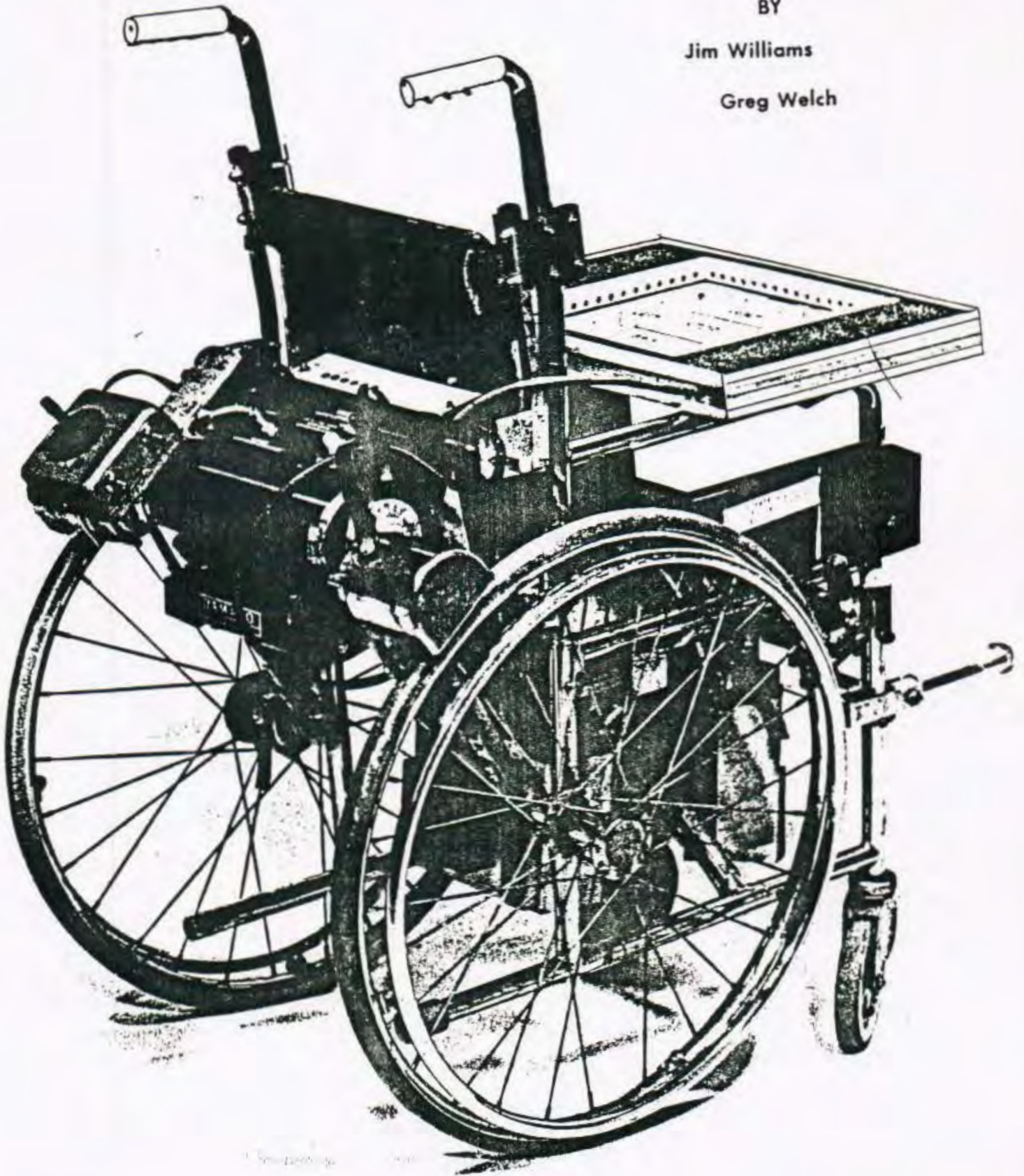


THE EASY CHAIR

BY

Jim Williams

Greg Welch



THE EASY CHAIR

James P. Williams - Gregory F. Welch

May 5, 1986

WHAT IS IT?

