AN ULTRASONIC TRANSDUCER FIELD SCANNER SYSTEM

BY

DAVID GARY PADGITT

B.S., University of Illinois, 1983

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1985

Urbana, Illinois

DEDICATION

This thesis is dedicated to my parents, who have provided me with many things, both visible and invisible.

ACKNOWLEDGEMENTS

This space is all mine, to write whatever I wish. I would like to share it with the following people who in some specific or remote way played a part in my life during the development of this thesis. This list is not necessarily complete.

First of all, thanks to Dr. Richard Magin who is not only a graduate advisor but a friend, who always invited me to happy hour, and let me help assemble his kitchen table. Similar thanks to Drs. Clif Burdette, Charles Cain, Leon Frizzell, and Steve Goss, but without the kitchen table. Thanks to Dr. Steve (Toroid Man) Foster for being both a good and bad influence. Thanks to Wanda Elliott for helping prepare this manuscript.

Thanks also to the following people:

HSL and BRL

Paul Benkeser
Tim Benson
Bob Cargnoni
Tom Cavicchi
Joe Cobb
Paul Embree
Aaron Field
Dave Grethen
Panos Hadjimitsos
John McCarthy
Bob Morimoto
Ken Ocheltree
Mary Scontras

URI THERM-X

Chris Badger
John Gedymin
Stephenie Graham
Jeff Kouzmanoff
Therese Liston
Paul Neubauer
Scott Walter

Steve Silverman

Harold Underwood

Miscellaneous Friends

Diane Agemura

Carol Gura

John Jaksch

Mike Kulas

Arnold Levitan

Tracy Mugishima

Dean Mumme

Carol Nakamura

Minh Nguyen

Janet Padgitt

Rudy Reavis

Mike Tanquary

Chuck Webster

Robert Wilson

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Current cancer research has shown that the controlled application of thermal energy, by means of ultrasonic waves, to certain cancerous tumors can be an effective method of therapy [1-5]. In order to precisely control the application of these sound waves to a cancerous tumor, detailed knowledge of its physical geometry, thermal density, and blood flow is required. Additionally, the heating characteristics of the ultrasonic applicator must be quantified. The purpose of this thesis is to describe the development of an automated field scanner system which has been designed to aid in characterizing such ultrasonic transducers. Although such scanners are commercially available, they are specialized pieces of test equipment, and not surprisingly very expensive. Furthermore, they lack the versatility of a custom designed system. This system has been created, in part, to characterize the 16-element applicator to be used in the URI THERM-X Sonotherm 1000 cancer therapy system. The research described in this thesis has been primarily supported by URI THERM-X.

Briefly stated, the goal of this project is to develop the necessary hardware and software to implement a semi-automated computer controlled system that gathers the spatial field patterns of a given ultrasonic applicator. Such a system is very useful for characterizing ultrasonic applicators in cancer therapy. This system could also be used for microwave antenna field characterization.

The specifications for the scanner system were developed through close interaction with researchers at URI THERM-X, who will ultimately be the users of the system. The overall requirements of the system were first discussed and decided upon, before any actual construction began. These were broken down into three major areas: mechanical hardware, electronic hardware, and computer software. A brief discussion of the decisions involved in developing the specifications for the three areas follows.

Mechanical Hardware Specifications

After considering the dimensions of typical applicators and the range of their fields, it was determined that a positioning system capable of locating an ultrasonic hydrophone probe (the device used to detect ultrasonic waves) within roughly a 45 cm x 45 cm x 45 cm (centimeter) water tank with at least 0.1 mm (millimeter) resolution and a linear positioning speed of approximately 2.5 cm per second would satisfy current and projected needs. Lead screw type positioners, such as the UniSlide series of motor drive assemblies manufactured by Velmex Inc. (Bloomfield, NY), are well-suited for such applications and were therefore chosen for this task. The scanner configuration would be designed in conjunction with a Velmex application engineer (Alan Brennan). He confirmed the fact that the UniSlide assemblies could easily be coupled to stepper motors, which are ideally suited for computerized control. The stepper motors specified would be in accordance with torque requirements of the Velmex UniSlide assemblies selected for the scanner system.

Electronic Hardware Specifications

The Apple IIe computer with 128 K bytes of RAM (Random Access Memory) was selected to be used for positional control and real time data collection and display. It was chosen because of its proven performance as a laboratory computer, ease in interfacing to external hardware, availability, and low cost. Based on the author's previous experience in developing an electronic system employing several stepper motors, it was clear that the computer to stepper motor interface would best be achieved by using a custom designed interface board connecting the Apple IIe to a stepper motor driver manufactured by Anaheim Automation (Anaheim, CA) which in turn drives the stepper motors.

In general terms, the collection portion of the electronic hardware involves amplifying the AC signal from the hydrophone probe, directing it to an RMS (Root Mean Squared) to DC converter to get a DC value proportional to the signal's amplitude, scaling and buffering this DC signal, and finally feeding this into an ADC (Analog to Digital Converter) interfaced to the Apple IIe. This specialized circuitry would be designed for use with a commercially available hydrophone probe. The probe for this task was a Dapco NP-10 needle transducer micro-probe with a detachable micro-dot connector manufactured by Dapco Industries, Inc. (Ridgefield, CT). Initially the ADC to Apple IIe interface circuit would use the relatively inexpensive and readily available ICL7109 12 bit ADC manufactured by Intersil (Cupertino, CA,) interfaced to the Apple IIe using a 6522 VIA (Versatile Interface Adapter). It was soon apparent, however, that the maximum conversion rate of 7.5 samples per second would severely limit the

system's performance. For this reason, the higher performance (and more expensive) 12-bit AD572 ADC, manufactured by Analog Devices, (Norwood, MA) was selected instead. Interfacing it to the Apple IIe would easily be accomplished using a 6522 VIA. The chip would also be ideal for interfacing the Apple IIe computer to the stepper motor driver pack and to the limit switches located on the Velmex UniSlide positioners.

Amplification of the low amplitude high frequency signal received by the hydrophone probe would best be achieved using a monolithic wideband amplifier. The Motorola MC1590 was a logical choice for the application. Coupling the hydrophone with the amplifier with minimal loss would necessitate winding a custom matching transformer, which could be performed at URI THERM-X. Conversion of the amplified hydrophone AC signal to a DC value proportional to its amplitude initially presented a problem. A high speed (1 MHz) highly linear rectification and filtering circuit would have to be designed to accomplish the task. Upon further investigation, however, it was found that an Analog Devices 442J RMS to DC converter module was ideal for such applications, and it became the logical choice. It would probably be necessary to buffer and scale the DC output of the 442J RMS to DC converter before connecting it to the AD572 ADC. The general purpose LM158 operational amplifier was chosen for this purpose.

Computer Software Specifications

Because of the real-time control constraints of the system, most of the stepper motor control and data acquisition portion of the software would require assembly language programming. The non time-critical software, such as user interface and system control

software, could be written using the Apple IIe's Basic language. Specifically, the division of tasks into assembly language and high level language categories was broken down as follows. Stepper motor control signals, real time display of x, y, and z coordinates, real time display of sampled data, and data acquisition would all be controlled directly using an assembly language program. User interface, such as prompting for positional information, error message displays, storage of acquired data to the disk drive and dumping to the printer would all be handled by a Basic program. Additionally, the Basic program would also act as an interface between the user and the assembly language program.

Thesis Organization

In the following chapters the specific details of the project will be elaborated upon. Chapter 2 presents a brief overview of the system development--the progression from conceptualization to finalized state. Chapter 3 discusses the final system design, briefly describing the mechanical hardware, and concentrating heavily on the electronic hardware and computer software package from a system-level perspective. Chapter 4 gives a detailed component level description of both the digital and analog electronic circuitry. Chapter 5 gives a detailed description of the assembly language and Basic programs. Finally, Chapter 6 presents the results of the project, and Chapter 7 gives recommendations for further system development and conclusions. Standard engineering notation and abbreviations are used whenever possible with clarifications added when necessary.

CHAPTER 2

SYSTEM DEVELOPMENT

Once all of the general system specifications had been decided upon, the actual development of the system began. A simplified functional diagram of the system is shown in Figure 1. (Figures are shown at the end of the thesis.) Since the UniSlide assemblies had a long delivery time, it was necessary to concentrate initial efforts on finalizing the mechanical hardware design. This involved an iterative procedure, going between Dr. E. C. Burdette and Dr. S. Goss who are researchers at URI THERM-X, and Alan Brennan at Velmex. A final design was decided upon, the required parts were ordered, and the appropriate stepper motors were specified by Velmex. The stepper motors specified were the MO92-FD09 manufactured by Superior Electric (Bristol, CT). It was now possible to choose a suitable stepper motor driver pack to power them. Consultation with Bud McNally at Anaheim Automtion resulted in the selection of the DPF09 three drive bilevel stepper motor driver pack. He also recommended using the economical Astrosyn 34PM-C110 stepper motor, distributed by Astrosyn America, Inc. (Van Nuys, CA), as a direct replacement for the costlier MO92-FD09 motor specified by Velmex.

In consultation with URI THERM-X, it was decided that two of the Astrosyn motors would be purchased along with the driver pack from Anaheim Automation (which is also a sales representative for the Astrosyn line of motors), and that the third motor would be a Slo-Syn to be purchased along with the UniSlide assemblies from

Velmex (which is also a sales representative for Superior Electric).

The reasons were fourfold: First,, it turned out that one of the motors needed to be double shafted, and the double shafted version of the Astrosyn Motor was currently out of stock at Anaheim Automation. On the other hand, Velmex could deliver a double shafted version of the Slo-Syn motor along with the rest of the order. Second, Anaheim Automation could deliver within ten days of receiving a purchase order, whereas turn-around time at Velmex was quoted at six to eight weeks. In order to develop operative software prior to delivery of all of the parts, it would be necessary to have at least one motor to use with the driver pack during software development. Third, there is the economic advantage of using as many Astrosyns as possible. And finally, having both types of motor would allow for performance comparisons to aid in future system modifications.

With mechanical hardware, stepper motors, and drivers on order, it was now necessary to design the interface and data collection circuitry. This circuit was designed, built, and tested; it provided the necessary interface between the Apple IIe computer and the stepper motor driver pack, as well as the 12-bit ADC. A simple test program was written to verify operation of its VIAs and ADC, with a potentiometer used to simulate the analog input.

With some of the hardware completed, design attention was now turned towards software development. Work was begun on writing the assembly language program which would control the stepper motors. Based on previous experience with stepper motors, it was

clear that some form of angular acceleration, or ramping, would be required to drive the stepper motors at a fast rate. This is necessary because the physical inertia of the positioner prohibits an instantaneous change of speed. This was conceptually feasible using software, but would require an iterative fine-tuning process to find the optimum ramping curve for the stepper motors. Using an algebric equation to define the ramping curve was conceivable, but would be an awkward and cumbersome task to handle in assembly language. A simple approach, which was finally used, was to employ a data table which contained numbers that would sequentially be put into a delay loop, between pulses directed to the stepper motor. This would effectively allow any desired ramping curve to be implemented, by precalculating the desired sequence of delays and storing these values in a data table, or more specifically, a ramp table. The assembly language program used to control the steppers was developed using this approach.

Once the assembly language program was developed into a useable form, attention was focused on a simple way of entering and modifying the data in the ramp table. A Bit Pad One graphics tablet, manufactured by Summagraphics Corp. (Fairfield, CT), was available at URI THERM-X and proved to be a useful and simple tool for entering the ramp table data. A program which had previously been written to enter time vs temperature curves for a piece of demonstration software was modified and enhanced to become a ramp curve entry and test program. This program permitted a user to digitize a curve on the graphics tablet, and interactively see how it controlled the ramp rate of a stepper motor. While the finishing touches were being put on the program, the stepper motor

driver pack and two stepper motors were delivered. Once the necessary wiring harnesses were made to interconnect the Apple IIe to the stepper driver pack to the stepper motors, testing and debugging of the pieces of hardware and software began. There were a few problems concerning the minimum pulse width that the driver pack would accept; however, once these were discovered and eliminated, the software could control the hardware.

At this point, about a month of solid software development would be required so that initial testing could be performed shortly after the Velmex positioning system was delivered. This would require about 800 additional lines of assembly language source code, involving the addition of real time Cartesian coordinate display on the Apple IIe's text screen and a data collection routine, which would store data in a 48 K byte auxiliary memory buffer and display the values of the data in real time on the high resolution graphics screen. This additional code was written and partially debugged shortly after the Velmex system arrived. A few more bugs surfaced as initial testing of a near mechanically complete system progressed. Once things began to look optimistic from a hardware and assembly language software viewpoint, it was now time to begin developing the system control software.

As mentioned earlier, the system control software was essentially going to be a large Basic program which would handle all user interface and linkage to the assembly language program. More specific details regarding the exact program flow were discussed with URI THERM-X personnel before writing the program began. The initial estimate of the scope and complexity proved to

be largely underestimated, as unexpected problems emerged. Of these problems, memory management became a primary concern. As the size of the Basic program increased, it soon became too large for the memory space normally allocated to Basic programs within the Apple IIe. When the assembly language program did a real time display of data on the high resolution graphics screen, part of the Basic program would be destroyed. The problem was resolved by entirely rearranging where the programs are stored in memory. Relocating the assembly language program simply meant reassembling it at a new starting address. Relocating the Basic program, however, required the writing of a Basic relocation program which could be run prior to loading the Basic program into RAM.

Once these programs were relocated in nonconflicting areas of memory, attention was now focused again on hardware. The amplifier circuit was built, tested, and debugged, with problems encountered with layout, and winding of the toroidal matching transformer. Originally, the entire circuit was wire-wrapped, with wire-wrap sockets used for both operational amplifiers. Wire-wrapping and careless placement of bypass capacitors both lead to noise problems. These were solved by trimming down the wire-wrap leads of the MC1590's socket and carefully rerouting the bypass capacitors and the wires carrying high frequency signals. Winding the toroidal matching network proved to be an exacting and time-consuming process, and resulted in a transformer with a very narrow bandwidth near 1 MHz. With the hardware essentially complete and with software partially operative, it was possible to run the system under experimental conditions with a functioning applicator in a water tank.

Using this setup, the system was able to collect and display data on the high resolution screen and dump it to the printer. The primary problem encountered, aside from several major bugs in the CONTROLLER program which had yet to be resolved, was that the motion of the axis positioners was accompanied by excessive vibration. At first, it was thought that these vibrations were due to a physical resonance of the system. Upon further investigation, however, it was observed that the vibrations occurred only when data were being collected and displayed on the high resolution screen. It was therefore assumed that the added time delay required to display the data on the graphics screen was slowing down the clock rate to the steppers so that it was at the motor's resonant frequency. To eliminate this suspected cause, the following actions were taken.

The high resolution graphics display subroutine of STEPPER was completely overhauled. The previous routine had displayed the data on the graphics screen in the form of histograms. This is very wasteful of time when compared to simply plotting a single point; thus, the routine had to be changed. Additionally, to access individual pixels on the Apple IIe's high resolution screen requires a fairly complex algorithm since adjacent pixels are not necessarily in contiguous memory addresses. In the previous routine, this had been accomplished by performing the algorithm in real time. Again, this is wasteful of time, and would be accomplished quickly (but at the expense of additional memory) by simply using a precalculated look-up table. In the improved version of this routine, using dots to display the data rather

than lines and using a look-up table to calculate screen addresses were the methods employed.

The result was little to no noticeable improvement in the system's performance. The mechanical vibrations were still present. Much time was spent contemplating this unexpected turn of events. The hardware was scrutinized, the software was sifted through until finally a possible explanation was discovered.

In the STEPPER program there is a subroutine name OUTPLS with only about a dozen lines of code, but it is perhaps the most important because it is responsible for outputting all the signals to all of the clock and direction control lines going to the driver pack. In this subroutine, it was found that the data for the direction control lines were set up only a few microsceconds before the rising edge of the clock pulse was output. If the driver pack did not have sufficient setup time prior to outputting a clock pulse, this could explain the system's erratic performance.

The suspected code was fixed and the preceding hypothesis was tested. The result was that the motors moved smoothly. With this major obstacle overcome, it was now possible to spend the remaining time on debugging and testing the CONTROLLER program. The debugging proceeded slowly, without incident, or additional major obstacles. The system was then used in an experimental setup. The results are detailed briefly in Chapter 6.

CHAPTER 3

FINAL SYSTEM DESCRIPTION

In this section a specific itemized and illustrated description of the entire Ultrasonic Transducer Field Scanner System is presented. From the information provided, and from the supplementary illustrations and tables, the reader should have enough knowledge to build a similar system. Again, the areas of mechanical hardware, electronic hardware, and computer software will be treated separately, with the last two receiving the primary emphasis.

Mechanical Hardware Description

The mechanical hardware for the scanner system is illustrated in Figure 2. It should be noted that a standard aquarium was used rather than what was originally to be a custom built Plexiglas water tank. Additionally, what was originally to be a custom machined hydrophone support arm was improvised using standard plumbing pieces purchased from the local hardware store. A complete parts list of the mechanical hardware is shown in Table 1. (Tables are shown at the end of the thesis.)

Electronic Hardware Description

The schematic for the ADC and VIA interface portion of the electronic hardware is illustrated in Figure 3 and will subsequently be referred to as the interface circuit. the corresponding parts list is shown in Table 2. The circuit was built on an Apple IIe computer Vector Board and requires a ± 15 V regulated power supply. This supply is also required for the

amplifier circuit used to condition the hydrophone's signal. Interconnection of the interface circuit to the stepper driver pack and positioner limit switches is accomplished using a 40 pin card edge connector and 40 conductor ribbon cable, fanned out to a 14 pin female 3M (Minnesota Mining and Manufacturing Co.) and a 10 pin female 3M connector. The exact pinout is shown in Table 3.

Interconnection of the clock, direction, and enable lines of the stepper motor driver pack is via a 14 pin male 3M connector which is housed in an aluminum mini box mounted to the driver pack. This box connects, via the 14 pin connector, to the ribbon cable going to the interface circuit. Interconnection of the three stepper motors to the driver pack is via male 6 pin Molex connectors, which connect to mating female connectors mounted in the aforementioned mini box. The schematic for these interconnections is shown in Figure 4, and a parts list is provided in Table 4.

Interconnection of the x, y, and z axis limit switches is accomplished using three cables with male and female 7 pin Amphenol connectors on either end: the female ends mate with the limit switch housing and the male ends mate with a mini distribution box. This box connects three female 7 pin Amphenol connectors to a male 10 pin 3M connector so that it can mate with the ribbon cable going to the interface circuit. A schematic for this limit switch distribution box is shown in Figure 5, with a parts list given in Table 5.

The schematic for the amplifier circuit is shown in Figure 6, with the corresponding parts list in Table 6. As mentioned

earlier, the schematic shares a ±15 V regulated DC power supply with the ADC in the interface circuit. The power supply used is a Power One #1HAA15-0.8 (Camarello, CA), and is connected to the two boards via color coded wires. Interconnection of the analog signal from the amplifier circuit board to the interface circuit board is accomplished using a shielded wire. Interconnection of the hydrophone probe to the amplifier circuit is achieved using standard male and female BNC connectors, respectively. The complete system-level interconnections of all the electronic hardware is shown in Figure 7.

Computer Software Description

A catalog listing of the System Software and Utilities Disk is shown in Figure 8. These text, binary, and Basic program files make up the complete software package. This software is designed to run on an Apple IIe configured in the following manner. The Apple IIe must be equipped with an Apple IIe computer extended 80-column text card in the auxiliary memory slot. A Videx (Corvallis, OR) PSIO card must be located in slot #1, with its parallel port phantomed into slot #2 which must be empty. The required protocol configuration for text and graphics is shown in Figure 9. This configuration information is identical to that stored in the PSIO.BP&GRAFIX300BAUD file on the System Software and Utilities Disk. The user should refer to the PSIO card's installation and operation manual for instructions on loading the configuration into the PSIO card. The parallel port of the PSIO card (slot #1) must be connected to either an Epson FX-80 or FX-100 parallel dot matrix printer (or a functionally compatable

printer), and the serial input port of the PSIO card must be connected to the Bit Pad One digitizing tablet. Slot #4 must contain the interface circuit board, and slot #6 must contain an Apple Disk Controller Card connected to at least one 5 1/4" floppy disk drive.

Referring to Figure 8, the function of each file on the System Software and Utilities Disk will now be presented. The HELLO program is a standard Basic program which, consistent with Apple IIe DOS 3.3 (Disk Operating System version 3.3), will be booted into upon power-up of the computer just after the DOS has been loaded. This program first adjusts the appropriate zero page pointers so that the start of Basic program memory will now occur at the memory address 16385 (hexadecimal address = $4001). Once this has been completed, it loads the main Basic CONTROLLER program starting at this new memory address, and then starts executing it. The first thing the program does after displaying a title page is to prompt the user to load the machine language routines. If the user chooses to load them, the CONTROLLER program will then load STEPPER.OBJ0.,RAMP, and finally TRANSFER.OBJ0 into the computer's memory. If this is the first running of CONTROLLER following power-up, or if a serious program "crash" has occurred, failure to load these programs will result in improper operation of CONTROLLER.

STEPPER.OBJ0 is the assembled binary object file of the text source file STEPPER. It contains the stepper motor control, real-time display, and data collection software. The internal details of this program will be described in Chapter 5. RAMP is a binary file that contains the 256 byte ramp table used by the

STEPPER.OBJ0 program and must be loaded prior to running this program. The data in RAMP were generated using the Basic DRIVER program. This is a utility program that allows the user to enter a ramp curve using a digitizing tablet and interactively to see its effect on motor control, as well as to obtain a hard copy and store the data to disk. HISCAN.OBJ0 is the assembled binary object file of the text source file HISCAN and is used by DRIVER to scan a curve on the Apple IIe's high resolution graphics screen and store the data in a 256 byte buffer. TRANSFER.OBJ0 is the assembled binary object file of the text source file TRANSFER. It moves 8 K byte blocks of data from the 48 K byte auxiliary memory buffer to the main memory high resolution graphic screen area where it may then be transferred to disk.

TEMPLATE is a Basic utility program which creates a hard copy of a scaled coordinate axis that the user may employ as an aid when entering a ramp curve using the DRIVER program mentioned earlier. Finally, as stated earlier, the PSIO.BP&GRAFIX300BAUD file contains the configuration data required by the Videx PSIO card. The listings for all of the aforementioned programs are included in Appendices A through C.

CHAPTER 4

ELECTRONIC HARDWARE

Earlier, the electronic hardware of the scanner system was described at the system level. In this section the specific operation of the circuitry will be explained. First, the interface circuit will be discussed from the component level followed by a similar discussion of the amplifier circuit.

Interface Circuit

As mentioned earlier, the schematic for the interface circuit is shown in Figure 3, and will be referred to often in the following text. Three inverters from a 74LS05 hex inverter integrated circuit form a circuit which generates a $\phi 1$ (phase 1) clock line available from the Apple IIe's peripheral connector. $\phi 2$ is not provided on this connector, thus necessitating its synthesis on the interface circuit board. It is created by inverting and delaying the $\phi 1$ clock by approximately 150 ns (nanoseconds). Inverting is accomplished using an odd number of inverters, and the delay is created by the 4.7 K ohm resistors and the 10 pF (picofarad) capacitor. The time constant for this RC circuit is 47 ns, and the time for a logical transition to occur, assuming a 3.4 V swing, is given by

$$t = -47 \ln[(5 - 3.4)/10] \text{ ns}$$
$$= 86 \text{ ns.} \qquad (1)$$

Assuming the nominal propagation delay of each inverter is 16 ns results in a total delay of 144 ns. This $\phi 2$ signal is applied to

both of the 6522 clock inputs (pin 25). An additional inverter is connected to the chip select 1 input of 6522 #1 to enable it when 6522 #2 is disabled. This will occur when the A4 address line from the peripheral connector is low. The remaining four address lines A3 - A0 are used to address the 16 internal registers of the 6522s by connecting them to the RS0 - RS3 address inputs. Table 7 shows these register addresses assuming the interface circuit card is plugged into slot #4 of the Apple IIe motherboard. Further details of this chip's internal registers and their functions are available in Rockwell International's R6522 data sheet (Document #29000D47).

The data lines D0 - D7 for the 6522s are tied directly to the corresponding data lines from the peripheral connector as well as the reset and read/write lines. The chip select 2 inputs of the 6522s are tied to the I/O select line from the peripheral connector, enabling them only when the slot (peripheral connector) is addressed.

The remaining 6522 lines are the power (Vcc and Vss) and parallel port (PA and PB) lines. Power is connected in the usual manner, with bypass capacitors to ground out any noise, and is provided by pin 25 (5 V DC) and pin 26 (ground) of the peripheral connector. The functions of the PA and PB parallel I/O lines are summarized in Table 8.

The most significant byte of data from the ADC is applied to PB1 (PB of port 1). The least significant nibble (lowest 4 bits) of the ADC is applied to PA1 bits 0 - 3. All 12 of these digital lines are outputs from the ADC and inputs to the VIA. Whether a given bit is an input or an output must be known in order to

correctly initialize the VIA's DDRs (Data Direction Registers). PA1 bits 4 - 6 are outputs from port 1 and are used to control the x, y, and z enable inputs of the driver pack, respectively. PA1 bit 7 is an output bit of port 1 and has a special control function. It is tied directly to the convert start input of the ADC (pin 21) and must be brought high (under software control) to initialize a conversion. The end of a conversion is signaled by the falling edge of the ADC's status line (pin 20) which is tied to the CA1 and CB1 inputs (pins 40 and 18) of port 1.

PA2 bits 0 - 5 are used as limit switch inputs to port 2. These switches limit the travel near each end of the x, y, and z axis positioners, to prevent the hydrophone probe from smashing into the walls of the water tank, which would be undesirable in most experiments. These inputs are normally low, but are pulled high through 1 K ohm pullup resistors when their corresponding NC (Normally Closed) limit switch becomes activated. PB2 bits 0 - 5 are used as clock and direction outputs for x, y, and z motion control from port 2 to the driver pack. Refer to Table 8 for the exact bit assignments.

Focusing on the remaining interconnections of the AD572 ADC, -15 V and +15 V are applied from the external regulated DC supply to pins 31 and 28, respectively, providing power for the analog portion of its integrated circuit, and are bypassed accordingly with 1 uF (microfarad) tantalum capacitors to ground. Digital and analog ground are applied to pins 15 and 26, respectively, and are common to both pin 26 (ground) of the peripheral connector and the external power supply ground. Pin 25 (5 V DC) of the peripheral connector provides power for the digital portion of the integrated

circuit via pin 16, and the Short Cycle Input (SCI-pin 14) is tied high to enable 12 bit conversion. The analog signal from the amplifier circuit is applied to the ADC (pin 24) through a simple low pass filter which removes any high frequency noise. The 100 ohm trimmer potentiometer connected between pins 18 and 27 is used to adjust for a full scale reading when exactly 10 V is applied to the ADC's input (pin 24). Similarly, the reference voltage from the 20 K ohm trimmer potentionmeter is used to adjust for a zero reading when the ADC's input (pin 24) is grounded. The exact procedure for calibrating this chip is described in full detail on page 5 of the AD572 data sheet. The amplifier circuit will now be described. The reader should refer to Figure 6 when necessary.

Amplifier Circuit

The hydrophone and its cable are connected to the primary side of a custom made toroidal transformer using a standard BNC connector. The secondary side of this transformer is applied to the differential inputs (pins 1 and 3) of a Motorola MC1590 wideband amplifier. This integrated circuit is single ended and is powered by +15 V applied to pin 7 and ground connected to pins 4 (substrate ground) and 8 (case ground). The AGC (Automatic Gain Control) bias is set by the 5 K ohm trimmer potentiometer and resistor combination, with the 6.8 uF capacitor grounding high frequency noise, and is applied to pin 2 (AGC input) of the integrated circuit through a 5.6 K ohm resistor. This voltage divider circuit allows a range of voltages from approximately 4 to 13. Varying this voltage between 6 and 9 will allow a gain reduction from 0 to 60 dB (decibels). This range was determined

from the gain versus voltage graph found on the Motorola MC1590 data sheet.

Pins 5 and 6 are the open collectors of a differential current amplifier which is the output stage of the integrated circuit. Pin 5 is tied to +15 V; pin 6 is grounded through a 1 K ohm resistor which converts the amplified current into voltage. The AC component of this voltage is passed through a 0.1 uF capacitor to the input (pin 8) of the 442J RMS to DC converter. The +15, 0, and -15 V are applied to pins 5, 4, and 3 of the 442J, respectively. Offset adjust is provided by the 20 K ohm trimmer potentiometer applied to pin 9. Pin 6 is tied to pin 2 to provide unity gain, and the output signal from pin 2 is applied to the noninverting buffer amplifier formed by the LM158 operational amplifier. The gain of this amplifier is set by the 100 K ohm trimmer potentiometer and ranges from 0 - 11.

CHAPTER 5

SOFTWARE

As mentioned earlier, the software portion of the scanner system consists primarily of the assembly language STEPPER program and the CONTROLLER program written in Basic. In this section, the inner workings of these two programs will be explained. Their listings are included in Appendices A and B and will be referred to frequently in the program descriptions which follow. It should be noted that within the listings of both programs, descriptive comments and remarks are included to aid in understanding the program's flow. These comments, as well as the following text, are included to simplify the modification and further enhancement of the system's software by an experienced programmer. A description of STEPPER is first, followed by a description of CONTROLLER.

STEPPER

STEPPER is an assembly language program of approximately 1200 lines, which acts as a link between the high level user interface CONTROLLER program and the external scanner hardware. The transfer of information between CONTROLLER and STEPPER is accomplished using a set of twelve linkage registers. The register addresses, names, and brief functional descriptions are shown in Table 9. Of these registers, the MOVE register and the XYZ register are the most important and the most used.

The MOVE register is a 3 byte (24 bit) register starting at memory address $804 and stored in low to high byte order. Before

one of the three scanner axis positioners can be moved to a new location, the MOVE register must be loaded with a number that corresponds to the desired destination location. This number is calculated by taking the distance between the destination and "home" positions in centimeter and dividing it by the constant 0.00127. This constant is derived from the fact that each pulse sent to the stepper motor driver turns the motor's shaft 1/400 revolution, and that each revolution moves the linear positioner 0.2 inches, with 1 inch equal to 2.54 cm.

$$1/400 \times 0.2 \times 2.54 = 0.00127 \qquad (2)$$

Once a number has been placed in the MOVE register, the program must be told which axis (x, y, or z) is to be moved, whether data are to be collected, if the data are to be displayed on the graphic screen, and so forth. This type of information is referred to as "command" information; it is transferred to the STEPPER program in the form of a 1 byte (8 bit) command code. This code is entered into the XYZ register which is the command register for the STEPPER program (the other eleven registers are considered "data" registers). The XYZ register is located at memory address $803. As shown in Table 9, each bit position in the XYZ register has a particular significance. For all command codes which move a scanner axis positioner, except for the HOME command code which is a special case, bits 0-2 are used to specify which axis (x, y, or z, respectively) is to be activated. It should be noted that only one axis may be moved at a time; therefore, only one of these three bits may be set in a given command code.

Bit 3 of the XYZ register is used to specify manual control of a given positioner. When this bit is set, and STEPPER is called, manual control of the positioner is achieved using the left and right arrow keys on the Apple IIe's keyboard to move the positioner towards and away from the home position, respectively, and the "return" key to exit.

Bit 4 of the XYZ register is used to clear all three of the 24 bit XCOUNT, YCOUNT, and ZCOUNT data registers which contain the distance (in number of pulses from home in binary) to the current x, y, and z axis positions. In most applications it is desirable to have the Cartesian coordinate origin be the home position. Therefore, after executing a HOME command, a "clear count register" command (bit 4 set, all others cleared) should be executed.

Setting bit 5 of the XYZ register will disable both the real time display of coordinate positions on the text screen and graphical data display on the high resolution graphics screen during data collection. The current x, y, and z coordinate positions (cm) are stored in the 4 byte geometric coordinate registers GCX, GCY, and GCZ, respectively. Each of these is stored in a packed BCD (Binary Coded Decimal) excess-500 format, as shown in Table 9. These registers must be initialized from the main user interface program (in this case CONTROLLER) to the appropriate value. Once initialized, STEPPER will increment or decrement the geometric coordinate register corresponding to the currently active axis positioner by 0.00127 cm per pulse depending on whether the motion is away from or towards home, respectively. This updating of the geometric coordinate registers will occur

whether or not the geometric coordinates are being displayed on the text screen (i.e., independent of the XYZ register bit 5 setting).

Setting bit 6 of the XYZ register will disable power to the stepper motors upon completion of a scan. This is useful to prevent overheating of the motors when the system is undergoing a prolonged period of use.

Finally, setting bit 7 of the XYZ register will enable the storage of the data during a given scan. The 12 bit digitized data are stored sequentially in 48 K byte (48,640 byte) buffer, located from $0200 to $BFFF in auxiliary memory space. The data are stored in 2 byte pairs (low byte, high byte order) with the least significant nibble of the low byte containing the lower 4 bits, and the high byte containing the upper 8 bits. Data are stored sequentially in this buffer starting at the current location pointed to by the 2 byte zero page pointer DATPTR ($EB-$EC, low byte--high byte). DATPTR may be reset to $0200 by loading the XYZ register with $FF and executing it (calling STEPPER). It is the responsibility of the user interface program to ensure that the value in DATPTR never exceeds $BFFF. DATPTR can be considered the 13th stepper linkage register, located below the rest in zero page.

The distance between data samples is determined by the value stored in the 2 byte (16 bit) DATINC register. Specifically, the desired distance between samples (in terms of pulses) should be entered into the DATINC register prior to executing a data storage scan. If zero is stored in DATINC, data collection will be disabled during a given scan. It should be emphasized that

disabling bit 7 of the XYZ register disables data storage, whereas storing a zero in DATINC disables data collection. This will be apparent once the functions of the MAXCNT and MAXVAL registers have been presented.

