION FROM DATA TAKEN BY SSTPUL OR
C     SSTPUF MODIFIED FOR 2.0 SOFTWARE AND BINARY DATA FILES
C     WILL ALARM ON POSITIVE OR NEGATIVE DEFECTS
C     NSIZ=SLOPE ALARM SETTING
C     IAXM=NUMBER OF STEPS ALONG THE X AXIS
C     IATHM=NUMBER OF STEPS IN THE THETA AXIS
C     STEP SIZE = 0.02375 FOR TUBES SCANNED AFTER 11/13/85
C         = 0.025   FOR TUBES SCANNED BEFORE
      INTEGER*2 IVOL(50,1801,8)
      DIMENSION IV(8),IYP(50,1801,1),DFSIZ(200),NDFS(200,7),NDFSS(200,7)
      DIMENSION DFLOC(200),IDFOR(200),DFLEN(200),NI(50),IAXPD(50,99)
      DIMENSION IYA(50),NDFSIZ(50,99),IAXSD(50,99),IAXED(50,99)
      CHARACTER ADUM*82,TUBE*10,TFILE*14,TNAME*10
      DATA NFT/1/,NSIZ/10/,DFL/0.3/,NDD/0/
      DATA NGR/39/,PA/0./,STPSIZ/0.02375/
      DATA LOU/6/,NDAT/9/,INV/1/
      DATA C0 /-3.176336E+02/,C11/-2.255382E-02/,C21/-2.219561E-03/,
     *C31/ 4.673161E-03/,C32/ 6.612573E-07/,C33/0.0/,
     *C41/ 2.304575E-01/,C42/-4.317338E-05/,C43/ 2.623878E-09/
C     PRINTER FILE OPENED,LOU=6

      OPEN(LOU,FILE='#PR')
C     GRAPHICS FILE OPENED,NGR=39
      OPEN(NGR,FILE='#GR')
C     IDENTIFY TUBE AND GENERATE FILE NAME FOR INPUT DATA
   10 WRITE(0,*)' ENTER TUBE NUMBER (10 ALPHA NUMERICS) -INSERT DISK IF'
     *,' NEEDED'
      READ(0,20) TUBE
```

```
   20 FORMAT(A10)
      TFILE=TUBE//'.DAT'
      OPEN(NDAT,FILE=TFILE,STATUS='OLD',FORM='UNFORMATTED',ERR=10)
      CALL _GETTIME(IYR,IMO,IDA,IHR,IMN,ISE,IFR)
    1 CONTINUE
    2 FORMAT(A10,'  TIME ',I2,':',I2,':',I2,'  DATE ',I2,'/',I2,'/',I2)
      READ(NDAT)TNAME,IHR,IMN,ISE,IMO,IDA,IYR
      WRITE(LOU,2)TNAME,IHR,IMN,ISE,IMO,IDA,IYR
      READ(NDAT)IAXM,IATHM,NCH,ICMD
      WRITE(0,*)'POINTS;AXIAL ',IAXM,' ;CIRCUMFERENTIAL ',IATHM,' ;NCH '
     *,NCH,'  ICMD ',ICMD
C     READ IN ALL DATA LOOP OVER AXIAL DATA,THEN ROTATIONAL DATA
      DO 100 IAX=1,IAXM
      DO 90  IATH=1,IATHM
      READ(NDAT,ERR=105)(IVOL(IATH,IAX,NT),NT=1,NCH)
   90 CONTINUE
  100 CONTINUE
      GO TO 108
  105 WRITE(0,*)'READ ERROR AT IAX = ',IAX
      IAXM=IAX-1
  108 MAXIAX=IAXM-8
C     SET INITIAL IY VALUES FOR DIFFERENT ARRAYS
      DO 120 IATH=1,IATHM
      IYA(IATH)=0
      DO 110 IAX=1,IAXM
      DO 115 NT=1,NCH
      IV(NT)=IVOL(IATH,IAX,NT)
  115 CONTINUE
      YD=C0+IV(1)*C11+IV(2)*C21
     *+IV(3)*(C31+IV(3)*(C32+IV(3)*C33))
     *+IV(4)*(C41+IV(4)*(C42+IV(4)*C43))
      IYP(IATH,IAX,1)=YD*50
      IYA(IATH)=IYA(IATH)+IYP(IATH,IAX,1)
  110 CONTINUE
      IYA(IATH)=IYA(IATH)/IAXM
  120 CONTINUE
      NSIZN=NSIZ/4
      WRITE(LOU,*)'POINTS;AXIAL ',IAXM,' ;CIR ',IATHM,' ;NCH ',NCH
     *,'  ICMD ',ICMD,' SLOPE TRIG: START ',NSIZ,' STOP ',NSIZN
      DO 150 IATH=1,IATHM
      DO 140 IAX=1,IAXM
      IYP(IATH,IAX,1)=IYP(IATH,IAX,1)+125-IYA(IATH)
  140 CONTINUE
  150 CONTINUE
      DO 200 I=1,7
      DO 190 N=1,99
      NDFS(N,I)=0
      NDFSS(N,I)=0
  190 CONTINUE
  200 CONTINUE
      IATH=1
  220 CALL _CLRWIN(NGR)
```

```
      L=IATH
      IOFF=0
      NSP=0
      NSN=0
      NDFP=0
      NDFN=0
      NDD=0
      ND=1
      DO 225 N=1,99
      NDFSIZ(IATH,N)=0
  225 CONTINUE
C     AXIAL LOOP- START OF DEFECT SCANNING
      DO 300 IAX=8,MAXIAX
      IX1=IAX+IOFF
C     RESET X VALUE FOR LONGER SCANS TO GET PLOT ON SCREEN
      IF(IX1.EQ.780)CALL _CLRWIN(NGR)
      IF(IX1.EQ.780) IOFF=IOFF-750
      IX2=IX1
      IY2=IYP(IATH,IAX,1)
      IY1=IYP(IATH,IAX-1,1)
      NSIZ2=IY2-IY1
      IF(NDFN.EQ.1)GO TO 260
C     CHECK FOR POS SLOPE;IF SLOPE.GT.NSIZ MAKE NSP(SLOPE POSITIVE)=1,
C     NSN=0
      IF(NSIZ2.GE.NSIZ.OR.NDFP.EQ.1)GO TO 230
C     CHECK FOR NEG SLOPE;IF SLOPE.LT.-NSIZ MAKE NSN(SLOPE NEGATIVE)=1,
C     NSP=0
      IF(NSIZ2.LE.-NSIZ.OR.NDFN.EQ.1)GO TO 260
      GO TO 290
C     POSITIVE SLOPE DEFECT MEASUREMENT SECTION
  230 CONTINUE
      IF(NSIZ2.GE.NSIZ)NSP=1
      NDFN=0
C    CHECK FOR NEG SLOPE;IF SLOPE.LT.NSIZN AND WE ARE IN A DEFECT REGION
C     IF NOT,WE ARE ON THE DOWNSIDE- MAKE NSN=1,AND TURN OFF POS SLOPE
      IF(NSIZ2.LT.-NSIZN.AND.NDFP.EQ.1)NSN=1
      IF(NSIZ2.LT.-NSIZN)NSP=0
      IF(NDD.NE.0)GO TO 240
      IAXSD(IATH,ND)=IAX
      IAXPD(IATH,ND)=IAX
      NDFSIZ(IATH,ND)=IY2
  240 I=IAXSD(IATH,ND)-3
      IDA=(IYP(L,I-4,1)+IYP(L,I-3,1)+IYP(L,I-2,1)+IYP(L,I-1,1)+IYP(L,
     *I,1))/5
      IAXED(IATH,ND)=IAX
      IF(IY2.GT.NDFSIZ(IATH,ND))IAXPD(IATH,ND)=IAX
      IF(IY2.GT.NDFSIZ(IATH,ND))NDFSIZ(IATH,ND)=IY2
C     TURN NEG SLOPE BACK ON IF SIGNAL HAS NOT GONE BACK UP PER DEFECT
C    *LENGTH8
      NDFSL=NSIZ*(2+(IAX-IAXSD(IATH,ND))/3)
      IF(IYP(IATH,IAX,1)-IDA.GT.NDFSL.AND.NSP.EQ.0)NSN=1
      NDFP=NSP+NSN
```

```
C     DRAW A LINE UNDER DEFECT REGION, DETERMINE DEFECT START(IAXSD),
C     DEFECT END(IAXED),THE PEAK VALUE(NDFSIZ) AND PEAK LOCATION(IAXPD)
      CALL GRAPH(IX1,100,IX2,100,NGR)
      CALL GRAPH(IX1,99,IX2,99,NGR)
      CALL GRAPH(IX1,98,IX2,98,NGR)
      NSN=0
      NDD=1
C     INCREMENT THE DEFECT COUNT AS WE LEAVE THE DEFECT REGION
      IF(NDD.EQ.1.AND.NDFP.EQ.0)ND=ND+1
      IF(NDD.EQ.1.AND.NDFP.EQ.0)NDD=0
      GO TO 290
  260 CONTINUE
C     NEGATIVE SLOPE DEFECT MEASUREMENT SECTION
      IF(NSIZ2.LE.-NSIZ)NSN=1
      NDFP=0
C    CHECK FOR POS SLOPE;IF SLOPE.GT.NSIZN AND WE ARE IN A DEFECT REGION
C     IF NOT,WE ARE ON THE UPSIDE- MAKE NSP=1,AND TURN OFF NEG SLOPE
      IF(NSIZ2.GT.NSIZN.AND.NDFN.EQ.1)NSP=1
      IF(NSIZ2.GT.NSIZN)NSN=0
      IF(NDD.NE.0)GO TO 270
      IAXSD(IATH,ND)=IAX
      IAXPD(IATH,ND)=IAX
      NDFSIZ(IATH,ND)=IY2
  270 I=IAXSD(IATH,ND)-3
      IDA=(IYP(L,I-4,1)+IYP(L,I-3,1)+IYP(L,I-2,1)+IYP(L,I-1,1)+IYP(L,I,
     *1))/5
      IAXED(IATH,ND)=IAX
      IF(IY2.LT.NDFSIZ(IATH,ND))IAXPD(IATH,ND)=IAX
      IF(IY2.LT.NDFSIZ(IATH,ND))NDFSIZ(IATH,ND)=IY2
C     TURN POS SLOPE BACK ON IF SIGNAL HAS NOT GONE BACK UP PER DEFECT
C     LENGTH
      NDFSL=NSIZ*(2+(IAX-IAXSD(IATH,ND))/3)
      IF(IDA-IYP(IATH,IAX,1).GT.NDFSL.AND.NSN.EQ.0)NSP=1
      NDFN=NSP+NSN
C     TERMINATE NEGATIVE DEFECT REGION IF WE HAVE A LARGE POSITIVE RDG
C     AND SLOPE
      IF(IY2-IDA.GT.(4*NSIZ).AND.NSIZ2.GT.NSIZ)NDFN=0
C     DRAW LINE UNDER DEFECT REGION,DETERMINE DEFECT START(IAXSD),DEFECT
C     END(IAXED),THE PEAK VALUE(NDFSIZ) AND THE PEAK LOCATION(IAXPD)
      CALL GRAPH(IX1,100,IX2,100,NGR)
      CALL GRAPH(IX1,98,IX2,98,NGR)
      CALL GRAPH(IX1,96,IX2,96,NGR)
      NSP=0
      NDD=1
C     INCREMENT THE DEFECT COUNT AS WE LEAVE THE DEFECT REGION
      IF(NDD.EQ.1.AND.NDFN.EQ.0)ND=ND+1
      IF(NDD.EQ.1.AND.NDFN.EQ.0)NDD=0
  290 CALL GRAPH(IX1,IY1,IX2,IY2,NGR)
  300 CONTINUE
      IF(NDD.EQ.0)ND=ND-1
      NI(IATH)=ND
      IF(ND.EQ.0)GO TO 500
```