- Microprocessor controlled wheelchair
- Aid to handicapped (cerebral palsy victims)
- Learning tool for children with inexperience in mobility
- Effective means of introduction to powered mobility
- Funded by The Wabash Center (for handicapped children) in West Lafayette, Indiana

SPECIFICATIONS

- Must add safeguards to powered mobility
- Must introduce a "force-free" method of input
- Should be removable without defacing the wheelchair
- Should be adaptable as child develops motor skills

DESCRIPTIONS

- Overall block diagram
 - 6 > Touch pad - *brood probes etc.*
 - J > Ultrasonics
 - 6 > Computer - *New (re-written) monitor*
 - J > Tone generator
 - 6 > Motor control - *SA (4 bit) zero span*
 - J > Power supply

TEST RESULTS

- 6 - Touch Pad & Ultrasonics
 - > Verified hardware operation (general)
 - > Used software test routines
- 6 - Motor control
 - > Monitored with oscilloscope
 - = Initial design produced incorrect references
 - = Modified to better meet specifications, and allow for offset & range adjustments
- J - Power supply
 - > Monitored battery voltage with storage scope
 - = Developed plots and determined that current design was sufficient
 - = Regulator moved outside of enclosure for reduced temperature

TIME ACTION PLAN

- 6 - Major portion completed on time (or ahead of schedule)

COST

- J - Development cost slightly exceeded projected cost (due to miscellaneous development expenses)

THE EASY CHAIR

James P. Williams - Gregory F. Welch

May 5, 1986

WHAT IS IT?

- Microprocessor controlled wheelchair
- Aid to handicapped (cerebral palsy victims)
- Learning tool for children with inexperience in mobility
- Effective means of introduction to powered mobility
- Funded by The Wabash Center (for handicapped children) in West Lafayette, Indiana

SPECIFICATIONS

- Must add safeguards to powered mobility
- Must introduce a "force-free" method of input
- Should be removable without defacing the wheelchair
- Should be adaptable as child develops motor skills

DESCRIPTIONS

- Overall block diagram
 - > Touch pad
 - > Ultrasonics
 - > Computer
 - > Tone generator
 - > Motor control
 - > Power supply

TEST RESULTS

- Touch Pad & Ultrasonics
 - > Verified hardware operation (general)
 - > Used software test routines
- Motor control
 - > Monitored with oscilloscope
 - = Initial design produced incorrect references
 - = Modified to better meet specifications, and allow for offset & range adjustments
- Power supply
 - > Monitored battery voltage with storage scope
 - = Developed plots and determined that current design was sufficient
 - = Regulator moved outside of enclosure for reduced temperature

TIME ACTION PLAN

- Major portion completed on time (or ahead of schedule)

COST

- Development cost slightly exceeded projected cost (due to miscellaneous development expenses)

TABLE OF CONTENTS

TOPIC	PAGE
FIGURE LIST	ii
ABSTRACT	1
INTRODUCTION	2
THE INFRARED TOUCH PAD	
Specifications	5
Block Diagram	7
Detailed Block Descriptions	
The Row Decoding Block	7
The Column Decoding Block	8
The Extra Decoding Block	8
The Touch Pad Block	9
The Row/column Detect Block	10
The Menu Select Detect Block	11
General Discussion	13
THE ULTRASONIC RANGING SYSTEM	
Specifications	14
Block Diagram	16
Detailed Block Descriptions	
The Directional Transducer Block	16
The Tone Generator Block	18
The Additional PIA and Timer Blocks	18
General Discussion	19
COMPUTER AND MOTOR CONTROL	
Specifications	20
Block Diagram	21
Detailed Block Descriptions	
The Computer Block	21
The Motor Control Block	22
General Discussion	23
CONCLUSION	24
Appendix A: EXTRA FIGURES	25
Appendix B: SOFTWARE OUTLINES	40
Appendix C: SOFTWARE LISTING	53
Appendix D: COSTING	107
Appendix E: BIBLIOGRAPHY	108