When data are being collected (DATINC is nonzero), each time that a sample is input from the ADC its magnitude is compared to the number stored in the 2 byte (24 bit) MAXVAL register. If the sampled value is larger, it is stored in MAXVAL and the current position of the axis positioner is stored in the 3 byte (24 bit) MAXCNT register. It should be noted that MAXVAL is reinitialized to zero each time STEPPER is called. If the real time display on the text screen is enabled (bit 5 of the XYZ register is zero), then every time STEPPER stores a new value in MAXCNT, this location is displayed (in cm) on the text screen directly below the geometric coordinate display corresponding to the current axis being scanned. It is therefore through the MAXCNT register that the user interface program can determine where a peak has occurred within a given scan.

The only remaining register which has not yet been discussed is the 1 byte (8 bit) ERFLG register. As its mnemonic suggests, it acts as an error flag; its purpose is to signal to the user interface program when an error occurs. An error has occurred during a given STEPPER operation when a nonzero value (typically 255) is returned in the ERFLG register. Typical errors include driving a positioner to a limit switch before a scan has been completed and attempting to execute an illegal command code in the XYZ register.

Now that the operation of the STEPPER linkage registers (Table 9) has been described, the structure of STEPPER will be presented. The general program flow of STEPPER is shown in the flowchart in Figure 10. It should be noted that this flowchart is only meant to provide an overview of the program flow of STEPPER, and not to illustrate every intricate detail of every subroutine. To understand the function and interrelation between the major subroutines in STEPPER, the following brief descriptions are provided. Using these in conjunction with the commented listing in Appendix A should develop sufficient information to provide a thorough understanding of the program.

START -

Initializes pointers and registers prior to entry into the XYZ Register Command Interpreter.

XYZ REGISTER COMMAND INTERPRETER -

Executes appropriate routines based on the XYZ command register.

XYORZ -

Sets up the appropriate count registers, geometric coordinate registers, screen pointers, etc., for either x, y, or z control based on the status of bits 0-2 of the XYZ register, respectively.

MANUAL CONTROL ROUTINE -

Allows manual control of the axis specified by bits 0-2 of the XYZ register using the left and right arrow keys for counterclockwise or clockwise motor motion (when facing the motor's shaft), respectively, and the "return" key to exit.

Each depression of the arrow keys advances the motor by 200 pulses in the appropriate direction.

HOME -

Drives the x, y, then z axes counterclockwise until the corresponding "in" limit switch has been reached.

MAIN -

Calculates the difference in number of pulses between the current position and the position in the MOVE register, determines the required direction of rotation, and whether or not ramping is to be used. If ramping is required, control is passed to RAMP. Otherwise, pulses are sent out at a constant rate determined by the constant NOMVAL.

RAMP -

Uses values in the ramp table pointed to by TBLPTR to ramp up to a constant speed, stay at that rate for a predetermined amount of time, then ramp down by going through the ramp table in reverse.

DELAY -

Uses the value in the X register nondestructively as a variable in a simple delay loop.

OUTPLS -

Outputs a clock pulse to the stepper motor specified through CKMASK, with a positive pulse width equal to the time spent in DELAY.

INCCNT -

Increments the 3 byte (24 bit) count register pointed to by REGPTR (XCOUNT, YCOUNT, or ZCOUNT) by 1, and the corresponding 4 byte (32 bit) geometric coordinate register

(GCX, GCY, or GCZ) pointed to by GCPTR by the value GCINC (0.00127 cm).

DECCNT -

Same as INCCNT, except it decrements rather than increments.

DCRDIF -

Decrements the 3 byte (24 bit) DIFF register which contains the difference (in pulses) between the current position and the destination (MOVE) position.

DATCOL -

Collects data if the 2 byte (24 bit) DATINC register is nonzero and displays them on the high resolution graphics screen (by calling HIPLOT) if bit 5 of the XYZ register is a "0." If bit 7 of the XYZ register is set, then data will be stored at the address pointed to by DATPTR every DATINC+1 number of pulses, with DATPTR being incremented each time. If the collected DATA are greater than MAXVAL, the current position is displayed on the text screen directly below the currently active geometric coordinate display.

RTSISP -

Displays the contents of the geometric coordinate register currently pointed to by GCPTR, starting at the location pointed to by SCRPTR on the text screen, if bit 5 of the XYZ command register is not set.

HIPLOT -

Plots the real time data in a 128 x 128 pixel window on the high resolution graphics screen (using the XTBL and XDTBL

look-up tables to maximize speed performance) if bit 5 of the XYZ command register is not set.

Now that the operation of the STEPPER program has been described, attention will be focused on the operation of the user interface program called CONTROLLER. As mentioned earlier, it is the Basic program which interfaces between the user and the STEPPER program as well as the Apple IIe's DOS. A complete listing of CONTROLLER is provided in Appendix B and should be referred to when necessary.

CONTROLLER

CONTROLLER is structured into a main calling program and several modular subroutines for easy debugging and modification. The flowchart of this program is shown in Figure 11. Numbers shown in parentheses indicate the starting line number of a given subroutine. In the following paragraphs, the operation of these subroutines will be detailed. Comments appearing within the program listing provide additional information. The main subroutines will be presented in numerical order, starting with the INITIALIZATIONS subroutine (2000).

INITALIZATIONS is actually a subroutine which calls several smaller subroutines. Its function is to take care of the initial set up of registers, title page display, and the loading of STEPPER.OBJ0, RAMP, and TRANSFER.OBJ0. The reader should understand the operation of subroutine 2300. This subroutine provides a simple example of communication between CONTROLLER and STEPPER. The Basic variable BASE is a constant of value 2048

(line 410) corresponding to the starting address of STEPPER ($0800). The variable name BASE was chosen rather than the more logical name "STEPPER" because the latter contains the Basic command "STEP" and is therefore an illegal variable under the rules of Applesoft Basic. The variable XYZ is equal to BASE + 3 (line 420), or in this case 2051 ($0803), and corresponds to the XYZ command register address. With these facts in mind, line 2310 places a 16 in the XYZ command register. Referring to Table 9, this is the code for a "clear count registers" command. Once this command has been loaded, all that remains is to transfer control to the assembly language program STEPPER. This is accomplished in line 2320. This illustrates one of the simplest interactions between CONTROLLER and STEPPER and will be seen repeatedly in various forms throughout the CONTROLLER program listing.

The USER POSITIONING subroutine (3000) is a simple prompting routine and command interpreter which allows manual keyboard control of any of the three axis positioners. As can be seen in Figure 11, this routine is called when a geometric coordinate system is desired, and allows the user to place an origin at the geometric center of the applicator being tested. Figure 12 shows the positive axes for both the geometric and tank coordinate systems. Returning to the subroutine, the calls to subroutines 12700 and 12100 should be noted. Subroutine 12700 sets up the DATINC linkage register (Table 9) with the value of the Basic variable SI. In line 3115, a zero is placed in the DATINC register. As noted in the discussion of the DATCOL routine in STEPPER, this has the effect of disabling data collection. This is done several times within this program, whenever it is

desirable to position the applicator without collecting data. Subroutine 12100 is used to display the legends around the real time coordinate data on the text screen; and again, it is called frequently throughout the program.

The HOME THE DETECTOR subroutine prompts the user to enter the z axis distance to the applicator and then homes the system if tank coordinates are used, or homes the detector and then saves the distance travelled (cm) in the Basic variables XM, YM, and ZM if geometric coordinates are used. These variables represent the offset between the two different coordinate systems and are used in several of the CONTROLLER's subroutines.

The PROMPT USER FOR Z COORDS. AND GO THERE subroutine (5000) does what its name implies. If geometric coordinates are used, this value must be negative, and if tank coordinates are used it must be positive (refer to Figure 12). The magnitude of the value entered is tested against the variable ZM, and if it is less than ZM, it is accepted. In lines 5350 and 5400, the variable MV and subroutine 12400 are used in a similar manner to SI and subroutine 12700 mentioned earlier, only in this case the MOVE register rather than the DATINC register is affected.

The MANUALLY FIND THE PEAK subroutine (6000) is similar in operation to the USER POSITIONING subroutine except that data collection is enabled and displayed on the high resolution graphics screen in real time, the location of peak data points is displayed and saved for scans in x and y, and the z position is fixed. The purpose of this subroutine is to allow the user to manually find either a local or global peak, which may then be used in subsequent subroutines as a new frame of reference around

which scans can be made. This is useful in determining if a peak is off center relative to the center of the applicator.

The AUTOMATICALLY FIND THE PEAK subroutine (7000), automatically finds a global peak in x and y (with z fixed) using the following scan algorithm. First, a coarse raster scan is made with samples taken every 1 mm along x, with y increments of 2 cm. Then a fine raster scan is made within a 4 cm square region around the peak found in the coarse raster scan. This fine scan is made with samples taken every 1 mm along y, with x increments of 5 mm. These scan paths are shown in Figure 13. The peak value found in the fine raster scan is used as the global peak.

Regarding the structure of this subroutine, the loop between lines 7190 and 7250 performs the coarse raster scan, and the loop between lines 7580 and 7630 performs the fine raster scan. Inside each of these loops is a call to subroutine 7300 which performs the actual scans by setting up the appropriate linkage registers and then calling STEPPER. The distance between limit switches for each axis is stored in the Basic constants XT, YT, and ZT (lines 610, 620, and 630). This subroutine, as well as others, uses these values to limit the scan range and must be less than or equal to the actual distance between limit switches for correct operation.

Line 7500 tests to see if the 4 cm square fine raster scan will be within the tank's limits; and if not, the "Peak is too close to tank limits" error message in line 7910 will be displayed.

The TRANSVERSE SCAN subroutine (8000) performs a scan of a fixed z location along x and y. The bulk of this subroutine

(lines 8000 to 8655) consists of prompts for user entry of coordinates, limit checking on these coordinates, and displaying of error messages. If tank coordinates are used, the user must enter x and y center coordinates, scan ranges, and scan increments. The physical meaning of each of these values is illustrated in Figure 14. The x and y scan ranges will default to the maximum value permitted by the tank limits. If geometric coordinates are used, the center coordinates, ranges, and increments must also be entered, but there is also the option of centering a scan around the peak coordinates, or defaulting to the geometric coordinate origin. It should be noted that if a zero is entered for either the x or y scan range, a one-dimensional scan is implied, whereas if both are nonzero, a two-dimensional raster scan is implied. The program lines which handle this logic and scan control are lines 8670 to 8990.

The LONGITUDINAL SCAN subroutine is essentially a simplified version of the TRANSVERSE SCAN subroutine. It performs a one-dimensional scan along the z axis at the peak x and y coordinates. Prompts are made for the z scan range and increment, and the entered values are tested against the appropriate limits.

The STORE TO DISK subroutine transfers the data collected in either the TRANSVERSE SCAN or LONGITUDINAL SCAN subroutines to disk. First, the user is prompted for a filename, then a sequential text file with this filename is created containing all of the important scan parameters. (The format of this file is shown in Table 10.) The data are then transferred from auxiliary memory in 8 K byte blocks to binary files with a filename made up of the filename input by the user concantenated to a decimal point

followed by the block number. As an example, suppose that a user has run a scan and has collected 10,000 bytes of data in the auxiliary memory buffer. Assume that TESTDATA has been chosen as the filename for these data. This subroutine will create a text file containing the scan information with the filename TESTDATA, and two binary files with filenames TESTDATA.0 and TESTDATA.1, which contain 8,192 and 1,808 bytes, respectively, (10,000 - 8,192 = 1,808).

The final subroutine to be discussed is the IMMEDIATE DATA SCAN AND DISPLAY subroutine. This subroutine allows the user to chose either x, y, or z axis scan and enter the coordinate to which a scan is to be made from the current position. If the coordinate is within tank limits, a scan will be made to that point, with the sampling interval set so that the scan will fill the display window on the high resolution display screen. The user then has the option of dumping the screen to the printer or performing more scans.

CHAPTER 6

SYSTEM TESTS AND RESULTS

System Tests

Testing of the system was begun once there was a high level of confidence that no physical damage to the transducer, probe, water tank, or user could occur as a result of inoperative software. The bulk of the testing consisted of running the CONTROLLER program through all possible paths of program flow shown in Figure 11, and correcting any remaining software bugs. Most remaining bugs were related to cosmetic defects in the user prompts and illegal branching paths. There were also a few bugs in the automatic peak detection subroutine and the immediate mode data collection subroutine. With all of the bugs presumably removed, and with program flow identical to Figure 11, test scans were run and sample data collected.

Results

Data were taken while in the immediate data scan and display mode with a 1 MHz sinsuoidal signal from a Wavetek model 191 function generator applied to applicator element number 6. Ten x-axis scans 8 cm long and spaced every 0.5 cm in the y direction were performed near this element at a distance of approximately 4 cm from the applicator (see Figure 15). The data collected were dumped to the printer, and the resulting plots are shown in Figure 16 (a - j). These plots indicate the relative intensities of various points near the active element. Such plots can provide the user with a good estimate of where the main areas of interest are within a given applicator's field. Once this is known, the

user may then proceed to collect data at these areas using the non-immediate data collection mode which provides a disk data storage option for later retrieval by a post-processing program.

Although the development of data post-processing programs was not a primary specified objective of this thesis, it was still desirable to verify that data are being stored to disk correctly. A test program named TESTPLOT was written to verify data storage to disk by retrieving a set of data files (both the text header file and the binary data file), displaying the file parameters and plotting the data on the printer. The listing for this program is provided in Appendix D, and the output for x-axis scan data retrieved from a test file named TEST1 (text) and TEST1.0 (binary) is shown in Figure 17. Similarly, the output for raster scan data stored in files TEST2 and TEST2.0 is shown in Figure 18. This plot shows that the successive x-axis scans in the raster are stored sequentially on disk, and when printed out graphically by TESTPLOT appear as several scans linked together horizontally.

CHAPTER 7

RECOMMENDATIONS AND CONCLUSIONS

Recommendations

Regarding the further development and enhancement of the system, a number of recommendations can be made. First, it is highly recommended that a failsafe system be added to protect against possible failure of the system's limit switches. This system should be designed so that when triggered, it immediately turns off the power to the stepper drivers. Such an enhancement would be relatively inexpensive, yet could prevent damage to costly components.

Second, to exploit the full capabilities of the system, post-processing software for the data stored on disk should be developed. The system has the capability of generating two-dimensional raster data; however, without any software to process and display this data, this feature is useless.

Third, additional amplifier circuits might be designed for measuring pulsed (rather than continuous wave) ultrasonic waves and for field measurements of microwave applicators.

Fourth, some form of data averaging might be added to STEPPER's data collection routine to reduce any noise that may be present, and produce smoother plots. Finally, an additional output driver routine for CONTROLLER may be desirable, to allow data output to a pen plotter which would produce higher quality plots than a standard printer.

Conclusions

Much time and effort were put into specifying and ordering the mechanical hardware, and designing and debugging the electronic hardware and software for the system just described. Conceptually, the project was simple, with similar systems already in existence being proof of its feasibility. The actual implementation of the system, however, proved to be nontrivial as with any system development. Significant effort was required to properly integrate the mechanical, electromechanical, electronic, and computer software components into an operational system.

Overall, the system is very much a real-world engineering development project, with primary emphasis on design rather than theory. In the context of the company for which this work was done, the system is a high technology tool which will be used in the development of a much larger and more complex cancer therapy system, which is based heavily on theory. Ultimately, the usefulness of both systems will be determined by the role that hyperthermia plays in future cancer therapy.

TABLES

Table 1

Parts List for Mechanical Hardware

QTY	DESCRIPTION
1	x-axis parallel coupled B4027P5J UniSlide assemblies with 6" sliders and a c-c of 23 1/2". Adjustable gibs for mounting MO92 type motor, scale, pointer, and crank
1	Y axis B4024P5J UniSlide assembly with 6" slider, adjustable gibs for mounting MO92 type motor, scale, pointer, and crank
1	Z axis B6027P5J UniSlide assembly with 8" slider, adjustable gibs for mounting MO92 type motor, scale, pointer, and crank
1	3-piece Y axis reinforcement bracket
1	Superior Electric Slo-Syn MO92-FD08E stepper motor (double shafted version of FD09)
2	Shinkoh Communications Industry Co., Ltd. Astrosyn 34PM-C110 stepper motor

Table 2

Parts List for the Interface Circuit

Note: All resistors are 1/4 W 10% unless noted.

QTY	DESCRIPTION
1	Analog Devices AD572, 12 bit analog to digital converter
2	6522 versatile interface adapters
1	74LS05 hex inverter with open collector outputs
1	10 pF ceramic disk capacitor
3	0.1 uF ceramic disk capacitors
4	1.0 uF tantalum capacitors
9	1 K ohm resistors
2	4.7 K ohm resistors
1	3.9 M ohm resistor
1	100 ohm 10-turn trimmer potentiometer
1	20 K ohm 10-turn trimmer potentiometer
Miscellaneous:	A #4609 Vector Board, IC sockets, wire, etc.

Table 3

40 Conductor Ribbon Connector Pin-Out

40 pin edge connector
(3M Scotchflex #3464-0000)

Pin	
1	
2	
3	
4	
5	
6	To 14 pin connector (female) (3M Scotchflex #3385-6014)
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	To 10 pin connector (female) (3M Scotchflex #3473-6010)
20	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	
32	
33	Unused
34	
35	
36	
37	
38	
39	
40	

Table 4

Parts List for the Driver Pack Mini Box

QTY	DESCRIPTION
1	Bud Radio Inc. #CU-3002-A aluminum mini box (4"L x 2 1/8"W x 1 5/8"H)
1	14 pin connector (male) 3M Scotchflex #3414-4305
3	6 pin connectors (female) Molex #03-09-1061
Miscellaneous:	2 sheet metal screws, 1 rubber grommet, 28 spade lug connectors, wire, 3 male Molex connectors (#03-09-2061) for the stepper motors.

Table 5

Parts List for the Limit Switch Distribution Box

QTY	DESCRIPTION
1	Bud Radio Inc, #CU-3002-A aluminum mini box (4"L x 2 1/8"W x 1 5/8"H)
1	10 pin connector (male) 3M Scotchflex #3446-4305
3	7 pin connectors (female) Amphenol #126-192 (optional: locking clip #126-1069)
Miscellaneous:	wire, male Amphenol connectors for cable assemblies (#126-191, hood and clamp: #126-1063, locking sleeve: #126-1064)

Table 6

Parts List for the Amplifier Circuit

Note: All resistors are 1/4 W 10% unless noted.

QTY	DESCRIPTION
1	MC1590 wideband amplifier
1	LM158 op amp
1	Analog Devices 442J RMS to DC converter
1	0.1 uF ceramic disk capacitor
3	6.8 uF tantalum capacitors
2	1 K ohm resistors
1	2 K ohm resistor
1	5.6 K ohm resistor
1	10 K ohm resistor
1	5 K ohm 10-turn trimmer potentiometer
1	20 K ohm 10-turn trimmer potentiometer
1	100 K ohm 10-turn trimmer potentiometer
1	Amidon Assoc. Inc. T-94-15 Torroid
1	BNc connector (female)
Miscellaneous:	perf board, IC sockets, wires, mounting hardware, etc.

Torroid winding information:

Primary: approximately 100-turns enameled 28 gauge copper wire

Secondary: approximately 20-turns enameled 24 gauge copper wire

Use a vector impedance meter to fine-tune at 1 MHz

Table 7

Interface Card Registers

ADDRESS	REGISTER	DESCRIPTION
50176 ($C400)	HDATA	high byte of digitized data (input)
50177 ($C401)	LDATA	Bit# 0-3 least significant nibble of digitized data (input) 4-6 x,y,z motor enable bits (output) '1' to enable 7 convert start bit (output) pulse A '1' to start to A to D conversion
50178 ($C402)	DDR (HDATA)	Data Direction Register for HDATA
50179 ($C403)	DDR (LDATA)	Data Direction Register for LDATA
50188 ($C40C)	PCR1	Peripheral Control Register (Port #1)
50189 ($C40D)	IFR1	Interrupt Flag Register (Port #1)
50192 ($C410)	CKDIR	Bit# 0 - 1 ck & dir x (output) 2 - 3 ck & dir y (output) 4 - 5 ck & dir z (output) 6 - 7 unused
50193 ($C411)	LIMIT	Bit# 0-2 x,y,z 'in' limits (input) 'in' 'in' is towards actual limit switch housing 3-5 x,y,z 'out'limits (input) 6-7 unused
50194 ($C412)	DDR (CKDIR)	Data Direction Register for CKDIR
50195 ($C413)	DDR (LIMIT)	Data Direction Register for LIMIT

Table 8

Port 1 and 2 I/O Lines

Port #1

I/O BIT	INPUT OR OUTPUT	DESCRIPTION
PA0-PA3	Inputs	least significant nibble of digitized data
PA4-PA6	Outputs	x, y, and z motor enable
PA7	Output	start Convert line
PB0-PB7	Inputs	most significant byte of digitized data

Port #2

PA0-PA2	Inputs	x, y, and z "in" limits
PA3-PA5	Inputs	x, y, and z "out" limits
PA6-PA7	------	unused
PB0-PB1	Outputs	x clock and direction lines
PB2-PB3	Outputs	y clock and direction lines
PB4-PB5	Outputs	z clock and direction lines
PB6-PB7	-------	unused

Table 9

STEPPER and CONTROLLER Linkage Registers

STARTING ADDRESS		REGISTER	DESCRIPTION
2051	($803)	XYZ	Bit# 0 '1' for X 1 '1' for Y 2 '1' for Z 3 '1' for manual positioning using arrow keys 4 '1' to clear count registers 5 '1' to supress real time display 6 '1' to disable motors when done 7 '1' to enable data storage Example Codes (D = don't care, # = active bit position) D##00000 = home 11111111 = reset DATPTR 00010000 = clear count registers ###01### = manual control ###00### = scan
2052	($804)	MOVE	3 byte absolute move value
2055	($807)	DATING	2 byte sampling increment register
2057	($809)	ERFLG	error if nonzero
2058	($80A)	GCX	geometric coordinate registers
2062	($80E)	GCY	4 byte packed BCD #'s
2066	($812)	GCZ	(ABC.DEFGH) stored in excess-500 format: AB CD EF GH
2070	($816)	XCOUNT	contain 24 bit absolute
2073	($819)	YCOUNT	pulse count relative to
2076	($81C)	ZCOUNT	home reference
2079	($81F)	MAXCNT	contains 24 bit absolute position where a peak has been found
2082	($822)	MAXVAL	12 bit peak data value

Table 10

Text File Format

ITEM#	VARIABLE NAME	DESCRIPTION
1	GC$	a "Y" or "N" indicating whether geometric coordinates are used
2	LD	length of data buffer (# bytes of data samples)
3	TS$	a "Y" or "N" indicating whether data are from a transverse scan (if not, longitudinal scan)
4-6	---	x, y, and z geometric coordinate offset from home (pulses); these values will be zero if tank coordinates are used
7-9	SX,SY,SZ	center coordinates of scan (pulses)
10-12	RX,RY,RZ	scan ranges (pulses)
13-15	IX,IY,IZ	scan increments (pulses)

FIGURES

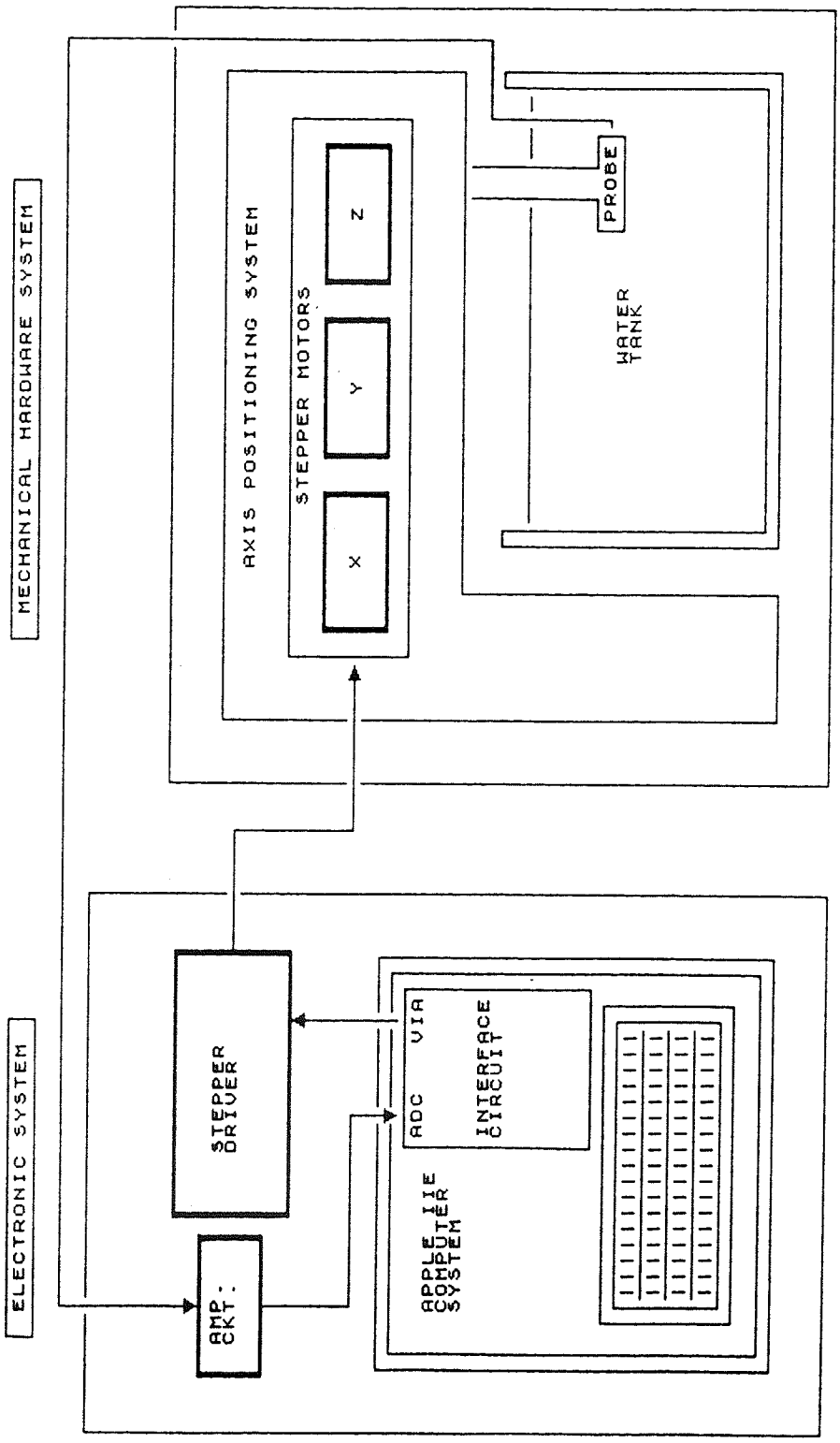


Figure 1. Functional system diagram.

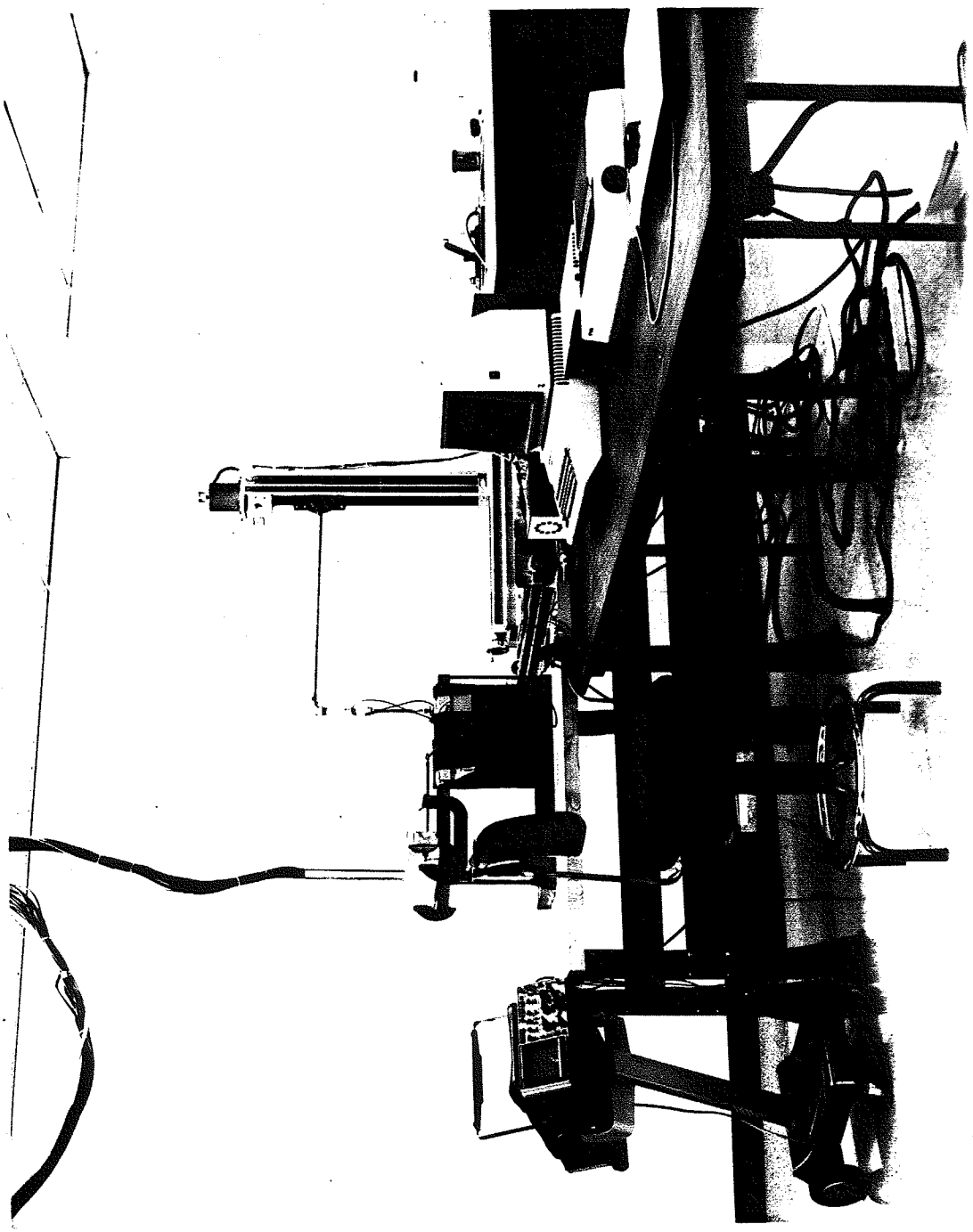


Figure 2. Mechanical system.

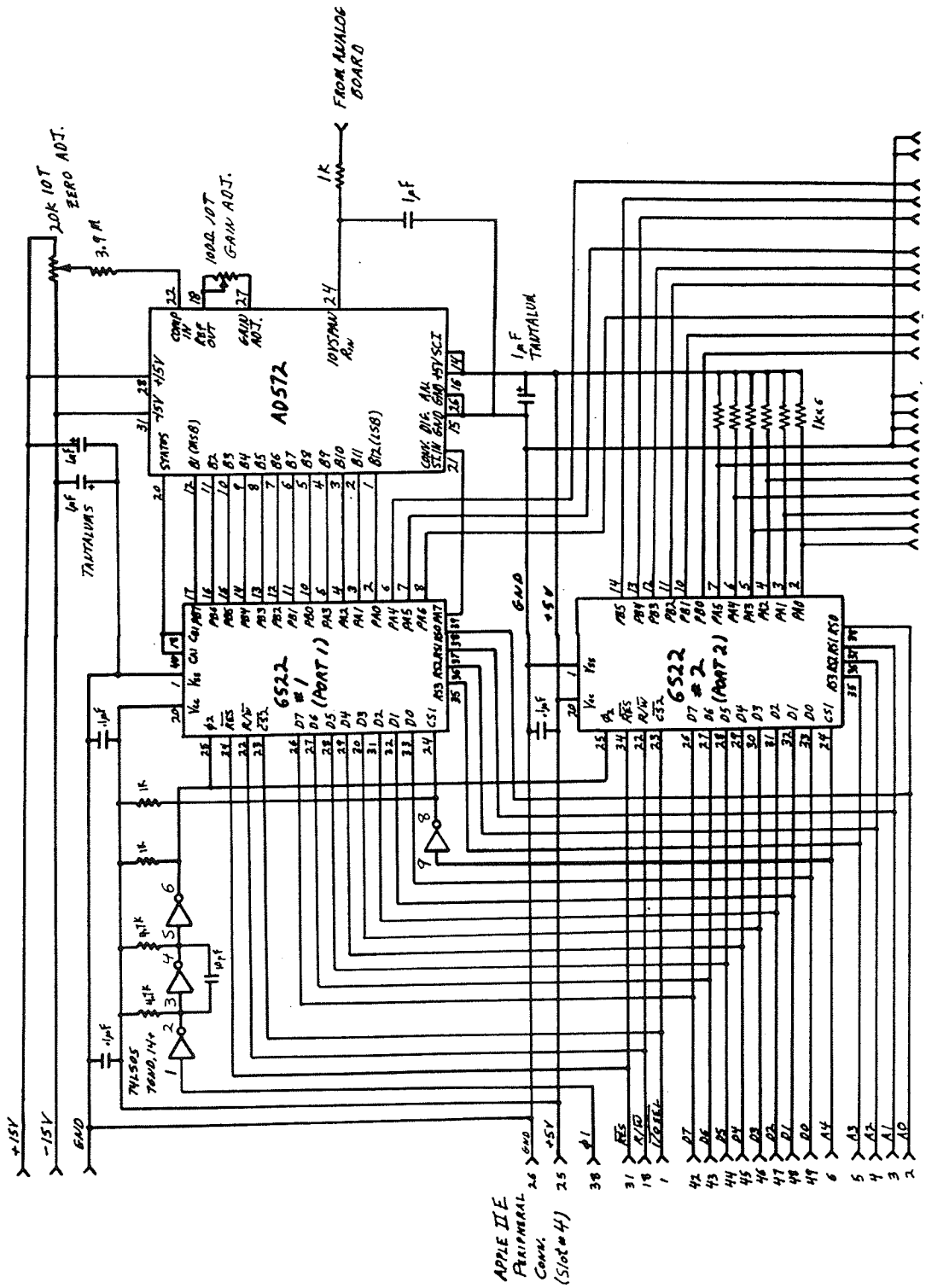


Figure 3. Interface circuit schematic.