```
      DO 310 N=1,ND
      I=IAXSD(IATH,N)-2
      J=IAXED(IATH,N)+1
      IDA=IYP(L,I-4,1)+IYP(L,I-3,1)+IYP(L,I-2,1)+IYP(L,I-1,1)+IYP(L,I,1)
      IDA=IYP(L,J+4,1)+IYP(L,J+3,1)+IYP(L,J+2,1)+IYP(L,J+1,1)+IYP(L,J,
     *1)+IDA
      NDFSIZ(IATH,N)=NDFSIZ(IATH,N)-(IDA/10)
      DFSIZ(N)=(NDFSIZ(IATH,N))/71.
      DFLOC(N)=0.955+STPSIZ*(IAXPD(IATH,N)-1)
      DFLEN(N)=STPSIZ*(IAXED(IATH,N)-IAXSD(IATH,N))
  310 CONTINUE
         CALL _GOTOXY(0,0)
      WRITE(0,315)IATH,(NDFSIZ(IATH,N),N=1,ND)
      WRITE(0,320)ND,(DFLOC(N),N=1,ND)
      WRITE(0,330)(DFLEN(N),N=1,ND)
c     WRITE(0,340)(IAXPD(IATH,N),N=1,ND)
c     WRITE(0,340)(IAXED(IATH,N),N=1,ND)
c     WRITE(0,340)(IAXSD(IATH,N),N=1,ND)
  315 FORMAT(I2,16(I5))
  320 FORMAT(I2,16(F5.2))
  330 FORMAT(2X,16(F5.2))
  340 FORMAT(2X,16(I5))
C     PROGRAM WILL WAIT HERE UNTIL A KEY IS HIT. PRINT SCREEN CAN
C     NOW BE DONE WITHOUT ANY EXTRA CHARACTERS ON THE SCREEN.
  400 CALL _GETKEY(IKY,IS,KBF1,KBF2,IRR)
C      IF(IKY.EQ.80.AND.IRR.EQ.0)CLOSE(LOU)
       IF(IRR.NE.0)GO TO 400
  500 IATH=IATH+1
      IF(IATH.LE.IATHM)GO TO 220
C   CONSOLIDATE THE DEFECTS FOR EACH IATH INTO SUMMARY ARRAY,NDFS(NDC,7)
  600 NDC=0
      DO 700 IATH=1,IATHM
      ND=NI(IATH)
      IF(ND.EQ.0)GO TO 700
      IF(NDC.GE.1)GO TO 620
C     FIRST DEFECTS ARE PLACED INTO THE ARRAY
      DO 610 N=1,ND
      NDFS(N,1)=IAXSD(IATH,N)
      NDFS(N,2)=IAXED(IATH,N)
      NDFS(N,3)=IAXPD(IATH,N)
      NDFS(N,4)=NDFSIZ(IATH,N)
      NDFS(N,5)=IATH
      NDFS(N,6)=IATH
      NDFS(N,7)=IATH
  610 CONTINUE
      NDC=NI(IATH)
      GO TO 700
C     NEXT DEFECTS ARE EITHER ADDED TO THE ARRAY OR MERGED WITH THE
C     PRIOR DEFECTS
  620 DO 680 N=1,ND
      NC=1
  630 IF(NDFS(NC,1).LE.IAXED(IATH,N).AND.NDFS(NC,2).GE.IAXSD(IATH,N))
```

```
     *GO TO 640
      NC=NC+1
      IF(NC.LE.NDC)GO TO 630
      NDC=NDC+1
      NDFS(NDC,1)=IAXSD(IATH,N)
      NDFS(NDC,2)=IAXED(IATH,N)
      NDFS(NDC,3)=IAXPD(IATH,N)
      NDFS(NDC,4)=NDFSIZ(IATH,N)
      NDFS(NDC,5)=IATH
      NDFS(NDC,6)=IATH
      NDFS(NDC,7)=IATH
      GO TO 680
C     MERGE DEFECT WITH OLD DEFECT-THE DEFECTS OVERLAPPED ALONG THE AXIS
C     DETERMINE VALUE AND LOCATION OF LARGEST PART OF DEFECT SIGNAL
  640 IF(ABS(NDFSIZ(IATH,N)).LT.ABS(NDFS(NC,4)))GO TO 650
      NDFS(NC,4)=NDFSIZ(IATH,N)
      NDFS(NC,3)=IAXPD(IATH,N)
      NDFS(NC,7)=IATH
  650 NDFS(NC,6)=IATH
C     SET THE LENGTH OF THE DEFECT TO INCLUDE THE MAXIMUM VALUES
      IF(IAXSD(IATH,N).LT.NDFS(NC,1))NDFS(NC,1)=IAXSD(IATH,N)
      IF(IAXED(IATH,N).GT.NDFS(NC,2))NDFS(NC,2)=IAXED(IATH,N)
  680 CONTINUE
  700 CONTINUE
C     SORT THE DEFECT ARRAY ACCORDING TO AXIAL LOCATION OF THE DEFECTS
      NS=1
      DO 730 IAX=8,MAXIAX
      DO 720 NC=1,NDC
      IF(IAX.NE.NDFS(NC,3))GO TO 720
      DO 710 J=1,7
      NDFSS(NS,J)=NDFS(NC,J)
  710 CONTINUE
      NS=NS+1
  720 CONTINUE
  730 CONTINUE
      DO 900 N=1,NDC
      DFSIZ(N)=(NDFSS(N,4))/71.
      IF(DFSIZ(N).GT.5.0)DFSIZ(N)=5.0
      IF(DFSIZ(N).LT.-5.0)DFSIZ(N)=-5.0
      DFLOC(N)=1.00+STPSIZ*(NDFSS(N,3)-1)
      DFLEN(N)=STPSIZ*(NDFSS(N,2)-NDFSS(N,1))
      IDFOR(N)=NDFSS(N,7)*360/IATHM
      IF(IDFOR(N).EQ.360)IDFOR(N)=0
  900 CONTINUE
      WRITE(LOU,*)'DF NO  SIZE  XLOC  OREN   LEN'
      DO 920 N=1,NDC
      IF(ABS(DFSIZ(N)).GT.DFL)WRITE(LOU,910)N,DFSIZ(N),DFLOC(N),IDFOR(N)
     *,DFLEN(N)
  920 CONTINUE
  910 FORMAT(I6,F6.2,F6.2,I6,F6.2,I6)
      WRITE(LOU,950)
  950 FORMAT(/,/,/,/)
```

```
      GO TO 1000
 1990 WRITE(LOTEK,*)'ERROR-CHECK IF PRINTER OFF-LINE HIT CR TO CONTINUE'
      READ(0,1991)ADUM
 1991 FORMAT(A2)
      GO TO 1
 1000 CALL _CLRWIN(NGR)
      STOP
      END
      SUBROUTINE GRAPH(IX1,IY1,IX2,IY2,NGR)
C     TEST PROGRAM TO DO GRAPHICS IN FORTRAN
      CALL _SETCOP(NGR,IX1,IY1)
      CALL _VECTOR(NGR,IX2,IY2)
      RETURN
      END
```

PULG2

This program is used to set up and test the new computer-adjusted instrument. It allows the selection of pulse on- and off-times, the pulse amplitude, the trigger levels of the voltage comparators, and the balance voltages of the filter amplifiers. A listing of PULG2 follows.

```
      PROGRAM PULG2
C     (28 MAY 1986)
C
C   THIS PROGRAM READS THE VOLTAGE LEVEL USING THE PULSED EDDY CURRENT
C   INSTRUMENT.THE INSTRUMENT IS SET UP USING A CALL TO COMMAND 0.
C     CMD0   SET INSTRUMENT UP
C     CMD1   FAST READING, MADE ON PULSE TURN ON,RDG THEN TRANSMIT.
C     CMD2   SAME AS CMD1 EXCEPT TRANSMITS OLD DATA WHILE NEW RDGS MADE
C     CMD3   USED FOR CALIBRATIONS
C     CMD4   SLOWER READINGS,USES THE SUBROUTINES THAT SUPPORT AVERAGING
C     PROGRAM PUTS OUT 4 DATA BYTES FOR EACH OUTPUT NUMBER IN SETUP
C     ROUTINE HAS AN AUTO-BALANCE LOOP THAT SETS BALANCE VOLTAGE TO THE
C     INPUTTED VALUE
      CHARACTER*1 CR
      DIMENSION IN(8),RDGC(8,4)
      DATA MFEC/45/,NCH/8/,NRDG/1000/,ICMD/1/
      DATA LOU/6/
      WRITE(0,*)'PULG2   AUTO BALANCE PROGRAM,4 BYTE SET-UP DATA'
      OPEN(LOU,FILE='#PR')
      OPEN(MFEC,FILE='#BUSE?')
C     SET FOR DMA WITH MORE THAN 4 CHARACTERS READ OR WRITE
      CALL _SETDMA(MFEC,2)
      CALL _STIMEOUT(MFEC,10)
      CR=CHAR(13)
    2 FORMAT('PULG TIME ',I2,':',I2,':',I2,' DATE ',I2,'/',I2,'/',I2,
     *' RDGS',I7)
      CALL SETUP(MFEC,NCH,NRDG,IER)
```

```
      WRITE(0,*)'HIT ANY KEY TO EXIT FROM LOOP'
   20 WRITE(0,*)' 1.SINGLE RDGS 2.AVG RDG 3.STOP ,NUMBER OF DATA ',
     *'CHANNELS,NO RDGS?'
      READ(0,*)NEXT,NCH,NRDG
      CALL _GETTIME(IYR,IMO,IDA,IHR,IMN,ISE,IFR)
      IF(NEXT.EQ.2)WRITE(LOU,2)IHR,IMN,ISE,IMO,IDA,IYR,NRDG
      GO TO (100,200,999),NEXT
  100 CALL _GETTIME(IYR,IMO,IDA,IHR,IMN,ISE,IFR)
      DO 90 K=1,NRDG
      TIM1=FLOAT(IHR)*3600.+FLOAT(IMN)*60.+FLOAT(ISE)+FLOAT(IFR)/32768.
      CALL PRDG(IN,MFEC,NCH,ICMD,IER)
      IF(IER.NE.0)GO TO 20
   90 CONTINUE
      CALL _GETTIME(IYR,IMO,IDA,IHR,IMN,ISE,IFR)
      TIM2=FLOAT(IHR)*3600.+FLOAT(IMN)*60.+FLOAT(ISE)+FLOAT(IFR)/32768.
      TIMT=((TIM2-TIM1)/FLOAT(NRDG))*1.0E3
      WRITE(0,60)(IN(NT),NT=1,8),TIMT
   60 FORMAT(8I5,F7.1)
C     WRITE(0,220)TIMT
  220 FORMAT('TOTAL TIME (MILI SEC)',F12.2)
      CALL _GETKEY(IKY,IS,KBF1,KBF2,IRR)
      IF(IRR.NE.0)GO TO 100
      GO TO 20
  200 CALL AVGR(IN,MFEC,NCH,ICMD,NRDG,LOU,IER)
      CALL _GETKEY(IKY,IS,KBF1,KBF2,IRR)
      IF(IRR.NE.0)GO TO 200
      GO TO 20
  999 STOP
      END
      SUBROUTINE SETUP(MFEC,NCH,NRDG,IER)
C     PROGRAM TO SET UP PULSE INSTRUMENT OVER THE GPIB BUSS
      CHARACTER*1 CR,C,O,P,D,R,A,L,B,G,CM
      DIMENSION IN(8),IB(8)
      DIMENSION V(10),IV(10),NRT(10),NRH(10),NRM(10),NRL(10)
      DATA ICMD/0/,C/'C'/,O/'O'/,P/'P'/,R/'R'/,A/'A'/,L/'L'/,B/'B'/,
     *G/'G'/
      DATA IVREF/3000/,ICMD1/1/,VCON/819.2/
      CR=CHAR(13)
   10 IER=0
      WRITE(0,*)'WHAT IS COMMAND ? C O P R A L B G S '
      READ(0,5) CM
    5 FORMAT(A1)
      NRDA=NCH
      IF(CM.EQ.'O'.OR.CM.EQ.'R')NRDA=1
      IF(CM.EQ.'O')WRITE(0,*)'OUTPUT VOLTAGE LEVEL,0 TO 10 VOLTS'
      IF(CM.EQ.'R')WRITE(0,*)'REFERENCE VOLTAGE LEVEL,-5 TO +5 VOLTS'
      IF(CM.EQ.'O')READ(0,*)V(1)
      IF(CM.EQ.'R')READ(0,*)V(1)
      IF(CM.EQ.'O')IV(1)=204.8*V(1)+2048
      IF(CM.EQ.'R')IV(1)=409.6*(5.+V(1))
      IF(CM.EQ.'P')NRDA=NCH+2
      IF(CM.EQ.'P')WRITE(0,*)'INPUT PULSE ONTIME,OFFTIME ',NCH,
```