FIGURE LIST

FIGURE	DESCRIPTION	PAGE
THE EASY CHAIR		
1.0	The Easy Chair	1
1.1	The Original Electric Wheelchair	2
1.2	Added Devices	3
1.3	The Easy Chair Block Diagram	4
THE INFRARED TOUCH PAD		
2.0	The Infrared Touch Pad	6
2.1	Touch Pad Block Diagram	7
2.2	Touch Pad Control Word Diagram	8
2.3	Sample LED/Transistor Circuitry	10
2.4.0	Touch Pad Schematic	25
2.4.1	(Touch Pad Schematic)	26
2.5	Printed Circuit Board and Components	12
2.6	Pictorial with cut-away	27
THE ULTRASONIC RANGING SYSTEM		
3.0	Ranging Module	15
3.1	Ultrasonic System Block Diagram	16
3.2	Ranging Module Schematic	28
3.3	Timing Diagram	17
3.4	Tone Generator Schematic	29
3.5	Additional Parallel Group Schematic	30
COMPUTER AND MOTOR CONTROL		
4.0	SCCS-85 Single Board Computer	22
4.1	Block Diagram	21
4.2	CPU Schematic	31
4.3	Memory Schematic	32
4.4	Timer Group, I/O Addressing Group Schematic	33
4.5	Serial Group Schematic	34
4.6	Parallel Group, Interrupt Group Schematic	35
4.7	Bus Connector Schematic	36
4.8	Power Supply Schematic	37
4.9	Motor Control Schematic	38
4.10	Status LED's and Control Switches	23
4.11	Connectors and Jacks	39

ABSTRACT

The Easy Chair is a microprocessor controlled electric wheelchair for small children with muscular disorders.

Because of the unique methods of control, this special wheelchair can be used by children with both limited muscular dexterity and strength.

Also, because of several safeguards incorporated into the design, even children with limited experience in mobility can operate the powerful wheelchair safely.

The following report details the design and theory of The Easy Chair. It is assumed that the reader has some degree of knowledge in the field of electrical engineering.

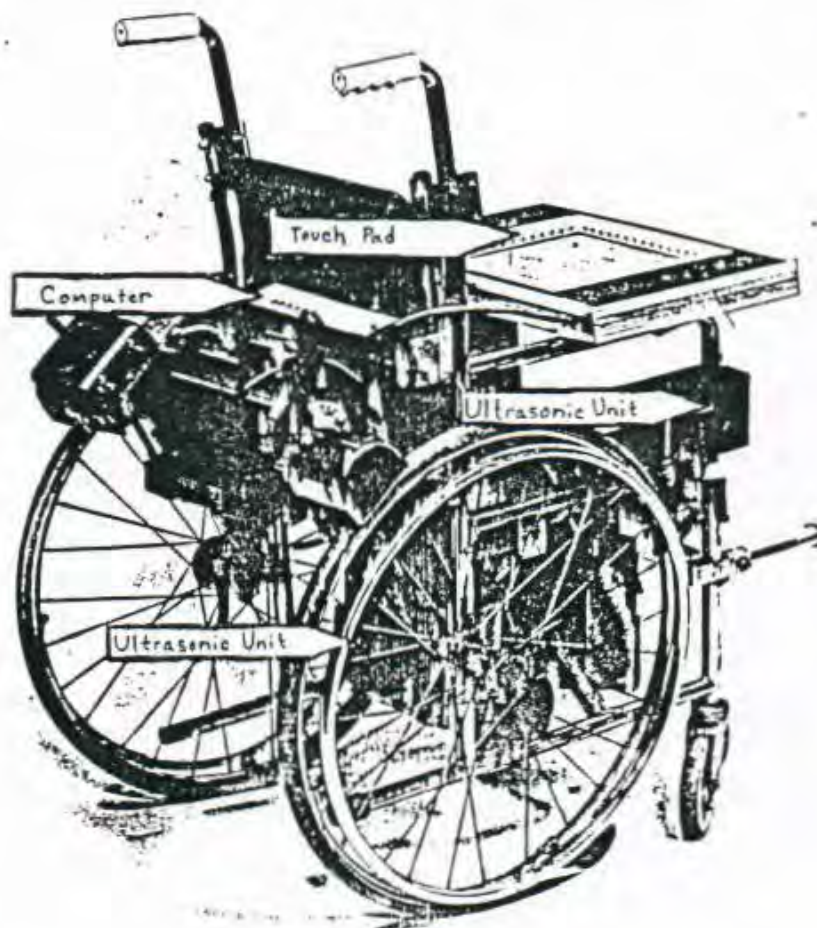


Figure 1.0 The Easy Chair

INTRODUCTION

The development of The Easy Chair is a very significant advancement for three main reasons. First of all, for many years small children with muscular disorders have had severely limited opportunities to acquire any experience in mobility. Secondly, this lack of mobility limited the opportunities to initiate communication with others. Thirdly, this lack of communication limited further their learning capabilities.