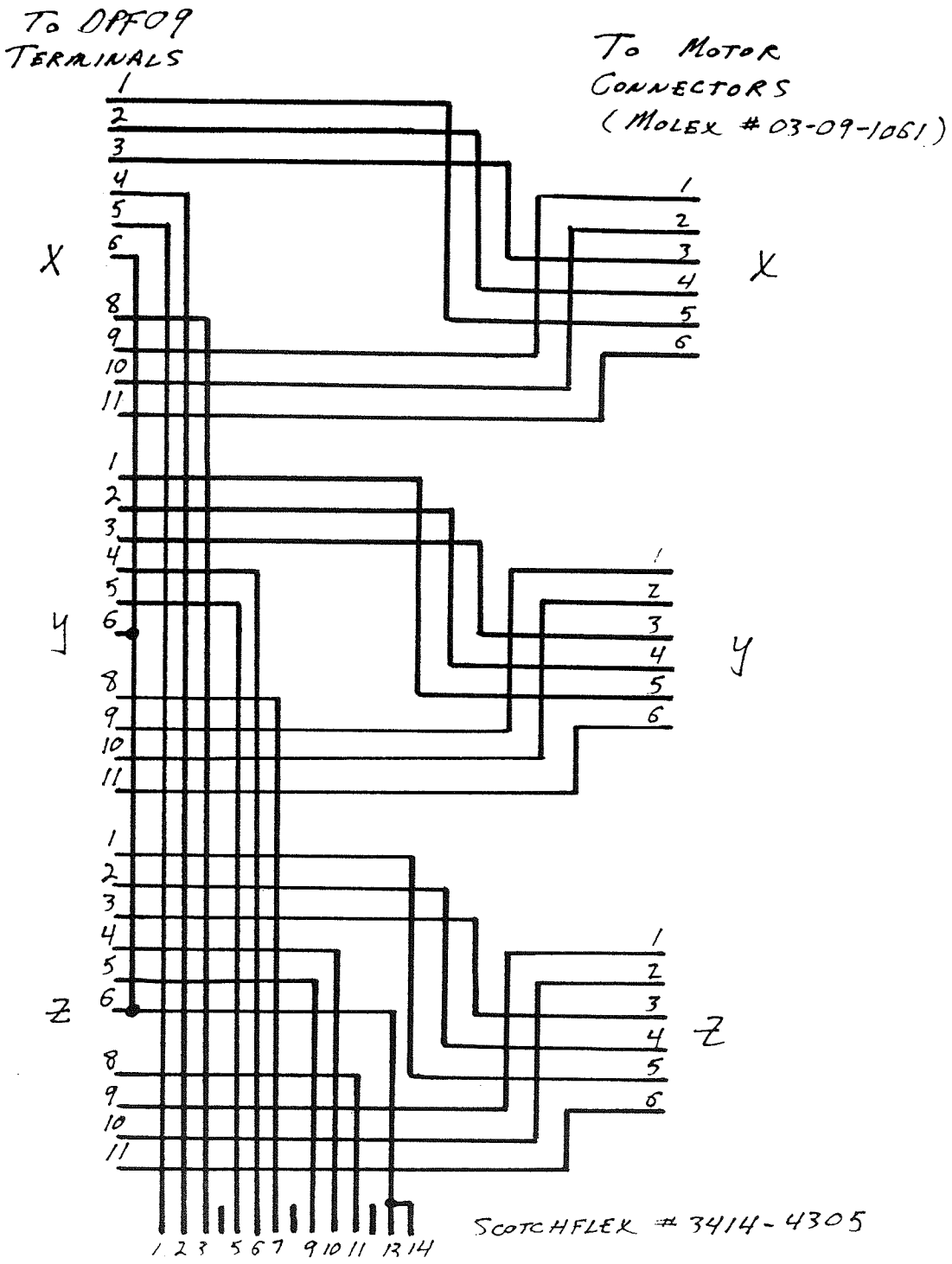


Figure 4. Driver pack mini box schematic.

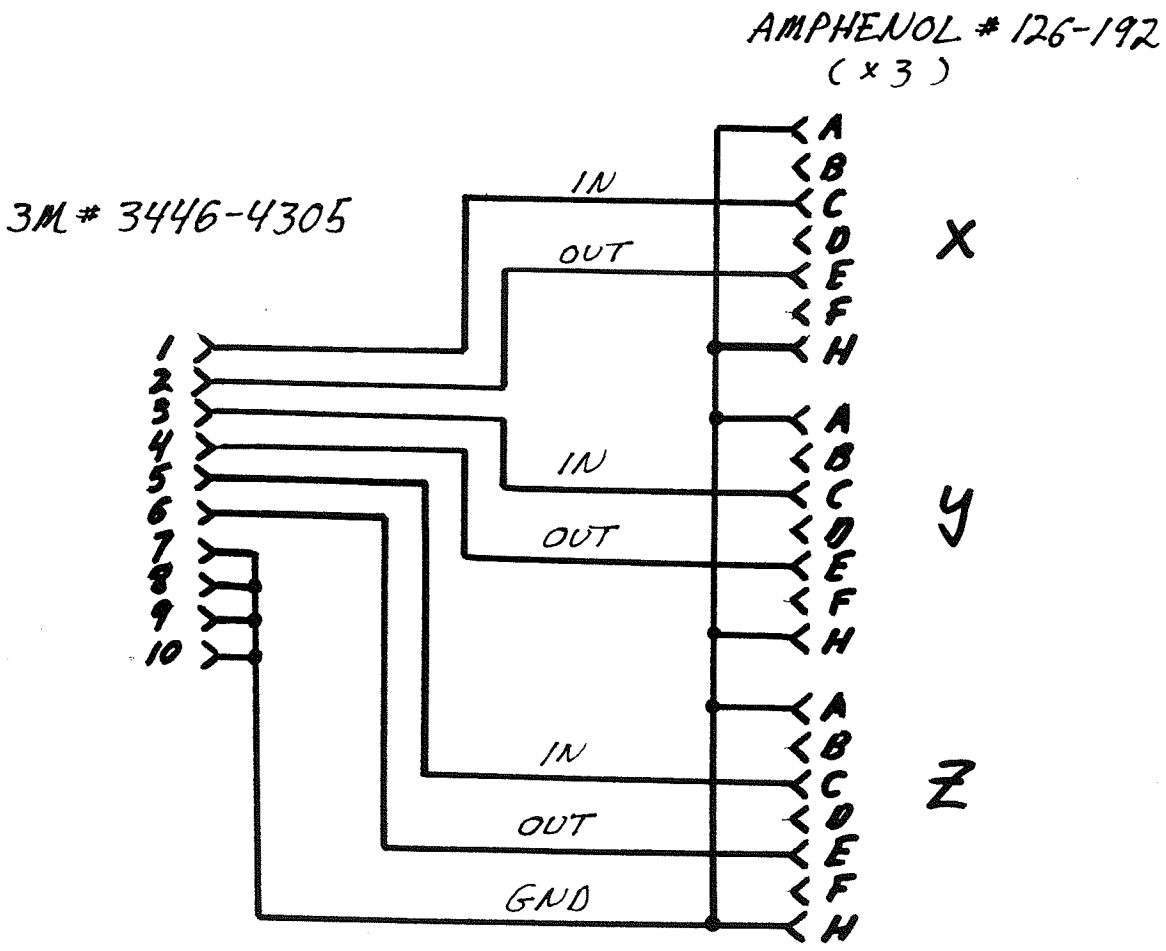


Figure 5. Limit switch distribution box schematic.

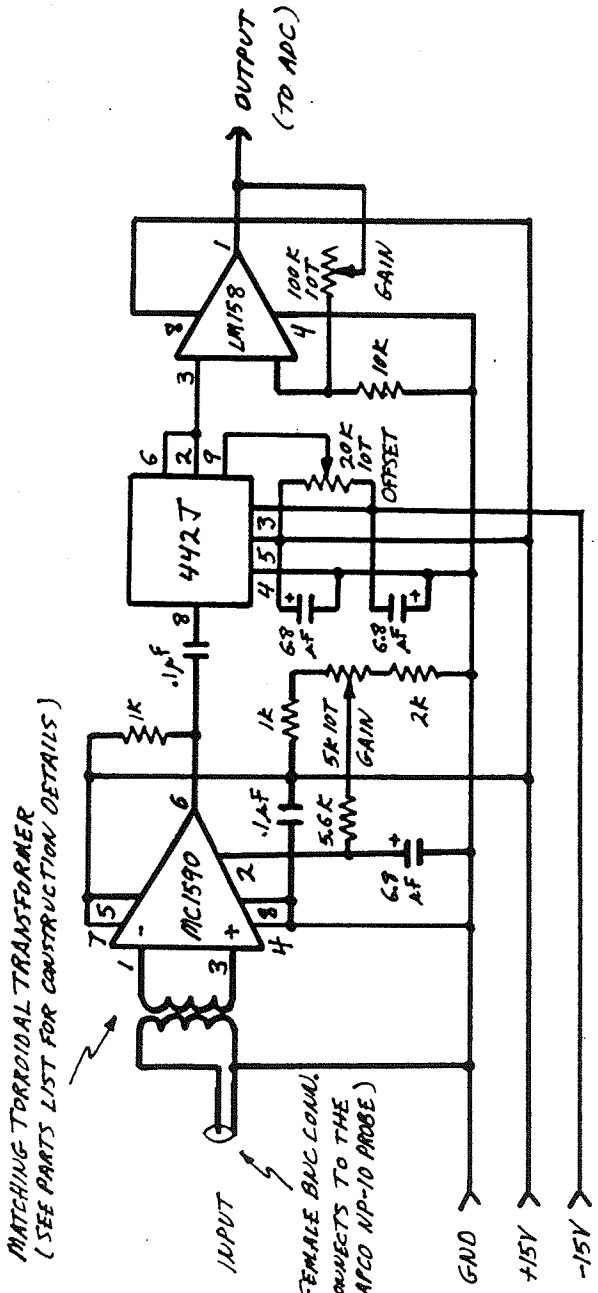


Figure 6. Amplifier circuit schematic.

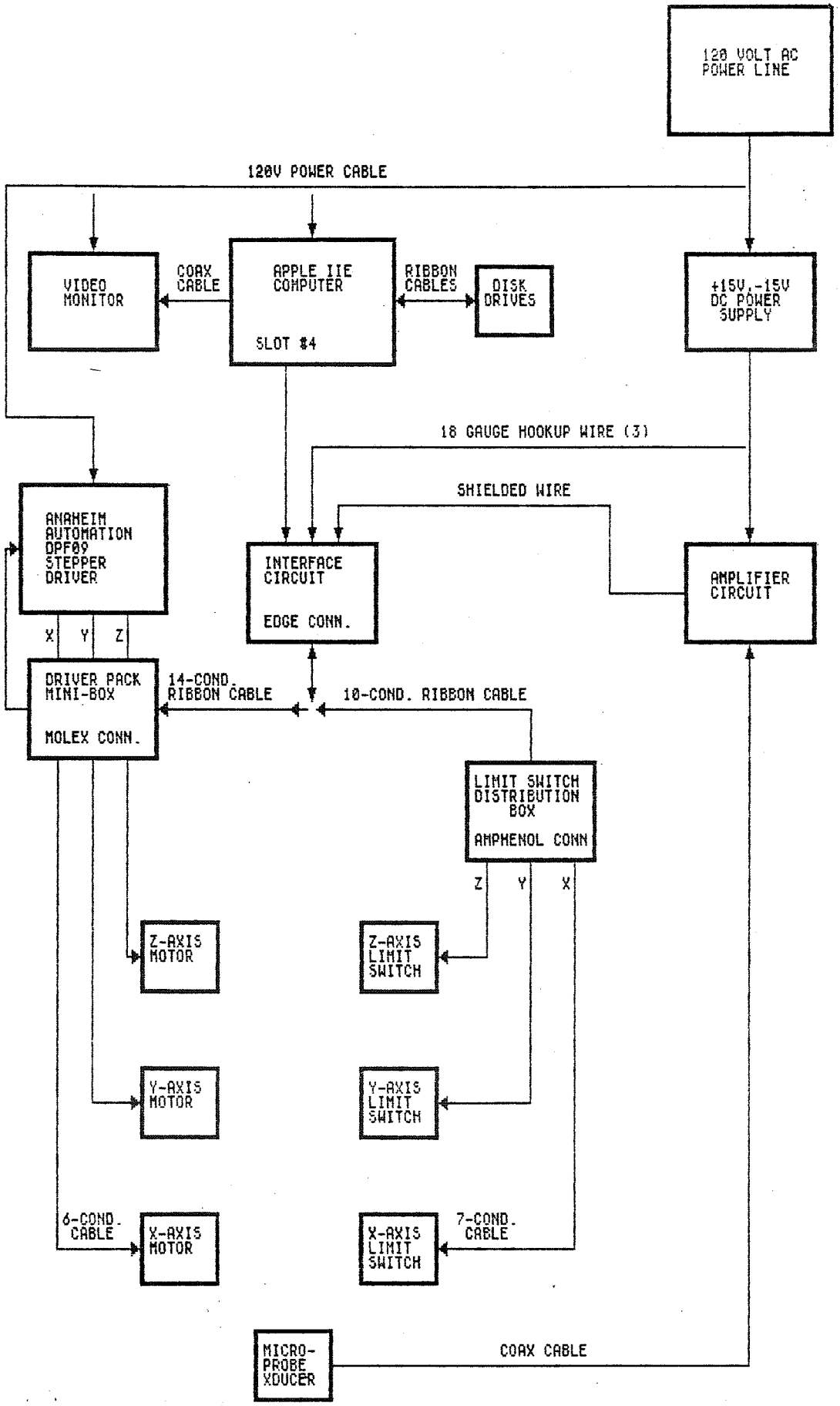


Figure 7. System level electronic interconnections.

```
]CATALOG
DISK VOLUME 254
*A 004 HELLO
*A 068 CONTROLLER
*T 102 STEPPER
*B 010 STEPPER.OBJO
*B 003 RAMP
*T 005 TRANSFER
*B 002 TRANSFER.OBJO
*A 020 DRIVER
*T 007 HISCAN
*B 002 HISCAN.OBJO
*A 005 TEMPLATE
*T 002 PSIO.BP&GRAFIX300BAUD

]
]
```

Figure 8. Catalog listing of system disk.

```
P.S.I.O. TEXT CONFIGURATION: PSIO.BP&GRAFIX300BAUD

SLOT NUMBER          PHAN 2     REAL 1
COLD START STRING

SEND STRING TO       PARALLEL
FORM WIDTH           80         80
FORM LENGTH          60         60
AUTO LINEFEED        ENABLED    ENABLED
VIDEO MODE           DISABLED   DISABLED
DELAY AFTER CR       NONE
LINEFEED MASK        ENABLED
CONVERT LOWER CASE   DISABLED
SHIFT MOD            DISABLED
XON/XOFF             ENABLED
DUPLEX MODE          HALF
BAUD RATE            300
PARITY               EVEN
DATA FORMAT          7 DATA 2 STOP

P.S.I.O. GRAPHIC CONFIGURATION

GRAPHICS SETUP STRING          [AH
                               ^ ^
TEXT RESET STRING              [2
                               ^
GRAPHICS ENTRY STRING          [K
                               ^
GRAPHICS EXIT STRING

SEND GR. ENTRY BEFORE COMMANDS  NO
SEND COUNT AFTER ENTRY STRING   YES
GRAPHIC COUNT SENT AS           16 BITS
GRAPHIC LINE FEED STRING        J
                                ^
USE GR. MODE TO ADVANCE PAPER   NO
NUMBER OF BITS PRINTED (1-8)    8
TOP OF GRAPHICS BYTE            BIT 0
GRAPHIC MASK (BINARY)           00000000
SERIAL OR PARALLEL DUMP         PARALLEL

NOTE: CHARACTERS WITH ^'S UNDER THEM ARE CONTROL CHARACTERS - [ IS 'ESC'
                                                               ^
```

Figure 9. PSIO card configuration printout.

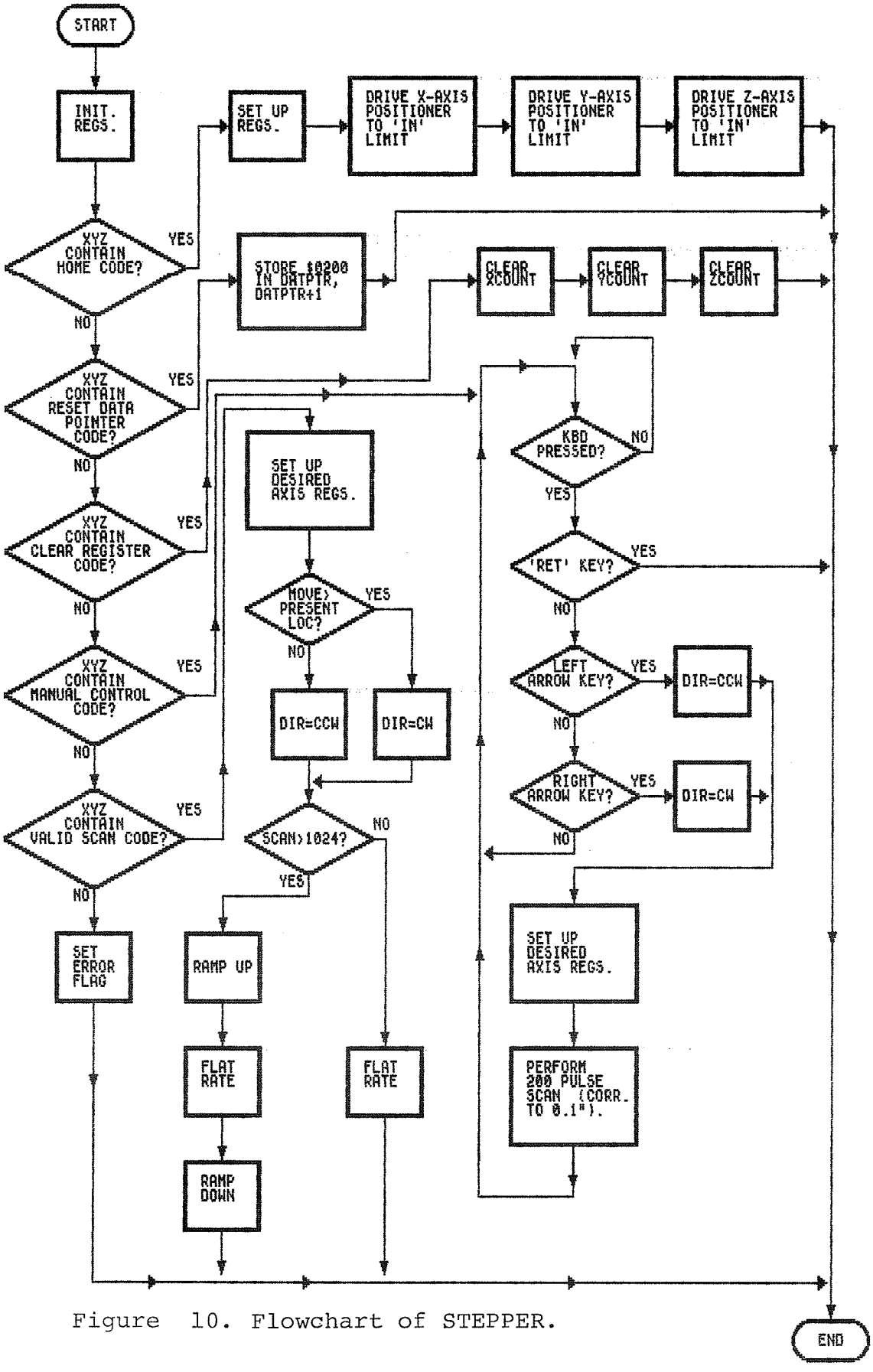


Figure 10. Flowchart of STEPPER.

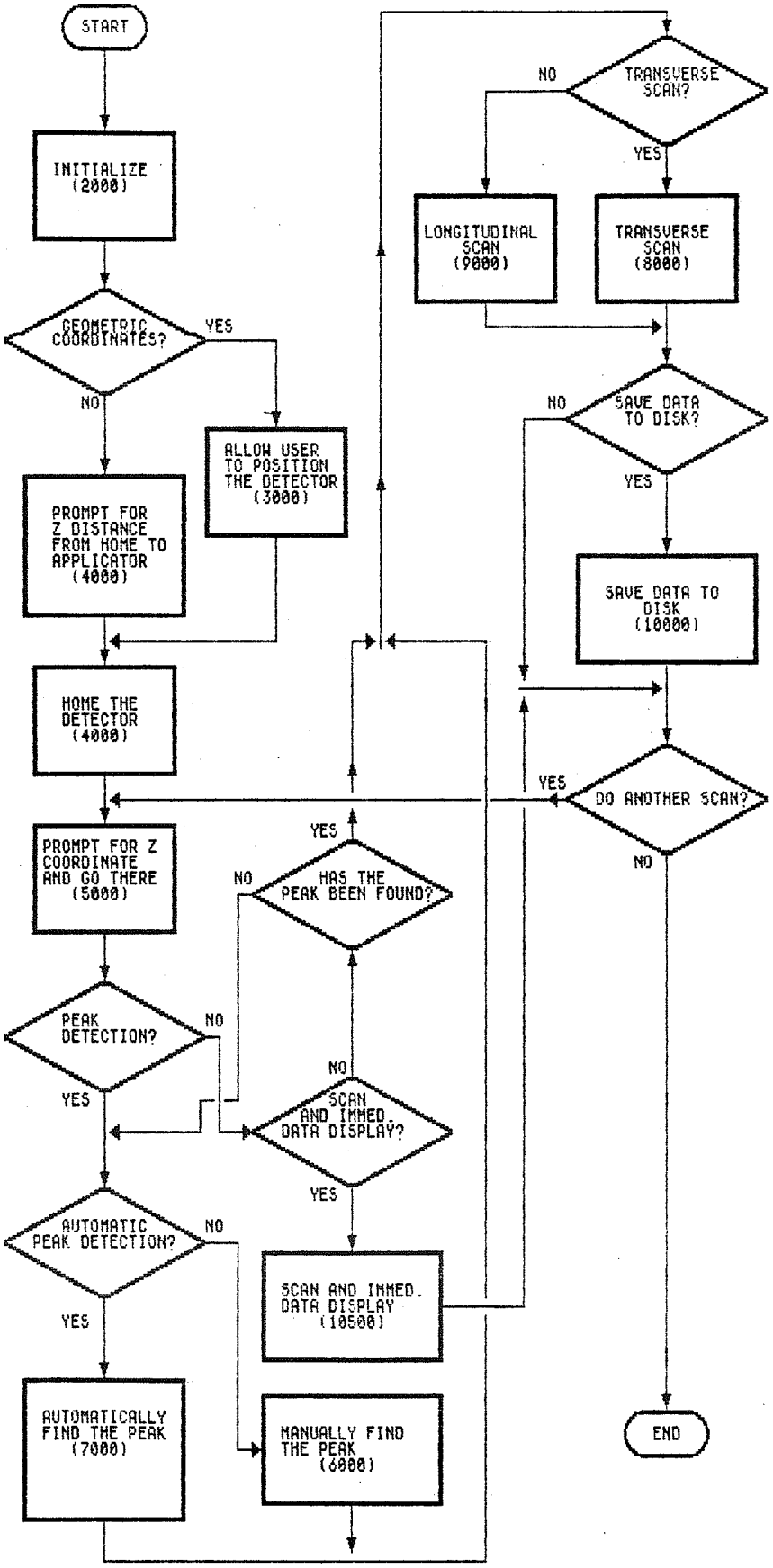


Figure 11. Flowchart of CONTROLLER.

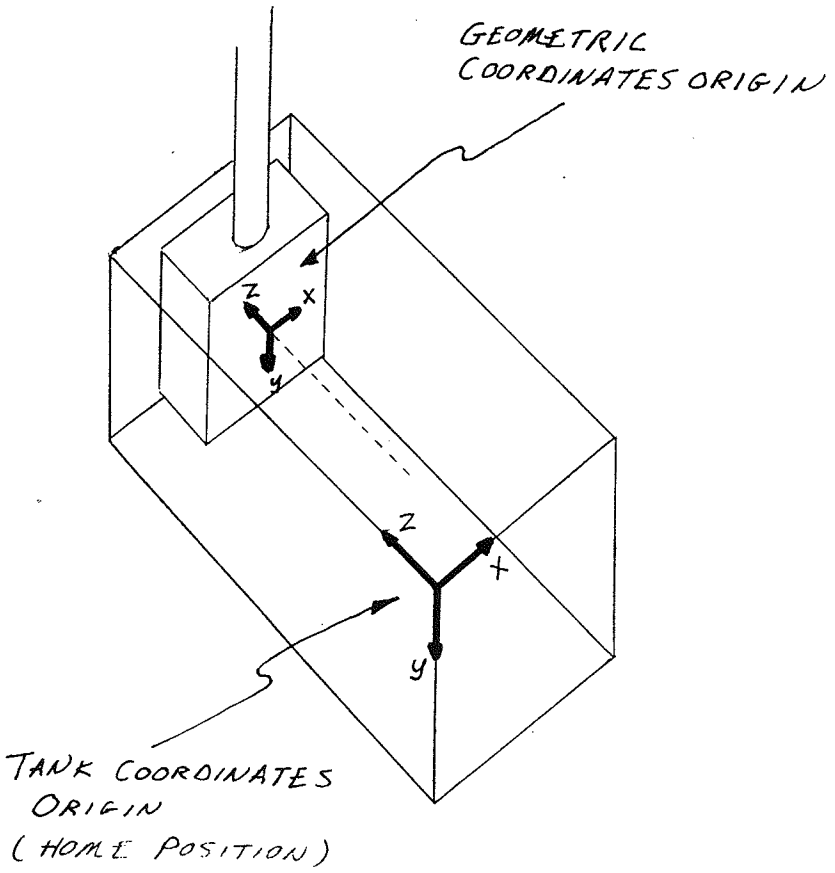
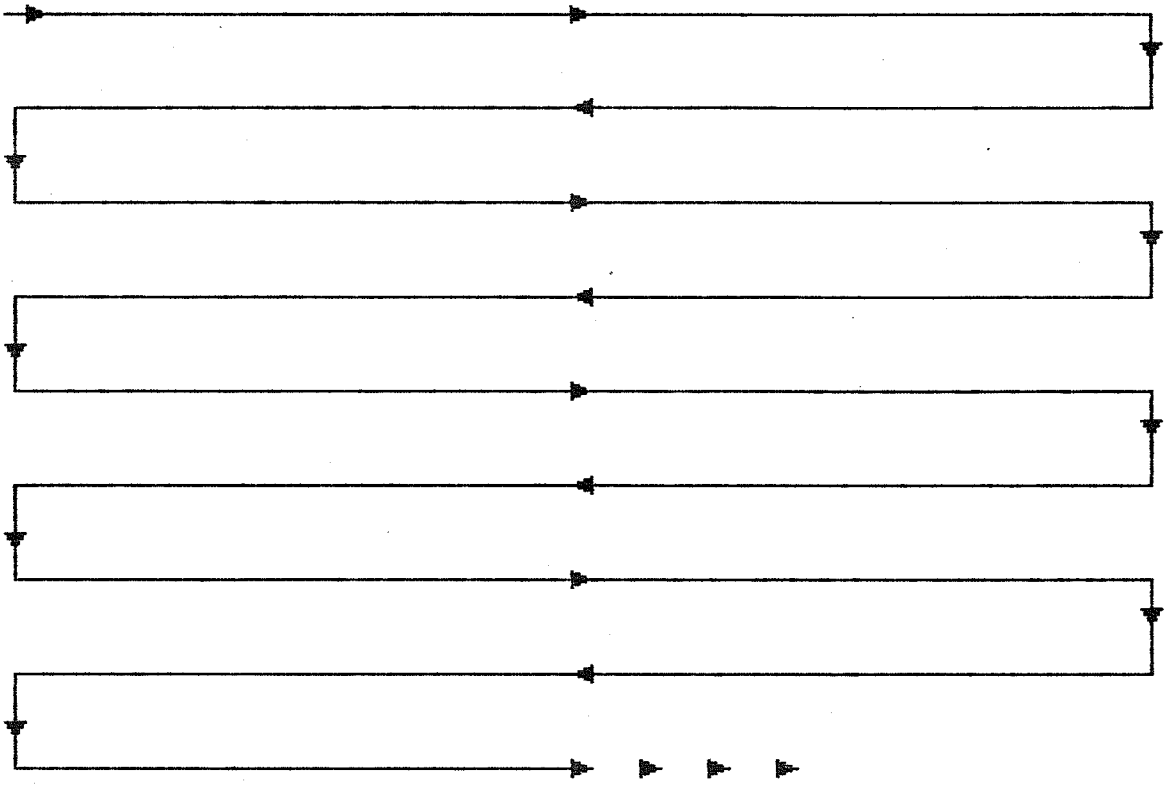
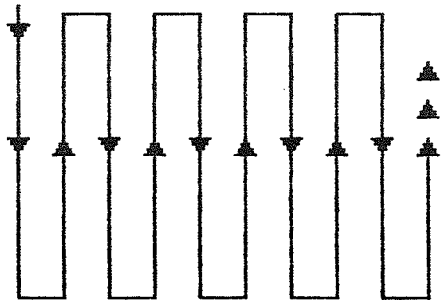


Figure 12. Geometric and tank coordinate positive axes.



(a)



(b)

Figure 13. Scan paths for auto peak detection (a) course scan and (b) fine scan.

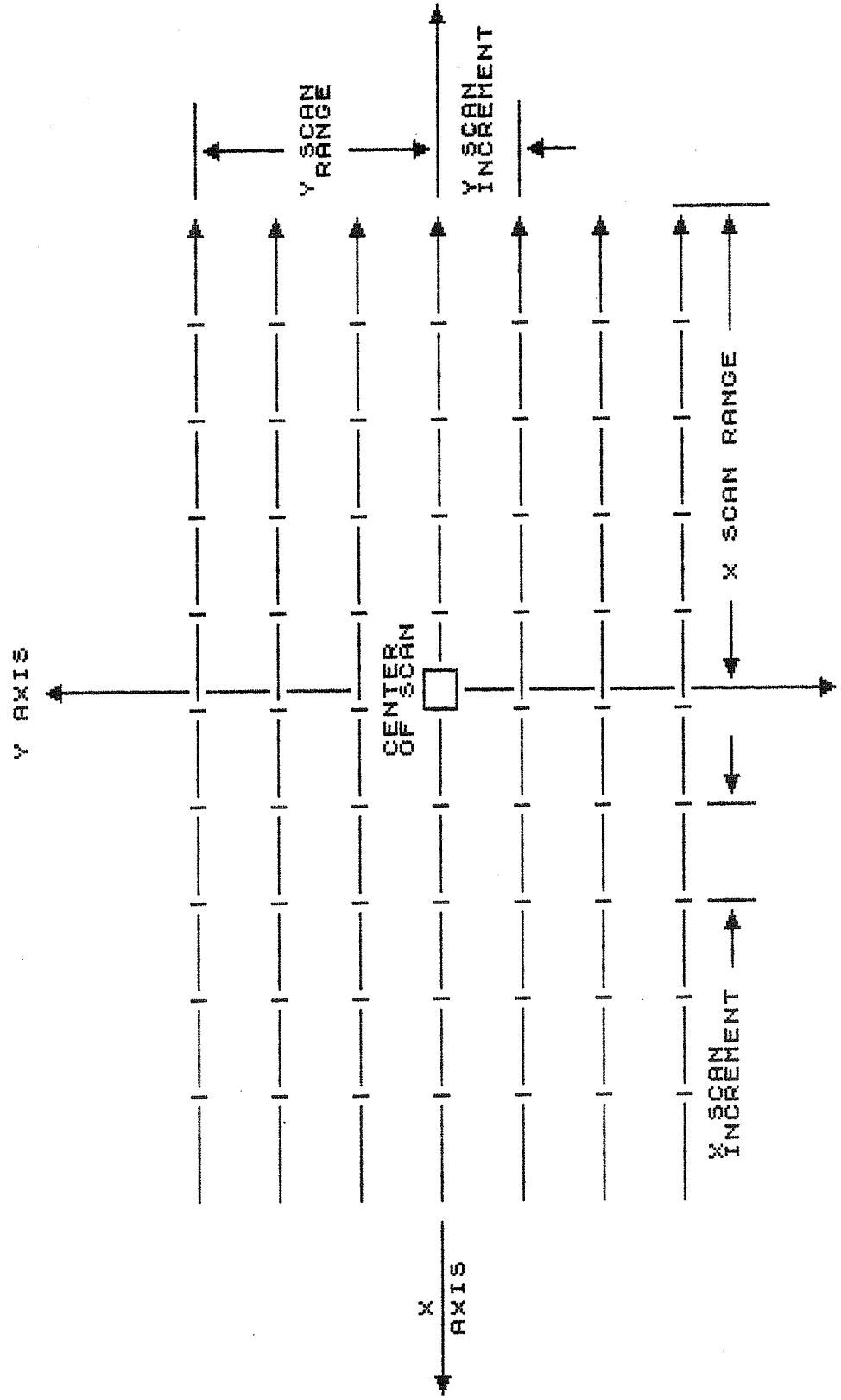


Figure 14. Transverse scan center, range, and increment.