```
     *' READING TIMES(MICROSECONDS)'
      IF(CM.EQ.'P')READ(0,*)(V(N),N=1,NRDA)
      IF(CM.EQ.'B')WRITE(0,*)'INPUT ',NCH,' VOLTAGES BETWEEN 0 AND 5 ',
     *'VOLTS'
      IF(CM.EQ.'L')WRITE(0,*)'INPUT ',NCH,' VOLTAGES BETWEEN -5 AND ',
     *'+5 VOLTS'
      IF(CM.EQ.'L'.OR.CM.EQ.'B')READ(0,*)(V(N),N=1,NCH)
      IF(CM.EQ.'G')WRITE(0,*)'INPUT ',NCH,' GAIN VALUES'
      IF(CM.EQ.'G')READ(0,*)(V(N),N=1,NCH)
      IF(CM.EQ.'S')GO TO 600
      DO 20 NC=1,NRDA
      IF(CM.EQ.'L')IV(NC)=(409.6*(5.+V(NC)))
      IF(CM.EQ.'G')IV(NC)=(4095.*V(NC))
      IF(IV(NC).GT.4095)IV(NC)=4095
      IF(CM.EQ.'P')IV(NC)=V(NC)/.1 -1
      IF(IV(NC).GT.65535)IV(NC)=65535
      IF(IV(NC).LT.0)IV(NC)=0
C     CONVERT INTEGER NUMBER TO 4 HEX (BASE 16) NUMBERS
      NRT(NC)=IV(NC)/4096
      NRH(NC)=(IV(NC)-NRT(NC)*4096)/256
      NRM(NC)=(IV(NC)-NRT(NC)*4096-NRH(NC)*256)/16
      NRL(NC)=IV(NC)-NRH(NC)*256-16*NRM(NC)
   20 CONTINUE
C     SET THE CHANNEL NO = NCH
      WRITE(MFEC,35,ERR=400)ICMD,C,NCH,CR
      IF(CM.EQ.'P')GO TO 50
      IF(CM.EQ.'O'.OR.CM.EQ.'R'.OR.CM.EQ.'A')GO TO 60
      IF(CM.EQ.'B')GO TO 100
      WRITE(MFEC,40,ERR=400)ICMD,CM,(NRT(N),NRH(N),NRM(N),NRL(N),N=1,
     *NRDA),CR
      WRITE(0,40,ERR=400)ICMD,CM,(NRT(N),NRH(N),NRM(N),NRL(N),N=1,NRDA),
     *CR
      GO TO 10
   50 WRITE(MFEC,45,ERR=400)ICMD,CM,(NRT(N),NRH(N),NRM(N),NRL(N),N=1,
     *NRDA),CR
      WRITE(0,45,ERR=400)ICMD,CM,(NRT(N),NRH(N),NRM(N),NRL(N),N=1,NRDA),
     *CR
      GO TO 10
   60 WRITE(MFEC,30,ERR=400)ICMD,CM,NRT(1),NRH(1),NRM(1),NRL(1),CR
      WRITE(0,30,ERR=400)ICMD,CM,NRT(1),NRH(1),NRM(1),NRL(1),CR
   35 FORMAT(I1,A1,I1,A1)
   30 FORMAT(I1,A1,4(Z1),A1)
   40 FORMAT(I1,A1,32(Z1),A1)
   45 FORMAT(I1,A1,40(Z1),A1)
      GO TO 10
  100 CONTINUE
      NSTEP=1000
      DO 110 NC=1,NRDA
      IB(NC)=(819.2*V(NC))
      IV(NC)=4095
  110 CONTINUE
  115 DO 120 NC=1,NRDA
```

```
      NRT(NC)=IV(NC)/4096
      NRH(NC)=(IV(NC)-NRT(NC)*4096)/256
      NRM(NC)=(IV(NC)-NRT(NC)*4096-NRH(NC)*256)/16
      NRL(NC)=IV(NC)-NRH(NC)*256-16*NRM(NC)
  120 CONTINUE
      WRITE(MFEC,40,ERR=400)ICMD,CM,(NRT(N),NRH(N),NRM(N),NRL(N),N=1,
     *NRDA),CR
      CALL PRDG(IN,MFEC,NCH,ICMD1,IER)
      NSTOP=0
      DO 130 NC=1,NRDA
      IF(IN(NC).GT.IB(NC).AND.IV(NC).GT.0)IV(NC)=IV(NC)-NSTEP
      IF(IN(NC).GT.IB(NC).AND.IV(NC).GT.0)NSTOP=1
  130 CONTINUE
      IF(NSTOP.NE.0)GO TO 115
      DO 140 NC=1,NRDA
      IV(NC)=IV(NC)+NSTEP
  140 CONTINUE
      WRITE(0,40,ERR=400)ICMD,CM,(NRT(N),NRH(N),NRM(N),NRL(N),N=1,NRDA),
     *CR
      NSTEP=NSTEP/10
      IF(NSTEP.NE.0)GO TO 115
      GO TO 10
  400 IER=1
      WRITE(0,*)'BUSS WRITE ERROR,HIT RESET AND CONTINUE'
  600 RETURN
      END
      SUBROUTINE PRDG(IN,MFEC,NCH,ICMD,IER)
C     PROGRAM TO READ PULSE DATA OVER THE GPIB BUSS NO AVERAGING
      CHARACTER*1 CR
      DIMENSION IN(8)
      CR=CHAR(13)
      IER=0
      WRITE(MFEC,35,ERR=400)ICMD,CR
      READ(MFEC,70,ERR=500)(IN(I),I=1,NCH)
   35 FORMAT(I1,A1)
   70 FORMAT(8I4)
      GO TO 600
  400 IER=1
      WRITE(0,*)'BUSS WRITE ERROR,HIT RESET AND CONTINUE'
      GO TO 600
  500 IER=2
      WRITE(0,*)'BUSS READ ERROR,HIT RESET AND CONTINUE'
  600 RETURN
      END
      SUBROUTINE AVGR(IN,MFEC,NCH,ICMD,NRDG,LOU,IER)
C     PROGRAM TO AVERAGE NRDG READINGS AND PRINT AVG RDG,STD DEV & MAX
C     VARIATION
      CHARACTER*1 CR
      DIMENSION IN(NCH),IVAR(8),IMAX(8),IMIN(8),SDV(8)
      REAL*8 VOLSQR(8),VOLSUM(8)
      CR=CHAR(13)
      CALL PRDG(IMAX,MFEC,NCH,1,IER)
```

```
      CALL PRDG(IMAX,MFEC,NCH,1,IER)
      DO 50 NC=1,NCH
      VOLSUM(NC)=0.
      VOLSQR(NC)=0.
      IMIN(NC)=IMAX(NC)
   50 CONTINUE
      DO 100 NR=1,NRDG
      WRITE(MFEC,35,ERR=400)ICMD,CR
      READ(MFEC,70,ERR=500)(IN(I),I=1,NCH)
   35 FORMAT(I1,A1)
   70 FORMAT(8I4)
      IF(IER.NE.0)GO TO 999
      DO 65 NC=1,NCH
      VOLSUM(NC)=VOLSUM(NC)+IN(NC)
      VOLSQR(NC)=VOLSQR(NC)+IN(NC)*IN(NC)
      IF(IN(NC).GT.IMAX(NC))IMAX(NC)=IN(NC)
      IF(IN(NC).LT.IMIN(NC))IMIN(NC)=IN(NC)
   65 CONTINUE
  100 CONTINUE
      DO 150 NC=1,NCH
      IVAR(NC)=IMAX(NC)-IMIN(NC)
      VOLSUM(NC)=VOLSUM(NC)/FLOAT(NRDG)
      SDV(NC)=SQRT((VOLSQR(NC)-VOLSUM(NC)*VOLSUM(NC)*FLOAT(NRDG))/FLOAT
     *(NRDG-1))
  150 CONTINUE
      WRITE(0,60)(IN(NC),NC=1,NCH)
      WRITE(0,75)(IVAR(NC),NC=1,NCH)
      WRITE(0,80)(SDV(NC),NC=1,NCH)
      WRITE(LOU,60)(IN(NC),NC=1,NCH)
      WRITE(LOU,75)(IVAR(NC),NC=1,NCH)
      WRITE(LOU,80)(SDV(NC),NC=1,NCH)
   60 FORMAT('AVG ',8I8)
   75 FORMAT('VAR ',8I8)
   80 FORMAT('SDV ',8F8.3)
      GO TO 999
  400 IER=1
      WRITE(0,*)'BUSS WRITE ERROR,HIT RESET AND CONTINUE'
      GO TO 999
  500 IER=2
      WRITE(0,*)'BUSS READ ERROR,HIT RESET AND CONTINUE'
  999 RETURN
      END
```

PULG

This program is written in Intel 8080 Assembly language and is used to run the NDT-COMP9B microcomputer. This program will work with either the manual set-up instrument or the computer set-up instrument. There are comments listed in the program that describe some of the different functions that are performed. This program is linked to two other

assembly language programs, MATH9 and COMP9, described in an earlier reference. A listing of PULG follows.

```
            TITLE 'PULG PROGRAM VERSION   JUNE 5 1986'
            NAME PULG
            LIST   B,G,O,T
;           NLIST   I,M,S,T
;
;           PROGRAM TO CONFIGURE THE COMP9B MICROCOMPUTER
;           AS A TALKER/LISTENER ON THE IEEE488/GPIB BUSS
;           USING AN 8291A& 2 8293'S.
;           TRANSMIT SET UP DATA TO COMP9B,THEN RUN DESIRED PROG
;
            ORG 0800H        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,HILO
;
;
```

```
;CONSTANTS ARE SET
;
NCHM        EQU 08H             NCHM=MAX NUMBER OF CHANNELS TO BE READ
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,TIM1,TIM2,...HI-LO
OLDSW       EQU RAWDA+4*NCHM    CALIBRATION SWITCH POSITION FROM CONTROLLER
ASCDAT      EQU OLDSW+1         ASCII DATA,8*NCHM+1 BYTES
DIAD        EQU ASCDAT+8*NCHM+1  ;ADDRESS FOR INPUT DATA
DIA0        EQU DIAD+2          ADDRESS FOR DATA USED IN PULSER SETUP
NCH         EQU DIA0+2          CHANNEL NUMBER COUNT FOR ADC RDGS
CNTSH       EQU NCH+1           SAMPLE AND HOLD COUNT NUMBER
SHDAT       EQU CNTSH+1         SAMPLE AND HOLD DATA BYTE
NCHAA       EQU SHDAT+1         ADDRESS WHWRE NO OF CHANNELS STORED
NCH4A       EQU NCHAA+1         ADDRESS WHERE 4*NCHA STORED
NCH8A       EQU NCH4A+1         ADDRESS WHERE 8*NCHA STORED
ASC4NA      EQU NCH8A+1         ADDRESS WHERE ASCDAT+4*NCHA IS STORED,2 BYTES
NRDG        EQU ASC4NA+2        ADDR OF NUMBER OF RDGS AVERAGED
DACSEL      EQU NRDG+2          ADDRESS OF DAC SELECT DATA,1 BYTE
DATIN       EQU DACSEL+1        START OF DATA READ IN FROM GPIB,256 BYTES
RAMADR      EQU 3800H           ADDRESS OF RAM PROGRAM,CALLED BY CMD6
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 ENDMK           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 0FH         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 ORDER 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,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,080H      LOAD PRG WRD,A=OUTPUT,B&C=OUTPUT
            OUT PORT2       SEND TO PORT2
            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,0DH       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
            XRA A           ZERO ACC,SET OUTPUT VOLTAGE TO 0
            OUT PORT2A      BYTE SENT TO DACS
            MVI A,08H       LOAD 08 INTO ACC
            OUT PORT2B      HI ORDER NIB IS SENT TO DAC
            MVI A,06FH      LOAD 0110/1111 INTO ACC
            OUT PORT2C      STROBE PULSE HEIGHT DAC LOW
            MVI A,07FH      LOAD 0111/1111 INTO ACC
            OUT PORT2C      LOCK PULSE HEIGHT VALUE INTO DAC
```

```
            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
;           THE SECOND BYTE DETERMINES THE DELAY TIME BETWEEN S/H PULSES
CADDR       DW CMD0            PULSER SET UP SUBROUTINE
            DW CMD1            FAST READINGS AND CONVERSION,ASCII XMIT
            DW CMD2            FAST RDG,SEND OLD RDG WHILE PROCSNG NXT
            DW CMD3            FAST CALIBRATOR READINGS
            DW CMD4            NORMAL PULSE READING,LONG CONVERSION
;
;           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               SUBROUTINE TO TAKE DATA WRITTEN BY
;                              THE BUSS CONTROLLER AND SET UP THE
;                              PULSER.DATA STORED AT RAM ADR DATIN
;           CHARACTER          DATA FOR           FORMAT
;                   C          NO OF CHANNELS     1 CHR,1 BYTE
;                   O          OUTPUT VOLTAGE     1 CHR,4 BYTES
;                   P          PULSE ON/OFFTIME   1 CHR,8 BYTES
;                              READING INTERVALS   4*NCHS BYTES
;                   R          REF COMP VOLT LEVEL 1 CHR,4 BYTES
;                   A          READINGS AVERAGED 1 CHR,4 BYTES
;                   L          LEVEL SETTINGS     1 CHR,4*NCHS BYTES
```