The original idea for such a wheelchair belongs to Professor George Karlin of the Special Education department at Purdue University. Professor Karlin first conceived such a device while working with cerebral palsy victims at The Wabash Center, Lafayette, Indiana. George Karlin also acted as an intermediary between the designers and the physical therapists at the center.

The idea behind a microprocessor controlled wheelchair (The Easy Chair) is to provide a safe mode of transportation for young children with muscular disorders such as cerebral palsy. Because the users will be so young, typically two to six years old, the chair was envisioned as being equipped with a variety of special devices. These devices would not only allow them to control wheelchair movement with only limited muscular force, but will also protect them from any undesirable circumstances.

The original electric wheelchair comes equipped with a Damaco DBB Add-On power unit. This unit includes batteries, the drive units (motors and controllers), and a proportional joystick controller.

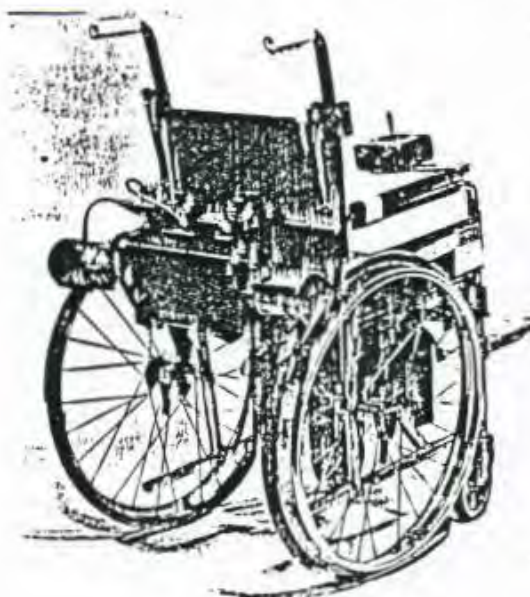


Figure 1.1 The Original Electric Wheelchair

The Easy Chair consists of this original wheelchair, with the addition of three extra devices:

- (1) An infrared touch pad
- (2) An ultrasonic ranging system
- (3) A computer control system

These three additional devices not only make operation by handicapped children more feasible, but they also give the wheelchair an added measure of control and safety.

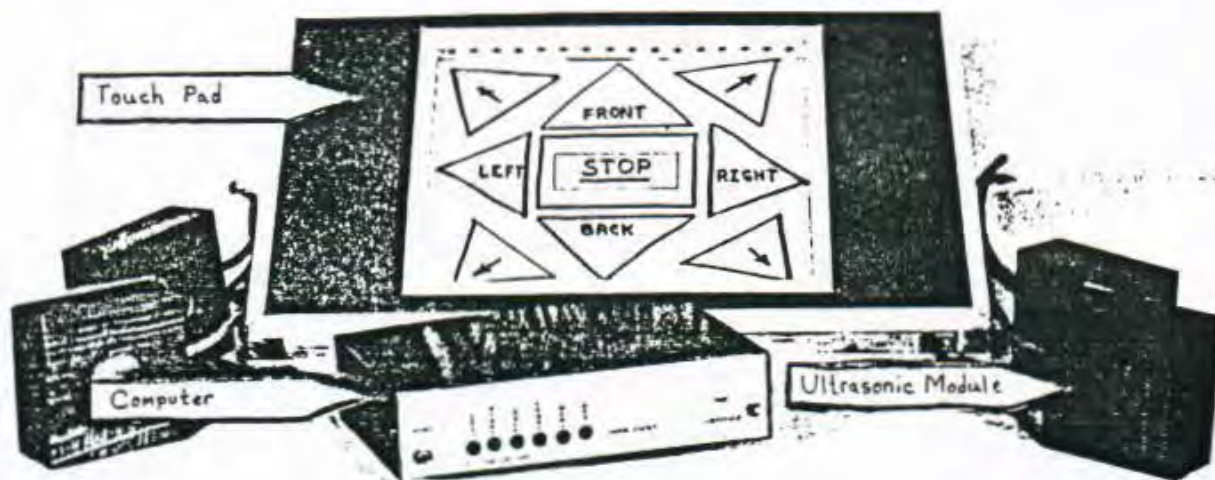


Figure 1.2 Added Devices

Shown below is a general block diagram for The Easy Chair which should give the reader an overall idea of how the different devices interact.