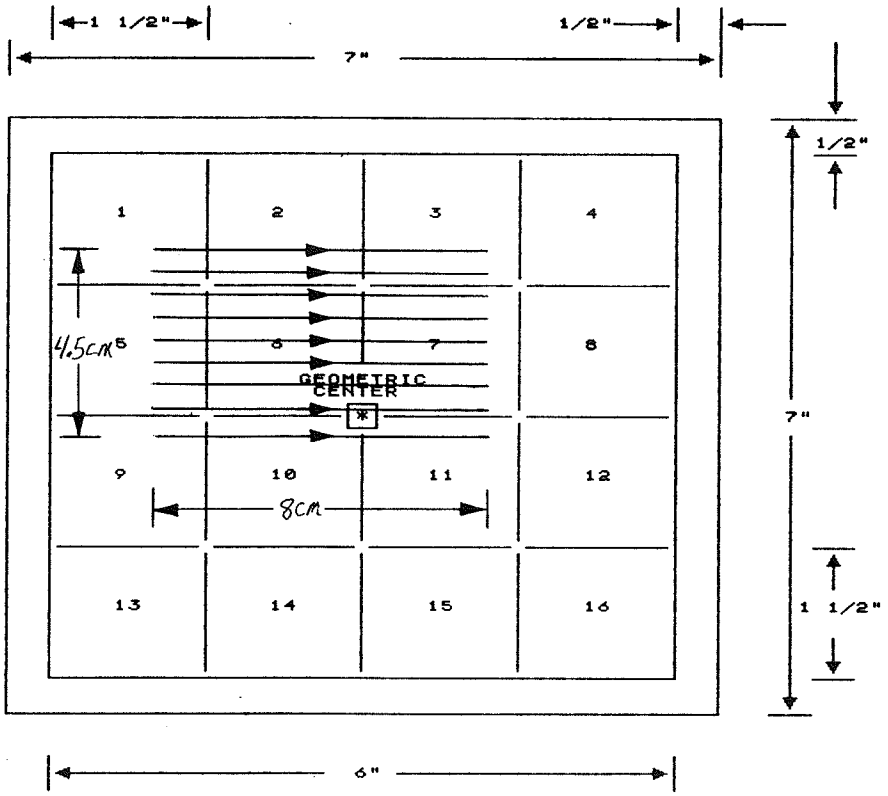
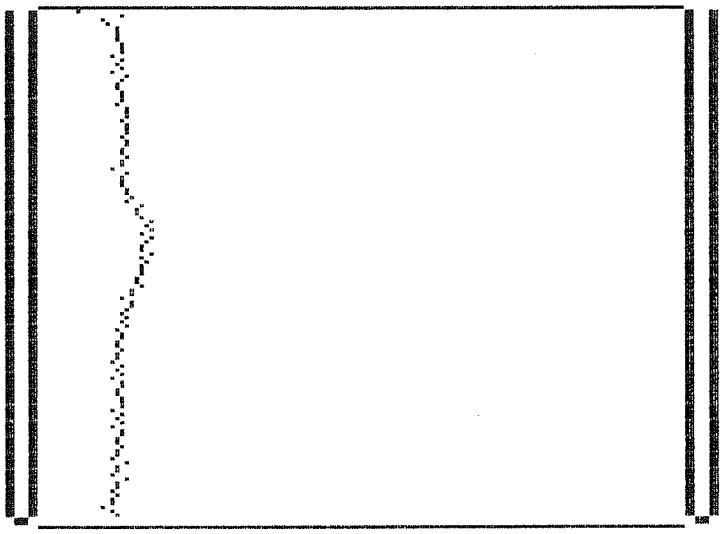


Figure 15. Applicator and scan paths for test setup.

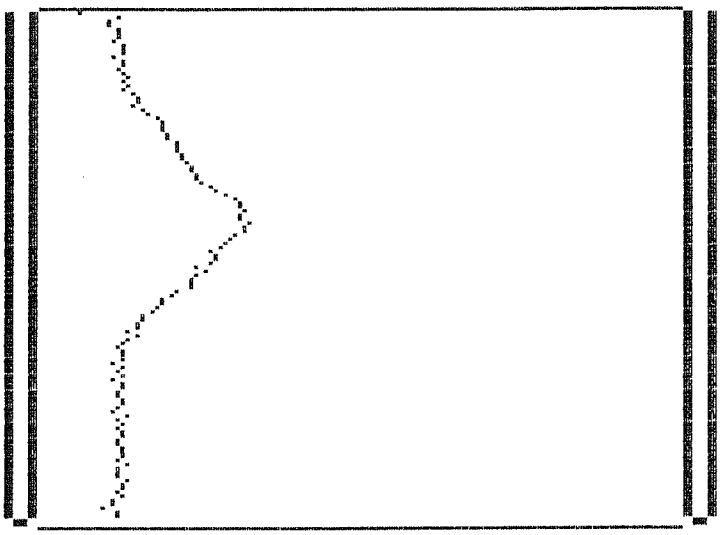
```
]X-AXIS SCAN
GEOMETRIC COORDINATES
SCAN FROM X=-5.00126 (CM)
TO X=3 (CM)
Y=-4.00050001 (CM)
Z=-4.0005 (CM)
PEAK VALUE AT X=-1.69799 (CM)
WINDOW HEIGHT=7.96544 (CM)
```



(a)

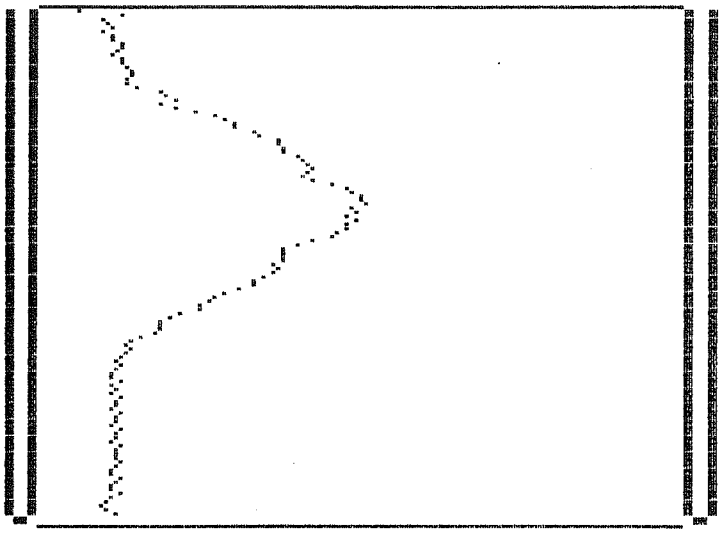
Figures 16a - j. X axis scan plots for y =-4.0 cm to +0.5 cm at 0.5 cm increments.

X-AXIS SCAN
GEOMETRIC COORDINATES
SCAN FROM X=-5.00126 (CM)
TO X=3 (CM)
Y=-3.50012 (CM)
Z=-4.0005 (CM)
PEAK VALUE AT X=-1.69799 (CM)
WINDOW HEIGHT=7.96544 (CM)



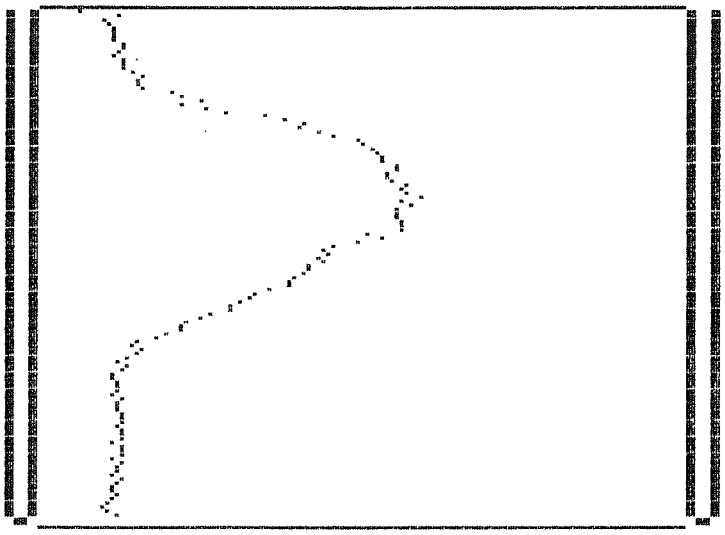
(b)

X-AXIS SCAN
GEOMETRIC COORDINATES
SCAN FROM X=-5.00126 (CM)
TO X=3 (CM)
Y=-3.00101 (CM)
Z=-4.0005 (CM)
PEAK VALUE AT X=-1.95199 (CM)
WINDOW HEIGHT=7.96544 (CM)



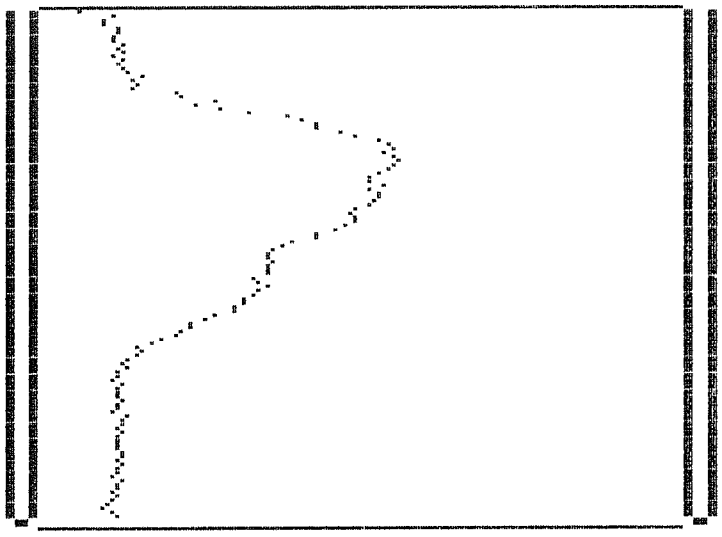
(c)

X-AXIS SCAN
GEOMETRIC COORDINATES
SCAN FROM X=-5.00126 (CM)
TO X=3 (CM)
Y=-2.50063 (CM)
Z=-4.0005 (CM)
PEAK VALUE AT X=-2.07899 (CM)
WINDOW HEIGHT=7.96544 (CM)



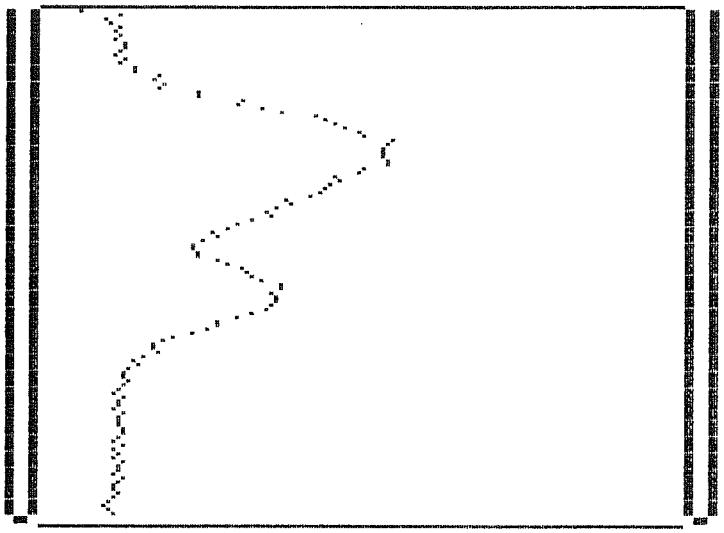
(d)

X-AXIS SCAN
GEOMETRIC COORDINATES
SCAN FROM X=-5.00126 (CM)
TO X=3 (CM)
Y=-2.00025 (CM)
Z=-4.0005 (CM)
PEAK VALUE AT X=-2.65049 (CM)
WINDOW HEIGHT=7.96544 (CM)



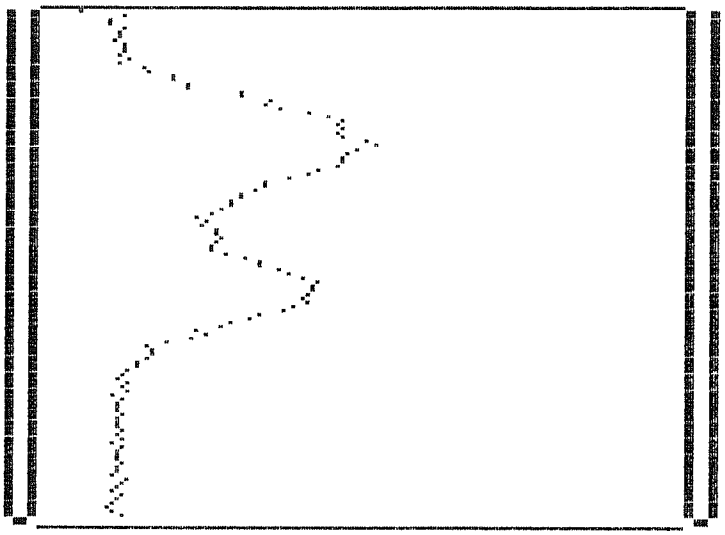
(e)

X-AXIS SCAN
GEOMETRIC COORDINATES
SCAN FROM X=-5.00126 (CM)
TO X=3 (CM)
Y=-1.50114001 (CM)
Z=-4.0005 (CM)
PEAK VALUE AT X=-2.96799 (CM)
WINDOW HEIGHT=7.96544 (CM)



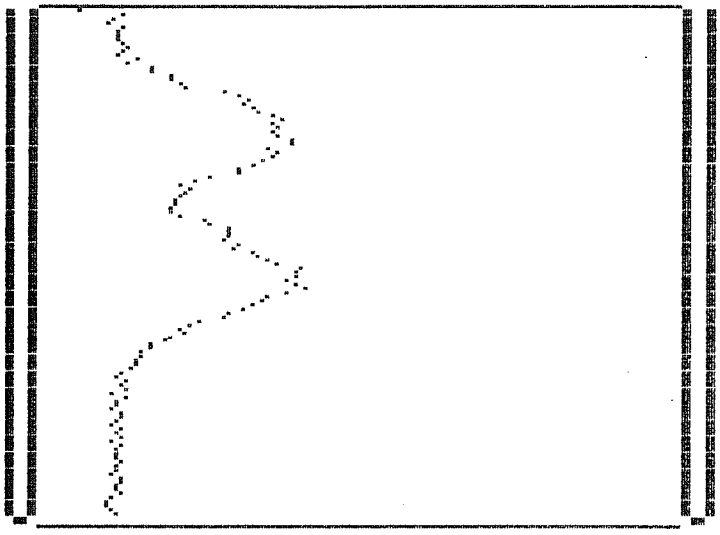
(f)

X-AXIS SCAN
GEOMETRIC COORDINATES
SCAN FROM X=-5.00126 (CM)
TO X=3 (CM)
Y=-1.00076001 (CM)
Z=-4.0005 (CM)
PEAK VALUE AT X=-2.90449 (CM)
WINDOW HEIGHT=7.96544 (CM)



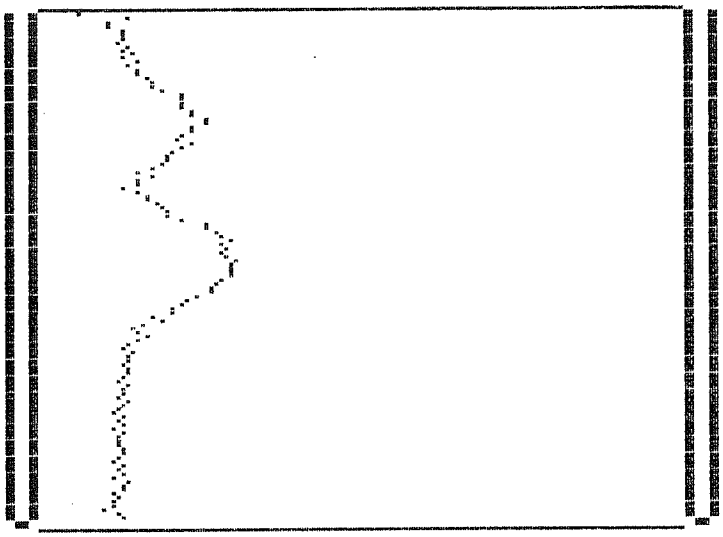
(g)

X-AXIS SCAN
GEOMETRIC COORDINATES
SCAN FROM X=-5.00126 (CM)
TO X=3 (CM)
Y=-.500380005 (CM)
Z=-4.0005 (CM)
PEAK VALUE AT X=-.61849 (CM)
WINDOW HEIGHT=7.96544 (CM)



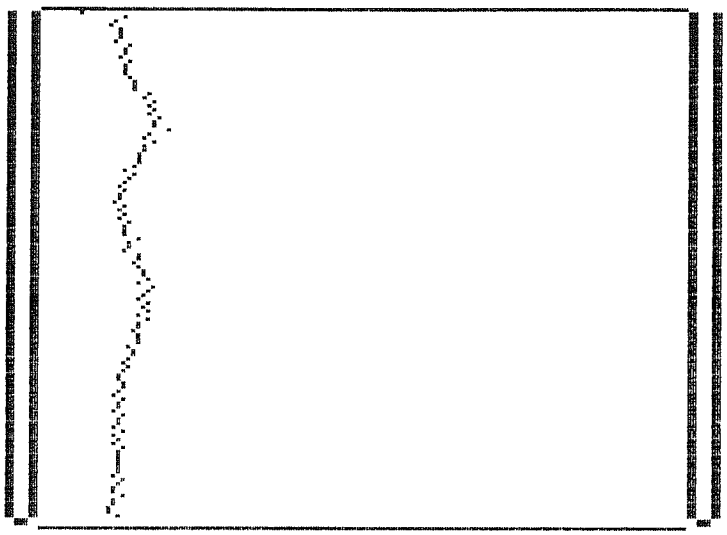
(h)

X-AXIS SCAN
GEOMETRIC COORDINATES
SCAN FROM X=-5.00126 (CM)
TO X=3 (CM)
Y=-4.84466553E-09 (CM)
Z=-4.0005 (CM)
PEAK VALUE AT X=-1.12649 (CM)
WINDOW HEIGHT=7.96544 (CM)



(i)

X-AXIS SCAN
GEOMETRIC COORDINATES
SCAN FROM X=-5.00126 (CM)
TO X=3 (CM)
Y=.499109995 (CM)
Z=-4.0005 (CM)
PEAK VALUE AT X=-3.15849 (CM)
WINDOW HEIGHT=7.96544 (CM)



(j)

```
FILENAME: TEST1
GEOMETRIC COORDINATES
162 DATA SAMPLES (324 BYTES)
TRANSVERSE SCAN

CENTER OF SCAN COORDINATES:
X=-1.47574
Y=-2.17297
Z=-4

SCAN RANGES:
X RANGE=4
Y RANGE=0
Z RANGE=0

SCAN INCREMENTS:
X INCREMENT=.05
Y INCREMENT=0
Z INCREMENT=0
```

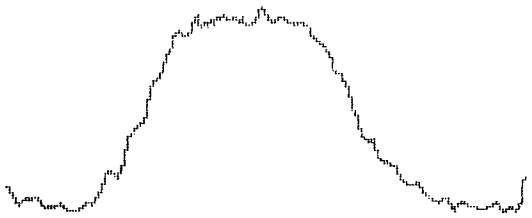


Figure 17. TEST1 output.

```
FILENAME: TEST2
GEOMETRIC COORDINATES
697 DATA SAMPLES (1394 BYTES)
TRANSVERSE SCAN

CENTER OF SCAN COORDINATES:
X=-.66802
Y=-2.07772
Z=-4

SCAN RANGES:
X RANGE=4
Y RANGE=4
Z RANGE=0

SCAN INCREMENTS:
X INCREMENT=.2
Y INCREMENT=.5
Z INCREMENT=0
```

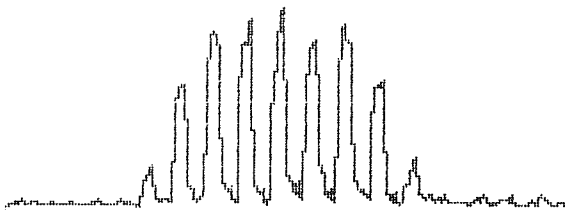


Figure 18. TEST2 output.