```
;                   B           BALANCE VOLTAGE    1 CHR,4*NCHA BYTES
;                   G           GAIN SETTINGS      1 CHR,4*NCHA BYTES
;
CMDO        MVI A,PRT1O         SET PORT1 FOR OUTPUT
            OUT PORT1           SEND TO CONTROL WORD
            LXI H,DATIN+1       LOAD ADDRESS OF 2ND DATA IN BYTE
CMDOL       MOV A,M             LOAD CHARACTER BYTE FROM MEMORY
            INX H               INCREMENT DATA IN SET COUNTER
            SHLD DIAO           STORE 2 BYTE DATA COUNTER IN DAIO
            CPI A'C'            COMPARE TO CHR C,SET CHANNEL NO
            JZ CHNOST           JUMP TO CHANNEL SETTING ROUTINE
            CPI A'O'            COMPARE TO CHR O,SET OUTPUT LEVEL
            JZ OPTLEV           JUMP TO OUTPUT LEVEL SET ROUTINE
            CPI A'P'            COMPARE TO CHR P,SET PULSE TIMES
            JZ PTSET            JUMP TO PULSE TIME SET ROUTINE
            CPI A'R'            COMPARE TO CHR R,SET REF DISC LEVEL
            JZ REFLEV           JUMP TO SET REFERENCE COMPARATOR VOLT LEVEL
            CPI A'L'            COMPARE TO CHR L,SET DISCRIMINATOR LEVEL
            JZ DISLVL           JUMP TO SET DISCRIMINATOR LEVEL
            CPI A'B'            COMPARE TO CHR B,SET BALANCE VOLTAGE
            JZ BALVOL           JUMP TO SET BALANCE VOLTAGE LEVEL
            CPI A'G'            COMPARE TO CHR G,SET VOLTAGE GAINS
            JZ GAINST           JUMP TO SET THE GAIN OF THE CHANNELS
            CPI A'A'            COMPARE TO CHR A,SET NO RDGS AVERAGED
            JZ AVGRD            JUMP TO SET AVG RDG NO ROUTINE
CMDOL1      MVI A,PRT1I
            OUT PORT1           SET PORT1 FOR INPUT
            LXI H,DATIN         STORE ADDRESS FOR INPUT DATA INTO HL
            SHLD DIAD           STORE IN RAM
            MVI A,013H          ENABLE READ,WRITE OR EOI
            OUT INTE1           SEND TO 8291 INT ENA
            EI                  ENABLE INTERRUPT ON 8080A
            HLT                 WAIT FOR INTERRUPT OVER GPIB
;
;           CHNOST              PROGRAM SETS THE CHANNEL NUMBER,NCHA,4*NCHA
;                               8*NCHA AND ASCDAT+4*NCHA
;
CHNOST      MOV A,M             MOVE CHANNEL NO BYTE INTO ACC
            INX H               INCREMENT COUNTER
            SHLD DIAO           STORE FOR NEXT TIME
            SUI 030H            CONVERT TO NUMBER
            STA NCHAA           STORE NUMBER OF CHANNELS
            ADD A               DOUBLE NUMBER OF CHANNELS
            ADD A               4X NUMBER OF CHANNELS
            STA NCH4A           STORE 4X NUMBER OF CHANNELS
            ADD A               8X NUMBER OF CHANNELS
            STA NCH8A           STORE 8X NUMBER OF CHANNELS
            RAR                 SHIFT RIGHT,DOWN TO 4X NO CHANNELS
            LXI H,ASCDAT        LOAD START OF ASCDAT INTO HL
            MOV E,A             4*NCHA IN E
            MVI D,0             DE CONTAINS 4*NCHA
            DAD D               HL CONTAINS ASCDAT+4*NCHA
```

```
            SHLD ASC4NA       ASC4NA IN RAM CONTAINS ASCDAT+4*NCHA
            JMP CMDOL1        CONSTANTS THAT USE NCHA SET,GO BACK
;
;           PTSET             SETS THE PULSE ONTIME,PULSE OFF TIME
;                             AND THE DELAY TIME FROM THE PULSE
;                             START TO THE SAMPLE FOR SAMPLE & HOLD IC'S
;                             DATA MUST BE IN THE ORDER GIVEN ABOVE
;
PTSET       MVI D,0           ZERO D COUNTER
            LDA NCHAA         LOAD THE NUMBER OF CHANNELS INTO A
            ADI 2             INCREASE THE CHANNELS BY TWO
            MOV E,A           STORE NO CHANS+2 IN E
            JNZ TIMLP         LOOP BACK IF NOT DONE
            MVI A,03FH        STROBE MR LOW,THEN BACK HI
            OUT PORT2C        SEND 0011/1111 OUT
            MVI A,07FH        SEND 2C6 BACK HI
            OUT PORT2C        SEND 0111/1111 OUT
TIMLP       CALL CONV4        CONVERT 4 NOS ADDR BY HL TO BINARY
            MOV A,C           LOAD LOW BYTE FROM C TO A
            OUT PORT2A        SEND OUT
            MOV A,B           LOAD HI BYTE INTO A
            OUT PORT2B        SEND 16 BITS TO TIMERS
            MVI A,070H        LOAD 0111/0000 INTO A
            ADD D             ADD CHN NO FROM REG D
            OUT PORT2C        SEND TO PORT 2C
            ORI 0FH           OR ACC WITH 0000/1111
            OUT PORT2C        LOAD RDG INTO ADC
            INR D             INCREMENT CHA COUNT
            DCR E             DECREMENT E COUNTER
            JNZ TIMLP         LOOP BACK IF NOT DONE
            MVI A,0FFH        STROBE XTR HI TO START
            OUT PORT2C        SEND TO PORT
            MVI A,07FH        SEND 2C7 BACK LOW
            OUT PORT2C        SEND 0111/1111 OUT
            SHLD DIA0         RESTORE COUNTER IN DIA0
            JMP CMDOL1        DATA SET,GO BACK TO CMD LOOP
;
;           GAINST            SET THE GAIN LEVEL FOR
;                             EACH CHANNEL
;
GAINST      MVI A,080H        LOAD 1000/0000 ,SELECT GAIN DAC
            STA DACSEL        STORE IN DACSEL IN RAM (1B7 BIT)
            JMP DACWRT        JUMP TO DAC WRITE SUBROUTINE
;
;           DISLVL            SET THE DISCRIMINATOR VOLTAGE LEVEL
;                             FOR EACH CHANNEL
;
DISLVL      MVI A,040H        LOAD 0100/0000 ,SELECT DISCRIMINATOR DAC
            STA DACSEL        STORE IN RAM (1B6 BIT)
            JMP DACWRT        JUMP TO DAC WRITE SUBROUTINE
;
;           BALVOL            SET THE BALANCE VOLTAGE LEVEL
```

```
;                                   FOR EACH CHANNEL
;
BALVOL      MVI A,010H              LOAD 0001/0000 ,SELECT BALANCE ADC
            STA DACSEL              STORE IN RAM (1B4 BIT)
            JMP DACWRT              JUMP TO DAC WRITE SUBROUTINE
;
;           DACWRT                  WRITES TO THE DAC ADDRESSED IN DACSEL

DACWRT      MVI D,0                 ZERO D COUNTER
            LDA NCHAA               LOAD THE NUMBER OF CHANNELS INTO A
            MOV E,A                 STORE NO CHANS IN E
DACLP       CALL CONV4              CONVERT 4 NOS ADDR BY HL TO BINARY
            MOV A,C                 LOAD LOW BYTE FROM C TO A
            OUT PORT1A              SEND OUT
            MOV A,B                 LOAD HI BYTE INTO A
            OUT PORT1C              SEND 12 BITS TO ADC
            LDA DACSEL              LOAD DAC SELECT FROM RAM
            ADD D                   ADD CHN NO FROM REG D
            OUT PORT1B              SEND TO PORT 1B
            ANI 0FH                 AND WITH 0000/1111,ZERO HI NIB
            OUT PORT1B              LOCK RDG INTO ADC
            INR D                   INCREMENT CHA COUNT
            DCR E                   DECREMENT E COUNTER
            JNZ DACLP               LOOP BACK IF NOT DONE
            SHLD DIA0               RESTORE COUNTER IN DIA0
            JMP CMD0L1              DATA SET,GO BACK TO CMD LOOP


            OTPLEV                  SUBROUTINE SETS THE OUTPUT VOLTAGE
                                    LEVEL OF THE PULSE   FORMAT O,IH,IM,IL

OPTLEV      CALL CONV4              CONVERT NEXT 4 NUMBERS
            MOV A,C                 MOVE LOW BYTE INTO ACC
            OUT PORT2A              BYTE SENT TO DACS
            MOV A,B                 MOVE HI NIB FROM REG BC
            OUT PORT2B              HI ORDER NIB IS SENT TO DAC
            MVI A,06FH              LOAD 0110/1111 INTO ACC
            OUT PORT2C              STROBE PULSE HEIGHT DAC LOW
            MVI A,07FH              LOAD 0111/1111 INTO ACC
            OUT PORT2C              LOCK PULSE HEIGHT VALUE INTO DAC
            SHLD DIA0               RESTORE COUNTER IN DIA0
            JMP CMD0L1              DATA SET GO BACK

            REFLEV                  VOLTAGE LEVEL FOR REFERENCE
                                  COMPARATOR R HINIB,HIDNIB LONIB

REFLEV      CALL CONV4              CONVERT NEXT 4 NUMBERS
            MOV A,C                 MOVE LOW BYTE INTO ACC
            OUT PORT2A              BYTE SENT TO DACS
            MOV A,B                 MOVE HI NIB FROM REG BC
            OUT PORT2B              HI ORDER NIB IS SENT TO DAC
            MVI A,05FH              LOAD 0101/1111 INTO ACC
```

```
            OUT PORT2C          STROBE REFERENCE LEVEL DAC LOW
            MVI A,07FH          LOAD 0111/1111 INTO ACC
            OUT PORT2C          LOCK REFERENCE VOLTAGE LEVEL VALUE INTO DAC
            SHLD DIA0           RESTORE COUNTER IN DIA0
            JMP CMD0L1          DATA SET GO BACK
;
;           AVGRD               SET THE NUMBER OF RDGS TO BE
;                               AVERAGED IN NRDG  FORMAT A,IH,IL
;
AVGRD       CALL CONV4          CONVERT 4 NIBS TO 16 BINARY BITS
            MOV A,C             MOVE LOW ORDER RESULT INTO A
            STA NRDG            STORE IN RAM AT NRDG ADDR
            SHLD DIA0           STORE COUNTER FOR NEXT CHARACTER
            JMP CMD0L1          DATA SET,GO BACK
;
;     CONV4         TAKES 4 ASCII CHARACTERS ADDRESSED BY HL
;                   IN RAM AS TOPNIB,HINIB,MIDNIB,LONIB AND CONVERTS
;                   THEM TO A 16 BIT BINARY NUMBER IN BC
;
CONV4       XRA A               ZERO ACCUMULATOR,CLEAR FLAGS
            MOV A,M             MOVE TOP NIB INTO A
            SUI 030H            CONVERT ASCII BYTE TO NUMBER
            CPI 10              COMPARE TO 10 DECIMAL
            JM CONVL0           JUMP AROUND IF .LT.10
            SUI 7               SUBTRACT 7 IF A,B,C,D,E,OR F
CONVL0      RLC                 ROTATE LEFT 4 TIMES
            RLC                 MOVE TOP NIB INTO HI ORDER
            RLC
            RLC
            MOV B,A             STORE IN REG B
            INX H               ADDRESS NEXT NIB
            MOV A,M             MOVE HI NIB INTO ACC
            SUI 030H            CONVERT ASCII BYTE TO NUMBER
            CPI 10              COMPARE TO 10 DECIMAL
            JM CONVL            JUMP AROUND IF .LT.10
            SUI 7               SUBTRACT 7 IF A,B,C,D,E,OR F
CONVL       ADD B               ADD TOP NIB TO HI NIB
            MOV B,A             STORE IN REGISTER B
            INX H               ADDRESS NEXT NIB
            MOV A,M             MOVE MID NIB INTO ACC
            SUI 030H            CONVERT ASCII BYTE TO NUMBER
            CPI 10              COMPARE TO 10 DECIMAL
            JM CONVL1           JUMP AROUND IF .LT.10
            SUI 7               SUBTRACT 7 IF A,B,C,D,E,OR F
CONVL1      RLC                 ROTATE ACC LEFT 4 TIMES
            RLC                 MOVE NIBBLE INTO HI ORDER
            RLC
            RLC
            MOV C,A             TEMP STORE IN C
            INX H               GO TO LOW ORDED NIB
            MOV A,M             LOAD IN ACC
            SUI 030H            CONVERT TO NUMBER
```

```
            CPI 10              COMPARE TO 10 DECIMAL
            JM CONVL2           JUMP AROUND IF .LT.10
            SUI 7               SUBTRACT 7 IF A,B,C,D,E,OR F
CONVL2      ADD C               ADD MID NIB TO LO NIB
            MOV C,A             STORE IN REG C
            INX H               SET UP NEXT COMMAND CHARACTER
            RET                 RETURN TO CALLER
;
;           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 START PULSE IF WE ARE STILL READING IT