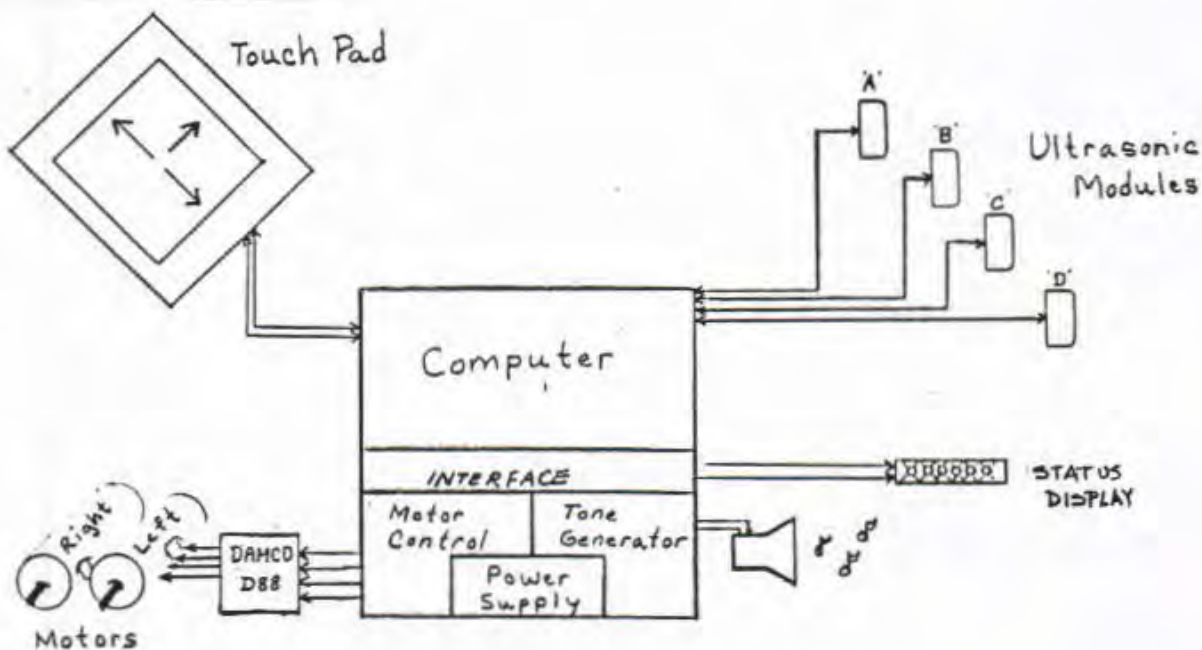


Figure 1.3 The Easy Chair Block Diagram

This report begins with the discussion of the infrared touch pad, including thoughts about why such a device was chosen. Then it explores the design and theory of the ultrasonic ranging system. Finally, it addresses the computer control system, along with the circuitry required to control the original wheelchair.

THE INFRARED TOUCH PAD

SPECIFICATIONS

The infrared touch pad is to be known as the input system for the control of the chair. It is thought of as the only real-time method of input to the computer control system. Therefore, it must meet several requirements which will allow it to be used to alter the current system configurations, or just to control the chair.

Specifications for the Easy Chair were outlined by an Occupational Therapist, Physical Therapist, and a classroom teacher from The Wabash Center in Lafayette, Indiana. This outlining was assisted by George Karlin, Special Education project coordinator at Purdue University, Lafayette, Indiana.