APPENDIX A

Listing of STEPPER

```
SOURCE FILE: STEPPER
0000:           1 *********************************
0000:           2 *                               *
0000:           3 *  SCANNER SYSTEM STEPPER MOTOR *
0000:           4 *  CONTROLLER PROGRAM           *
0000:           5 *                               *
0000:           6 *  COPYRIGHT 1984               *
0000:           7 *  DAVE PADGITT                 *
0000:           8 *  UNIVERSITY OF ILLINOIS       *
0000:           9 *  AND URI THERM-X INC.         *
0000:          10 *                               *
0000:          11 * 10/21/84                      *
0000:          12 *********************************
0000:          13 *
0000:          14 *
0000:          15 * MISCELLANEOUS NOTES
0000:          16 *
0000:          17 *
0000:          18 * A VALID COMMAND CODE MUST BE
0000:          19 * POKED INTO XYZ COMMAND REGISTER
0000:          20 * PRIOR TO ENTRY.
0000:          21 * ALSO, THE VIA DDR'S MUST BE
0000:          22 * INITIALIZED CORRECTLY PRIOR
0000:          23 * TO ENTERING THIS PROGRAM.
0000:          24 *
0000:          25 *********************************
0000:          26 *                               *
0000:          27 *  VARIABLE ASSIGNMENTS         *
0000:          28 *                               *
0000:          29 *********************************
0000:          30 * ZERO PAGE POINTERS
0006:          31 REGPTR  EQU  $06      ;POINTS TO COUNT REG.
0008:          32 TBLPTR  EQU  $08      ;POINTS TO RAMP TABLE
001A:          33 GCPTR   EQU  $1A      ;POINTS TO GEOMETRIC COORD. REG.
00CE:          34 SCRPTR  EQU  $CE      ;POINTS TO SCREEN ADDRESS
00EB:          35 DATPTR  EQU  $EB      ;POINTS TO CURRENT LOC. IN DATA TABLE
00ED:          36 HGRPTR  EQU  $ED      ;POINTS TO CURRENT HIRES LOC.
00F9:          37 XPTR    EQU  $F9      ;POINTS TO XTBL
00FB:          38 XDPTR   EQU  $FB      ;POINTS TO XDTBL
00FD:          39 WINPTR  EQU  $FD      ;POINTS TO WINTBL
0000:          40 *********************************
0000:          41 *                               *
0000:          42 * INTERFACE CARD REGISTERS      *
0000:          43 *                               *
0000:          44 * REG.   DESCRIPTION            *
0000:          45 *                               *
0000:          46 * HDATA  HIGH BYTE OF DIGITIZED *
0000:          47 *        DATA (INPUT)           *
0000:          48 *                               *
0000:          49 * LDATA  BIT#                   *
0000:          50 *         0-3 LEAST SIGNIFICANT *
0000:          51 *             NIBBLE OF DIGIT-  *
```

```
0000:          52 *            IZED DATA (INPUT) *
0000:          53 *         4-6 X,Y,Z MOTOR ENABLE*
0000:          54 *            BITS (OUTPUT).    *
0000:          55 *            '1' TO ENABLE.    *
0000:          56 *         7  CONVERT START BIT *
0000:          57 *            (OUTPUT). PULSE A *
0000:          58 *            '1' TO START A TO *
0000:          59 *            D CONVERSION.     *
0000:          60 *                              *
0000:          61 * IFR1    INTERRUPT FLAG REG   *
0000:          62 *         FOR PORT1            *
0000:          63 *                              *
0000:          64 * CKDIR  BIT#                  *
0000:          65 *         0&1 CK & DIR X       *
0000:          66 *             (OUTPUT)         *
0000:          67 *         2&3 CK & DIR Y       *
0000:          68 *             (OUTPUT)         *
0000:          69 *         4&5 CK & DIR Z       *
0000:          70 *             (OUTPUT)         *
0000:          71 *         6&7 UNUSED           *
0000:          72 *                              *
0000:          73 * LIMIT  BIT#                  *
0000:          74 *         0-2 X,Y,Z 'IN' LIMITS *
0000:          75 *             (INPUT).         *
0000:          76 *             'IN' IS TOWARDS  *
0000:          77 *             ACTUAL LIMIT     *
0000:          78 *             SWITCH HOUSING.  *
0000:          79 *         3-5 X,Y,Z 'OUT' LIMITS*
0000:          80 *             (INPUT).         *
0000:          81 *         6&7 UNUSED           *
0000:          82 *                              *
0000:          83 ********************************
C400:          84 SLOT    EQU  $C400     ;SLOT ADDRESS
0000:          85 * VIA #1 (A/D)
C400:          86 HDATA   EQU  SLOT+0    ;PORT B
C401:          87 LDATA   EQU  SLOT+1    ;PORT A
C40D:          88 IFR1    EQU  SLOT+13   ;IFR OF VIA #1
0000:          89 * VIA #2 (CK & DIR & LIMITS)
C410:          90 CKDIR   EQU  SLOT+16   ;PORT B
C411:          91 LIMIT   EQU  SLOT+17   ;PORT A
0000:          92 * CONSTANT VALUES
0015:          93 TWEAK   EQU  21        ;USED TO TWEAK RANGE OF DELAY
0060:          94 NOMVAL  EQU  $60       ;SHOULD BE LARGER THAN TWEAK
0127:          95 GCINC   EQU  $127      ;INC. BY .00127 CM/PULSE
0000:          96 * REAL TIME DISPLAY SCREEN ADDRESSES
06D4:          97 XSCR    EQU  $6D4
06E1:          98 YSCR    EQU  $6E1
06EE:          99 ZSCR    EQU  $6EE
0000:         100 * APPLE SOFT SWITCH LOCATIONS
C000:         101 KEYBD   EQU  $C000     ;SAME AS 80STORE
C010:         102 KEYDWN  EQU  $C010
0000:         103 * AUX. MEM. SOFT SWITCH
C004:         104 RAMWRT  EQU  $C004     ;RAMWRT=OFF,RAMWRT+1=ON
0000:         105 * KEYBOARD KEYS
```

```
000D:          106 RET     EQU  $0D       ;RETURN KEY
0008:          107 LARROW  EQU  $08       ;LEFT ARROW
0015:          108 RARROW  EQU  $15       ;RIGHT ARROW
0000:          109 * ASCII VALUES
00AB:          110 PLUS    EQU  $AB       ;PLUS SIGN
00AD:          111 MINUS   EQU  $AD       ;MINUS SIGN
00AE:          112 DP      EQU  $AE       ;DECIMAL POINT
----- NEXT OBJECT FILE NAME IS STEPPER.OBJ0
0800:          113         ORG  $0800     ;2048 DECIMAL
0800:          114 *********************************
0800:          115 *                               *
0800:          116 *  ENTRY POINT OF PROGRAM       *
0800:          117 *  (ENTER FROM BASIC USING A    *
0800:          118 *   'CALL 2048').               *
0800:          119 *                               *
0800:          120 *********************************
0800:          121 *
0800:          122 *
0800:4C 32 0A  123         JMP  START
0803:          124 *
0803:          125 *
0803:          126 *********************************
0803:          127 *                               *
0803:          128 *         BASIC PROGRAM         *
0803:          129 *       LINKAGE REGISTERS       *
0803:          130 *                               *
0803:          131 * REG.  DESCRIPTION             *
0803:          132 *                               *
0803:          133 * XYZ   BIT#                    *
0803:          134 *        0  '1' FOR X           *
0803:          135 *        1  '1' FOR Y           *
0803:          136 *        2  '1' FOR Z           *
0803:          137 *        3  '1' FOR MANUAL POS- *
0803:          138 *              ITIONING USING   *
0803:          139 *              ARROW KEYS.      *
0803:          140 *        4  '1' TO CLEAR COUNT  *
0803:          141 *           REGISTERS           *
0803:          142 *        5  '1' TO SUPRESS      *
0803:          143 *           REAL TIME DISPLAY   *
0803:          144 *        6  '1' TO DISABLE      *
0803:          145 *           MOTORS WHEN DONE.   *
0803:          146 *        7  '1' TO ENABLE DATA  *
0803:          147 *           STORAGE             *
0803:          148 *                               *
0803:          149 *        EXAMPLE CODES          *
0803:          150 *        (D=DON'T CARE, *=      *
0803:          151 *         ACTIVE BIT POS.)      *
0803:          152 *                               *
0803:          153 *        D**00000 = HOME        *
0803:          154 *        11111111 = RESET DATPTR*
0803:          155 *        00010000 = CLEAR COUNT *
0803:          156 *                   REGS.       *
0803:          157 *        ***01*** = MANUAL      *
0803:          158 *                   CONTROL     *
```

```
0803:          159 *        ***00*** = SCAN        *
0803:          160 *                               *
0803:          161 * MOVE  3 BYTE ABSOLUTE MOVE    *
0803:          162 *       VALUE.                  *
0803:          163 *                               *
0803:          164 * DATINC 2 BYTE SAMPLING        *
0803:          165 *        INCREMENT REGISTER.    *
0803:          166 *                               *
0803:          167 * ERFLG ERROR IF NONZERO        *
0803:          168 *                               *
0803:          169 * GCX   GEOM. COORD. REGISTERS  *
0803:          170 * GCY   4 BYTE PACKED BCD #'S   *
0803:          171 * GCZ   (ABC.DEFGH) STORED IN   *
0803:          172 *       EXCESS-500 FORMAT: AB   *
0803:          173 *                          CD   *
0803:          174 *                          EF   *
0803:          175 *                          GH   *
0803:          176 *                               *
0803:          177 * XCOUNT CONTAIN 24 BIT ABS.    *
0803:          178 * YCOUNT PULSE COUNT REL. TO    *
0803:          179 * ZCOUNT HOME REF.              *
0803:          180 *                               *
0803:          181 * MAXCNT CONTAINS 24 BIT ABS.   *
0803:          182 *        POS. WHERE A PEAK HAS  *
0803:          183 *        BEEN FOUND.            *
0803:          184 *                               *
0803:          185 * MAXVAL 12 BIT PEAK DATA VAL.  *
0803:          186 *                               *
0803:          187 *********************************
0803:          188 * GLOBAL REGISTERS
0803:          189 XYZ     DS   1
0804:          190 MOVE    DS   3
0807:          191 DATINC  DS   2
0809:          192 ERFLAG  DS   1
080A:          193 * GEOMETRIC COORDINATE REGISTERS
080A:          194 *(HIGH TO LOW BYTE ORDER)
080A:          195 GCX     DS   4
080E:          196 GCY     DS   4
0812:          197 GCZ     DS   4
0816:          198 * 24 BIT COUNT REGISTERS
0816:          199 XCOUNT  DS   3
0819:          200 YCOUNT  DS   3
081C:          201 ZCOUNT  DS   3
081F:          202 MAXCNT  DS   3
0822:          203 MAXVAL  DS   2
0824:          204 * LOCAL REGISTERS
0824:          205 DIFF    DS   3
0827:          206 RTREG   DS   3        ;REAL TIME DISP. REG.
082A:          207 CKMASK  DS   1
082B:          208 LIMASK  DS   1
082C:          209 NEGFLG  DS   1
082D:          210 LSTFLG  DS   1
082E:          211 INCREM  DS   2        ;LOCAL COPY OF DATINC
0830:          212 * HIRES POINTER
```

```
0830:           213 HIRESY  DS   1
0831:           214 HGRIND  DS   1
0832:           215 ********************************
0832:           216 *                              *
0832:           217 * DATA TABLES                  *
0832:           218 *    XTBL-X PIXEL COORD. CONV. *
0832:           219 *         TABLE                *
0832:           220 *   XDTBL-X PIXEL VALUE CONV.  *
0832:           221 *         TABLE                *
0832:           222 *                              *
0832:           223 ********************************
0832:0A 0A 0A   224 XTBL    DFB  10,10,10,10,10,10,10
0835:0A 0A 0A
0838:0A
0839:0B 0B 0B   225         DFB  11,11,11,11,11,11,11
083C:0B 0B 0B
083F:0B
0840:0C 0C 0C   226         DFB  12,12,12,12,12,12,12
0843:0C 0C 0C
0846:0C
0847:0D 0D 0D   227         DFB  13,13,13,13,13,13,13
084A:0D 0D 0D
084D:0D
084E:0E 0E 0E   228         DFB  14,14,14,14,14,14,14
0851:0E 0E 0E
0854:0E
0855:0F 0F 0F   229         DFB  15,15,15,15,15,15,15
0858:0F 0F 0F
085B:0F
085C:10 10 10   230         DFB  16,16,16,16,16,16,16
085F:10 10 10
0862:10
0863:11 11 11   231         DFB  17,17,17,17,17,17,17
0866:11 11 11
0869:11
086A:12 12 12   232         DFB  18,18,18,18,18,18,18
086D:12 12 12
0870:12
0871:13 13 13   233         DFB  19,19,19,19,19,19,19
0874:13 13 13
0877:13
0878:14 14 14   234         DFB  20,20,20,20,20,20,20
087B:14 14 14
087E:14
087F:15 15 15   235         DFB  21,21,21,21,21,21,21
0882:15 15 15
0885:15
0886:16 16 16   236         DFB  22,22,22,22,22,22,22
0889:16 16 16
088C:16
088D:17 17 17   237         DFB  23,23,23,23,23,23,23
0890:17 17 17
0893:17
0894:18 18 18   238         DFB  24,24,24,24,24,24,24
```

```
0897:18 18 18
089A:18
089B:19 19 19  239          DFB  25,25,25,25,25,25,25
089E:19 19 19
08A1:19
08A2:1A 1A 1A  240          DFB  26,26,26,26,26,26,26
08A5:1A 1A 1A
08A8:1A
08A9:1B 1B 1B  241          DFB  27,27,27,27,27,27,27
08AC:1B 1B 1B
08AF:1B
08B0:1C 1C     242          DFB  28,28
08B2:01 02 04  243 XDTBL    DFB  1,2,4,8,$10,$20,$40
08B5:08 10 20
08B8:40
08B9:01 02 04  244          DFB  1,2,4,8,$10,$20,$40
08BC:08 10 20
08BF:40
08C0:01 02 04  245          DFB  1,2,4,8,$10,$20,$40
08C3:08 10 20
08C6:40
08C7:01 02 04  246          DFB  1,2,4,8,$10,$20,$40
08CA:08 10 20
08CD:40
08CE:01 02 04  247          DFB  1,2,4,8,$10,$20,$40
08D1:08 10 20
08D4:40
08D5:01 02 04  248          DFB  1,2,4,8,$10,$20,$40
08D8:08 10 20
08DB:40
08DC:01 02 04  249          DFB  1,2,4,8,$10,$20,$40
08DF:08 10 20
08E2:40
08E3:01 02 04  250          DFB  1,2,4,8,$10,$20,$40
08E6:08 10 20
08E9:40
08EA:01 02 04  251          DFB  1,2,4,8,$10,$20,$40
08ED:08 10 20
08F0:40
08F1:01 02 04  252          DFB  1,2,4,8,$10,$20,$40
08F4:08 10 20
08F7:40
08F8:01 02 04  253          DFB  1,2,4,8,$10,$20,$40
08FB:08 10 20
08FE:40
08FF:01 02 04  254          DFB  1,2,4,8,$10,$20,$40
0902:08 10 20
0905:40
0906:01 02 04  255          DFB  1,2,4,8,$10,$20,$40
0909:08 10 20
090C:40
090D:01 02 04  256          DFB  1,2,4,8,$10,$20,$40
0910:08 10 20
0913:40
```

```
0914:01 02 04  257         DFB  1,2,4,8,$10,$20,$40
0917:08 10 20
091A:40
091B:01 02 04  258         DFB  1,2,4,8,$10,$20,$40
091E:08 10 20
0921:40
0922:01 02 04  259         DFB  1,2,4,8,$10,$20,$40
0925:08 10 20
0928:40
0929:01 02 04  260         DFB  1,2,4,8,$10,$20,$40
092C:08 10 20
092F:40
0930:01 02 04  261         DFB  1,2,4,8,$10,$20,$40
0933:08 10 20
0936:40
0937:01 02 04  262         DFB  1,2,4,8,$10,$20,$40
093A:08 10 20
093D:40
093E:01 02 04  263         DFB  1,2,4,8,$10,$20,$40
0941:08 10 20
0944:40
0945:01 02 04  264         DFB  1,2,4,8,$10,$20,$40
0948:08 10 20
094B:40
094C:01 02 04  265         DFB  1,2,4,8,$10,$20,$40
094F:08 10 20
0952:40
0953:01 02 04  266         DFB  1,2,4,8,$10,$20,$40
0956:08 10 20
0959:40
095A:01 02 04  267         DFB  1,2,4,8,$10,$20,$40
095D:08 10 20
0960:40
0961:01 02 04  268         DFB  1,2,4,8,$10,$20,$40
0964:08 10 20
0967:40
0968:01 02 04  269         DFB  1,2,4,8,$10,$20,$40
096B:08 10 20
096E:40
096F:01 02 04  270         DFB  1,2,4
0972:          271 WINTBL  DS   192
0A32:          272 *********************************
0A32:          273 *                               *
0A32:          274 *       START OF PROGRAM        *
0A32:          275 *                               *
0A32:          276 *********************************
0A32:A9 12     277 START   LDA  #$12      ;PUT RAMP TABLE ADDR. IN TBLPTR
0A34:85 09     278         STA  TBLPTR+1
0A36:A9 00     279         LDA  #0
0A38:85 08     280         STA  TBLPTR    ;RAMP TABLE POINTER @ $1200
0A3A:8D 22 08  281         STA  MAXVAL
0A3D:8D 23 08  282         STA  MAXVAL+1  ;CLEAR MAXVAL
0A40:8D 1F 08  283         STA  MAXCNT
0A43:8D 20 08  284         STA  MAXCNT+1
```

```
0A46:8D 21 08  285         STA  MAXCNT+2  ;CLEAR MAXCNT
0A49:8D 09 08  286         STA  ERFLAG    ;NO ERROR
0A4C:A9 7F     287         LDA  #$7F      ;INIT CODE
0A4E:8D 2D 08  288         STA  LSTFLG
0A51:A9 15     289         LDA  #21
0A53:8D 30 08  290         STA  HIRESY
0A56:AD 07 08  291         LDA  DATINC
0A59:8D 2E 08  292         STA  INCREM
0A5C:AD 08 08  293         LDA  DATINC+1
0A5F:8D 2F 08  294         STA  INCREM+1  ;INITIALIZE INCREM.
0A62:8D 00 C0  295         STA  KEYBD     ;MAKE SURE 80STORE IS OFF
0A65:A9 32     296         LDA  #>XTBL
0A67:85 F9     297         STA  XPTR
0A69:A9 08     298         LDA  #<XTBL
0A6B:85 FA     299         STA  XPTR+1
0A6D:A9 B2     300         LDA  #>XDTBL
0A6F:85 FB     301         STA  XDPTR
0A71:A9 08     302         LDA  #<XDTBL    ;MORE TABLE POINTER INITIALIZATIONS
0A73:85 FC     303         STA  XDPTR+1
0A75:A9 72     304         LDA  #>WINTBL
0A77:85 FD     305         STA  WINPTR
0A79:A9 09     306         LDA  #<WINTBL
0A7B:85 FE     307         STA  WINPTR+1
0A7D:          308 *********************************
0A7D:          309 *                               *
0A7D:          310 *        XYZ REGISTER           *
0A7D:          311 *     COMMAND INTERPRETER       *
0A7D:          312 *        (BITS 0-4)             *
0A7D:          313 *                               *
0A7D:          314 *********************************
0A7D:AD 03 08  315         LDA  XYZ
0A80:29 1F     316         AND  #$1F      ;LOOK ONLY AT 5 LSB'S
0A82:C9 00     317         CMP  #0        ;CODE FOR HOME
0A84:D0 0B     318         BNE  CMPFF
0A86:AD 03 08  319         LDA  XYZ
0A89:29 60     320         AND  #$60
0A8B:8D 03 08  321         STA  XYZ       ;DISABLE DATA COLL.
0A8E:4C 8D 0B  322         JMP  HOME
0A91:AD 03 08  323 CMPFF   LDA  XYZ
0A94:C9 FF     324         CMP  #$FF      ;CODE FOR RESET DATA POINTER
0A96:D0 0B     325         BNE  CMP16
0A98:A9 02     326         LDA  #$02
0A9A:85 EC     327         STA  DATPTR+1
0A9C:A9 00     328         LDA  #0
0A9E:85 EB     329         STA  DATPTR
0AA0:4C CE 0D  330         JMP  ENDEND
0AA3:C9 10     331 CMP16   CMP  #16       ;CODE FOR CLEARING REG'S
0AA5:F0 02     332         BEQ  CNT16
0AA7:D0 06     333         BNE  CMP8
0AA9:20 7B 0B  334 CNT16   JSR  CLRREG
0AAC:4C CE 0D  335         JMP  ENDEND
0AAF:29 08     336 CMP8    AND  #8        ;CODE FOR MANUAL CONTROL
0AB1:F0 03     337         BEQ  XXYYZZ
0AB3:4C EF 0A  338         JMP  GETKEY
```

```
0AB6:20 C1 0A  339 XXYYZZ  JSR  XYORZ
0AB9:D0 03     340         BNE  CMPCNT
0ABB:4C BD 0D  341         JMP  ERROR
0ABE:4C 4C 0C  342 CMPCNT  JMP  MAIN
0AC1:          343 *********************************
0AC1:          344 *                               *
0AC1:          345 * XYORZ SETS UP FOR EITHER X,Y, *
0AC1:          346 * Z, AND THEN RETURNS WITH ZERO *
0AC1:          347 * BIT NOT SET. OTHERWISE A SET  *
0AC1:          348 * ZERO BIT INDICATES INVALID    *
0AC1:          349 * CODE.                         *
0AC1:          350 *                               *
0AC1:          351 *********************************
0AC1:AD 03 08  352 XYORZ   LDA  XYZ
0AC4:29 01     353         AND  #1        ;X BIT
0AC6:F0 06     354         BEQ  BITY
0AC8:20 C2 0B  355         JSR  XSETUP
0ACB:4C EC 0A  356         JMP  CMPEND
0ACE:AD 03 08  357 BITY    LDA  XYZ
0AD1:29 02     358         AND  #2        ;Y BIT
0AD3:F0 06     359         BEQ  BITZ
0AD5:20 E8 0B  360         JSR  YSETUP
0AD8:4C EC 0A  361         JMP  CMPEND
0ADB:AD 03 08  362 BITZ    LDA  XYZ
0ADE:29 04     363         AND  #4        ;Z BIT
0AE0:F0 06     364         BEQ  CMPERR
0AE2:20 0E 0C  365         JSR  ZSETUP
0AE5:4C EC 0A  366         JMP  CMPEND
0AE8:A9 00     367 CMPERR  LDA  #0        ;SET ZERO BIT IN STATUS REG.
0AEA:F0 02     368         BEQ  CMPFIN
0AEC:A9 FF     369 CMPEND  LDA  #$FF      ;CLEAR ZERO BIT IN STATUS REG.
0AEE:60        370 CMPFIN  RTS
0AEF:          371 *********************************
0AEF:          372 *                               *
0AEF:          373 *    MANUAL CONTROL ROUTINE     *
0AEF:          374 *                               *
0AEF:          375 *********************************
0AEF:AD 00 C0  376 GETKEY  LDA  KEYBD
0AF2:10 FB     377         BPL  GETKEY    ;WAIT FOR KEYBOARD
0AF4:29 7F     378         AND  #$7F      ;STRIP OFF MSBIT
0AF6:2C 10 C0  379         BIT  KEYDWN
0AF9:C9 0D     380         CMP  #RET      ;RETURN KEY?
0AFB:D0 03     381         BNE  CONT1
0AFD:4C C2 0D  382         JMP  END
0B00:C9 08     383 CONT1   CMP  #LARROW   ;LEFT ARROW KEY?
0B02:D0 20     384         BNE  CONT2
0B04:A9 00     385         LDA  #0        ;DIRECTION IS CCW
0B06:8D 10 C4  386         STA  CKDIR
0B09:A9 FF     387         LDA  #$FF
0B0B:8D 2C 08  388         STA  NEGFLG
0B0E:20 C1 0A  389         JSR  XYORZ
0B11:D0 03     390         BNE  CONT1A
0B13:4C BD 0D  391         JMP  ERROR
0B16:20 6D 0B  392 CONT1A  JSR  FLINIT
```

```
0B19:AD 2B 08  393         LDA  LIMASK
0B1C:29 07     394         AND  #7
0B1E:8D 2B 08  395         STA  LIMASK    ;LOOK ONLY AT 'IN' LIMIT
0B21:4C 45 0B  396         JMP  CONT3
0B24:C9 15     397 CONT2   CMP  #RARROW   ;RIGHT ARROW KEY?
0B26:D0 42     398         BNE  CONT5
0B28:A9 2A     399         LDA  #$2A      ;DIRECTION IS CW
0B2A:8D 10 C4  400         STA  CKDIR
0B2D:A9 00     401         LDA  #0
0B2F:8D 2C 08  402         STA  NEGFLG
0B32:20 C1 0A  403         JSR  XYORZ
0B35:D0 03     404         BNE  CONT2A
0B37:4C BD 0D  405         JMP  ERROR
0B3A:20 6D 0B  406 CONT2A  JSR  FLINIT
0B3D:AD 2B 08  407         LDA  LIMASK
0B40:29 38     408         AND  #$38
0B42:8D 2B 08  409         STA  LIMASK    ;LOOK ONLY AT 'OUT' LIMIT
0B45:AD 11 C4  410 CONT3   LDA  LIMIT
0B48:2D 2B 08  411         AND  LIMASK
0B4B:F0 03     412         BEQ  CONT4
0B4D:4C BD 0D  413         JMP  ERROR
0B50:A9 02     414 CONT4   LDA  #$02      ;DATA TABLE STARTS AT $0200 OF AUX. MEM.
0B52:85 EC     415         STA  DATPTR+1
0B54:A9 00     416         LDA  #0
0B56:85 EB     417         STA  DATPTR    ;RESET DATA POINTER
0B58:A9 C8     418         LDA  #200      ;200 PULSES=1/2 REV.
0B5A:8D 24 08  419         STA  DIFF
0B5D:A9 00     420         LDA  #0
0B5F:8D 25 08  421         STA  DIFF+1
0B62:8D 26 08  422         STA  DIFF+2
0B65:A2 60     423         LDX  #NOMVAL
0B67:20 EF 0C  424         JSR  SLUP1     ;JUMP INTO FLAT RATE ROUTINE
0B6A:4C EF 0A  425 CONT5   JMP  GETKEY
0B6D:          426 *********************************
0B6D:          427 *                               *
0B6D:          428 * FLAG INITIALIZATION FOR FIRST *
0B6D:          429 * ENTRY- USED MAINLY IN MANUAL  *
0B6D:          430 * CONTROL ROUTINE.              *
0B6D:          431 * FLAGS USED MAINLY IN HIPLOT.  *
0B6D:          432 *                               *
0B6D:          433 *********************************
0B6D:AD 2D 08  434 FLINIT  LDA  LSTFLG
0B70:C9 7F     435         CMP  #$7F      ;INIT CODE
0B72:D0 06     436         BNE  FLGRET
0B74:AD 2C 08  437         LDA  NEGFLG
0B77:8D 2D 08  438         STA  LSTFLG
0B7A:60        439 FLGRET  RTS
0B7B:          440 *********************************
0B7B:          441 *                               *
0B7B:          442 *  CLRREG CLEARS XCOUNT,YCOUNT  *
0B7B:          443 *  AND ZCOUNT                   *
0B7B:          444 *                               *
0B7B:          445 *********************************
0B7B:A9 16     446 CLRREG  LDA  #>XCOUNT  ;SET UP REGPTR
```

```
0B7D:85 06     447         STA  REGPTR
0B7F:A9 08     448         LDA  #<XCOUNT
0B81:85 07     449         STA  REGPTR+1
0B83:A9 00     450         LDA  #0
0B85:A0 09     451         LDY  #9
0B87:88        452 CLRLUP  DEY
0B88:91 06     453         STA  (REGPTR),Y
0B8A:D0 FB     454         BNE  CLRLUP
0B8C:60        455         RTS
0B8D:          456 *********************************
0B8D:          457 *                               *
0B8D:          458 *  HOME RETURNS ALL MOTORS TO   *
0B8D:          459 *  'IN' LIMITS AND RETURNS      *
0B8D:          460 *  DISTANCE TRAVELLED IN XCOUNT *
0B8D:          461 *  YCOUNT & ZCOUNT.             *
0B8D:          462 *  THESE COUNTERS MAY THEN BE   *
0B8D:          463 *  CLEARED BY RE-ENTERING WITH  *
0B8D:          464 *  AN XYZ CODE 16.              *
0B8D:          465 *                               *
0B8D:          466 *********************************
0B8D:A9 FF     467 HOME    LDA  #$FF
0B8F:8D 2C 0B  468         STA  NEGFLG     ;ASSUMES XYZ TO BE TURNED CCW
0B92:A9 00     469         LDA  #0
0B94:8D 10 C4  470         STA  CKDIR
0B97:A2 60     471         LDX  #NOMVAL
0B99:20 C2 0B  472         JSR  XSETUP
0B9C:20 AE 0B  473         JSR  HLUP1
0B9F:20 E8 0B  474         JSR  YSETUP
0BA2:20 AE 0B  475         JSR  HLUP1
0BA5:20 0E 0C  476         JSR  ZSETUP
0BA8:20 AE 0B  477         JSR  HLUP1
0BAB:4C C2 0D  478         JMP  END
0BAE:          479 *
0BAE:AD 11 C4  480 HLUP1   LDA  LIMIT
0BB1:2D 2B 08  481         AND  LIMASK
0BB4:29 07     482         AND  #7         ;LOOK ONLY AT 'IN' LIMITS
0BB6:D0 09     483         BNE  HLEND
0BB8:20 DF 0D  484         JSR  OUTPLS
0BBB:20 F5 0D  485         JSR  INCCNT     ;NOTE: HOME INCREMENTS COUNTERS
0BBE:4C AE 0B  486         JMP  HLUP1
0BC1:60        487 HLEND   RTS
0BC2:          488 *********************************
0BC2:          489 *                               *
0BC2:          490 * XSETUP,YSETUP, AND ZSETUP     *
0BC2:          491 * SET UP POINTERS AND MASKS     *
0BC2:          492 * FOR THE APPROPRIATE MOTOR     *
0BC2:          493 * CONTROL.                      *
0BC2:          494 *                               *
0BC2:          495 *********************************
0BC2:A9 16     496 XSETUP  LDA  #>XCOUNT  ;LOW BYTE OF X COUNT REG.
0BC4:85 06     497         STA  REGPTR
0BC6:A9 08     498         LDA  #<XCOUNT  ;HIGH BYTE OF X COUNT REG.
0BC8:85 07     499         STA  REGPTR+1
0BCA:A9 0A     500         LDA  #>GCX     ;LOW BYTE OF GCX REG.
```

```
0BCC:85 1A     501         STA  GCPTR
0BCE:A9 08     502         LDA  #<GCX     ;HIGH BYTE OF GCX REG.
0BD0:85 1B     503         STA  GCPTR+1
0BD2:A9 D4     504         LDA  #>XSCR    ;LOW BYTE OF X SCREEN POINTER
0BD4:85 CE     505         STA  SCRPTR
0BD6:A9 06     506         LDA  #<XSCR    ;HIGH BYTE OF X SCREEN POINTER
0BD8:85 CF     507         STA  SCRPTR+1
0BDA:A9 01     508         LDA  #1
0BDC:8D 2A 08  509         STA  CKMASK
0BDF:A9 09     510         LDA  #$9
0BE1:8D 2B 08  511         STA  LIMASK
0BE4:20 34 0C  512         JSR  ENABLE
0BE7:60        513         RTS
0BE8:          514 *
0BE8:A9 19     515 YSETUP  LDA  #>YCOUNT
0BEA:85 06     516         STA  REGPTR
0BEC:A9 08     517         LDA  #<YCOUNT
0BEE:85 07     518         STA  REGPTR+1
0BF0:A9 0E     519         LDA  #>GCY
0BF2:85 1A     520         STA  GCPTR
0BF4:A9 08     521         LDA  #<GCY
0BF6:85 1B     522         STA  GCPTR+1
0BF8:A9 E1     523         LDA  #>YSCR
0BFA:85 CE     524         STA  SCRPTR
0BFC:A9 06     525         LDA  #<YSCR
0BFE:85 CF     526         STA  SCRPTR+1
0C00:A9 04     527         LDA  #4
0C02:8D 2A 08  528         STA  CKMASK
0C05:A9 12     529         LDA  #$12
0C07:8D 2B 08  530         STA  LIMASK
0C0A:20 34 0C  531         JSR  ENABLE
0C0D:60        532         RTS
0C0E:          533 *
0C0E:A9 1C     534 ZSETUP  LDA  #>ZCOUNT
0C10:85 06     535         STA  REGPTR
0C12:A9 08     536         LDA  #<ZCOUNT
0C14:85 07     537         STA  REGPTR+1
0C16:A9 12     538         LDA  #>GCZ
0C18:85 1A     539         STA  GCPTR
0C1A:A9 08     540         LDA  #<GCZ
0C1C:85 1B     541         STA  GCPTR+1
0C1E:A9 EE     542         LDA  #>ZSCR
0C20:85 CE     543         STA  SCRPTR
0C22:A9 06     544         LDA  #<ZSCR
0C24:85 CF     545         STA  SCRPTR+1
0C26:A9 10     546         LDA  #$10
0C28:8D 2A 08  547         STA  CKMASK
0C2B:A9 24     548         LDA  #$24
0C2D:8D 2B 08  549         STA  LIMASK
0C30:20 34 0C  550         JSR  ENABLE
0C33:60        551         RTS
0C34:          552 *********************************
0C34:          553 *                               *
0C34:          554 * ENABLE APPROPRIATE MOTORS    *
```

```
0C34:          555 * BASED ON STATUS OF XYZ BIT 6  *
0C34:          556 *                              *
0C34:          557 ********************************
0C34:AD 03 08  558 ENABLE  LDA  XYZ
0C37:29 40     559         AND  #64       ;BIT 6
0C39:F0 0B     560         BEQ  ENCNT1
0C3B:AD 03 08  561         LDA  XYZ
0C3E:29 07     562         AND  #7        ;LOOK ONLY AT 3 LSB'S
0C40:0A        563         ASL  A
0C41:0A        564         ASL  A
0C42:0A        565         ASL  A
0C43:0A        566         ASL  A
0C44:D0 02     567         BNE  ENEND     ;ENABLE ONLY X,Y,OR Z
0C46:A9 70     568 ENCNT1  LDA  #$70      ;ENABLE X,Y,AND Z
0C48:8D 01 C4  569 ENEND   STA  LDATA
0C4B:60        570         RTS
0C4C:          571 ********************************
0C4C:          572 *                              *
0C4C:          573 *  MAIN STEPPER MOTOR CONTROL  *
0C4C:          574 *  PORTION OF PROGRAM          *
0C4C:          575 *                              *
0C4C:          576 ********************************
0C4C:A0 00     577 MAIN    LDY  #0        ;SUBTRACT REG POINTED TO
0C4E:38        578         SEC             ;BY REGPTR FROM MOVE
0C4F:AD 04 08  579         LDA  MOVE      ;& STORE RESULT IN DIFF
0C52:F1 06     580         SBC  (REGPTR),Y
0C54:8D 24 08  581         STA  DIFF
0C57:C8        582         INY
0C58:AD 05 08  583         LDA  MOVE+1
0C5B:F1 06     584         SBC  (REGPTR),Y
0C5D:8D 25 08  585         STA  DIFF+1
0C60:C8        586         INY
0C61:AD 06 08  587         LDA  MOVE+2
0C64:F1 06     588         SBC  (REGPTR),Y
0C66:8D 26 08  589         STA  DIFF+2
0C69:B0 40     590         BCS  POS       ;DIFF IS POS (CW)
0C6B:A9 FF     591         LDA  #$FF      ;DIFF IS NEG (CCW)
0C6D:8D 2C 08  592         STA  NEGFLG
0C70:AD 2B 08  593         LDA  LIMASK
0C73:29 07     594         AND  #7
0C75:8D 2B 08  595         STA  LIMASK    ;LOOK ONLY AT 'IN' LIMITS
0C78:AD 26 08  596         LDA  DIFF+2    ;CHANGE DIFF TO POS #
0C7B:49 FF     597         EOR  #$FF      ;BY TAKING 2'S COMPLEMENT
0C7D:8D 26 08  598         STA  DIFF+2    ;AND ADDING 1. NOTE: MOVE
0C80:AD 25 08  599         LDA  DIFF+1    ;MUST HAVE ONLY 23 SIGNIFICANT DIGITS
0C83:49 FF     600         EOR  #$FF      ;FOR CORRECT OPERATION
0C85:8D 25 08  601         STA  DIFF+1
0C88:AD 24 08  602         LDA  DIFF
0C8B:49 FF     603         EOR  #$FF
0C8D:18        604         CLC
0C8E:69 01     605         ADC  #1
0C90:8D 24 08  606         STA  DIFF
0C93:A9 00     607         LDA  #0
0C95:6D 25 08  608         ADC  DIFF+1
```

```
0C98:8D 25 08  609         STA  DIFF+1
0C9B:A9 00     610         LDA  #0
0C9D:6D 26 08  611         ADC  DIFF+2
0CA0:8D 26 08  612         STA  DIFF+2
0CA3:A9 00     613         LDA  #0        ;DIRECTION IS NEG (CCW)
0CA5:8D 10 C4  614         STA  CKDIR
0CA8:4C BD 0C  615         JMP  MCONT
0CAB:A9 00     616 POS     LDA  #0
0CAD:8D 2C 08  617         STA  NEGFLG    ;FALSE FOR POS.
0CB0:AD 2B 08  618         LDA  LIMASK
0CB3:29 38     619         AND  #$38
0CB5:8D 2B 08  620         STA  LIMASK    ;LOOK ONLY AT 'OUT' LIMITS
0CB8:A9 2A     621         LDA  #$2A      ;DIRECTION IS POS (CW)
0CBA:8D 10 C4  622         STA  CKDIR
0CBD:A9 00     623 MCONT   LDA  #0        ;TRAP OUT ZERO DIFF
0CBF:0D 24 08  624         ORA  DIFF
0CC2:0D 25 08  625         ORA  DIFF+1
0CC5:0D 26 08  626         ORA  DIFF+2
0CC8:D0 03     627         BNE  MCONT1
0CCA:4C C2 0D  628         JMP  END
0CCD:38        629 MCONT1  SEC             ;SUBTRACT 1024 FROM DIFF
0CCE:AD 25 08  630         LDA  DIFF+1    ;TO SEE IF RAMPING
0CD1:48        631         PHA            ;IS REQUIRED
0CD2:E9 04     632         SBC  #4
0CD4:8D 25 08  633         STA  DIFF+1
0CD7:AD 26 08  634         LDA  DIFF+2
0CDA:48        635         PHA
0CDB:E9 00     636         SBC  #0
0CDD:8D 26 08  637         STA  DIFF+2
0CE0:B0 34     638         BCS  RAMP      ;BRANCH IF DIFF >= 1024
0CE2:68        639         PLA             ;OTHERWISE RESTORE DIFF
0CE3:8D 26 08  640         STA  DIFF+2
0CE6:68        641         PLA
0CE7:8D 25 08  642         STA  DIFF+1    ;AND STEP RIGHT INTO NOMINAL VALUE
0CEA:20 6D 0B  643         JSR  FLINIT    ;SET UP FLAGS USED IN HIPLOT
0CED:A2 60     644         LDX  #NOMVAL
0CEF:AD 11 C4  645 SLUP1   LDA  LIMIT
0CF2:2D 2B 08  646         AND  LIMASK
0CF5:F0 03     647         BEQ  SCONT0
0CF7:4C BD 0D  648         JMP  ERROR
0CFA:20 DF 0D  649 SCONT0  JSR  OUTPLS
0CFD:AD 2C 08  650         LDA  NEGFLG
0D00:F0 06     651         BEQ  SCONT1    ;BRANCH IF POS
0D02:20 48 0E  652         JSR  DECCNT
0D05:4C 0B 0D  653         JMP  SCONT2
0D08:20 F5 0D  654 SCONT1  JSR  INCCNT
0D0B:20 8E 0E  655 SCONT2  JSR  DCRDIF
0D0E:D0 DF     656         BNE  SLUP1
0D10:4C C2 0D  657         JMP  END
0D13:4C BD 0D  658 SERROR  JMP  ERROR
0D16:          659 *********************************
0D16:          660 *                               *
0D16:          661 *    RAMPING ROUTINE            *
0D16:          662 *                               *
```

```
0D16:          663 * RLUP1-RAMP UP                *
0D16:          664 * RLUP2-FLAT PULSE RATE        *
0D16:          665 * RLUP3-RAMP DOWN              *
0D16:          666 *                              *
0D16:          667 ********************************
0D16:68        668 RAMP    PLA            ;DISCARD SAVED DIFF.
0D17:68        669         PLA
0D18:20 6D 0B  670         JSR  FLINIT    ;SET UP FLAGS USED IN HIPLOT
0D1B:A0 00     671         LDY  #0
0D1D:AD 11 C4  672 RLUP1   LDA  LIMIT     ;RAMP UP
0D20:2D 2B 08  673         AND  LIMASK
0D23:D0 EE     674         BNE  SERROR
0D25:B1 08     675         LDA  (TBLPTR),Y
0D27:AA        676         TAX
0D28:20 DF 0D  677         JSR  OUTPLS
0D2B:AD 2C 08  678         LDA  NEGFLG
0D2E:F0 06     679         BEQ  RCONT1    ;BRANCH IF POS
0D30:20 48 0E  680         JSR  DECCNT
0D33:4C 39 0D  681         JMP  RCONT2
0D36:20 F5 0D  682 RCONT1  JSR  INCCNT
0D39:AD 11 C4  683 RCONT2  LDA  LIMIT
0D3C:2D 2B 08  684         AND  LIMASK
0D3F:D0 D2     685         BNE  SERROR
0D41:20 DF 0D  686         JSR  OUTPLS
0D44:AD 2C 08  687         LDA  NEGFLG
0D47:F0 06     688         BEQ  RCONTA
0D49:20 48 0E  689         JSR  DECCNT
0D4C:4C 52 0D  690         JMP  RCONTB
0D4F:20 F5 0D  691 RCONTA  JSR  INCCNT
0D52:C8        692 RCONTB  INY
0D53:D0 C8     693         BNE  RLUP1
0D55:AD 24 08  694         LDA  DIFF      ;NEXT RAMP AT CONSTANT RATE
0D58:0D 25 08  695         ORA  DIFF+1
0D5B:0D 26 08  696         ORA  DIFF+2
0D5E:F0 1E     697         BEQ  RDWN      ;TRAP OUT ZERO DIFF.
0D60:AD 11 C4  698 RLUP2   LDA  LIMIT
0D63:2D 2B 08  699         AND  LIMASK
0D66:D0 55     700         BNE  ERROR
0D68:20 DF 0D  701         JSR  OUTPLS
0D6B:AD 2C 08  702         LDA  NEGFLG
0D6E:F0 06     703         BEQ  RCONT3    ;BRANCH IF POS
0D70:20 48 0E  704         JSR  DECCNT
0D73:4C 79 0D  705         JMP  RCONT4
0D76:20 F5 0D  706 RCONT3  JSR  INCCNT
0D79:20 8E 0E  707 RCONT4  JSR  DCRDIF
0D7C:D0 E2     708         BNE  RLUP2
0D7E:A0 00     709 RDWN    LDY  #0        ;RAMP DOWN
0D80:AD 11 C4  710 RLUP3   LDA  LIMIT
0D83:2D 2B 08  711         AND  LIMASK
0D86:D0 35     712         BNE  ERROR
0D88:88        713         DEY
0D89:B1 08     714         LDA  (TBLPTR),Y
0D8B:AA        715         TAX
0D8C:20 DF 0D  716         JSR  OUTPLS
```

```
0D8F:AD 2C 08  717         LDA  NEGFLG
0D92:F0 06     718         BEQ  RCONT5    ;BRANCH IF POS
0D94:20 48 0E  719         JSR  DECCNT
0D97:4C 9D 0D  720         JMP  RCONT6
0D9A:20 F5 0D  721 RCONT5  JSR  INCCNT
0D9D:AD 11 C4  722 RCONT6  LDA  LIMIT
0DA0:2D 2B 08  723         AND  LIMASK
0DA3:D0 18     724         BNE  ERROR
0DA5:20 DF 0D  725         JSR  OUTPLS
0DA8:AD 2C 08  726         LDA  NEGFLG
0DAB:F0 06     727         BEQ  RCONTC
0DAD:20 48 0E  728         JSR  DECCNT
0DB0:4C B6 0D  729         JMP  RCONTD
0DB3:20 F5 0D  730 RCONTC  JSR  INCCNT
0DB6:98        731 RCONTD  TYA
0DB7:09 00     732         ORA  #0        ;SET FLAGS
0DB9:D0 C5     733         BNE  RLUP3
0DBB:F0 05     734         BEQ  END
0DBD:A9 FF     735 ERROR   LDA  #$FF
0DBF:8D 09 08  736         STA  ERFLAG
0DC2:AD 03 08  737 END     LDA  XYZ
0DC5:29 40     738         AND  #64       ;BIT 6
0DC7:F0 05     739         BEQ  ENDEND
0DC9:A9 00     740         LDA  #0
0DCB:8D 01 C4  741         STA  LDATA     ;SHUT OFF MOTORS
0DCE:60        742 ENDEND  RTS
0DCF:          743 *********************************
0DCF:          744 *                               *
0DCF:          745 *     TOTAL WASTE OF TIME       *
0DCF:          746 *                               *
0DCF:          747 *********************************
0DCF:8A        748 DELAY   TXA             ;SAVE X
0DD0:48        749         PHA
0DD1:38        750         SEC
0DD2:E9 15     751         SBC  #TWEAK    ;USE TO ADJUST DELAY RANGE
0DD4:90 06     752         BCC  DELRET    ;BRANCH IF NEG. RESULT
0DD6:F0 04     753         BEQ  DELRET    ;OR IF EQUAL TO ZERO
0DD8:AA        754         TAX
0DD9:CA        755 DELLUP  DEX
0DDA:D0 FD     756         BNE  DELLUP
0DDC:68        757 DELRET  PLA
0DDD:AA        758         TAX             ;RESTORE X
0DDE:60        759         RTS
0DDF:          760 *********************************
0DDF:          761 *                               *
0DDF:          762 *  OUTPUT STEPPER MOTOR PULSE   *
0DDF:          763 *           SUBROUTINE          *
0DDF:          764 *                               *
0DDF:          765 *********************************
0DDF:AD 10 C4  766 OUTPLS  LDA  CKDIR
0DE2:4D 2A 08  767         EOR  CKMASK
0DE5:8D 10 C4  768         STA  CKDIR     ;GO HIGH
0DE8:20 CF 0D  769         JSR  DELAY     ;STAY HIGH
0DEB:AD 10 C4  770         LDA  CKDIR
```

```
0DEE:4D 2A 08  771         EOR  CKMASK
0DF1:8D 10 C4  772         STA  CKDIR     ;GO LOW
0DF4:60        773         RTS
0DF5:          774 *********************************
0DF5:          775 *                               *
0DF5:          776 * INCCNT AND DECCNT INCREMENT,  *
0DF5:          777 * OR DECREMENT RESPECTIVELY THE *
0DF5:          778 * COUNT REGISTER POINTED TO BY  *
0DF5:          779 * REGPTR BY ONE, AND THE GEOM.  *
0DF5:          780 * COORD. REGISTER POINTED TO BY *
0DF5:          781 * GCPTR BY GCINC                *
0DF5:          782 *                               *
0DF5:          783 *********************************
0DF5:98        784 INCCNT  TYA
0DF6:48        785         PHA             ;SAVE Y
0DF7:18        786         CLC
0DF8:A0 00     787         LDY  #0
0DFA:B1 06     788         LDA  (REGPTR),Y
0DFC:69 01     789         ADC  #1
0DFE:91 06     790         STA  (REGPTR),Y
0E00:C8        791         INY
0E01:B1 06     792         LDA  (REGPTR),Y
0E03:69 00     793         ADC  #0
0E05:91 06     794         STA  (REGPTR),Y
0E07:C8        795         INY
0E08:B1 06     796         LDA  (REGPTR),Y
0E0A:69 00     797         ADC  #0
0E0C:91 06     798         STA  (REGPTR),Y
0E0E:AD 03 08  799         LDA  XYZ
0E11:29 1F     800         AND  #$1F      ;CHECK TO SEE IF 'HOME' CODE
0E13:D0 06     801         BNE  NOHOME
0E15:20 6D 0E  802         JSR  DECGC     ;DEC. (NOT INC.) GEOM. COORDS. ONLY WHEN HOMING
0E18:4C 1E 0E  803         JMP  INCCON
0E1B:20 27 0E  804 NOHOME  JSR  INCGC     ;INC. GEOM. COORDS. UNDER USUAL CONDS.
0E1E:20 AE 0E  805 INCCON  JSR  DATCOL    ;COLLECT DATA
0E21:20 C5 0F  806         JSR  RTDISP    ;PUT COORDS. ON SCREEN
0E24:68        807         PLA             ;RESTORE Y
0E25:A8        808         TAY
0E26:60        809         RTS
0E27:          810 *
0E27:F8        811 INCGC   SED
0E28:A0 03     812         LDY  #3
0E2A:18        813         CLC
0E2B:A9 27     814         LDA  #>GCINC
0E2D:71 1A     815         ADC  (GCPTR),Y ;(---.---GH)
0E2F:91 1A     816         STA  (GCPTR),Y
0E31:88        817         DEY
0E32:A9 01     818         LDA  #<GCINC
0E34:71 1A     819         ADC  (GCPTR),Y ;(---.-EF--)
0E36:91 1A     820         STA  (GCPTR),Y
0E38:88        821         DEY
0E39:A9 00     822         LDA  #0
0E3B:71 1A     823         ADC  (GCPTR),Y ;(--C.D----)
0E3D:91 1A     824         STA  (GCPTR),Y
```

```
0E3F:88       825         DEY
0E40:A9 00    826         LDA  #0
0E42:71 1A    827         ADC  (GCPTR),Y ;(AB-.-----)
0E44:91 1A    828         STA  (GCPTR),Y
0E46:D8       829         CLD
0E47:60       830         RTS
0E48:         831 *
0E48:98       832 DECCNT  TYA
0E49:48       833         PHA             ;SAVE Y
0E4A:38       834         SEC
0E4B:A0 00    835         LDY  #0
0E4D:B1 06    836         LDA  (REGPTR),Y
0E4F:E9 01    837         SBC  #1
0E51:91 06    838         STA  (REGPTR),Y
0E53:C8       839         INY
0E54:B1 06    840         LDA  (REGPTR),Y
0E56:E9 00    841         SBC  #0
0E58:91 06    842         STA  (REGPTR),Y
0E5A:C8       843         INY
0E5B:B1 06    844         LDA  (REGPTR),Y
0E5D:E9 00    845         SBC  #0
0E5F:91 06    846         STA  (REGPTR),Y
0E61:20 6D 0E 847         JSR  DECGC
0E64:20 AE 0E 848         JSR  DATCOL     ;COLLECT DATA
0E67:20 C5 0F 849         JSR  RTDISP     ;PUT COORDS. ON SCREEN
0E6A:68       850         PLA              ;RESTORE Y
0E6B:A8       851         TAY
0E6C:60       852         RTS
0E6D:         853 *
0E6D:F8       854 DECGC   SED
0E6E:A0 03    855         LDY  #3
0E70:38       856         SEC
0E71:B1 1A    857         LDA  (GCPTR),Y ;(---.---GH)
0E73:E9 27    858         SBC  #>GCINC
0E75:91 1A    859         STA  (GCPTR),Y
0E77:88       860         DEY
0E78:B1 1A    861         LDA  (GCPTR),Y ;(---.-EF--)
0E7A:E9 01    862         SBC  #<GCINC
0E7C:91 1A    863         STA  (GCPTR),Y
0E7E:88       864         DEY
0E7F:B1 1A    865         LDA  (GCPTR),Y ;(--C.D----)
0E81:E9 00    866         SBC  #0
0E83:91 1A    867         STA  (GCPTR),Y
0E85:88       868         DEY
0E86:B1 1A    869         LDA  (GCPTR),Y ;(AB-.-----)
0E88:E9 00    870         SBC  #0
0E8A:91 1A    871         STA  (GCPTR),Y
0E8C:D8       872         CLD
0E8D:60       873         RTS
0E8E:         874 *
0E8E:         875 *********************************
0E8E:         876 *                               *
0E8E:         877 * DCRDIF DECREMENTS DIFF. REG.  *
0E8E:         878 * AND RETURNS WITH A ZERO IN    *
```

```
0E8E:          879 * THE ACC. IF DIFF IS ZERO     *
0E8E:          880 *                              *
0E8E:          881 ********************************
0E8E:38        882 DCRDIF  SEC
0E8F:AD 24 08  883         LDA  DIFF
0E92:E9 01     884         SBC  #1
0E94:8D 24 08  885         STA  DIFF
0E97:AD 25 08  886         LDA  DIFF+1
0E9A:E9 00     887         SBC  #0
0E9C:8D 25 08  888         STA  DIFF+1
0E9F:AD 26 08  889         LDA  DIFF+2
0EA2:E9 00     890         SBC  #0
0EA4:8D 26 08  891         STA  DIFF+2
0EA7:0D 25 08  892         ORA  DIFF+1
0EAA:0D 24 08  893         ORA  DIFF
0EAD:60        894         RTS
0EAE:          895 ********************************
0EAE:          896 *                              *
0EAE:          897 * DATCOL COLLECTS DATA (AND    *
0EAE:          898 * WILL STORE IT AT THE ADDR.   *
0EAE:          899 * POINTED TO BY DATPRT, IF XYZ  *
0EAE:          900 * BIT 7 IS SET) EVERY DATINC+1  *
0EAE:          901 * # OF PULSES IN ONE DIRECTION  *
0EAE:          902 * AND UPDATES THE RTD          *
0EAE:          903 * IF XYZ BIT 5 IS A '0'.       *
0EAE:          904 * NOTE: IF DATINC,DATINC+1 ARE  *
0EAE:          905 *       EQUAL TO ZERO, THE ROU- *
0EAE:          906 *       TINE WILL BE EXITED.   *
0EAE:          907 *                              *
0EAE:          908 ********************************
0EAE:AD 07 08  909 DATCOL  LDA  DATINC
0EB1:0D 08 08  910         ORA  DATINC+1
0EB4:D0 03     911         BNE  DATST
0EB6:4C 5B 0F  912         JMP  DATEND    ;EXIT IF DATINC=0
0EB9:AD 2C 08  913 DATST   LDA  NEGFLG
0EBC:F0 06     914         BEQ  DATUP
0EBE:20 98 0F  915         JSR  DECREM    ;DECREMENT INCREM IF NEGFLG IS SET
0EC1:4C C7 0E  916         JMP  DATCON
0EC4:20 6A 0F  917 DATUP   JSR  UPCREM    ;INCREMENT INCREM IF NEGFLG IS CLEARED
0EC7:F0 03     918 DATCON  BEQ  DATCNT    ;SAMPLE IF INCREM HAS REACHED A COUNT OF ZERO
0EC9:4C 5B 0F  919         JMP  DATEND
0ECC:AD 01 C4  920 DATCNT  LDA  LDATA
0ECF:09 80     921         ORA  #$80      ;SET MSBIT
0ED1:8D 01 C4  922         STA  LDATA
0ED4:49 80     923         EOR  #$80      ;CLEAR MSBIT
0ED6:8D 01 C4  924         STA  LDATA     ;START CONVERSION
0ED9:AD 0D C4  925 COLLUP  LDA  IFR1
0EDC:29 12     926         AND  #$12
0EDE:F0 F9     927         BEQ  COLLUP    ;WAIT LOOP
0EE0:AD 03 08  928         LDA  XYZ
0EE3:29 80     929         AND  #$80      ;LOOK AT BIT 7
0EE5:F0 1A     930         BEQ  DATDIS
0EE7:A0 00     931         LDY  #0
0EE9:AD 01 C4  932         LDA  LDATA
```

```
0EEC:29 0F     933         AND  #$0F      ;STRIP OFF JUNK
0EEE:8D 05 C0  934         STA  RAMWRT+1  ;SWITCH TO AUX. MEM.
0EF1:91 EB     935         STA  (DATPTR),Y
0EF3:20 5C 0F  936         JSR  INCDAT
0EF6:AD 00 C4  937         LDA  HDATA
0EF9:91 EB     938         STA  (DATPTR),Y
0EFB:8D 04 C0  939         STA  RAMWRT    ;BACK TO MAIN MEM
0EFE:20 5C 0F  940         JSR  INCDAT
0F01:20 5A 10  941 DATDIS  JSR  HIPLOT    ;HIRES GRAPH
0F04:38        942         SEC
0F05:AD 01 C4  943         LDA  LDATA
0F08:29 0F     944         AND  #$0F
0F0A:ED 22 08  945         SBC  MAXVAL
0F0D:AD 00 C4  946         LDA  HDATA
0F10:ED 23 08  947         SBC  MAXVAL+1
0F13:90 46     948         BCC  DATEND    ;BRANCH IF DATA<MAXVAL
0F15:F0 44     949         BEQ  DATEND    ;BRANCH IF ZERO TOO
0F17:AD 01 C4  950         LDA  LDATA
0F1A:29 0F     951         AND  #$0F
0F1C:8D 22 08  952         STA  MAXVAL
0F1F:AD 00 C4  953         LDA  HDATA
0F22:8D 23 08  954         STA  MAXVAL+1  ;SET NEW MAXVAL
0F25:A0 00     955         LDY  #0
0F27:B1 06     956         LDA  (REGPTR),Y
0F29:8D 1F 08  957         STA  MAXCNT
0F2C:C8        958         INY
0F2D:B1 06     959         LDA  (REGPTR),Y
0F2F:8D 20 08  960         STA  MAXCNT+1
0F32:C8        961         INY
0F33:B1 06     962         LDA  (REGPTR),Y
0F35:8D 21 08  963         STA  MAXCNT+2  ;SET NEW MAX. COUNT
0F38:AD 03 08  964         LDA  XYZ
0F3B:29 1F     965         AND  #$1F
0F3D:F0 1C     966         BEQ  DATEND    ;SKIP IF 'HOME' CODE
0F3F:A5 CE     967         LDA  SCRPTR
0F41:48        968         PHA
0F42:A5 CF     969         LDA  SCRPTR+1
0F44:48        970         PHA            ;SAVE SCREEN POINTER
0F45:18        971         CLC
0F46:A5 CE     972         LDA  SCRPTR
0F48:69 80     973         ADC  #$80
0F4A:85 CE     974         STA  SCRPTR
0F4C:A5 CF     975         LDA  SCRPTR+1
0F4E:69 00     976         ADC  #0
0F50:85 CF     977         STA  SCRPTR+1  ;POINT TO NEXT LINE OF TEXT
0F52:20 C5 0F  978         JSR  RTDISP    ;DISPLAY NEW PEAK LOCATION
0F55:68        979         PLA
0F56:85 CF     980         STA  SCRPTR+1
0F58:68        981         PLA
0F59:85 CE     982         STA  SCRPTR    ;RESTORE SCREEN POINTER
0F5B:60        983 DATEND  RTS
0F5C:          984 *
0F5C:18        985 INCDAT  CLC            ;INCREMENTS DATPTR
0F5D:A5 EB     986         LDA  DATPTR
```

```
0F5F:69 01     987         ADC  #1
0F61:85 EB     988         STA  DATPTR
0F63:A5 EC     989         LDA  DATPTR+1
0F65:69 00     990         ADC  #0
0F67:85 EC     991         STA  DATPTR+1
0F69:60        992         RTS
0F6A:          993 *********************************
0F6A:          994 *                               *
0F6A:          995 * UPCREM INCREMENTS INCREM (MOD *
0F6A:          996 * DATINC) AND DECREM DECRENENTS *
0F6A:          997 * INCREM (MOD DATINC). BOTH     *
0F6A:          998 * RETURN A '0' IN ACC. IF INCREM*
0F6A:          999 * IS A '0'.                     *
0F6A:         1000 *                               *
0F6A:         1001 *********************************
0F6A:AD 2E 08 1002 UPCREM  LDA  INCREM
0F6D:CD 07 08 1003         CMP  DATINC
0F70:D0 11    1004         BNE  CRMCNT
0F72:AD 2F 08 1005         LDA  INCREM+1
0F75:CD 08 08 1006         CMP  DATINC+1
0F78:D0 09    1007         BNE  CRMCNT    ;BRANCH IF INCREM<>DATINC
0F7A:A9 00    1008         LDA  #0        ;OTHERWISE CLEAR INCREM
0F7C:8D 2E 08 1009         STA  INCREM
0F7F:8D 2F 08 1010         STA  INCREM+1
0F82:60       1011         RTS
0F83:18       1012 CRMCNT  CLC
0F84:AD 2E 08 1013         LDA  INCREM
0F87:69 01    1014         ADC  #1
0F89:8D 2E 08 1015         STA  INCREM
0F8C:AD 2F 08 1016         LDA  INCREM+1
0F8F:69 00    1017         ADC  #0
0F91:8D 2F 08 1018         STA  INCREM+1
0F94:0D 2E 08 1019         ORA  INCREM
0F97:60       1020         RTS
0F98:         1021 *
0F98:AD 2E 08 1022 DECREM  LDA  INCREM
0F9B:0D 2F 08 1023         ORA  INCREM+1
0F9E:D0 10    1024         BNE  CREMCN
0FA0:AD 07 08 1025         LDA  DATINC
0FA3:8D 2E 08 1026         STA  INCREM
0FA6:AD 08 08 1027         LDA  DATINC+1
0FA9:8D 2F 08 1028         STA  INCREM+1
0FAC:0D 2E 08 1029         ORA  INCREM
0FAF:60       1030         RTS
0FB0:38       1031 CREMCN  SEC
0FB1:AD 2E 08 1032         LDA  INCREM
0FB4:E9 01    1033         SBC  #1
0FB6:8D 2E 08 1034         STA  INCREM
0FB9:AD 2F 08 1035         LDA  INCREM+1
0FBC:E9 00    1036         SBC  #0
0FBE:8D 2F 08 1037         STA  INCREM+1
0FC1:0D 2E 08 1038         ORA  INCREM
0FC4:60       1039         RTS
0FC5:         1040 *********************************
```

```
0FC5:          1041 *                               *
0FC5:          1042 * REAL TIME GEOM. COORD. DISPLAY*
0FC5:          1043 * DISPLAYS (SABC.DE) ON SCREEN  *
0FC5:          1044 * IF XYZ BIT 5 IS A '0'         *
0FC5:          1045 * (WHERE S=SIGN)                *
0FC5:          1046 * STARTING AT LOC. POINTED TO BY*
0FC5:          1047 * SCRPTR                        *
0FC5:          1048 *                               *
0FC5:          1049 *********************************
0FC5:AD 03 08 1050 RTDISP  LDA  XYZ
0FC8:29 20     1051        AND  #32       ;BIT 5
0FCA:F0 03     1052        BEQ  RT0       ;GET OUT IF DISABLED
0FCC:4C 59 10 1053        JMP  RT3
0FCF:A0 00     1054 RT0    LDY  #0
0FD1:B1 1A     1055        LDA  (GCPTR),Y
0FD3:A0 02     1056        LDY  #2
0FD5:38        1057        SEC
0FD6:F8        1058        SED
0FD7:E9 50     1059        SBC  #$50
0FD9:B0 1F     1060        BCS  RT1       ;BRANCH IF GC IS POS.
0FDB:38        1061        SEC
0FDC:A9 00     1062        LDA  #0        ;OTHERWISE SUB GC FROM 500.00
0FDE:F1 1A     1063        SBC  (GCPTR),Y ;(---.-E(F))
0FE0:8D 29 08 1064        STA  RTREG+2
0FE3:88        1065        DEY
0FE4:A9 00     1066        LDA  #0
0FE6:F1 1A     1067        SBC  (GCPTR),Y ;(--C.D-)
0FE8:8D 28 08 1068        STA  RTREG+1
0FEB:A9 50     1069        LDA  #$50
0FED:88        1070        DEY
0FEE:F1 1A     1071        SBC  (GCPTR),Y ;(AB-.--)
0FF0:8D 27 08 1072        STA  RTREG
0FF3:A9 AD     1073        LDA  #MINUS
0FF5:91 CE     1074        STA  (SCRPTR),Y ;OUTPUT A '-'
0FF7:4C 12 10 1075        JMP  RT2
0FFA:B1 1A     1076 RT1    LDA  (GCPTR),Y ;COPY LS DIGITS
0FFC:8D 29 08 1077        STA  RTREG+2
0FFF:88        1078        DEY
1000:B1 1A     1079        LDA  (GCPTR),Y
1002:8D 28 08 1080        STA  RTREG+1
1005:88        1081        DEY
1006:B1 1A     1082        LDA  (GCPTR),Y
1008:38        1083        SEC
1009:E9 50     1084        SBC  #$50      ;SUBTRACT 500 FROM GC
100B:8D 27 08 1085        STA  RTREG
100E:A9 AB     1086        LDA  #PLUS
1010:91 CE     1087        STA  (SCRPTR),Y
1012:D8        1088 RT2    CLD
1013:AD 27 08 1089        LDA  RTREG
1016:29 F0     1090        AND  #$F0
1018:4A        1091        LSR  A
1019:4A        1092        LSR  A
101A:4A        1093        LSR  A
101B:4A        1094        LSR  A
```

```
101C:18       1095         CLC
101D:69 B0    1096         ADC  #$B0
101F:C8       1097         INY
1020:91 CE    1098         STA  (SCRPTR),Y
1022:AD 27 08 1099         LDA  RTREG
1025:29 0F    1100         AND  #$0F      ;(-B-.--)
1027:18       1101         CLC
1028:69 B0    1102         ADC  #$B0      ;ASCII-IZE
102A:C8       1103         INY
102B:91 CE    1104         STA  (SCRPTR),Y ;OUTPUT ASC(B)
102D:AD 28 08 1105         LDA  RTREG+1   ;(--C.--)
1030:4A       1106         LSR  A
1031:4A       1107         LSR  A
1032:4A       1108         LSR  A
1033:4A       1109         LSR  A
1034:18       1110         CLC
1035:69 B0    1111         ADC  #$B0
1037:C8       1112         INY
1038:91 CE    1113         STA  (SCRPTR),Y ;OUTPUT ASC(C)
103A:C8       1114         INY
103B:A9 AE    1115         LDA  #DP
103D:91 CE    1116         STA  (SCRPTR),Y ;OUTPUT A '.'
103F:AD 28 08 1117         LDA  RTREG+1
1042:29 0F    1118         AND  #$0F      ;(---.D-)
1044:18       1119         CLC
1045:69 B0    1120         ADC  #$B0
1047:C8       1121         INY
1048:91 CE    1122         STA  (SCRPTR),Y ;OUTPUT ASC(D)
104A:AD 29 08 1123         LDA  RTREG+2
104D:29 F0    1124         AND  #$F0      ;(---.-E)
104F:4A       1125         LSR  A
1050:4A       1126         LSR  A
1051:4A       1127         LSR  A
1052:4A       1128         LSR  A
1053:18       1129         CLC
1054:69 B0    1130         ADC  #$B0
1056:C8       1131         INY
1057:91 CE    1132         STA  (SCRPTR),Y ;OUTPUT ASC(E)
1059:60       1133 RT3     RTS
105A:         1134 *
105A:         1135 *********************************
105A:         1136 *                               *
105A:         1137 * HIPLOT PLOTS REAL TIME DATA   *
105A:         1138 * IN A 128*128 WINDOW ON THE    *
105A:         1139 * HGR SCREEN 1 BETWEEN X COORDS*
105A:         1140 * 70,198 & YCOORDS 21,150 WHEN *
105A:         1141 * XYZ BIT 5 IS '0'.            *
105A:         1142 *                               *
105A:         1143 *********************************
105A:AD 03 08 1144 HIPLOT  LDA  XYZ
105D:29 20    1145         AND  #$20      ;BIT 5
105F:F0 03    1146         BEQ  HICONT
1061:4C FB 10 1147         JMP  HIEND
1064:98       1148 HICONT  TYA
```

```
1065:48       1149         PHA
1066:8A       1150         TXA
1067:48       1151         PHA             ;SAVE Y & X
1068:AD 2C 08 1152         LDA  NEGFLG
106B:D0 16    1153         BNE  HIB        ;BRANCH IF NEG.
106D:AD 2D 08 1154         LDA  LSTFLG
1070:D0 25    1155         BNE  HID        ;BRANCH IF DIRECTION JUST CHANGED
1072:AD 30 08 1156         LDA  HIRESY
1075:18       1157         CLC
1076:69 01    1158         ADC  #1
1078:C9 96    1159         CMP  #150       ;BOTTOM OF HIRES WINDOW?
107A:90 02    1160         BCC  HIA
107C:A9 16    1161         LDA  #22        ;TOP OF HIRES WINDOW
107E:8D 30 08 1162 HIA     STA  HIRESY
1081:D0 14    1163         BNE  HID
1083:AD 2D 08 1164 HIB     LDA  LSTFLG
1086:F0 0F    1165         BEQ  HID        ;BRANCH IF DIRECTION JUST CHANGED
1088:AD 30 08 1166         LDA  HIRESY
108B:38       1167         SEC
108C:E9 01    1168         SBC  #1
108E:C9 16    1169         CMP  #22        ;TOP OF HIRES WINDOW?
1090:B0 02    1170         BCS  HIC
1092:A9 95    1171         LDA  #149       ;BOTTOM OF HIRES WINDOW
1094:8D 30 08 1172 HIC     STA  HIRESY
1097:AD 30 08 1173 HID     LDA  HIRESY
109A:48       1174         PHA
109B:29 C0    1175         AND  #$C0
109D:85 ED    1176         STA  HGRPTR
109F:4A       1177         LSR  A
10A0:4A       1178         LSR  A
10A1:05 ED    1179         ORA  HGRPTR
10A3:85 ED    1180         STA  HGRPTR
10A5:68       1181         PLA
10A6:85 EE    1182         STA  HGRPTR+1
10A8:0A       1183         ASL  A
10A9:0A       1184         ASL  A
10AA:0A       1185         ASL  A
10AB:26 EE    1186         ROL  HGRPTR+1
10AD:0A       1187         ASL  A
10AE:26 EE    1188         ROL  HGRPTR+1
10B0:0A       1189         ASL  A
10B1:66 ED    1190         ROR  HGRPTR
10B3:A5 EE    1191         LDA  HGRPTR+1
10B5:29 1F    1192         AND  #$1F
10B7:09 20    1193         ORA  #$20
10B9:85 EE    1194         STA  HGRPTR+1
10BB:         1195 *
10BB:AC 30 08 1196         LDY  HIRESY
10BE:B1 FD    1197         LDA  (WINPTR),Y
10C0:10 01    1198         BPL  HIE
10C2:4A       1199         LSR  A          ;DIV BY 2 IF GARBAGE
10C3:A8       1200 HIE     TAY
10C4:B1 F9    1201         LDA  (XPTR),Y
10C6:A8       1202         TAY
```

```
10C7:A9 00    1203         LDA  #0
10C9:91 ED    1204         STA  (HGRPTR),Y ;ERASE PREVIOUS PIXEL
10CB:         1205 *
10CB:AD 00 C4 1206         LDA  HDATA
10CE:4A       1207         LSR  A         ;DIV BY 2
10CF:AC 30 08 1208         LDY  HIRESY
10D2:91 FD    1209         STA  (WINPTR),Y ;PUT NEW VALUE IN WINDOW TABLE
10D4:A8       1210         TAY
10D5:B1 F9    1211         LDA  (XPTR),Y
10D7:8D 31 08 1212         STA  HGRIND    ;SET UP X-AXIS INDEX
10DA:B1 FB    1213         LDA  (XDPTR),Y ;GET PIXEL BYTE
10DC:AC 31 08 1214         LDY  HGRIND
10DF:91 ED    1215         STA  (HGRPTR),Y ;PUT PIXEL ON SCREEN
10E1:A0 09    1216         LDY  #9        ;PUT L & R TRAVELLING WALLS ON SCREEN
10E3:B1 ED    1217         LDA  (HGRPTR),Y
10E5:49 FF    1218         EOR  #$FF
10E7:91 ED    1219         STA  (HGRPTR),Y
10E9:A0 1D    1220         LDY  #29
10EB:B1 ED    1221         LDA  (HGRPTR),Y
10ED:49 FF    1222         EOR  #$FF
10EF:91 ED    1223         STA  (HGRPTR),Y
10F1:AD 2C 08 1224         LDA  NEGFLG
10F4:8D 2D 08 1225         STA  LSTFLG
10F7:68       1226         PLA
10F8:AA       1227         TAX
10F9:68       1228         PLA
10FA:A8       1229         TAY            ;RESTORE X & Y
10FB:60       1230 HIEND   RTS
10FC:         1231 *

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