            MVI A,020H          STROBE ADC 0010/0000,BIT B5 HI
            OUT PORT1B
            XRA A               THEN STROBE 0000/0000,BIT B5 LO
            OUT PORT1B          A START COMMAND HAS BEEN SENT TO ADC'S
            LHLD ASC4NA         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
            JMP RDGOUT          READ ADC'S,SET UP DATA FOR TRANSMIT,HALT
;
;           CMD3                LOOP FOR FAST CALIBRATION READINGS WITH NO
;                               STORAGE IN THE COMP9B MICROCOMPUTER
;
CMD3        MVI A,00            SET FOR CALIBRATE
            OUT PORT2A          SET RELAY FOR CALIBRATE
            CALL ADBUS          DON'T START PULSE IF WE ARE STILL READING IT

            LDA NCHAA           SET THE NUMBER OF S&H CHANNELS IN RAM
            STA CNTSH           STORE
            LDA CNTSH           LOOK AT COUNTER VALUE
            DCR A               DECREMENT COUNTER
            STA CNTSH           RESTORE COUNTER
            MVI A,PRT1I
            OUT PORT1           SET PORT1 FOR INPUT
;           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
            MVI A,020H          STROBE ADC 0010/0000,BIT B5 HI
            OUT PORT1B
            XRA A               THEN STROBE 0000/0000,BIT B5 LO
            OUT PORT1B          A START COMMAND HAS BEEN SENT TO ADC'S
```

```
            LHLD ASC4NA       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
            CALL ADBUS        WAIT FOR RDG TO END BEFORE DATA IS READ
            JMP RDGOUT        READ ADC'S,SET UP DATA FOR TRANSMIT,HALT
;
;           CMD4              LOOP TO TAKE READINGS AND SEND THEM OUT
;
CMD4        MVI A,020H        STROBE ADC 0010/0000,BIT B5 HI
            OUT PORT1B
            XRA A             THEN STROBE 0000/0000,BIT B5 LO
            OUT PORT1B        A START COMMAND HAS BEEN SENT TO ADC'S
            CALL ADBUS        HANG IN LOOP UNTIL ADC READY
            LDA NRDG          LOAD THE NUMBER OF RDGS AVERAGED
            MOV B,A           STORED INTO B
            LDA NCHAA         LOAD THE NUMBER OF CHANNELS INTO A
            MOV C,A           MOVE NUMBER INTO C REG
            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
            LDA NCH4A         16 BYTES OF DATA TO BE CONVERTED
            MOV C,A           STORE IN C COUNTER
            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
            MVI A,0FFH        SET ACCUMULATOR TO 1111/1111
            OUT PORT2B        TURN ALL SAMPLE & HOLD CHIPS TO SAMPLE
            STA SHDAT         RESET SAMPLE AND HOLD DATA
            LXI H,ASCDAT      LOAD ADDRESS OF DATA TO BE SENT
            LDA NCH8A         NUMBER OF CHARACTERS TO BE SENT
            MOV C,A           NUMBER OF CHARACTERS IN C REG
            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
;
;             RDGOUT          READ EACH ADC AND CONVERT TO ASCII REP OF BCD
;                             SET THE SYSTEM FOR OUTPUT OVER THE BUSS
RDGOUT      LDA NCHAA         LOAD CHANNEL COUNT
            STA NCH           STORE IN RAM,CH NO GOES FROM 0 TO 7
ADRDLP      ADI 07H           CHANNEL SEL ON B0,B1,B2;ENABLE ON B3
            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 0FH         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 0F0H        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 0FH         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 0F0H        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 ADRDLP          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
            LDA NCH4A           NUMBER OF CHARACTERS TO BE SENT
            MOV C,A             NUMBER OF CHARACTERS MOVED TO C REG
            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
;
;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       LDA NCHAA           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 ADCONVERTER 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 08H             ADD 0000/1000 SELECT ADC'S,DESELECT DACS
            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.
            XRA A              LEAVE START LO,DESELECT ADCONVERTERS
            OUT PORT1B         SEND TO PORT1B
            RET                 RETURNS WITH POS. ANSWER ON AF STACK.
;
;           ADBUS              HANG HERE UNTIL ALL BUSY LINES OF THE DATEL
;                        ADC-ET12B ANALOG TO DIGITAL CONVERTERS GO 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        - GENERATES A TIME DELAY,DEPENDING ON THE CONSTANT LOADED
;                 INTO BC, # CYCLES = 20*FFFF/(BC)  ABOUT 10 USEC/LOOP
;           WITH 18.432 MHZ CLOCK,TIME=.640/BC SEC OR BC=.640/TIME(SEC)
DELAY       MOV A,C             MOVE CONTENTS OF C REG TO A
            ADD B               CHECK REG BC FOR ZERO
            RZ                  RETURN IF BC ZERO
            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 ARE OVER WRITTEN WITH BCD DATA
;
HEXDEC      LXI H,RAWDA+3       START WITH LO BYTE OF 1ST RAW DATA WRD
            LDA NCHAA           LOAD THE NUMBER OF CHANNELS TO BE CONVERTED
            MOV C,A             MOVE NUMBER OF CHANNELS TO BE CONVERTED TO C