- 1) It was determined that a touch sensitive input surface requiring minimal pressure would best suit the needs of the small children. The system needed to be adaptable to different children, some of whom are incapable of generating high response force.
- 2) The touch-pad should use a common medium for set-up, to increase the independance of the system and its users. This is to say that it should be possible to simply plug in or unplug the touch-pad, and to switch between the pad and the current joystick with little or no effort.
- 3) It should be totally self-contained as a unit, electronics and all. Again, this would increase the independance of the system.
- 4) The touch-pad should be constructed in such a way that it could be attached to the current center off-set mounting arm of the wheelchair (which swings out of the way of the user), with the option of resting on the lap tray of the chair. These two methods will result in the touch-pad being as ambidextrous as possible.
- 5) The unit should be large enough to be easily viewed and touched, but small enough so as not to be obtrusive to the user and the wheelchair. A general touch-pad area of ten inches by ten inches was set for initial dimensions.
- 6) The size and locations of the symbols on the touch-pad (used to control the wheelchair) must be programmable. This will accomodate different ranges of motion.

- 7) The touch pad must be moisture proof. Children with such handicaps as cerebral palsy frequently have oral motor problems which result in excessive drooling. Any reasonable amount of moisture should not cause the wheelchair to malfunction.

In the early design stages, it had been thought that a total hardware solution was the most reliable and consistent solution to the problems presented for a touch pad. However after carefully studying that route, and testing the results, it was determined that a combination of approximately equal amounts of hardware and software would allow the most flexible design. The following sections describe the present solution, and how it is implemented.

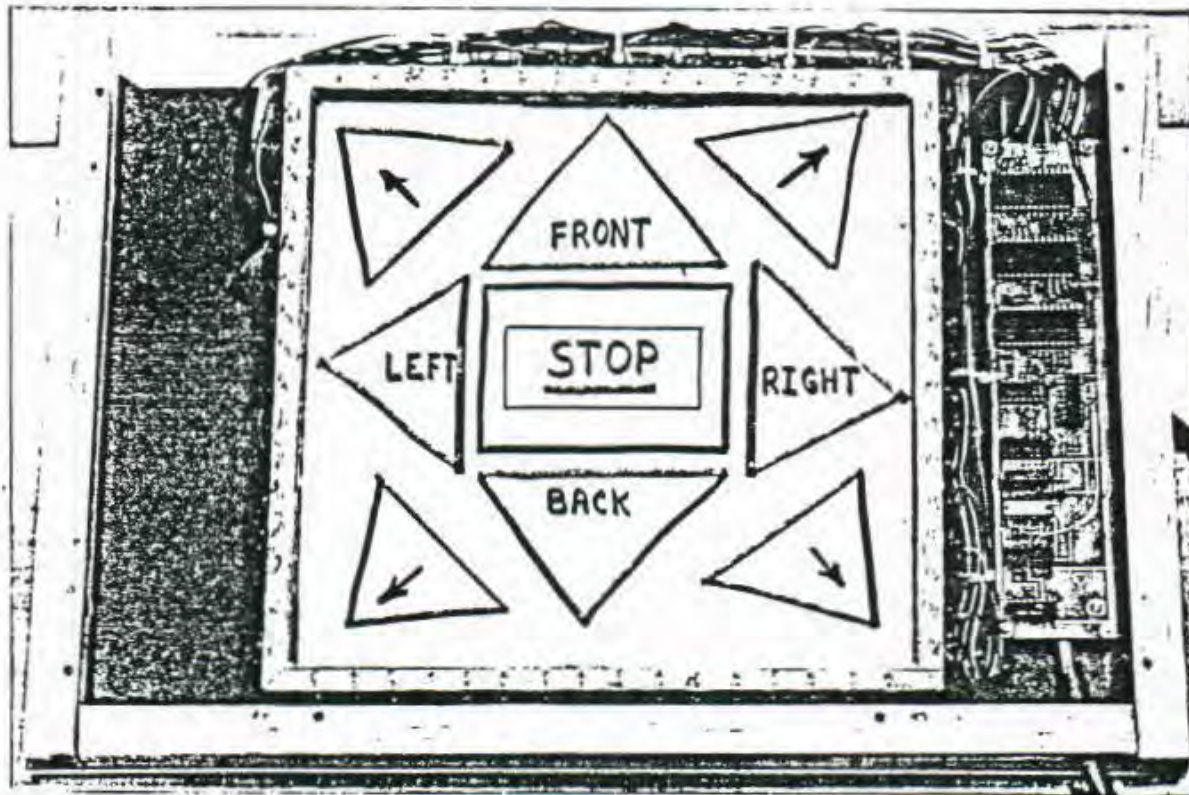


Figure 2.0 The Infrared Touch Pad

BLOCK DIAGRAM

The block diagram for the touch pad is shown below. It consists of six main blocks which include the row decoding (selecting) block, the column decoding block, the extra decoding block (which includes the menu-select decoding), the touch-pad block, the row/column detect block, and the menu-select detect block. Each of these blocks will be discussed in greater detail in the following sections. (See also Figure 2.4 Schematics)