```
0ACE BITY       0ADB BITZ       C410 CKDIR      082A CKMASK
0B87 CLRLUP     0B7B CLRREG     0AA3 CMP16      0AAF CMP8
0ABE CMPCNT     0AEC CMPEND     0AE8 CMPERR     0A91 CMPFF
0AEE CMPFIN     0AA9 CNT16      0ED9 COLLUP     0B16 CONT1A
0B00 CONT1      0B3A CONT2A     0B24 CONT2      0B45 CONT3
0B50 CONT4      0B6A CONT5      0FB0 CREMCN     0F83 CRMCNT
0ECC DATCNT     0EAE DATCOL     0EC7 DATCON     0F01 DATDIS
0F5B DATEND     0807 DATINC       EB DATPTR     0EB9 DATST
0EC4 DATUP      0E8E DCRDIF     0E48 DECCNT     0E6D DECGC
0F98 DECREM     0DCF DELAY      0DD9 DELLUP     0DDC DELRET
0824 DIFF         AE DP         0C34 ENABLE     0C46 ENCNT1
0DCE ENDEND     0DC2 END        0C48 ENEND      0809 ERFLAG
0DBD ERROR      0B7A FLGRET     0B6D FLINIT     0127 GCINC
  1A GCPTR      080A GCX        080E GCY        0812 GCZ
0AEF GETKEY     C400 HDATA      0831 HGRIND       ED HGRPTR
107E HIA        1083 HIB        1064 HICONT     1094 HIC
1097 HID        10C3 HIE        10FB HIEND      105A HIPLOT
0830 HIRESY     0BC1 HLEND      0BAE HLUP1      0B8D HOME
C40D IFR1       0DF5 INCCNT     0E1E INCCON     0F5C INCDAT
0E27 INCGC      082E INCREM     C000 KEYBD      C010 KEYDWN
  08 LARROW     C401 LDATA      082B LIMASK     C411 LIMIT
082D LSTFLG     0C4C MAIN       081F MAXCNT     0822 MAXVAL
0CBD MCONT      0CCD MCONT1       AD MINUS      0804 MOVE
082C NEGFLG     0E1B NOHOME       60 NOMVAL     0DDF OUTPLS
  AB PLUS       0CAB POS        0D16 RAMP       C004 RAMWRT
  15 RARROW     0D36 RCONT1     0D39 RCONT2     0D76 RCONT3
0D79 RCONT4     0D9A RCONT5     0D9D RCONT6     0D4F RCONTA
0D52 RCONTB     0DB3 RCONTC     0DB6 RCONTD     0D7E RDWN
  06 REGPTR       0D RET        0D1D RLUP1      0D60 RLUP2
0D80 RLUP3      0FCF RT0        0FFA RT1        1012 RT2
1059 RT3        0FC5 RTDISP     0827 RTREG      0CFA SCONT0
0D08 SCONT1     0D0B SCONT2       CE SCRPTR     0D13 SERROR
C400 SLOT       0CEF SLUP1      0A32 START        0B TBLPTR
  15 TWEAK      0F6A UPCREM       FD WINPTR     0972 WINTBL
0816 XCOUNT       FB XDPTR      08B2 XDTBL        F9 XPTR
06D4 XSCR       0BC2 XSETUP     0832 XTBL       0AB6 XXYYZZ
0AC1 XYORZ      0803 XYZ        0819 YCOUNT     06E1 YSCR
0BE8 YSETUP     081C ZCOUNT     06EE ZSCR       0C0E ZSETUP
```

```
  06 REGPTR       08 TBLPTR       08 LARROW       0D RET
  15 RARROW       15 TWEAK        1A GCPTR        60 NOMVAL
  AB PLUS         AD MINUS        AE DP           CE SCRPTR
  EB DATPTR       ED HGRPTR       F9 XPTR         FB XDPTR
  FD WINPTR     0127 GCINC      06D4 XSCR       06E1 YSCR
06EE ZSCR       0803 XYZ        0804 MOVE       0807 DATINC
0809 ERFLAG     080A GCX        080E GCY        0812 GCZ
0816 XCOUNT     0819 YCOUNT     081C ZCOUNT     081F MAXCNT
0822 MAXVAL     0824 DIFF       0827 RTREG      082A CKMASK
082B LIMASK     082C NEGFLG     082D LSTFLG     082E INCREM
0830 HIRESY     0831 HGRIND     0832 XTBL       08B2 XDTBL
0972 WINTBL     0A32 START      0A91 CMPFF      0AA3 CMP16
0AA9 CNT16      0AAF CMP8       0AB6 XXYYZZ     0ABE CMPCNT
0AC1 XYORZ      0ACE BITY       0ADB BITZ       0AE8 CMPERR
0AEC CMPEND     0AEE CMPFIN     0AEF GETKEY     0B00 CONT1
0B16 CONT1A     0B24 CONT2      0B3A CONT2A     0B45 CONT3
0B50 CONT4      0B6A CONT5      0B6D FLINIT     0B7A FLGRET
0B7B CLRREG     0B87 CLRLUP     0B8D HOME       0BAE HLUP1
0BC1 HLEND      0BC2 XSETUP     0BE8 YSETUP     0C0E ZSETUP
0C34 ENABLE     0C46 ENCNT1     0C48 ENEND      0C4C MAIN
0CAB POS        0CBD MCONT      0CCD MCONT1     0CEF SLUP1
0CFA SCONT0     0D08 SCONT1     0D0B SCONT2     0D13 SERROR
0D16 RAMP       0D1D RLUP1      0D36 RCONT1     0D39 RCONT2
0D4F RCONTA     0D52 RCONTB     0D60 RLUP2      0D76 RCONT3
0D79 RCONT4     0D7E RDWN       0D80 RLUP3      0D9A RCONT5
0D9D RCONT6     0DB3 RCONTC     0DB6 RCONTD     0DBD ERROR
0DC2 END        0DCE ENDEND     0DCF DELAY      0DD9 DELLUP
0DDC DELRET     0DDF OUTPLS     0DF5 INCCNT     0E1B NOHOME
0E1E INCCON     0E27 INCGC      0E48 DECCNT     0E6D DECGC
0E8E DCRDIF     0EAE DATCOL     0EB9 DATST      0EC4 DATUP
0EC7 DATCON     0ECC DATCNT     0ED9 COLLUP     0F01 DATDIS
0F5B DATEND     0F5C INCDAT     0F6A UPCREM     0F83 CRMCNT
0F98 DECREM     0FB0 CREMCN     0FC5 RTDISP     0FCF RT0
0FFA RT1        1012 RT2        1059 RT3        105A HIPLOT
1064 HICONT     107E HIA        1083 HIB        1094 HIC
1097 HID        10C3 HIE        10FB HIEND      C000 KEYBD
C004 RAMWRT     C010 KEYDWN     C400 HDATA      C400 SLOT
C401 LDATA      C40D IFR1       C410 CKDIR      C411 LIMIT
```

APPENDIX B

Listing of CONTROLLER

```
]LIST
100  REM ********************************
110  REM *                              *
120  REM * BASIC SCANNER SYSTEM AXIS    *
130  REM * CONTROLLER AND DATA ACQUIS-  *
140  REM * ITION PROGRAM                *
150  REM *                              *
160  REM * COPYRIGHT 1984               *
170  REM * DAVE PADGITT                 *
180  REM * UNIVERSITY OF ILLINOIS       *
190  REM * AND URI THERM-X INC.         *
200  REM *                              *
210  REM * 11/11/84                     *
220  REM *                              *
230  REM ********************************
290  REM VARIABLES AND CONSTANTS
300 D$ =  CHR$ (4)
303 I$ =  CHR$ (137)
305  REM VIA #1 (A/D)
310 SLOT = 50176: REM SLOT #4 ($C400)
315 CMPULSE = 0.00127: REM # OF CM/PULSE
320 HD = SLOT + 0: REM HDATA (PORT B)
330 LD = SLOT + 1: REM LDATA (PORT A)
340 FIR1 = SLOT + 13: REM IFR OF VIA #1
350  REM VIA #2 (CK & DIR & LIMITS)
360 CKDIR = SLOT + 16: REM PORT B
370 LIMIT = SLOT + 17: REM PORT A
400  REM MACHINE LANG. ROUTINES
405 XFER = 768: REM AUX. MEM. TO HIRES DATA TRANSFER ROUTINE @ ($301).
410 BASE = 2048: REM STEPPER DRIVER/DATA COLLECTION/REAL TIME DISPLAY ROUTINE ($0800)
415 BUFSIZE = 48640: REM DATA BUFFER IN AUX. MEM. $0200-$BFFF
417  REM GLOBAL REGS.
420 XYZ = BASE + 3: REM ADDRESS OF XYZ COMMAND REG.
430 MOVE = BASE + 4: REM ADDRESS OF 3-BYTE MOVE REG.
440 DINC = BASE + 7: REM ADDRESS OF DATA COLLECTION INCREMENT REG.
450 ERFLAG = BASE + 9: REM ERROR FLAG REGISTER
460 DPTR = 235: REM ZERO PAGE DATA POINTER DATPTR ($EB)
470 SRCBEG = 60: REM ZERO PAGE POINTER FOR XFER ROUTINE
500  REM GEOMETRIC COORDINATE REGISTERS
510 GX = BASE + 10
520 GY = BASE + 14
530 GZ = BASE + 18
550  REM 24 BIT COUNT REGISTERS
560 XCOUNT = BASE + 22
570 YCOUNT = BASE + 25
580 ZCOUNT = BASE + 28
590 MAXCNT = BASE + 31
600 VMAX = BASE + 34
605  REM TANK LIMITS
610 XT = 23: REM THESE VALUES CORR. TO ACTUAL RANGE BETWEEN LIMIT SWITCHES
620 YT = 29
630 ZT = 31
```