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
;
;           SHLD                PROGRAM TO RUN THE SAMPLE
;                               AND HOLD CIRCUITS (28.5 USECS)
;
SHLD        LDA SHDAT           LOAD SAMPLE AND HOLD DATA WORD
            ANI 07FH            SET HIGHEST BIT TO ZERO,KEEP OTHERS SAME
            RLC                 ROTATE ACCUMULATOR LEFT
            STA SHDAT           STORE FOR NEXT TIME
            OUT PORT2B          TURN NEXT S&H LOW,SWITCH TO HOLD
            RET                 GO BACK TO CALLER
;
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
```

PORT 1 INTERFACE OF NDT COMP 9B

Port ID.	Function	I/O Assignment
1C7		Input
1C6		Input
1C5		Input
1C4	ADC Busy, Input to Port	Input
1C3	MSB Data Line from ADC;Bal.,Level,Gain	Bi-directional
1C2	Data Line from ADC	Bi-directional
1C1	Data Line from ADC	Bi-directional
1C0	Data Line from ADC	Bi-directional
1A7	Data Line from ADC	Bi-directional
1A6	Data Line from ADC	Bi-directional
1A5	Data Line from ADC	Bi-directional
1A4	Data Line from ADC	Bi-directional
1A3	Data Line from ADC	Bi-directional
1A2	Data Line from ADC	Bi-directional
1A1	Data Line from ADC	Bi-directional
1A0	LSB Data Line	Bi-directional
1B7	Gain Enable	Output
1B6	Level Enable (74F138-E3)	Output
1B5	ADC Start	Output
1B4	Bal. Enable (74F138-E3)	Output
1B3	ADC Enable (E3)	Output
1B2	ADC A2, Bal.,Level,Gain,Chip Select	Output
1B1	ADC A1, Bal.,Level,Gain,Chip Select	Output
1B0	ADC A0, Bal.,Level,Gain,Chip Select	Output

PORT 2 INTERFACE OF NDT COMP 9B

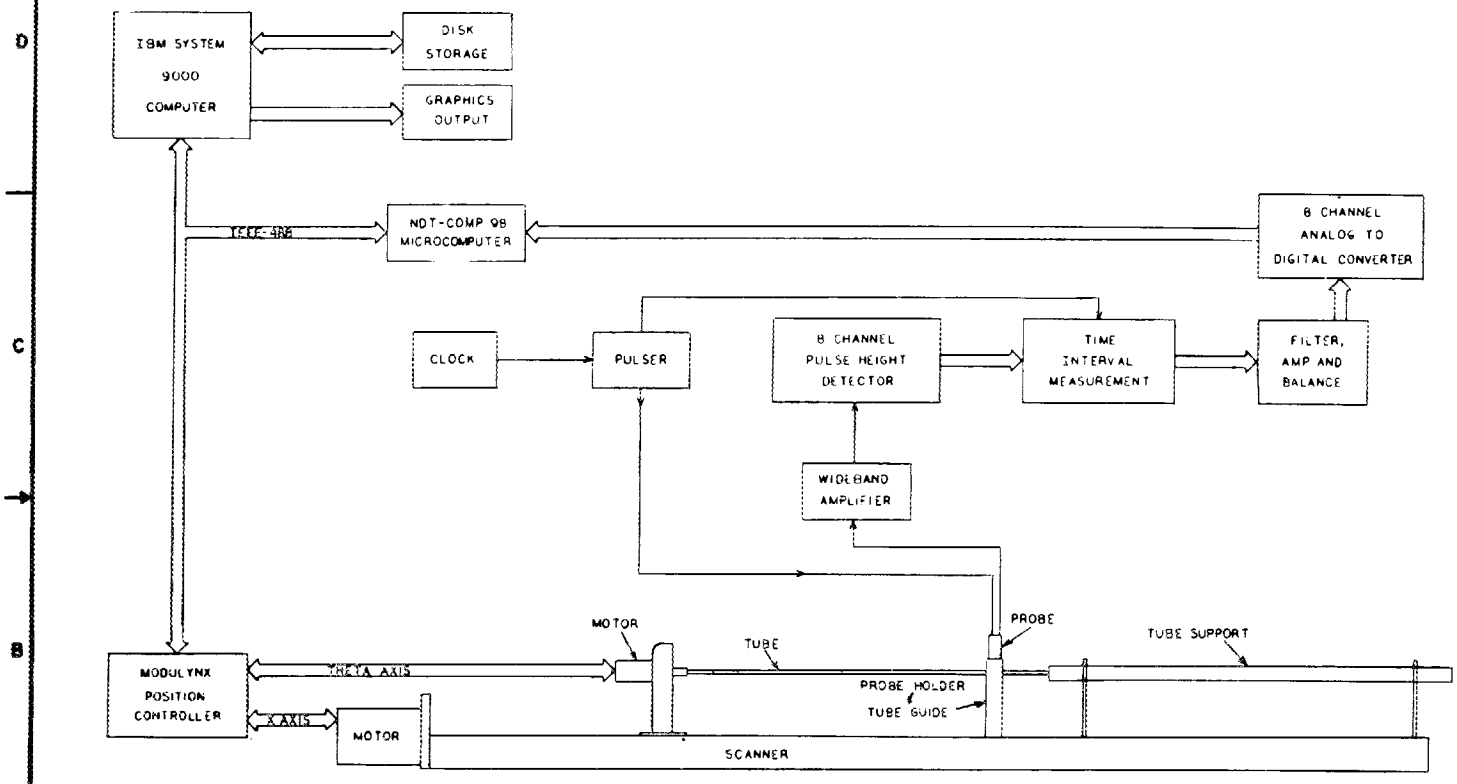
Port ID.	Function	I/O Assignment
2C7	Divider XTR	Output
2C6	Divider MR	Output
2C5	Reference Level (Chip Select AD394)	Output
2C4	Pulse Height (Chip Select AD394)	Output
2C3		Output
2C2		Output
2C1		Output
2C0		Output
2B7	MSB Data Line,Divider	Output
2B6	Data Line,Divider	Output
2B5	Data Line,Divider	Output
2B4	Data Line,Divider	Output
2B3	Data Line,Divider,MSB,Pulse Ht.,Ref.Level DACs	Output
2B2	Data Line,Divider,Pulse Height DAC,Ref.Level DAC	Output
2B1	Data Line,Divider,Pulse Height DAC,Ref.Level DAC	Output
2B0	Data Line,Divider,Pulse Height DAC,Ref.Level DAC	Output
2A7	Data Line,Divider,Pulse Height DAC,Ref.Level DAC	Output
2A6	Data Line,Divider,Pulse Height DAC,Ref.Level DAC	Output
2A5	Data Line,Divider,Pulse Height DAC,Ref.Level DAC	Output
2A4	Data Line,Divider,Pulse Height DAC,Ref.Level DAC	Output
2A3	Data Line,Divider,Pulse Height DAC,Ref.Level DAC	Output
2A2	Data Line,Divider,Pulse Height DAC,Ref.Level DAC	Output
2A1	Data Line,Divider,Pulse Height DAC,Ref.Level DAC	Output
2A0	LSB Data Line,Divider,Pulse Ht.DAC,Ref.Level DAC	Output

R20F Reflection Coil Specifications

(All lengths in inches; one inch=25.4 mm)

Quantity	Each Driver Coil	Each Pickup Coil	Copper Focusing Ring
I.D.	0.030	0.014	0.062
O.D.	0.050	0.028	0.100
Length	0.012	0.004	0.016
Min.Liftoff	0.002	0.000	
Turns	20	28	
Size Wire	#44 AWG	#52 AWG	
Turns/Layer	5	4	
No.Layers	4	7	
Inductance	0.39 uH	0.43 uH	
D.C.Resistance	0.56 Ohms	4.66 Ohms	

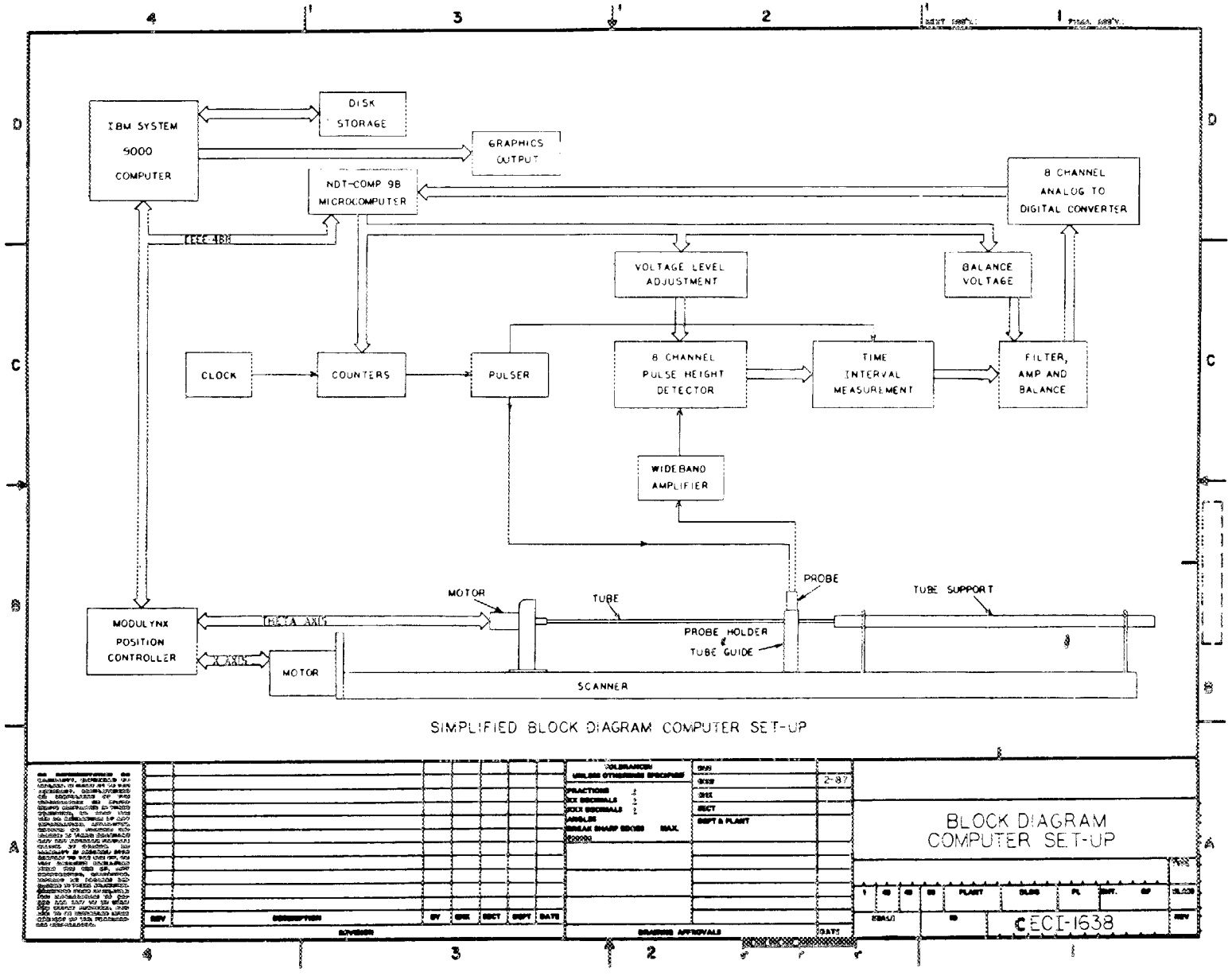
INSTRUMENT WIRING DIAGRAMS



SIMPLIFIED BLOCK DIAGRAM MANUAL SET-UP

BLOCK DIAGRAM
MANUAL SET-UP

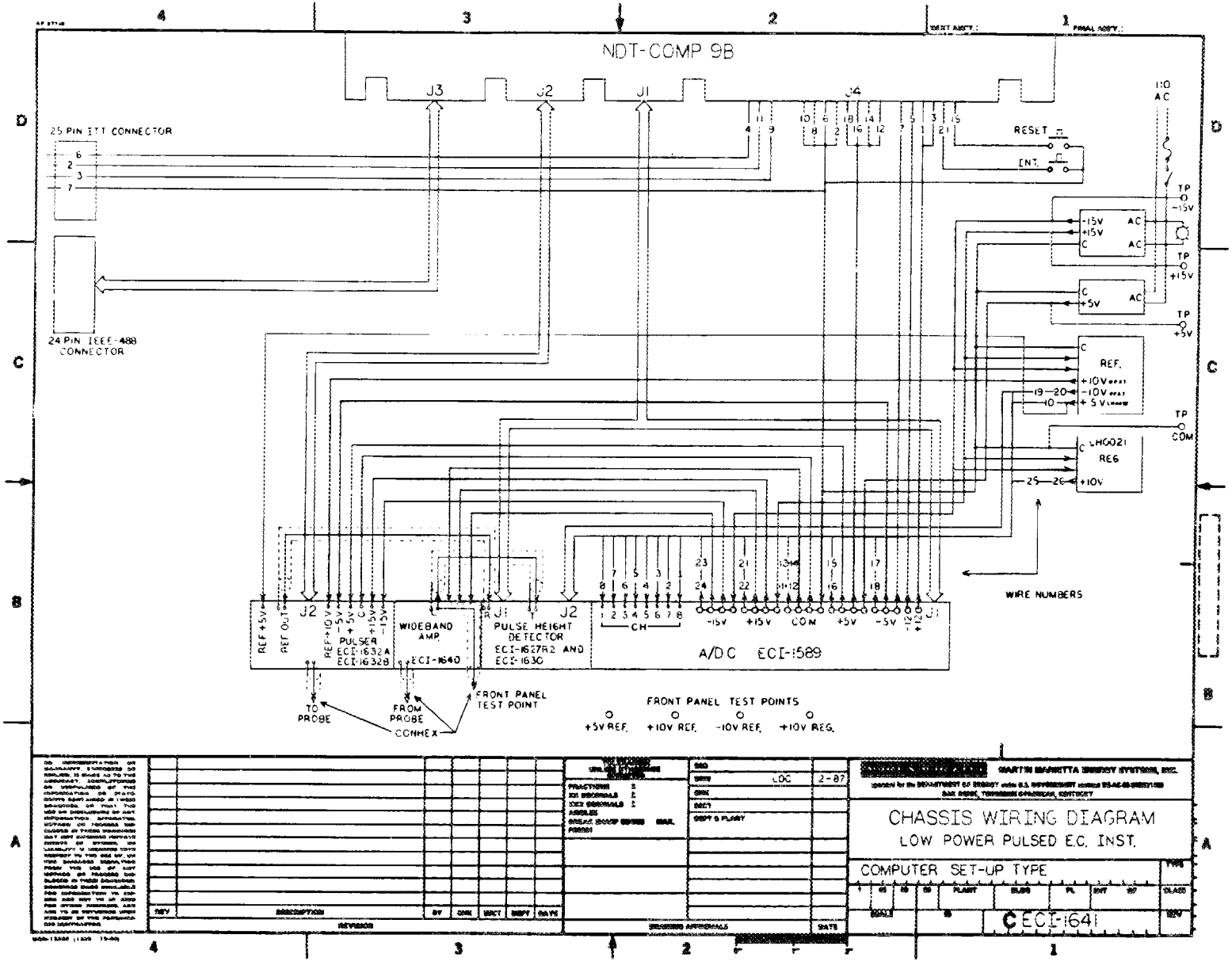