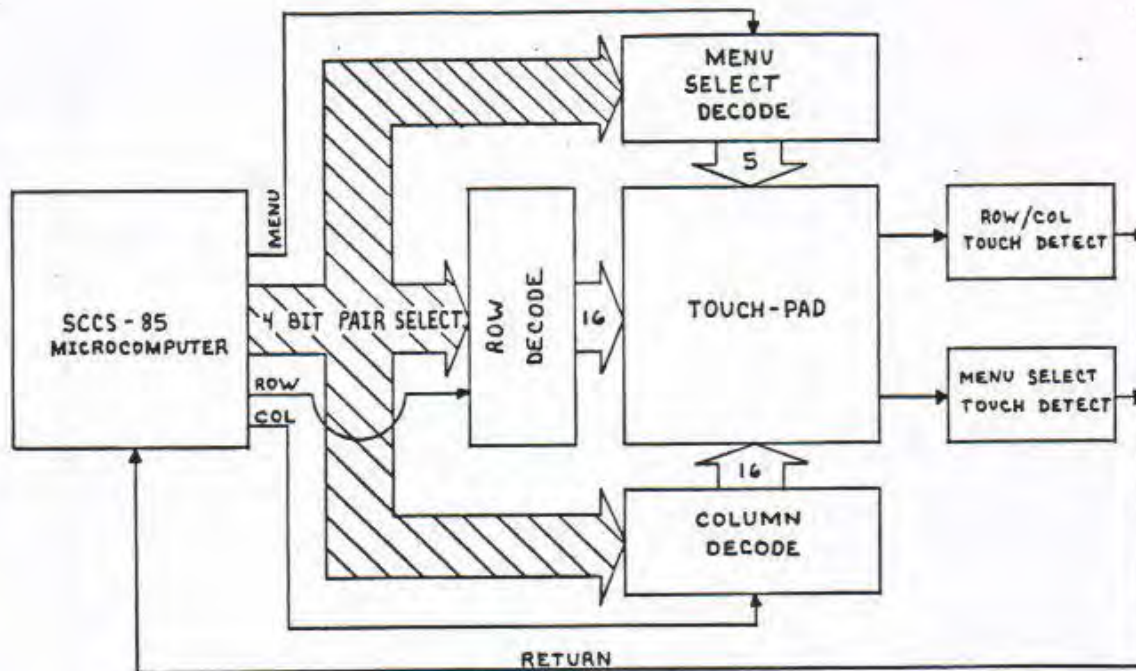


Figure 2.1 Touch Pad Block Diagram

I. THE ROW DECODING BLOCK

The row decoding block is one such block where the seven bit control word which is sent to the touch-pad circuitry is interpreted to select a certain LED/phototransistor pair.

The decoding is accomplished by sending the lower four bits of the seven bit touch-pad word to the pad. This nibble gives a zero through fifteen (F Hex) count which is used to select one of the sixteen *row*, *column* or *extra* LEDs. Then by using the upper three bits, one of three chip select lines is brought high.