```
640  REM TANK LIMITS (PULSES FROM HOME)
650 XP =  INT (XT / CMPULSE + .5)
660 YP =  INT (YT / CMPULSE + .5)
670 ZP =  INT (ZT / CMPULSE + .5)
1000  REM ******************************
1010  REM *                            *
1020  REM * MAIN CALLING PROGRAM       *
1030  REM *                            *
1037  PRINT  CHR$ (12)
1040  REM ******************************
1100  GOSUB 2000: REM INITIALIZE
1110  HOME
1115 PX = 0:PY = 0: REM CLEAR PEAKS
1120  PRINT "GEOMETRIC COORDINATES? (Y/N)"
1125  PRINT "(IF NOT, TANK COORDINATES ASSUMED): ";: INPUT "";GC$
1130  IF GC$ <  > "Y" AND GC$ <  > "N" THEN  GOSUB 13100: GOTO 1110
1140  IF GC$ <  > "Y" THEN 1160
1150  GOSUB 3000: REM USER POSITIONING
1160  GOSUB 4000: REM HOME THE DETECTOR
1170  GOSUB 5000: REM GET Z COORD.
1175  GOSUB 13320: REM DELAY
1180  IF PX = 0 AND PY = 0 THEN 1190
1183  HOME : PRINT "CONTINUE USING PREVIOUS PEAK? (Y/N): ";
1185  INPUT "";PP$
1187  IF PP$ = "Y" THEN 1250
1190  HOME : PRINT "DO YOU WANT TO FIND A PEAK? (Y/N)"
1193  PRINT "(IF NOT, IMMED. SCAN ASSUMED): ";: INPUT "";PD$
1195  IF PD$ <  > "Y" THEN 1250
1197  HOME : PRINT "AUTOMATIC PEAK DETECTION? (Y/N)"
1200  PRINT "(IF NOT, MANUAL PEAK DET. ASSUMED): ";: INPUT "";AP$
1210  IF AP$ <  > "Y" THEN 1240
1220  GOSUB 7000: REM AUTOMATICALLY FIND THE PEAK
1225  GOSUB 13330
1230  GOTO 1300
1240  GOSUB 6000: REM MANUALLY FIND THE PEAK
1245  GOTO 1300
1250  HOME
1260  PRINT "SCAN & IMMED. DATA DISPLAY? (Y/N): ";
1265  INPUT "";ID$
1270  IF ID$ = "Y" THEN 1280
1275  IF PX = 0 AND PY = 0 THEN 1197
1277  GOTO 1300
1280  GOSUB 10500: REM  IMMED. COLL. & DISP.
1290  GOTO 1420
1300  HOME : PRINT "TRANSVERSE SCAN? (Y/N)"
1305  PRINT "(IF NOT, LONGITUDINAL ASSUMED): ";: INPUT "";TS$
1307  IF TS$ <  > "Y" AND TS$ <  > "N" THEN  GOSUB 13100: GOTO 1300
1310  IF TS$ <  > "Y" THEN 1340
1320  GOSUB 8000: REM TRANSVERSE SCAN
1325  GOSUB 13330
1330  GOTO 1350
1340  GOSUB 9000: REM LONGITUDINAL SCAN
1345  GOSUB 13330
1350  TEXT : HOME
```

```
1360  INPUT "SAVE DATA TO DISK? (Y/N): ";SD$
1370  IF SD$ <  > "Y" THEN 1420
1380  GOSUB 10000
1390  GOSUB 13320
1420  HOME
1430  INPUT "DO ANOTHER SCAN? (Y/N): ";SC$
1450  IF SC$ = "N" THEN 1999
1460  GOTO 1170: REM GET NEW Z COORD.
1999  END
2000  REM ******************************
2010  REM *                            *
2020  REM * INITIALIZATIONS            *
2030  REM *                            *
2040  REM ******************************
2090  GOSUB 2200: REM SET UP VIA'S
2100  GOSUB 2500: REM DISPLAY TITLE PAGE
2110  GOSUB 13330: REM DELAY
2120  GOSUB 2600: REM LOAD MACH. LANG. PROGS. & RAMP TABLE
2130  GOSUB 2300: REM CLEAR COUNT REGS.
2140  GOSUB 2400: REM CLEAR GEOM. COORD. REGS.
2160  RETURN
2200  REM VIA DDR SETUPS
2210  POKE SLOT + 2,0: REM DDRB1
2220  POKE SLOT + 3,240: REM DDRA1
2230  POKE SLOT + 18,63: REM DDRB2
2240  POKE SLOT + 19,0: REM DDRA2
2250  POKE SLOT + 12,0: REM PCR1
2255  POKE LD,0
2260  RETURN
2300  REM CLEAR COUNT REGISTERS
2310  POKE XYZ,16: REM CLEAR CODE
2320  CALL BASE
2330  RETURN
2400  REM CLEAR GEOM. COORD. REGS.
2410  POKE GX,80: REM MSB=$80 (BCD=50) EXCESS-500 REP. OF ZERO
2420  POKE GY,80
2430  POKE GZ,80
2440  FOR I = 1 TO 3
2450  POKE GX + I,0
2460  POKE GY + I,0
2470  POKE GZ + I,0
2480  NEXT I
2490  RETURN
2500  REM TITLE PAGE DISPLAY
2510  HOME : VTAB 5: HTAB 2
2520  PRINT "URI THERM-X SCANNER SYSTEM CONTROLLER"
2530  VTAB 7: HTAB 12
2540  PRINT "BY DAVE PADGITT"
2550  VTAB 9: HTAB 12
2560  PRINT "COPYRIGHT 1984"
2570  RETURN
2600  REM LOAD MACHINE LANGUAGE ROUTINES
2610  HOME
2620  PRINT "LOAD MACHINE LANG. ROUTINES? (Y/N)"
```

```
2630  PRINT "(MUST BE DONE ON FIRST ENTRY): ";: INPUT "";ML$
2640  IF ML$ <  > "Y" THEN 2680
2650  PRINT D$;"BLOAD STEPPER.OBJ0,A$0800": REM ORG $0800
2660  PRINT D$;"BLOAD RAMP,A$1200"
2670  PRINT D$;"BLOAD TRANSFER.OBJ0,A$300"
2680  RETURN
3000  REM ******************************
3010  REM *                            *
3020  REM * USER POSITIONING           *
3030  REM *                            *
3040  REM ******************************
3100  HOME
3110  GOSUB 2400: REM CLEAR GEOM. COORD. REGS.
3115 SI = 0: GOSUB 12700
3120  GOSUB 12100: REM PUT UP REAL TIME DISPLAY LEGENDS
3130  HTAB 1: VTAB 1
3135  PRINT "MANUAL CENTER POSITIONING"
3140  PRINT "INPUT X,Y,Z AND USE LEFT AND"
3150  PRINT "RIGHT ARROWS FOR AUTO POSITIONING"
3155  PRINT "PRESSING 'RETURN' WHEN DONE"
3160  PRINT "OR MANUALLY POSITION AND PRESS"
3170  PRINT "'SPACE' WHEN FINISHED: ";
3180  GET A$
3190  IF A$ = "X" THEN  POKE XYZ,73: GOTO 3240: REM X-AXIS WITH SHUTOFF
3200  IF A$ = "Y" THEN  POKE XYZ,74: GOTO 3240: REM Y-AXIS WITH SHUTOFF
3210  IF A$ = "Z" THEN  POKE XYZ,76: GOTO 3240: REM Z-AXIS WITH SHUTOFF
3220  IF A$ = " " THEN 3300: REM RETURN IF 'SPACE'
3230  GOTO 3130
3235 SI = 0: GOSUB 12700
3240  CALL BASE: REM CALL STEPPER
3250  GOTO 3130
3300  RETURN
4000  REM ******************************
4010  REM *                            *
4020  REM * HOME THE DETECTOR          *
4030  REM *                            *
4040  REM ******************************
4050  IF GC$ = "Y" THEN 4090
4060  HOME
4065  PRINT "TANK COORDINATES"
4070  PRINT "INPUT Z DIST. FROM HOME"
4075  PRINT "TO APPLICATOR (CM)."
4076  PRINT "(WARNING:FAILURE TO ENTER A"
4077  PRINT "SUITABLE VALUE MAY RESULT"
4078  PRINT " IN PHYSICAL DAMAGE): ";: INPUT "";ZM
4080  IF (ZM > ZT) OR (ZM <  = 0) THEN  GOSUB 13000: GOTO 4060
4090  HOME
4100  GOSUB 12100
4110  VTAB 1
4130  PRINT "HOMING THE SYSTEM..."
4140  GOSUB 2300: REM CLEAR COUNT REGS.
4150  GOSUB 2400: REM CLEAR GEOM. COORD. REGS.
4155 SI = 0: GOSUB 12700
4160  POKE XYZ,64: REM CODE FOR HOME WITH MOTOR DISABLE
```

```
4170  CALL BASE
4180  IF GC$ <  > "Y" THEN  GOSUB 2400: GOTO 4450
4190  REM GET X,Y,Z (CM) IN BASIC VARIABLES XM,YM,ZM
4200 XM = ( PEEK (XCOUNT) +  PEEK (XCOUNT + 1) * 256 +  PEEK (XCOUNT + 2) * 65536) * CMPU
     LSE
4210 YM = ( PEEK (YCOUNT) +  PEEK (YCOUNT + 1) * 256 +  PEEK (YCOUNT + 2) * 65536) * CMPU
     LSE
4220 ZM = ( PEEK (ZCOUNT) +  PEEK (ZCOUNT + 1) * 256 +  PEEK (ZCOUNT + 2) * 65536) * CMPU
     LSE
4450  GOSUB 2300: REM CLEAR COUNT REGS.
4460  VTAB 1
4480  PRINT "                ...DONE"
4490  GOSUB 13320: REM DELAY
4500  RETURN
5000  REM ********************************
5010  REM *                              *
5020  REM * PROMPT USER FOR Z COORDS.    *
5030  REM * AND GO THERE                 *
5040  REM *                              *
5050  REM ********************************
5110  IF GC$ <  > "Y" THEN 5300
5120  HOME
5125  PRINT "DESIRED Z COORDINATE"
5130  PRINT "Z RANGE BETWEEN 0 & "; - ZM;" (CM)"
5140  PRINT "INPUT Z COORDINATE (CM): ";: INPUT "";SZ
5150  IF SZ <  - ZM OR SZ > 0 THEN  GOSUB 13000: GOTO 5120
5160 SZ = (ZM + SZ) / CMPULSE
5165 MV = SZ
5170  GOTO 5400
5300  HOME
5305  PRINT "DESIRED Z COORDINATE"
5310  PRINT "Z-RANGE BETWEEN 0 & ";ZM;" (CM)"
5320  PRINT "INPUT Z COORDINATE (CM): ";: INPUT "";SZ
5330  IF SZ < 0 OR SZ > ZM THEN  GOSUB 13000: GOTO 5300
5340 SZ = SZ / CMPULSE
5350 MV = SZ
5400  GOSUB 12400
5405 SI = 0: GOSUB 12700
5410  GOSUB 12100
5420  POKE XYZ,68
5430  CALL BASE
5500  RETURN
6000  REM ********************************
6010  REM *                              *
6020  REM * MANUALLY FIND THE PEAK       *
6030  REM *                              *
6040  REM ********************************
6050 SI = 16: REM DEFAULT SWEEP INCREMENT
6060 PX = 0:PY = 0: REM PEAKS SET INITIALLY TO 0
6100  HOME : VTAB 21
6105  PRINT "MANUAL PEAK DETECTION"
6110  PRINT "INPUT X OR Y, OR I TO CHANGE"
6115  PRINT "SWEEP INCREMENT, AND ANY OTHER"
6120  PRINT "KEY WHEN FINISHED: ";
```

```
6130  GET MA$
6140  IF MA$ = "X" THEN  POKE XYZ,201: GOTO 6300: REM X-AXIS WITH RTD. & DATA COLL. & DIS
     ABLE
6150  IF MA$ = "Y" THEN  POKE XYZ,202: GOTO 6400: REM Y-AXIS WITH RTD. & DATA COLL. & DIS
     ABLE
6160  IF MA$ = "I" THEN 6180
6170  GOTO 6700
6180  HOME : VTAB 21
6190  INPUT "INPUT SWEEP INCREMENT (CM): ";SI
6200  IF SI < 2 * CMPULSE THEN  PRINT "MUST BE GREATER THAN ";2 * CMPULSE;" CM": GOSUB 13
     330: GOTO 6180
6210 SI =  INT (SI / CMPULSE): REM CONVERT TO # OF PULSES
6220  GOTO 6100
6300  REM X-AXIS
6310  GOSUB 12600: REM SET UP RTD. & CALL BASE
6320  GOSUB 12550
6350  GOTO 6100
6400  REM Y-AXIS
6410  GOSUB 12600
6420  GOSUB 12560
6450  GOTO 6100
6700  REM MOVE TO CURRENT PEAKS
6705  HOME : VTAB 21
6710  IF GC$ = "Y" THEN KX = PX - (XM / CMPULSE):KY = PY - (YM / CMPULSE): GOTO 6720
6715 KX = PX:KY = PY
6720  PRINT "CURR. PEAK POS. ('0'= NOT SCANNED):"
6730  PRINT "X= ";KX * CMPULSE;" (CM)"
6740  PRINT "Y= ";KY * CMPULSE;" (CM)"
6750  PRINT "MOVE TO THESE PEAKS? (Y/N): ";
6755  INPUT "";EM$
6760  IF EM$ <  > "Y" THEN 6100
6800  REM MOVE TO PEAK X,Y
6805  IF PX = 0 OR PY = 0 THEN  PRINT "MUST FIND BOTH PEAKS": GOSUB 13310: GOTO 6100
6810  GOSUB 12300
6815 MV = PX
6820  GOSUB 12400: REM LOAD MOVE REG.
6825  HOME : TEXT : GOSUB 12100
6830  POKE XYZ,193
6835 SI = 0: GOSUB 12700
6840  CALL BASE
6850 MV = PY
6860  GOSUB 12400
6870  POKE XYZ,194
6880  CALL BASE
6883  GOSUB 13320
6885  HOME
6890  RETURN : REM LEAVE SUBROUTINE 6000
7000  REM ********************************
7010  REM *                              *
7020  REM * AUTOMATICALLY FIND THE PEAK *
7030  REM *                              *
7040  REM ********************************
7100  TEXT : HOME
7110  PRINT "AUTOMATIC PEAK DETECTION"
```

```
7130 MV = 1 / CMPULSE: GOSUB 12400
7140 SI = 0: GOSUB 12700
7150  GOSUB 12100
7160  POKE XYZ,65: CALL BASE
7170  POKE XYZ,66: CALL BASE
7180 PK = 0: REM CLEAR PEAK DATA VALUE
7190  FOR K = 2 / CMPULSE TO (YT - 1) / CMPULSE STEP 4 / CMPULSE
7195 KK = K
7200  FOR J = (XT - 1) / CMPULSE TO 0 STEP  - (XT - 2) / CMPULSE
7205 K1 = 66:K2 = 193
7210  GOSUB 7300
7212  IF FG = 0 THEN 7230
7215  GOSUB 12550: REM GET PX
7217  GOSUB 12500:PY = CY
7220 P1 = PX:P2 = PY
7230 KK = K + 2 / CMPULSE
7240  NEXT J
7250  NEXT K
7260  GOSUB 7500
7270  RETURN
7300 MV = KK: GOSUB 12400
7310 SI = 0: GOSUB 12700
7320  GOSUB 12100
7330  POKE XYZ,K1: CALL BASE
7340 MV = J: GOSUB 12400
7350 SI = 0.1 / CMPULSE: GOSUB 12700: REM 1MM RESOLUTION
7360  POKE XYZ,255: CALL BASE
7370  POKE XYZ,K2: GOSUB 12600
7380  IF  PEEK (VMAX) + 16 *  PEEK (VMAX + 1) <  = PK THEN FG = 0: RETURN
7390 PK =  PEEK (VMAX) + 16 *  PEEK (VMAX + 1)
7400 FG = 1
7410  RETURN
7500  IF P1 - 2 / CMPULSE < 0 OR P1 + 2 / CMPULSE > XP OR P2 - 2 / CMPULSE < 0 OR P2 + 2 /
    CMPULSE > YP THEN 7900
7510 MV = P1 - 2 / CMPULSE: GOSUB 12400
7520 SI = 0: GOSUB 12700
7530  GOSUB 12100
7540  POKE XYZ,65: CALL BASE
7550 MV = P2 - 2 / CMPULSE: GOSUB 12400
7560  POKE XYZ,66: CALL BASE
7570 PK = 0: REM CLEAR PEAK DATA VALUE
7580  FOR K = P1 - 2 / CMPULSE TO P1 + 2 / CMPULSE STEP 1 / CMPULSE
7585 KK = K
7590  FOR J = P2 + 2 / CMPULSE TO P2 - 3 / CMPULSE STEP  - 4 / CMPULSE
7595 K1 = 65:K2 = 194
7600  GOSUB 7300
7602  IF FG = 0 THEN 7610
7605  GOSUB 12560: REM GET PY
7607  GOSUB 12500:PX = CX
7610 KK = K + .5 / CMPULSE
7620  NEXT J
7630  NEXT K
7640  TEXT : HOME
7650  PRINT "MOVE TO PEAK": GOSUB 13310
```

```
7660 MV = PX: GOSUB 12400
7670 SI = 0: GOSUB 12700
7680  GOSUB 12100
7690  POKE XYZ,65: CALL BASE
7700 MV = PY: GOSUB 12400
7710  POKE XYZ,66: CALL BASE
7720  RETURN
7900  TEXT : HOME
7910  PRINT "PEAK IS TOO CLOSE TO TANK LIMITS"
7920  GOSUB 13330
7930  RETURN
8000  REM ******************************
8010  REM *                            *
8020  REM * TRANSVERSE SCAN            *
8030  REM *                            *
8040  REM ******************************
8050 RZ = 0:IZ = 0
8100  HOME
8110  IF GC$ <  > "Y" THEN 8300
8120  PRINT "CENTER SCAN AROUND PEAK (Y/N): ";
8130  INPUT "";CS$
8140  IF CS$ = "Y" THEN SX = PX:SY = PY: GOTO 8400
8150  HOME : PRINT "ENTER X & Y CENTER COORDS. (CM)"
8155  PRINT "(DEFAULTS TO THE ORIGIN)"
8157  PRINT
8160  PRINT "X COORDINATE: 0.0";: HTAB 15
8170  INPUT "";SX$
8180  IF SX$ = "" THEN  HTAB 15: VTAB 4: PRINT "0.0":SX = XM / CMPULSE: GOTO 8210
8190 SX =  VAL (SX$)
8200  IF ((SX + XM) <  = 0) OR (SX + XM <  = XT) THEN  GOSUB 13000: GOTO 8150
8205 SX = (SX + XM) / CMPULSE
8210  VTAB 6: PRINT "Y COORDINATE: 0.0";: HTAB 15
8220  INPUT "";SY$
8230  IF SY$ = "" THEN  HTAB 15: VTAB 6: PRINT "0.0":SY = YM / CMPULSE: GOTO 8400
8240 SY =  VAL (SY$)
8250  IF ((SY + YM) <  = 0) OR (SY + YM >  = YT) THEN  GOSUB 13000: GOTO 8210
8260 SY = (SY + YM) / CMPULSE
8270  GOTO 8400
8300  HOME : PRINT "ENTER X & Y CENTER COORDS. (CM)"
8305  PRINT
8310  INPUT "X COORDINATE: ";SX
8340  IF SX <  = 0 OR SX >  = XT THEN  GOSUB 13000: GOTO 8300
8350 SX = SX / CMPULSE
8360  INPUT "Y COORDINATE: ";SY
8370  IF SY <  = 0 OR SY >  = YT THEN 8360
8380 SY = SY / CMPULSE
8400  HOME : PRINT "ENTER X & Y SCAN RANGES (CM)"
8410  PRINT "(DEFAULTS TO MAXIMUM PERMISSABLE)"
8420  PRINT "(ENTER '0' FOR AT MOST ONE VALUE)"
8425  PRINT
8430  IF (XP - SX) > SX THEN RX = SX: GOTO 8450
8440 RX = XP - SX
8450  VTAB 5: PRINT "X RANGE: ";RX * CMPULSE;: HTAB 10
8460  INPUT "";RX$
```

```
8470  IF RX$ = "" THEN  HTAB 10: VTAB 5: PRINT RX * CMPULSE: GOTO 8500
8480  IF  VAL (RX$) < 0 OR  VAL (RX$) > RX * CMPULSE THEN  GOSUB 13000: GOTO 8450
8490 RX =  VAL (RX$) / CMPULSE
8500  IF (YP - SY) > SY THEN RY = SY: GOTO 8520
8510 RY = YP - SY
8520  VTAB 7: PRINT "Y RANGE: ";RY * CMPULSE;: HTAB 10
8530  INPUT "";RY$
8540  IF RY$ = "" THEN  HTAB 10: VTAB 7: PRINT RY * CMPULSE: GOTO 8570
8550  IF  VAL (RY$) < 0 OR  VAL (RY$) > RY * CMPULSE THEN  GOSUB 13000: GOTO 8520
8560 RY =  VAL (RY$) / CMPULSE
8570  IF (RX = 0) AND (RY = 0) THEN  PRINT "BOTH RANGES CANNOT EQUAL ZERO": GOSUB 13310: GOT
O
    8400
8600  HOME : PRINT "ENTER X & Y DIST. BETWEEN SAMPLES (CM)"
8610  IF RX = 0 THEN 8640
8620  INPUT "X INCREMENT: ";IX
8625  IF IX = 0 THEN  PRINT "MUST BE NONZERO": GOSUB 13310: GOTO 8600
8630  IF (IX < 2 * CMPULSE) OR (IX >  = RX * CMPULSE) THEN  GOSUB 13000: GOTO 8600
8631  IF 4 * RX * CMPULSE / IX > BUFSIZE THEN  GOSUB 13200: GOTO 8600
8633 IX = IX / CMPULSE
8635  IF RY = 0 THEN 8665
8640  INPUT "Y INCREMENT: ";IY
8645  IF IY = 0 THEN  PRINT "MUST BE NONZERO": FOR I = 1TO1500: NEXT : GOTO 8640
8650  IF (IY < 2 * CMPULSE) OR (IY >  = RY * CMPULSE) THEN  PRINT "OUT OF RANGE": FOR I =
    1 TO 1500: NEXT : GOTO 8600
8651  IF RX = 0 THEN 8654
8652  IF 8 * RX * CMPULSE * RY * CMPULSE / (IX * IY) > BUFSIZE THEN  GOSUB 13200: GOTO 86
    00
8653  GOTO 8655
8654  IF 4 * RY * CMPULSE / IY > BUFSIZE THEN  GOSUB 13200: GOTO 8600
8655 IY = IY / CMPULSE
8665  TEXT : HOME
8670  POKE XYZ,255: CALL BASE: REM CODE TO RESET DATA POINTER
8680  IF RY = 0 THEN 8750
8690  IF RX = 0 THEN 8800
8700  FOR Y = SY - RY TO SY + RY STEP IY
8710  GOSUB 8900
8720  NEXT Y
8730  RETURN
8750 Y = SY
8760  GOSUB 8900
8770  RETURN
8800 MV = SX: GOSUB 12400
8810 SI = 0: GOSUB 12700
8820  GOSUB 12100
8830  POKE XYZ,65: CALL BASE
8840 MV = SY - RY: GOSUB 12400
8850  POKE XYZ,66: CALL BASE
8860 MV = SY + RY: GOSUB 12400
8870 SI = IY: POKE XYZ,194
8880  GOSUB 12600
8890  RETURN
8900  TEXT : HOME : GOSUB 12100
8910 MV = Y: GOSUB 12400
8920 SI = 0: GOSUB 12700
```

```
8930  POKE XYZ,66: CALL BASE
8940 MV = SX - RX: GOSUB 12400
8950  POKE XYZ,65: CALL BASE
8960 MV = SX + RX: GOSUB 12400
8970 SI = IX: POKE XYZ,193: GOSUB 12600
8990  RETURN
9000  REM ******************************
9010  REM *                            *
9020  REM * LONGITUDINAL SCAN          *
9030  REM *                            *
9040  REM ******************************
9050 RX = 0:RY = 0:IX = 0:IY = 0
9100  IF (ZP - SZ) > SZ THEN RZ = SZ: GOTO 9115
9110 RZ = ZP - SZ
9115  HOME : PRINT "LONGITUDINAL SCAN": PRINT
9120  PRINT "ENTER Z SCAN RANGE (CM)"
9130  PRINT "(DEFAULT TO MAX. VALUE)"
9135  PRINT
9140  PRINT "Z RANGE: ";RZ * CMPULSE;: HTAB 10
9150  INPUT "";RZ$
9160  IF RZ$ = "" THEN  HTAB 10: VTAB 4: PRINT RZ * CMPULSE: GOTO 9200
9170  IF  VAL (RZ$) < CMPULSE OR  VAL (RZ$) > RZ * CMPULSE THEN  GOSUB 13000: GOTO 9100
9180 RZ =  VAL (RZ$) / CMPULSE
9200  HOME : PRINT "ENTER Z SCAN INCREMENT (CM)"
9205  PRINT
9210  INPUT "Z INCREMENT: ";IZ
9220  IF (IZ < 2 * CMPULSE) OR (IZ  = RZ * CMPULSE) THEN  GOSUB 13000: GOTO 9200
9225  IF 4 * RZ * CMPULSE / IZ > BUFSIZE THEN  GOSUB 13200: GOTO 9200
9230 IZ = IZ / CMPULSE
9300  TEXT : HOME : GOSUB 12100
9310 MV = PX: GOSUB 12400
9320 SI = 0: GOSUB 12700
9330  POKE XYZ,65: CALL BASE
9340 MV = PY: GOSUB 12400
9350  POKE XYZ,66: CALL BASE
9360 MV = SZ - RZ: GOSUB 12400
9370  POKE XYZ,68: CALL BASE
9380 MV = SZ + RZ: GOSUB 12400
9390 SI = IZ
9395  POKE XYZ,255: CALL BASE
9400  POKE XYZ,196: GOSUB 12600
9410  RETURN
10000  REM ******************************
10010  REM *                            *
10020  REM * STORE TO DISK              *
10030  REM *                            *
10040  REM ******************************
10050  TEXT : HOME
10060  INPUT "INPUT FILENAME: ";FF$
10070  IF  LEN (FF$) > 30 THEN  PRINT "TOO LONG": GOSUB 13310: GOTO 10050
10110  PRINT "INSERT DATA DISK & HIT A KEY ";
10120  GET SH$
10130  HOME
10140  PRINT "SAVING DATA TO DISK..."
```

```
10150  ONERR  GOTO 10110
10160 DL =  PEEK (DPTR) + 256 *  PEEK (DPTR + 1) - 512
10170  PRINT D$;"OPEN ";FF$
10180  PRINT D$;"WRITE ";FF$
10190  PRINT GC$: PRINT DL: PRINT TS$
10195  PRINT  INT (XM / CMPULSE): PRINT  INT (YM / CMPULSE): PRINT  INT (ZM / CMPULSE)
10200  PRINT SX: PRINT SY: PRINT SZ
10210  PRINT RX: PRINT RY: PRINT RZ
10220  PRINT IX: PRINT IY: PRINT IZ
10230  PRINT D$;"CLOSE ";FF$
10250  FOR I = 0 TO  INT ((( PEEK (DPTR + 1) - 2) / 32) + .5)
10270  POKE XFER,2 + (I * 32)
10280  CALL XFER + 1
10290  IF (DL - I * 8192) < 8192 THEN LE = DL - (I * 8192): GOTO 10320
10300 LE = 8192
10320  PRINT D$;"BSAVE ";FF$;".";I;",A$2000,L";LE
10330  NEXT I
10340  VTAB 1: PRINT "                    ...DONE"
10350  GOSUB 13320
10360  RETURN
10500  REM ******************************
10510  REM *                            *
10520  REM * IMMEDIATE DATA SCAN AND    *
10530  REM * DISPLAY                    *
10540  REM *                            *
10550  REM ******************************
10590 ID = 0
10600  IF GC$ = "Y" THEN X0 = XM / CMPULSE:Y0 = YM / CMPULSE:Z0 = ZM / CMPULSE: GOTO 1062
    0
10610 X0 = 0:Y0 = 0:Z0 = 0
10620  HOME
10630  VTAB 21
10640  PRINT "INPUT X,Y,OR Z TO MOVE,"
10650  PRINT "P TO PRINT OUT, OR ANY"
10660  PRINT "OTHER KEY WHEN DONE: ";
10670  GET ID$
10675  PRINT
10690  IF ID$ = "X" THEN  GOSUB 12500:CD = 193:CN = CX:CS = X0:CT = XP: GOTO 11000
10700  IF ID$ = "Y" THEN  GOSUB 12500:CD = 194:CN = CY:CS = Y0:CT = YP: GOTO 11000
10710  IF ID$ = "Z" THEN  GOSUB 12500:CD = 196:CN = CZ:CS = Z0:CT = ZP: GOTO 11000
10720  IF ID$ <  > "P" THEN  TEXT : HOME : RETURN
10730  IF ID = 0 THEN  PRINT "MUST MOVE BEFORE PRINTING": GOSUB 13320: GOTO 10620
10740  PRINT D$;"PR#1"
10750  PRINT IE$;"-AXIS SCAN"
10760  IF GC$ = "Y" THEN  PRINT "GEOMETRIC COORDINATES": GOTO 10780
10770  PRINT "TANK COORDINATES"
10780  PRINT "SCAN FROM ";IE$;"=";CC;" (CM)"
10790  PRINT "TO ";IE$;"=";MM * CMPULSE;" (CM)"
10800  IF IE$ = "X" THEN  PRINT "Y=";(CY - Y0) * CMPULSE;" (CM)": PRINT "Z=";(CZ - Z0) *
    CMPULSE;" (CM)": GOTO 10830
10810  IF IE$ = "Y" THEN  PRINT "X=";(CX - X0) * CMPULSE;" (CM)": PRINT "Z=";(CZ - Z0) *
    CMPULSE;" (CM)": GOTO 10830
10820  IF IE$ = "Z" THEN  PRINT "X=";(CX - X0) * CMPULSE;" (CM)": PRINT "Y=";(CY - Y0) *
    CMPULSE;" (CM)"
```

```
10830  GOSUB 12570
10835  PRINT "PEAK VALUE AT ";IE$;"=";(PP - CS) * CMPULSE;" (CM)"
10837  PRINT "WINDOW HEIGHT=";SI * 128 * CMPULSE;" (CM)"
10839  PRINT I$;"16H"
10840  PRINT  CHR$ (12)
10845  PRINT D$;"PR#0"
10850  GOTO 10620
11000  HOME : VTAB 21
11010  PRINT "CURRENT ";ID$;"-AXIS POSITION: ";
11020  PRINT (CN - CS) * CMPULSE;" (CM)"
11030  PRINT "ENTER MOVE ('RETURN' TO EXIT): ";
11040  INPUT "";MM$
11041  IF MM$ = "" THEN 10620
11042  IF  VAL (MM$) = 0 THEN 10620
11043 MM =  VAL (MM$) / CMPULSE
11050 MV = MM + CS
11060  IF MV < 0 OR MV > CT THEN  GOSUB 13000: GOTO 11000
11070  GOSUB 12400
11080  IF  ABS (MV - CN) <  = 256 THEN SI = 2: GOTO 11093
11090 SI =  INT ( ABS (MV - CN) / 128 + 0.5)
11093  POKE XYZ,255: CALL BASE: REM RESET DATPTR
11096  POKE XYZ,CD
11100  GOSUB 12600
11110 ID = 1
11115 IE$ = ID$
11120 CC = (CN - CS) * CMPULSE
11125  GOSUB 13320
11130  GOTO 10690
12000  REM ******************************
12010  REM *                            *
12020  REM * MISC. UTILITIES            *
12030  REM *                            *
12040  REM ******************************
12100  REM REAL TIME DISPLAY LEGENDS
12110  PRINT : VTAB 21
12120  PRINT "    X-AXIS       Y-AXIS       Z-AXIS    "
12130  VTAB 22
12140  PRINT "    ------- CM   ------- CM   ------- CM"
12150  VTAB 23
12170  RETURN
12200  REM HIRES INITIALIZATION & WINDOW
12250  HGR
12260  HCOLOR= 3
12270  HPLOT 70,21 TO 202,21: HPLOT 70,150 TO 202,150
12275  HPLOT 65,22 TO 65,149: HPLOT 66,22 TO 66,149: HPLOT 67,22 TO 67,149
12276  HPLOT 205,22 TO 205,149: HPLOT 206,22 TO 206,149: HPLOT 207,22 TO 207,149
12280  RETURN
12300  REM REAL TIME DISPLAY LEGENDS WITH MAX
12310  GOSUB 12100
12320  PRINT "MAX:"
12330  RETURN
12400  REM SET UP MOVE REG
12410 M2 =  INT (MV / 65536)
12420 M1 =  INT ((MV - 65536 * M2) / 256)
```