CECI-1639R1

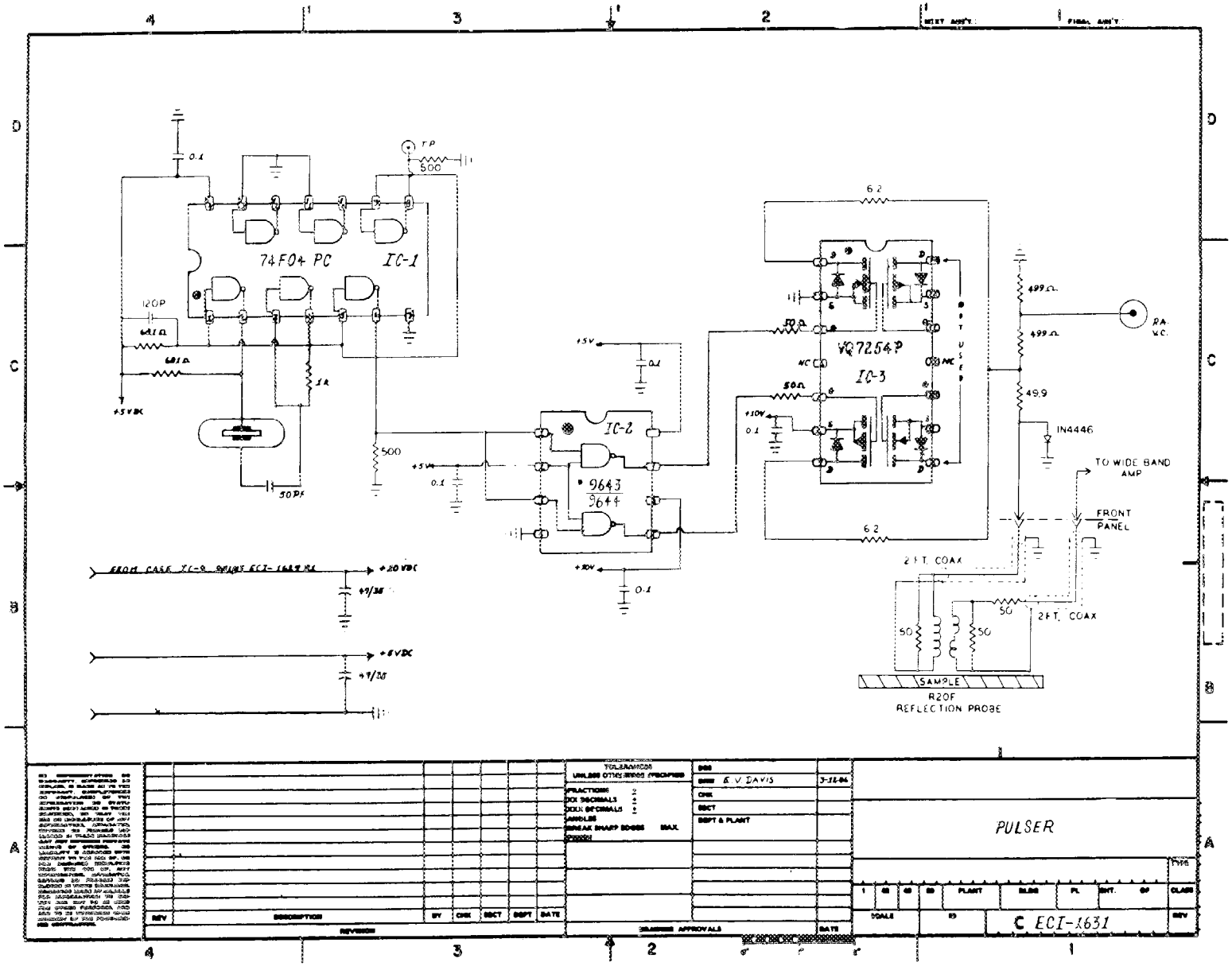


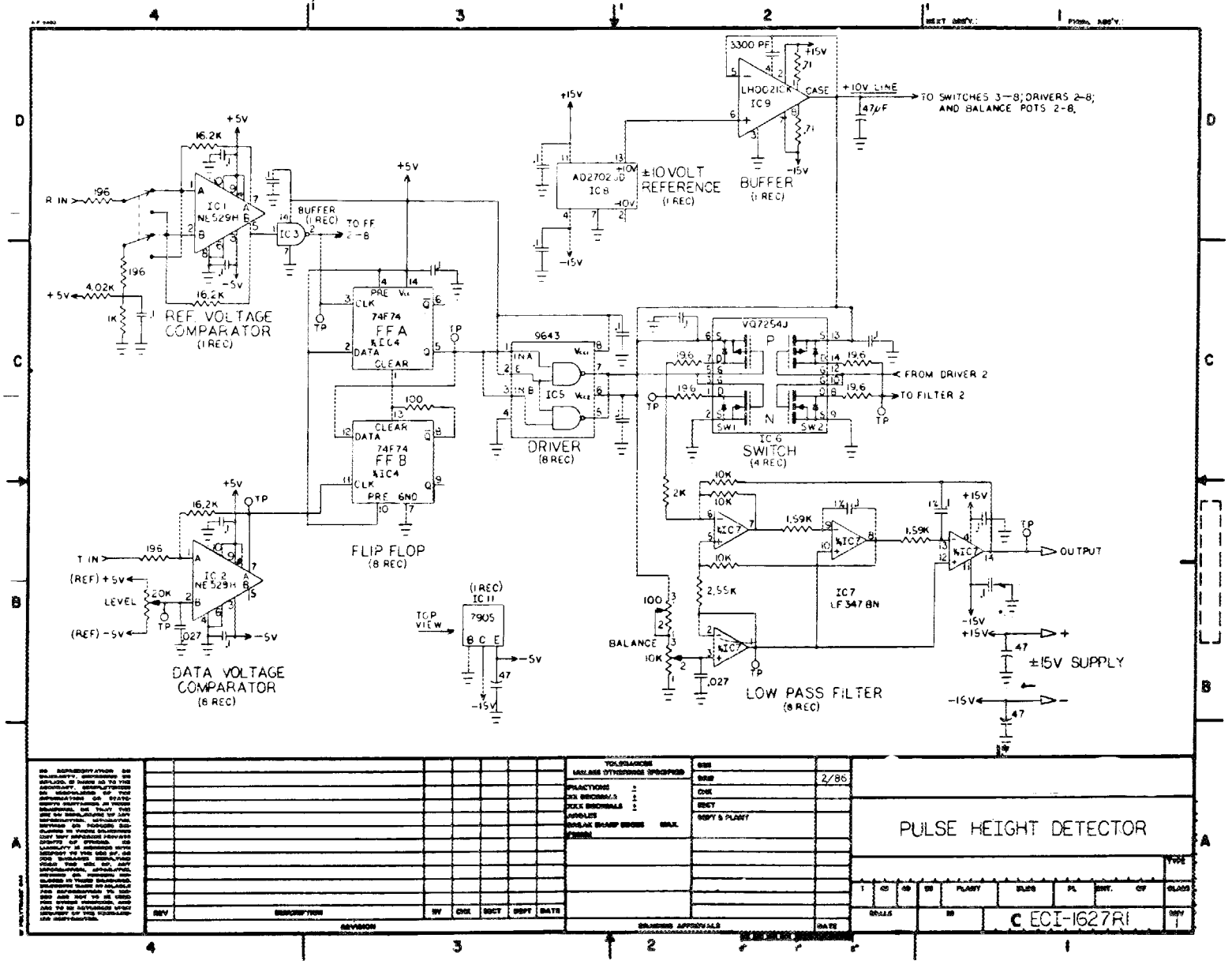
SIMPLIFIED BLOCK DIAGRAM COMPUTER SET-UP

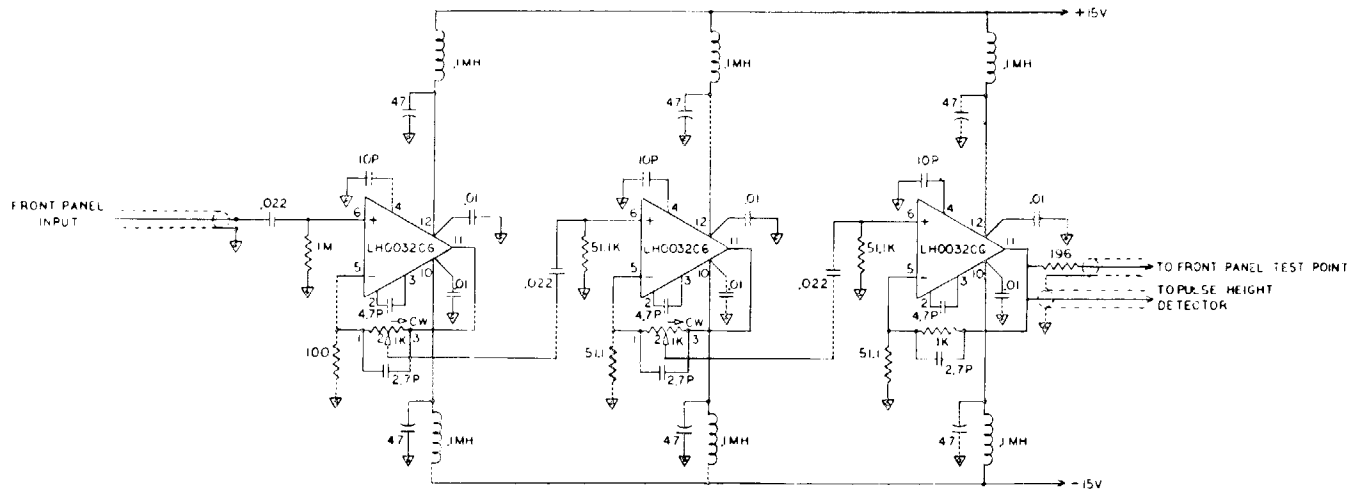
BLOCK DIAGRAM COMPUTER SET-UP

CECI-1638


NDT-COMP 9B
J3
J2
J1
J4
25 PIN ITT CONNECTOR
24 PIN IEEE-488 CONNECTOR
RESET
INT.
1:0 AC
TP -15V
TP +15V
TP +5V
TP COM
-15V
+15V
C
AC
+5V
REF.
+10V REF.
-10V REF.
+5V REF.
LH0021 REG
+10V
WIRE NUMBERS
REF +5V
REF OUT
REF +10V
PULSER ECI-1632A ECI-1632B
WIDEBAND AMP. ECI-1640
PULSE HEIGHT DETECTOR ECI-1627R2 AND ECI-1630
A/D C ECI-1589
CH
-15V
+15V
COM
+5V
-5V
-12V
+12V
TO PROBE
FROM PROBE
CONNEX
FRONT PANEL TEST POINT
FRONT PANEL TEST POINTS
+5V REF.
+10V REF.
-10V REF.
+10V REG.
MARTIN MARIETTA ENERGY SYSTEMS, INC.
LOC
2-87
CHASSIS WIRING DIAGRAM
LOW POWER PULSED E.C. INST.
COMPUTER SET-UP TYPE
ECI-1641
REVISION
DESCRIPTION
DRAWING APPROVALS
DATE


74F04 PC
IC-1
TP
500
0.1
120P
681Ω
681Ω
+5VDC
1K
50PF
500
+5V
0.1
IC-2
9643
9644
+5V
0.1
+10V
0.1
6.2
VQ7254P
IC-3
50Ω
NC
NC
6.2
499Ω
499Ω
49.9
IN4446
TO WIDE BAND AMP
FRONT PANEL
2 FT. COAX
50
50
50
2 FT. COAX
SAMPLE
R20F
REFLECTION PROBE
+20 VDC
+6VDC
PULSER
E. V. DAVIS
3-12-84
C ECI-1631


R IN
196
16.2K
IC1
NE529H
REF. VOLTAGE COMPARATOR
(1 REC)
4.02K
1K
BUFFER
(1 REC)
IC3
TO FF 2-8
74F74
FF A
&IC4
PRE
CLK
DATA
CLEAR
100
74F74
FF B
&IC4
FLIP FLOP
(8 REC)
T IN
(REF) +5V
LEVEL
20K
(REF) -5V
IC2
NE529H
DATA VOLTAGE COMPARATOR
(8 REC)
TOP VIEW
(1 REC)
IC11
7905
9643
IC5
DRIVER
(8 REC)
AD2702JD
IC8
±10 VOLT REFERENCE
(1 REC)
3300 PF
LH0021CK
IC9
BUFFER
(1 REC)
+10V LINE
TO SWITCHES 3-8; DRIVERS 2-8; AND BALANCE POTS 2-8.
VQ7254J
FROM DRIVER 2
TO FILTER 2
IC6
SWITCH
(4 REC)
IC7
LF347BN
OUTPUT
BALANCE
LOW PASS FILTER
(8 REC)
±15V SUPPLY
PULSE HEIGHT DETECTOR
C ECI-1627R1
2/86



FOR USE WITH ECI-1641 WIRING

MARTIN MARIETTA ENERGY SYSTEMS, INC.

operated for the DEPARTMENT OF ENERGY under U.S. GOVERNMENT contract DE-AC-05-84OR21400

OAK RIDGE, TENNESSEE 37830 OPERATION, KENTUCKY

WIDEBAND AMPLIFIER

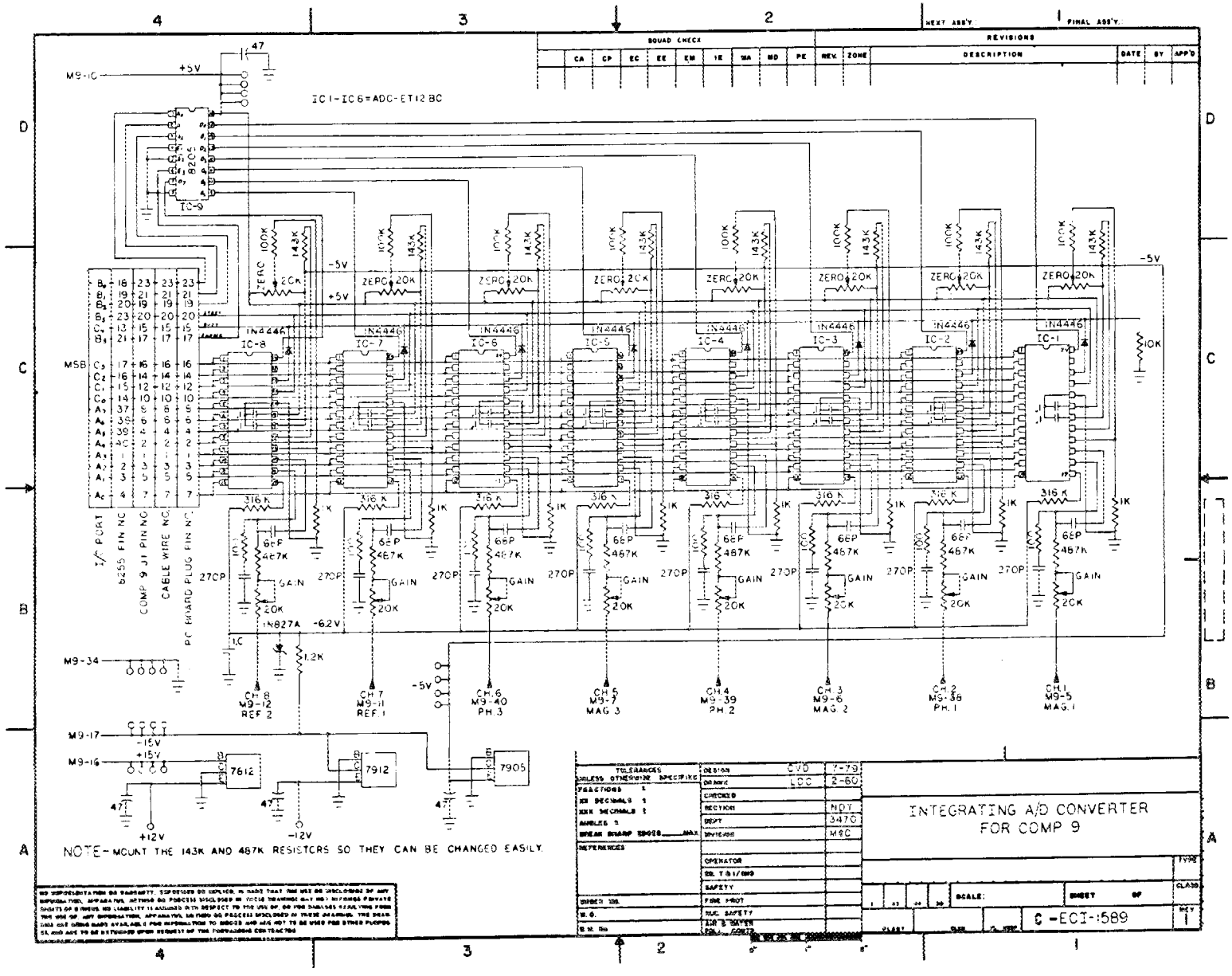
PULSED E.C. INST.

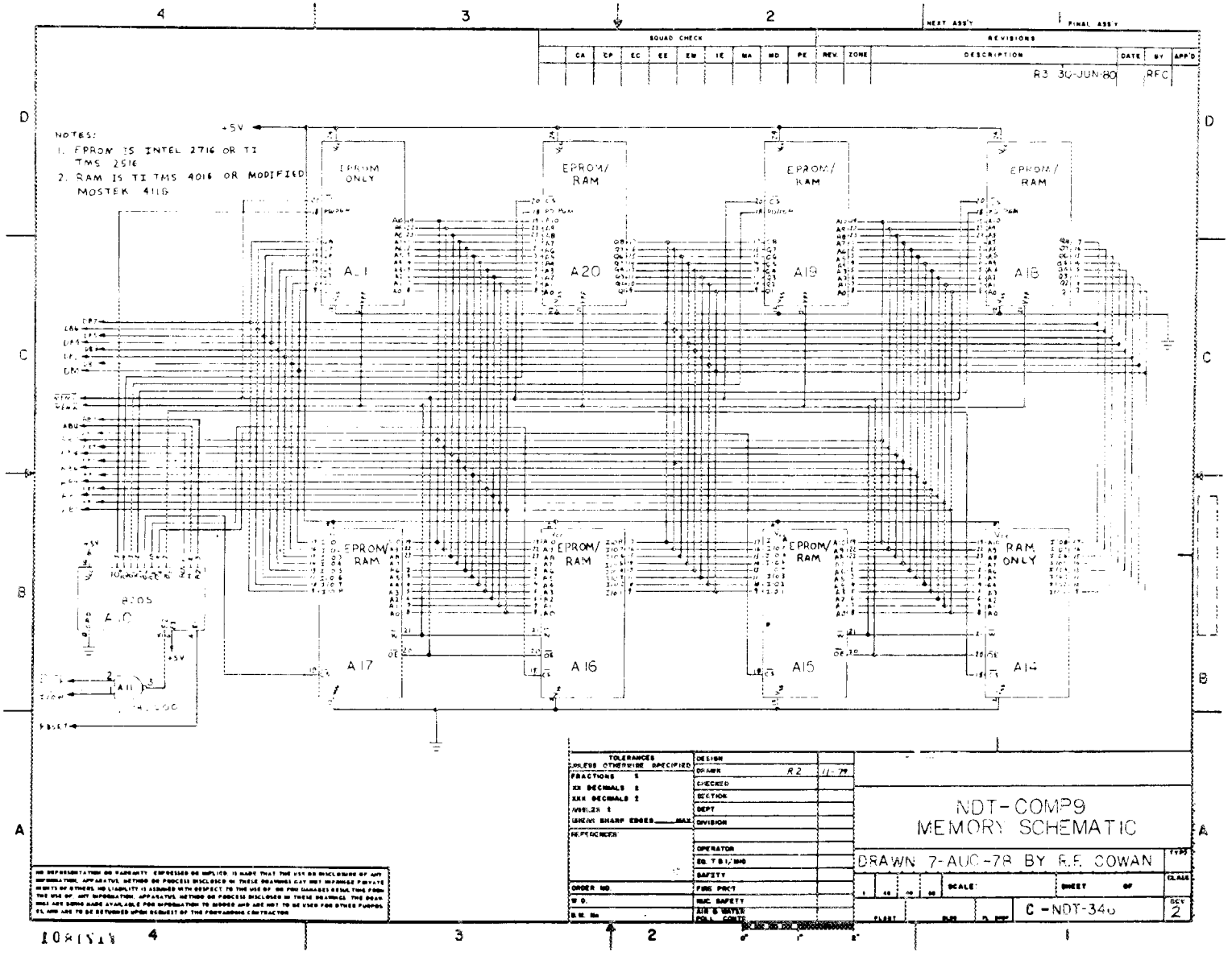
(LOW POWER INST)

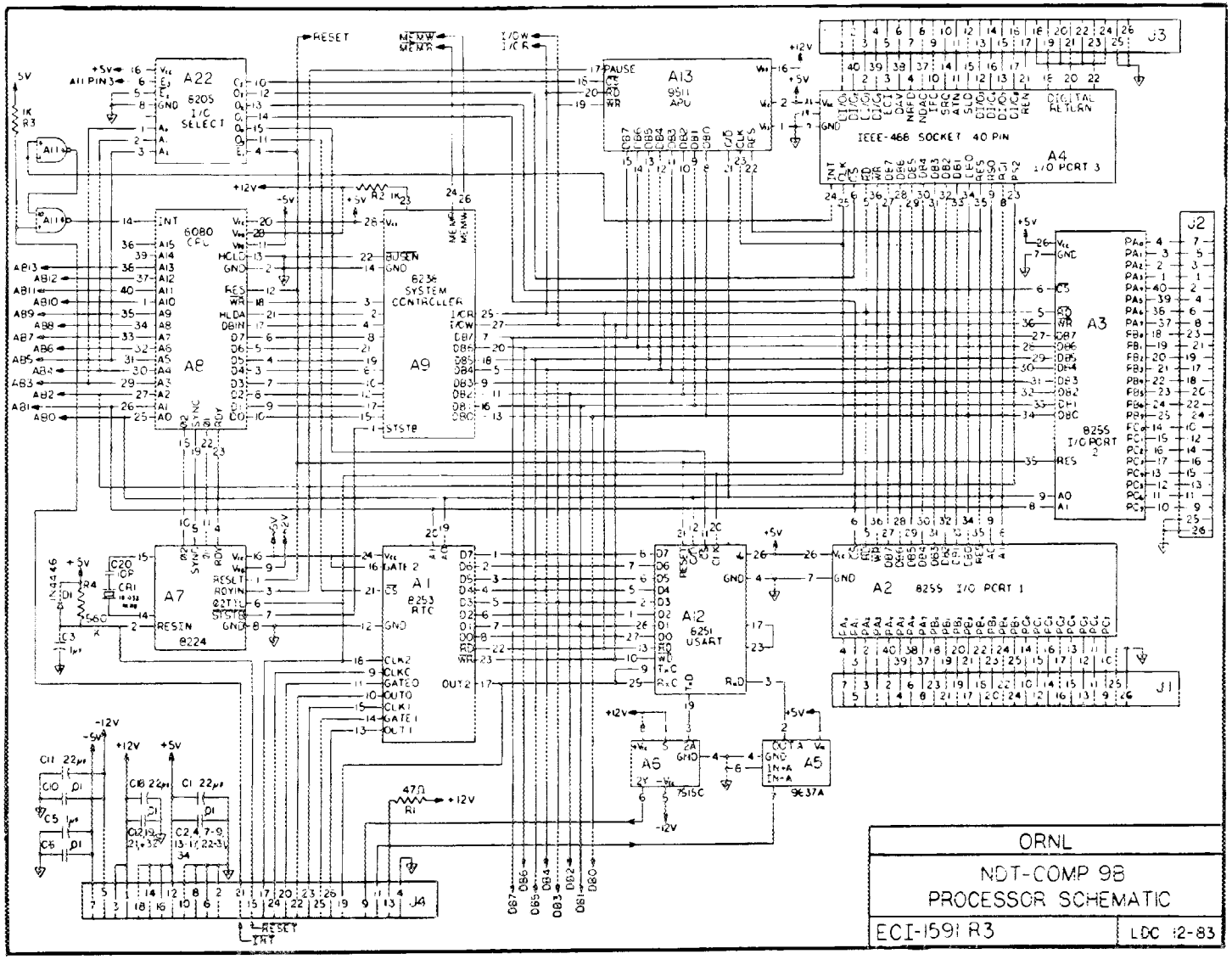
LDC 2-87

ECI-1640

REV	DESCRIPTION	BY	CHK	SECT	DEPT	DATE


IC1-IC6=ADC-ET12BC
M9-1C
+5V
47
8205
IC-9
MSB
I/O PORT
8255 PIN NO.
COMP 9 J1 PIN NO.
CABLE WIRE NO.
PC BOARD PLUG PIN NO.
ZERO 20K
100K
143K
-5V
+5V
1N4446
IC-8
IC-7
IC-6
IC-5
IC-4
IC-3
IC-2
IC-1
316K
1K
10K
68P
467K
270P
GAIN
20K
1N827A
-6.2V
1.2K
M9-34
CH.8 M9-12 REF.2
CH.7 M9-11 REF.1
CH.6 M9-40 PH.3
CH.5 M9-7 MAG.3
CH.4 M9-39 PH.2
CH.3 M9-6 MAG.2
CH.2 M9-38 PH.1
CH.1 M9-5 MAG.1
M9-17
-15V
M9-16
+15V
7612
7912
7905
+12V
-12V
NOTE-MOUNT THE 143K AND 467K RESISTORS SO THEY CAN BE CHANGED EASILY.
INTEGRATING A/D CONVERTER
FOR COMP 9
C-ECI-1589
REV 1


NOTES:
1. EPROM IS INTEL 2716 OR TI TMS 2516
2. RAM IS TI TMS 4016 OR MODIFIED MOSTEK 4116
+5V
EPROM ONLY
EPROM/ RAM
RAM ONLY
A1
A20
A19
A18
A17
A16
A15
A14
A10
A11
R205
RESET
NDT-COMP9
MEMORY SCHEMATIC
DRAWN 7-AUG-78 BY R.E. COWAN
C-NDT-340
R3 30-JUN-80 RFC
REVISIONS
SQUAD CHECK
TOLERANCES UNLESS OTHERWISE SPECIFIED
DESIGN
DRAWN
CHECKED
SECTION
DEPT
DIVISION
OPERATOR