```
12430 M0 =  INT (MV) -  INT (65536 * M2 + 256 * M1)
12440  POKE MOVE,M0
12450  POKE MOVE + 1,M1
12460  POKE MOVE + 2,M2
12470  RETURN
12500  REM GET CURRENT X,Y,Z COUNTS
12510 CX =  PEEK (XCOUNT) + (256 *  PEEK (XCOUNT + 1)) + (65536 *  PEEK (XCOUNT + 2))
12520 CY =  PEEK (YCOUNT) + (256 *  PEEK (YCOUNT + 1)) + (65536 *  PEEK (YCOUNT + 2))
12530 CZ =  PEEK (ZCOUNT) + (256 *  PEEK (ZCOUNT + 1)) + (65536 *  PEEK (ZCOUNT + 2))
12540  RETURN
12550 PX =  PEEK (MAXCNT) + (256 *  PEEK (MAXCNT + 1)) + (65536 *  PEEK (MAXCNT + 2)): RETUR
N

12560 PY =  PEEK (MAXCNT) + (256 *  PEEK (MAXCNT + 1)) + (65536 *  PEEK (MAXCNT + 2)): RETUR
N

12570 PP =  PEEK (MAXCNT) + (256 *  PEEK (MAXCNT + 1)) + (65536 *  PEEK (MAXCNT + 2)): RETUR
N

12600  REM SET UP SWEEP INC. AND CALL BASE
12605  HOME
12610  GOSUB 12700
12630  GOSUB 12200: REM  INIT. HIRES & WINDOW
12640  GOSUB 12300: REM RTD. LEGENDS & MAX.
12650  PRINT "WINDOW HEIGHT= ";SI * 128 * CMPULSE;" (CM)";
12660  CALL BASE
12670  RETURN
12700  REM SET UP SWEEP INCREMENT (SI=0 DISABLES DATA COLLECTION)
12701  REM (SI)=2 IS VALID INCREMENT VALUE)
12705  IF SI = 0 THEN  POKE DINC,0: POKE DINC + 1,0: RETURN
12710  POKE DINC, INT (SI - 1) - 256 *  INT ((SI - 1) / 256)
12720  POKE DINC + 1, INT ((SI - 1) / 256)
12730  RETURN
12995  REM ******************************
12996  REM *                            *
12997  REM * ERROR MESSAGES & DELAYS    *
12998  REM *                            *
12999  REM ******************************
13000  REM OUT OF RANGE MESSAGE
13010  PRINT "OUT OF RANGE"
13020  GOSUB 13310
13030  RETURN
13100  PRINT "MUST USE 'Y' OR 'N'"
13110  GOSUB 13310
13120  RETURN
13200  PRINT "EXCEEDS BUFFER CAPACITY"
13210  GOSUB 13310
13220  RETURN
13300  FOR I = 1 TO 1000: NEXT : RETURN
13310  FOR I = 1 TO 1500: NEXT : RETURN
13320  FOR I = 1 TO 2000: NEXT : RETURN
13330  FOR I = 1 TO 3000: NEXT : RETURN

1
```

APPENDIX C

Miscellaneous Program Listings

Listing of HELLO

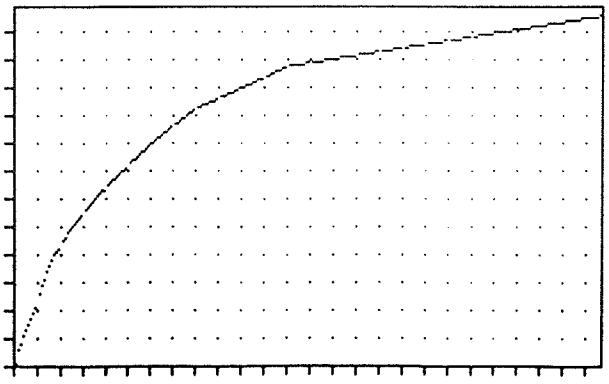
```
]LIST

10  REM ************************************
20  REM *                                  *
30  REM * ULTRASONIC FIELD SCANNER         *
40  REM * SCANNER SYSTEM HELLO PROGRAM     *
50  REM *                                  *
60  REM * COPYRIGHT 1984                   *
70  REM * DAVE PADGITT                     *
80  REM * UNIVERSITY OF ILLINOIS           *
90  REM * AND URI THERM-X INC.             *
95  REM *                                  *
99  REM ************************************
120  REM
130  REM RELOCATE BASIC TO 16385 ($4001)
150  POKE 16384,0
160  POKE 103,1
170  POKE 104,64
180  POKE 105,1
190  POKE 106,64
200  PRINT  CHR$ (4);"RUN CONTROLLER"

]
```

Data Dump and Plot of RAMP

```
1200- 00 90 8E 8B 89 87 85 83
1208- 81 7E 7C 79 77 74 72 70
1210- 6E 6D 6C 6A 69 68 66 65
1218- 64 63 62 61 60 5F 5E 5D
1220- 5C 5B 5A 59 58 57 57 56
1228- 55 54 53 53 52 51 50 50
1230- 4F 4E 4D 4C 4C 4B 4A 49
1238- 49 48 47 46 46 45 44 44
1240- 43 42 42 41 40 40 3F 3F
1248- 3E 3D 3D 3C 3B 3B 3A 3A
1250- 39 39 38 38 38 37 37 37
1258- 36 36 35 35 35 34 34 34
1260- 33 33 32 32 32 31 31 31
1268- 30 30 2F 2F 2F 2E 2E 2E
1270- 2D 2D 2C 2C 2C 2B 2B 2B
1278- 2A 2A 2A 2A 2A 2A 2A 29
1280- 29 29 29 29 29 29 29 28
1288- 28 28 28 28 28 28 27 27
1290- 27 27 27 27 27 27 26 26
1298- 26 26 26 26 26 25 25 25
12A0- 25 25 25 25 25 24 24 24
12A8- 24 24 24 24 23 23 23 23
12B0- 23 23 23 23 22 22 22 22
12B8- 22 22 22 21 21 21 21 21
12C0- 21 21 21 20 20 20 20 20
12C8- 20 20 1F 1F 1F 1F 1F 1F
12D0- 1F 1F 1E 1E 1E 1E 1E 1E
12D8- 1E 1D 1D 1D 1D 1D 1D 1D
12E0- 1D 1C 1C 1C 1C 1C 1C 1C
12E8- 1B 1B 1B 1B 1B 1B 1B 1B
12F0- 1A 1A 1A 1A 1A 1A 1A 19
12F8- 19 19 19 19 19 19 19 18
*
```



Listing of TRANSFER

```
SOURCE FILE: TRANSFER
0000:           1 ********************************
0000:           2 *                              *
0000:           3 * AUX MEMORY TO HGR1 DATA       *
0000:           4 * TRANSFER ROUTINE.            *
0000:           5 * (MODIFIED VERSION OF THE     *
0000:           6 *  HIRES PAGE MOVER FOUND IN   *
0000:           7 *  THE EXTENDED 80-COLUMN TEXT *
0000:           8 *  CARD SUPPLEMENT P43-44).    *
0000:           9 *                              *
0000:          10 * COPYRIGHT 1984               *
0000:          11 * DAVE PADGITT                 *
0000:          12 * UNIVERSITY OF ILLINOIS       *
0000:          13 * AND URI THERM-X INC.         *
0000:          14 *                              *
0000:          15 * 9/21/84                      *
0000:          16 ********************************
0000:          17 * ZERO PAGE POINTERS
0000:          18         DSECT
003C:          19         ORG  $3C
003C:          20 SRCBEG  DS   2
003E:          21 SRCEND  DS   2
0040:          22         DS   2
0042:          23 DSTBEG  DS   2
0000:          24         DEND
2000:          25 PG1BEG  EQU  $2000
C311:          26 AUXMOV  EQU  $C311
----- NEXT OBJECT FILE NAME IS TRANSFER.OBJ0
0300:          27         ORG  $300
0300:          28 PARM    DS   1         ;MUST BE FILLED WITH 2+PAGE*32 WHERE P
AGE=0 TO 5
0301:A9 00     29 XFER    LDA  #>PG1BEG  ;ENTRY POINT FOR TRANSFER ROUTINE
0303:85 42     30         STA  DSTBEG
0305:A9 20     31         LDA  #<PG1BEG
0307:85 43     32         STA  DSTBEG+1  ;DESTINATION IS HGR1
0309:A9 00     33         LDA  #0
030B:85 3C     34         STA  SRCBEG
030D:AD 00 03  35         LDA  PARM
0310:85 3D     36         STA  SRCBEG+1
0312:A9 FF     37         LDA  #$FF
0314:85 3E     38         STA  SRCEND
0316:18        39         CLC
0317:AD 00 03  40         LDA  PARM
031A:69 1F     41         ADC  #$1F
031C:85 3F     42         STA  SRCEND+1
031E:18        43         CLC            ;MOVE FROM AUX. TO MAIN MEM.
031F:20 11 C3  44         JSR  AUXMOV
0322:60        45         RTS

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

```
  3C SRCBEG       3E SRCEND       42 DSTBEG       0300 PARM
?0301 XFER        2000 PG1BEG     C311 AUXMOV
```

```
C311 AUXMOV       42 DSTBEG       0300 PARM       2000 PG1BEG
  3C SRCBEG       3E SRCEND       ?0301 XFER
```

Listing of DRIVER

```
]LIST

100  REM ********************************
110  REM *                              *
120  REM *    STEPPER MOTOR RAMPING     *
130  REM *    CURVE ENTRY AND TEST      *
140  REM *     PROGRAM (DRIVER)         *
150  REM *                              *
160  REM *    COPYRIGHT 1984            *
170  REM *    DAVE PADGITT              *
180  REM *    UNIVERSITY OF ILLINOIS    *
190  REM *    AND URI THERM-X INC.      *
200  REM *                              *
205  REM *      8/28/84                 *
210  REM ********************************
220  REM
230  REM  THIS PROGRAM USES A BIT-PAD ONE DIGITIZING TABLET IN SLOT#2 AS AN
240  REM  INPUT DEVICE FOR THE CURVE DATA ENTRY
270  REM ********************************
280  REM  INITIALIZATIONS AND PROMPTS
290  REM ********************************
300 D$ =  CHR$ (4): REM  CONTROL-D
310 I$ =  CHR$ (137): REM  CONTROL-I
320 XS = 6.7265625: REM   X SCALE FACTOR
330 YS = 5.625: REM   Y SCALE FACTOR
340 BASE = 2048: REM  ORG $0800
490  TEXT
500  HOME
510  PRINT "STEPPER MOTOR RAMPING CURVE "
520  PRINT "ENTRY AND TEST PROGRAM"
530  PRINT
540  INPUT "LOAD MACHINE LANGUAGE ROUTINES? (Y/N):";L$
550  IF L$ <  > "Y" THEN 600
560  PRINT D$;"BLOAD HISCAN.OBJ0,A$300": REM  LOAD MACHINE LANGUAGE ROUTINES
570  PRINT D$;"BLOAD STEPPER.OBJ0,A$B00"
600  HOME : VTAB 21: INVERSE
610  PRINT "LOAD A RAMPING CURVE? (Y/N):";: NORMAL
620  INPUT "";Q$
630  IF Q$ <  > "Y" THEN 700
640  HOME : VTAB 21: INVERSE
650  PRINT "RAMPING CURVE FILENAME:";: NORMAL
660  INPUT "";N$
670  PRINT D$;"BLOAD";N$;",A$1200": REM RAMP TABLE LOCATED AT $1200
680  GOSUB 5000
690  GOTO 750
700  REM ********************************
710  REM  MAIN PROGRAM
720  REM ********************************
730  REM YELLOW=1,WHITE=2,BLUE=4,GREEN=8
740  GOSUB 2000
750  PRINT "Y=CLR,W=ENT,B=REPLT/SAVE,G=STEP": GOSUB 1000: REM  INPUT DATA FROM TABLET
760  IF F = 1 THEN  GOSUB 2000: GOTO 750: REM  SCREEN SETUP SUBROUTINE
770  IF F = 2 THEN  GOSUB 3000: GOTO 750: REM  CURVE ENTRY ROUTINE
```

```
780  IF F = 4 THEN  GOSUB 4000: GOTO 750: REM  REPLOT/SAVE CURVE SUBROUTINE
790  IF F = 8 THEN  GOSUB 6000: REM  RUN STEPPER ROUTINE
800  GOTO 750: REM  EXIT PROGRAM WITH CTRL-RESET
990  END
1000  REM ********************************
1010  REM TABLET INPUT DATA ROUTINE
1020  REM ********************************
1030  PRINT "PRESS A BUTTON"
1040  PRINT D$;"IN#2": REM   TABLET IN SLOT #2
1050  INPUT "";X,Y,F
1060  PRINT D$;"IN#0": REM   BACK TO KEYBOARD INPUT
1070  RETURN
2000  REM ********************************
2010  REM  CLEAR SCREEN AND SET UP AXIS
2020  REM ********************************
2030  HGR
2040  HCOLOR= 7: REM WHITE
2050  HPLOT 12,21 TO 269,21 TO 269,150 TO 12,150 TO 12,21
2060  FOR YY = 150 TO 21 STEP  - 10: HPLOT 8,YY TO 11,YY: NEXT
2070  FOR XX = 12 TO 269 STEP 10: HPLOT XX,151 TO XX,153: NEXT
2080  RETURN
3000  REM ********************************
3010  REM  CURVE ENTRY SUBROUTINE
3020  REM ********************************
3025  PRINT D$;"IN#2": REM  TABLET IS SIN SLOT #2
3030  HOME : VTAB 21: FLASH
3040  PRINT "ENTER ORIGIN WITH YELLOW BUTTON": NORMAL
3050  GOSUB 3400
3060  IF F <  > 1 THEN 3300
3070 ZX = X:ZY = Y
3080  PRINT "ORIGIN ENTERED"
3100  REM GET FIRST POINT
3110  PRINT "ENTER POINTS WITH YELLOW BUTTON"
3120  GOSUB 3400
3130  IF F <  > 1 THEN 3300
3140 LX =  INT (13 + (X - ZX) / XS)
3150 LY =  INT (150 - (Y - ZY) / YS)
3160  IF LX < 0 OR LX > 279 OR LY < 22 OR LY > 149 THEN 3120: REM  BOX BOUNDARY
3170  PRINT "ENTER POINTS WITH YELLOW BUTTON"
3180  GOSUB 3400
3190  IF F <  > 1 THEN 3300
3200 NX =  INT (13 + (X - ZX) / XS)
3210 NY =  INT (150 - (Y - ZY) / YS)
3220  IF NX < 0 OR NX > 279 OR NY < 22 OR NY > 149 THEN 3180: REM  BOX BOUNDARY
3230  HPLOT LX,LY TO NX,NY: REM  DRAW LINE
3240 LX = NX: REM  UPDATE LASTX
3250 LY = NY: REM  UPDATE LASTY
3260  GOTO 3170
3300  PRINT D$;"PR#0": REM RETURN INPUT TO KEYBOARD
3310  RETURN
3400  REM POINT ENTRY ROUTINE
3410  INPUT "";X,Y,F
3420  RETURN
4000  REM ********************************
```

```
4010  REM REPLOT/SAVE CURVE SUBROUTINE
4020  REM ******************************
4030  INVERSE : PRINT "YELLOW TO REPLOT, BLUE TO SAVE": NORMAL
4040  GOSUB 1000
4050  IF F = 4 THEN 4200
4055  IF F <  > 1 THEN 4400
4060  GOSUB 2000: GOSUB 5050: REM  PLOT W/O DOTS
4070  GOTO 4300
4200  REM HISCAN CONTROL
4210  PRINT "DIGITIZING..."
4220  POKE 771,18: REM CURVE AT $1200
4230  CALL 768: REM  HISCAN.OBJ0
4240  GOSUB 5000
4300  INVERSE : PRINT "SAVE RAMP CURVE TO DISK? (Y/N):";: NORMAL
4310  INPUT "";SA$
4320  IF SA$ <  > "Y" THEN 4400
4330  INVERSE : PRINT "FILENAME:";: NORMAL
4340  INPUT "";N$
4360  IF  LEN (N$) > 31 THEN 4330
4370  PRINT D$;"BSAVE";N$;",A$1200,L$100"
4400  INVERSE : PRINT "PRINT OUT CURVE? (Y/N):";: NORMAL
4410  INPUT "";P$
4420  IF P$ <  > "Y" THEN 4500
4430  GOSUB 5000
4440  PRINT D$;"PR#1": REM PRINTER IN SLOT #1
4450  PRINT I$;"17H"
4460  PRINT D$;"PR#0"
4500  GOSUB 2000
4510  RETURN
5000  REM ******************************
5010  REM  PLOT BUFFER ROUTINE
5020  REM ******************************
5025  GOSUB 2000
5030  FOR YY = 150 TO 21 STEP  - 10
5035  FOR XX = 12 TO 269 STEP 10
5040  HPLOT XX,YY
5045  NEXT : NEXT
5050  REM  ENTRY POINT FOR REPLOT W/O DOTS
5060 K = BASE + 256 * 10 - 13: REM BUFFER AT $1200
5070  FOR I = 13 TO 268
5080 J =  PEEK (K + I)
5090  HPLOT I,J
5100  NEXT
5110  RETURN
6000  REM ******************************
6010  REM  STEPPER CONTROL SUBROUTINE
6020  REM ******************************
6025  REM  CONTROLS X-AXIS STEPPER ONLY
6027  HOME
6030  REM STEPPER CARD IN SLOT 4
6040  POKE 50195,0: REM DDRA2
6050  POKE 50194,63: REM DDRB2
6055  POKE 50179,240: REM DDRA1
6060  POKE BASE + 22,0: POKE BASE + 23,16: POKE BASE + 24,0: REM START X REG AT $4096
```

```
6065  POKE BASE + 10,80: POKE BASE + 11,0: POKE BASE + 12,0: POKE BASE + 13,0: REM SET GC
    X TO EXCESS-500 CM
6070  VTAB 23: PRINT "                                        ": VTAB 23: PRINT "INPUT ABS
    OLUTE POSITION:";: INPUT "";N
6080 N =  ABS ( INT (N))
6090 N2 =  INT (N / 65536)
6100 N1 =  INT ((N - N2 * 65536) / 256)
6110 N0 =  INT (N - (N1 * 256 + N2 * 65536))
6130  REM MOVE REG.
6140  POKE BASE + 4,N0: POKE BASE + 5,N1: POKE BASE + 6,N2
6150  POKE BASE + 3,65: REM XYZ REG.
6155  POKE BASE + 7,0: POKE BASE + 8,0
6160  CALL BASE: REM CALL STEPPER DRIVER MACHINE LANG. PROG.
6170  VTAB 23: PRINT "                                        ": VTAB 23: PRINT "CONTINUE?
     (Y/N):";: INPUT "";C$
6180  IF C$ <  > "N" THEN 6070
6200  RETURN

]
```

Listing of HISCAN

```
SOURCE FILE: HISCAN
0000:            1 ******************************
0000:            2 *                            *
0000:            3 *    HIRES SCREEN SCAN       *
0000:            4 *                            *
0000:            5 * THIS PROGRAM SCANS THE     *
0000:            6 * HIRES SCREEN AND STORES    *
0000:            7 * GRAPHICAL CURVE DATA IN    *
0000:            8 * TABLE FORM IN RAM          *
0000:            9 *                            *
0000:           10 *      DAVE PADGITT          *
0000:           11 *        2/18/84             *
0000:           12 *                            *
0000:           13 ******************************
0000:           14 *
0006:           15 SCRPTR  EQU  $06
0008:           16 TBLPTR  EQU  $08
0000:           17 *
----- NEXT OBJECT FILE NAME IS HISCAN.OBJ0
0300:           18         ORG  $0300
0300:           19 *
0300:4C 06 03   20         JMP  BEGIN
0303:           21 TEMP    DS   1          ;POKE TEMP CURVE HIGH BYTE OF ADDR.
HERE
0304:           22 XCRD    DS   1
0305:           23 YCRD    DS   1
0306:           24 *
0306:A5 06      25 BEGIN   LDA  $06        ;SAVE ZERO PAGE LOCATIONS
0308:48         26         PHA
0309:A5 07      27         LDA  $07
030B:48         28         PHA
030C:A5 08      29         LDA  $08
030E:48         30         PHA
030F:A5 09      31         LDA  $09
0311:48         32         PHA
0312:           33 *
0312:A0 00      34         LDY  #$00
0314:8C 04 03   35         STY  XCRD       ;CLEAR HCRD
0317:84 08      36         STY  TBLPTR     ;CLEAR LOW BYTE
0319:A2 20      37         LDX  #$20       ;SET MASK TO 0010000
031B:AD 03 03   38         LDA  TEMP
031E:85 09      39         STA  TBLPTR+1
0320:A9 16      40 HLOOP   LDA  #22        ;22 DECIMAL
0322:8D 05 03   41         STA  YCRD
0325:20 61 03   42 VLOOP   JSR  SETPTR
0328:8A         43         TXA             ; PUT MASK IN ACC
0329:31 06      44         AND  (SCRPTR),Y ;AND PIXEL WITH MASK
032B:D0 07      45         BNE  VCONT
032D:EE 05 03   46         INC  YCRD
0330:C0 96      47         CPY  #150
0332:D0 F1      48         BNE  VLOOP
0334:           49 *
```

```
0334:98         50 VCONT   TYA
0335:48         51         PHA            ; STORE Y
0336:AC 04 03   52         LDY  XCRD
0339:AD 05 03   53         LDA  YCRD
033C:91 08      54         STA  (TBLPTR),Y
033E:68         55         PLA
033F:A8         56         TAY            ;RESTORE Y
0340:EE 04 03   57         INC  XCRD
0343:F0 0F      58         BEQ  EXIT
0345:8A         59         TXA
0346:C9 40      60         CMP  #$40      ; MASK BIT 7 SET?
0348:D0 03      61         BNE  SLMASK
034A:C8         62         INY            ; INCR COL. POINTER
034B:18         63         CLC
034C:0A         64         ASL  A
034D:0A         65 SLMASK  ASL  A         ; SLIDE MASK
034E:69 00      66         ADC#$00
0350:AA         67         TAX            ; NEW MASK IN X
0351:18         68         CLC
0352:90 CC      69         BCC  HLOOP
0354:           70 *
0354:68         71 EXIT    PLA            ; RESTORE ZERO PAGE
0355:85 09      72         STA  $09
0357:68         73         PLA
0358:85 08      74         STA  $08
035A:68         75         PLA
035B:85 07      76         STA  $07
035D:68         77         PLA
035E:85 06      78         STA  $06
0360:           79 *
0360:60         80         RTS
0361:           81 *
0361:AD 05 03   82 SETPTR  LDA  YCRD
0364:48         83         PHA
0365:29 C0      84         AND  #$C0
0367:85 06      85         STA  SCRPTR
0369:4A         86         LSR  A
036A:4A         87         LSR  A
036B:05 06      88         ORA  SCRPTR
036D:85 06      89         STA  SCRPTR
036F:68         90         PLA
0370:85 07      91         STA  SCRPTR+1
0372:0A         92         ASL  A
0373:0A         93         ASL  A
0374:0A         94         ASL  A
0375:26 07      95         ROL  SCRPTR+1
0377:0A         96         ASL  A
0378:26 07      97         ROL  SCRPTR+1
037A:0A         98         ASL  A
037B:66 06      99         ROR  SCRPTR
037D:A5 07     100         LDA  SCRPTR+1
037F:29 1F     101         AND  #$1F
0381:09 20     102         ORA  #$20
0383:85 07     103         STA  SCRPTR+1
```

```
0385:60        104         RTS

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

0306 BEGIN	0354 EXIT	0320 HLOOP	06 SCRPTR
0361 SETPTR	034D SLMASK	08 TBLPTR	0303 TEMP
0334 VCONT	0325 VLOOP	0304 XCRD	0305 YCRD

06 SCRPTR	08 TBLPTR	0303 TEMP	0304 XCRD
0305 YCRD	0306 BEGIN	0320 HLOOP	0325 VLOOP
0334 VCONT	034D SLMASK	0354 EXIT	0361 SETPTR

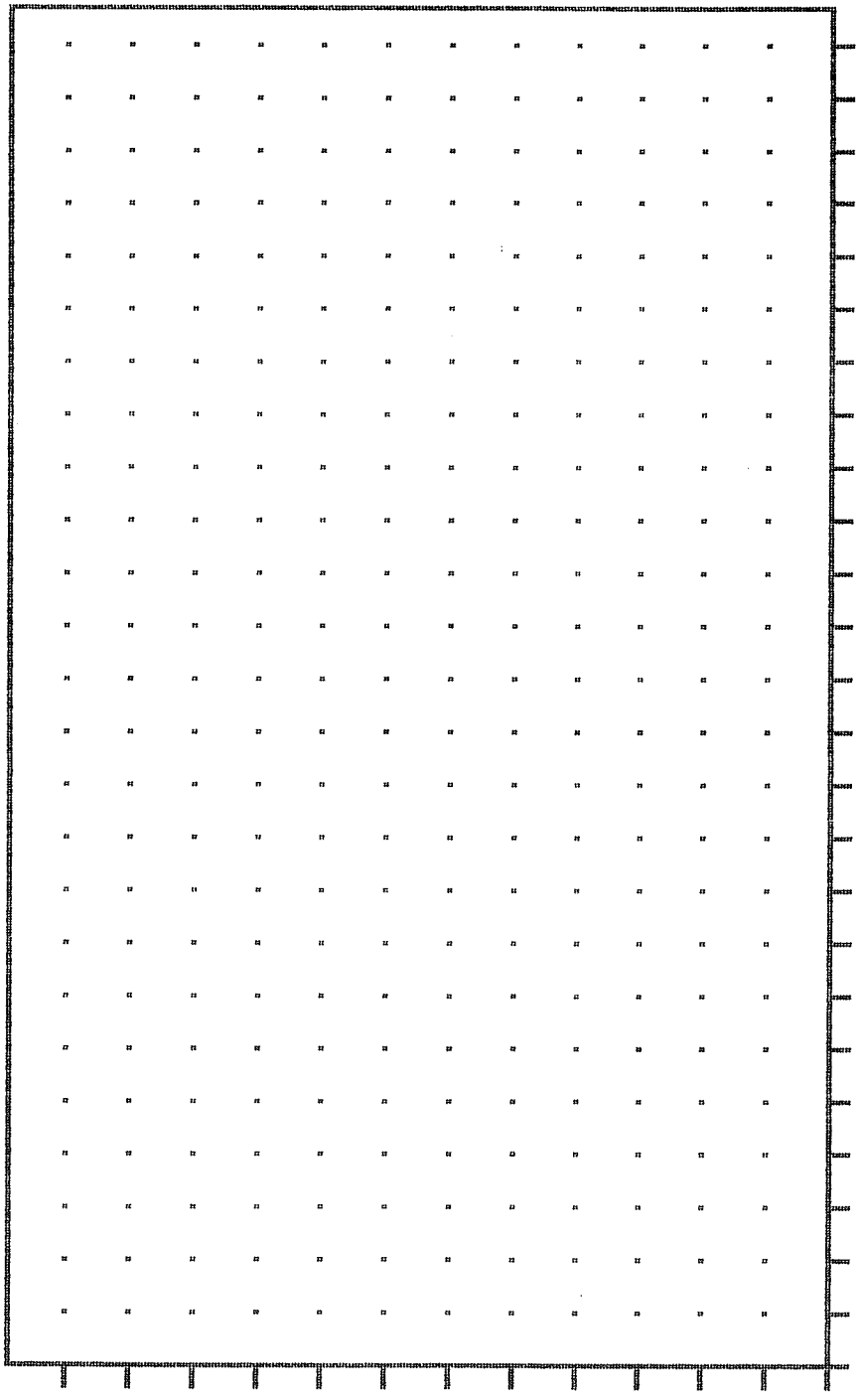
Listing of TEMPLATE

```
]LIST

 10  REM ********************************
 20  REM *                              *
 30  REM *  RAMP CURVE AXIS TEMPLATE    *
 40  REM *                              *
 50  REM *  COPYRIGHT 1984              *
 60  REM *  DAVE PADGITT                *
 70  REM *  UNIVERSITY OF ILLINOIS      *
 80  REM *  AND URI THERM-X INC.        *
 90  REM *                              *
 100  REM ********************************
 110 D$ =  CHR$ (4): REM  CONTROL-D
 120 I$ =  CHR$ (137): REM   CONTROL-I
 130  REM  THIS PROGRAM DISPLAYS A RAMP CURVE
 140  REM  AXIS TEMPLATE ON THE HIRES SCREEN
 150  REM  AND OUTPUTS IT TO THE PRINTER.
 160  REM  THIS TEMPLATE IS SCALED SO THAT IT
 170  REM  MAY BE USED DIRECTLY ON THE BIT
 180  REM  PAD ONE DIGITIZING TABLET.
 190  HOME
 200  HGR
 210  HPLOT 12,21 TO 269,21 TO 269,150 TO 12,150 TO 12,21
 220  FOR Y = 150 TO 21 STEP  - 10
 230  HPLOT 8,Y TO 11,Y
 240  NEXT Y
 250  FOR X = 12 TO 269 STEP 10
 260  HPLOT X,151 TO X,153
 270  NEXT X
 300  FOR Y = 150 TO 21 STEP  - 10
 310  FOR X = 12 TO 269 STEP 10
 320  HPLOT X,Y
 330  NEXT X
 340  NEXT Y
 400  PRINT D$;"PR#1"
 410  PRINT "RAMP CURVE AXIS TEMPLATE"
 420  PRINT
 440  PRINT
 460  PRINT I$;"17H"
 470  PRINT D$;"PR#0"
 480  TEXT
 500  END

]
```

Sample TEMPLATE Printout



APPENDIX D

TESTPLOT Listing

```
]LIST

10  REM ********************************
20  REM *                              *
30  REM * SCANNER SYSTEM DATA STORAGE  *
40  REM * TEST PROGRAM                 *
50  REM *                              *
60  REM * COPYRIGHT 1984               *
65  REM * DAVE PADGITT                 *
70  REM * UNIVERSITY OF ILLINOIS       *
75  REM * AND URI THERM-X INC.         *
80  REM *                              *
85  REM * 11/11/84                     *
90  REM *                              *
95  REM ********************************
100 D$ =  CHR$ (4)
110 CMPULSE = 0.00127
200  HOME
210  INPUT "FILENAME: ";F$
220  IF  LEN (F$) > 30 THEN 200
250  PRINT D$;"OPEN";F$
260  PRINT D$;"READ";F$
270  INPUT GC$,LD,TS$,GX,GY,GZ,SX,SY,SZ
280  INPUT RX,RY,RZ,IX,IY,IZ
290  PRINT D$;"CLOSE";F$
295  PRINT D$;"PR#1"
297  PRINT "FILENAME: ";F$
300  IF GC$ = "Y" THEN  PRINT "GEOMETRIC COORDINATES": GOTO 320
310  PRINT "TANK COORDINATES"
320  PRINT LD / 2;" DATA SAMPLES (";LD;" BYTES)"
330  IF TS$ = "Y" THEN  PRINT "TRANSVERSE SCAN": GOTO 350
340  PRINT "LONGITUDINAL SCAN"
350  PRINT
355  PRINT "CENTER OF SCAN COORDINATES: "
360  PRINT "X=";(SX - GX) * CMPULSE
370  PRINT "Y=";(SY - GY) * CMPULSE
380  PRINT "Z=";(SZ - GZ) * CMPULSE
390  PRINT
400  PRINT "SCAN RANGES: "
410  PRINT "X RANGE=";RX * CMPULSE
420  PRINT "Y RANGE=";RY * CMPULSE
430  PRINT "Z RANGE=";RZ * CMPULSE
440  PRINT
450  PRINT "SCAN INCREMENTS: "
460  PRINT "X INCREMENT=";IX * CMPULSE
470  PRINT "Y INCREMENT=";IY * CMPULSE
480  PRINT "Z INCREMENT=";IZ * CMPULSE
500  PRINT D$;"BLOAD";F$;".0,A$4000"
503 MU = 1
505 EX = LD
510 EX = EX / 2
515 MU = 2 * MU
520  IF EX > 279 THEN 510
```

```
530  HGR
540  HPLOT 0,128 -  PEEK (16385) / 2
550  FOR X = 1 TO LD / MU
560  HPLOT  TO X,128 -  PEEK (16385 + X * MU) / 2
570  NEXT
590  PRINT D$;"PR#1"
600  PRINT  CHR$ (137);"OH"
605  PRINT  CHR$ (12)
610  PRINT D$;"PR#0"
620  TEXT
1000  END

]
```

APPENDIX E

Manufacturers' Publications Referenced

Videx PSIO Dual Function Interface Card Installation and Operation Manual, Videx, Inc., 897 N.W. Grant Avenue, Corvallis, OR 97330.

Rockwell R6522 Data Sheet (Document #29000D47), Rockwell International, Midwest Regional Sales Office, 1011 E. Touhy Avenue, Suite 245, Des Plaines, IL 60018.

Analog Devices AD572 Data Sheet, Analog Devices, Route One Industrial Park, P. O. Box 280, Norwood, MA 02062.

Motorola Linear Integrated Circuits Data Book (Pages 6-51 to 6-59), Motorola Semiconductor Products Inc., Box 20912, Phoenix, AZ 85036.

REFERENCES

1. G. M. Hahn, Hyperthermia for the engineer: A short biological primer. IEEE Trans. Biomed. Eng. BME-31, 3, 1984.
2. F. W. Kremkau, Cancer therapy with ultrasound: A historical review. J. Clin. Ultrasound 7, 287-300, 1979.
3. J. Robert Stewart, Malcolm A. Bagshwa, Peter M. Corry, Eugene W. Gerner, Frederic A Gibbs, Jr., George M. Hahn, Pademaker P. Lele, and James R. Oleson, Hyperthermia as a Treatment of Cancer. Cancer Treatment Symposia 1, 135, 1984.
4. IEEE Transactions on Biomedical Engineering, Special Issue on Hyperthermia and Cancer Therapy, BME-31, 1984.
5. IEEE Tranacations on Biomedical Engineering, Special Issue on Ultrasound Hyperthermia, SU-31, 1984.