SAFETY
FIRE PROT
NUC SAFETY
AIR & WATER POLL. CONTR.
ORDER NO.
W.O.
SCALE:
SHEET OF
SIZE
C
REV
2


A22
8205
I/O
SELECT
A8
8080
CPU
A9
8236
SYSTEM
CONTROLLER
A13
9511
APU
A4
I/O PORT 3
IEEE-488 SOCKET 40 PIN
DIGITAL
RETURN
A3
8255
I/O PORT
2
A7
8224
A1
8253
RTC
A12
8251
USART
A2
8255 I/O PORT 1
A6
7515C
A5
9637A
J1
J2
J3
J4
RESET
INT
ORNL
NDT-COMP 98
PROCESSOR SCHEMATIC
ECI-1591 R3
LDC 12-83

ORNL-6408
Distribution
Categories UC-23,
-25, -38

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
8–12. W. E. Deeds
13–42. C. V. Dodd
43. C. R. Kennedy
44. K. A. Kitzke
45–49. D. W. Koerner
50. F. M. Haggag
51 O. F. Kimball
52. R. K. Nanstad
53. R. W. McClung
54–58. C. A. Pickett
59. P. L. Rittenhouse
60 G. M. Slaughter
61. J. H. Smith
62–64. P. T. Thornton
65. J. R. Weir
66. F. W. Wiffen
67. G. Y. Chin (Consultant)
68. H. E. Cook (Consultant)
69. F. F. Lange (Consultant)
70. T. E. Mitchell (Consultant)
71. W. D. Nix (Consultant)
72. J. C. Williams (Consultant)

EXTERNAL DISTRIBUTION

73. ARGONNE NATIONAL LABORATORY, 9700 S. Cass Avenue, Argonne, IL 60439

O. K. Chopra

74. BABCOCK AND WILCOX COMPANY, Fossil Power Generation Division, 20 South Van Buren Avenue, Barberton, OH 44203

M. Gold

75. BURNS AND ROE, INC., 700 Kinderkamack Road, Oradell, NJ 07469

C. S. Ehrman

76. COMBUSTION ENGINEERING, INC., 911 W. Main Street, Chattanooga, TN 37402

D. A. Canonico

77–78. COMBUSTION ENGINEERING, INC., 1000 Prospect Hill Road, Windsor, CT 06095

P. E. C. Bryant
A. L. Gaines

79. ELECTRIC POWER RESEARCH INSTITUTE, 3412 Hillview Avenue, P.O. Box 10412, Palo Alto, CA 94303

R. I. Jaffee

80. ELECTRIC POWER RESEARCH INSTITUTE-CONSOLIDATED MANAGEMENT ORGANIZATION, Suite 220, One Energy Center, Naperville, IL 60566

D. R. Riley

81. FOSTER WHEELER DEVELOPMENT CORPORATION, 12 Peach Tree Hill Road, Livingston, NJ 07039

W. R. Apblett

82. GA TECHNOLOGIES, INC., P.O. Box 86508, San Diego, CA 92138

83–84. GENERAL ELECTRIC COMPANY, Nuclear Systems Technology Operations, 310 DeGuigne Drive, P.O. Box 3508, Sunnyvale, CA 94088

P. J. Ring
P. Roy

85. HANFORD ENGINEERING DEVELOPMENT LABORATORY, P.O. Box 1970, Richland, WA 99352

L. D. Blackburn

86. MARTIN MARIETTA AEROSPACE, Department 3773, Michoud Division, P.O. Box 29304, New Orleans, LA 70189

G. W. Scott

87–88. ROCKWELL INTERNATIONAL, Rocketdyne Division, 6633 Canoga Avenue, Canoga Park, CA 91304

T. L. Anderson
W. T. Lee

89. WESTINGHOUSE ELECTRIC CORPORATION, Advanced Energy Systems Division, P.O. Box 10864, Pittsburgh, PA 15236

R. W. Buckman

90–93. DOE, Washington, DC 20545

Office of Fusion Energy
T. C. Reuther

Office of Reactor Systems Development and Technology
C. C. Bigelow
R. J. Neuhold
B. J. Rock

94. DOE, OAK RIDGE OPERATIONS OFFICE, P.O. Box E, Oak Ridge, TN 37831

Office of Assistant Manager for Energy Research and Development

95–328. DOE, TECHNICAL INFORMATION CENTER, Office of Information Services P.O. Box 62, Oak Ridge, TN 37831

For distribution as shown in DOE/TIC-4500, Distribution Categories UC-23 (Radioisotope and Radiation Applications), UC-25 (Materials), and UC-38 (Engineering and Equipment).

☆U.S. GOVERNMENT PRINTING OFFICE: 1987-748-168/60087