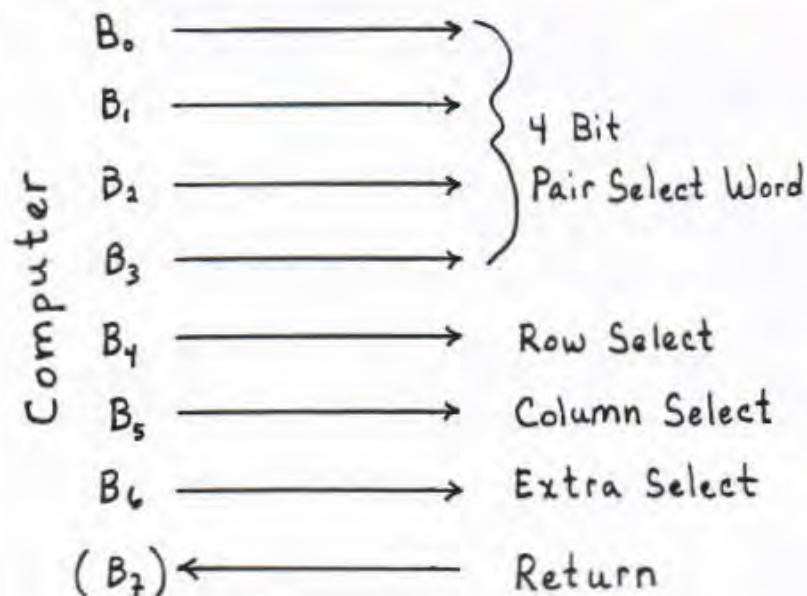


Figure 2.2 Touch Pad Control Word Diagram

To accomplish this, a 74154 4 to 16 line decoder is used. The outputs of this 74154 are low when they are selected, so they are used to provide a ground path for the infrared LEDs and phototransistors, thus allowing them to be turned on only when they are selected.

It is appropriate at this time to accent the fact that the select lines are used to select both an LED and a phototransistor. With this scheme, if there is nothing blocking the beam path from the LED to the phototransistor, then the phototransistor should be turned on.

II. THE COLUMN DECODING BLOCK

The column decoding block functions in almost the same fashion as the row decoding block. The only difference is that of the select line which is used to select the column decoding chip. Of the three select lines (which correspond to the upper three bits of the touch-pad word), one is used to select the row decoding chip, one the column decoding chip, and one the extra decoding chip. The select lines use a 'positive logic', so for instance to select the column pairs, the column select bit must be high (+5 volts).

Again, in the same fashion as the row decoding, this block selects certain LED/phototransistor pairs which are then monitored by the touch detection circuitry.

III. THE EXTRA DECODING BLOCK

Again, the basic function of the extra decoding block is the

same as that of the row and column decoding blocks. However, this block serves no one single function such as row or column decoding.

The term extra is meant to reflect the odd or "extra" decoding that is done by this block. At the present time, it serves to select one of the five menu-select LED/phototransistor pairs for observation.

In referring to figure 2.4, it should be noted that the three 'menu select' lines are passed through tri-state buffers before they are connected to the LED/phototransistor pairs. This is because smaller LEDs and phototransistors had to be used for the five menu select pairs (to fit between the column pairs in the pad).

These smaller phototransistors had lower off-state resistance, which caused problems when they were not selected. Normally when a pair is not selected, +5 volts is connected to the cathode of the LED and to the emitter of the phototransistor. This would not allow either to be turned on. With these five menu select pairs however, the +5 volts (seen when not selected) caused the menu-select detect circuitry to send a touch message to the computer. Therefore, the tri-state buffers were used, which present an open circuit in their non-selected state.

IV. THE TOUCH PAD BLOCK

This block contains the actual touch-pad with the LEDs and phototransistors mounted in it, and the slot for the selected menus to be inserted into (see figure 4). Along the vertical and horizontal sides of the sunken touch area, are alternately mounted 32 infrared LEDs and 32 phototransistors, one across from each LED. These pairs were alternated to reduce the amount of light being received in error.

The LEDs and phototransistors were carefully aligned so as to achieve the maximum signal recieved when a signal is sent. Each of the cathodes of the LEDs along with the emitters of the phototransistors across from them, are tied to the select lines of the 74154s (see also The Row Decoder Block and The Column Decoder Block).

The touch-pad also contains five seperate pairs which are mounted perpendicular to the row and column pairs, along the edge of the pad. These serve the purpose of allowing the computer to detect which menu is in the pad. The paper menus have five corresponding holes which can be cut open or left intact (closed), representing zeros and ones.

The anodes of all of the infrared LEDs (both row/column LEDs and menu-select LEDs) are tied high through a single series

limiting resistor. Also, the cathode of each LED is connected to the emitter of its corresponding phototransistor. Therefore, when the pair is selected, and the cathode and the emitter are both taken to ground, turning on the LED and allowing the phototransistor to be turned on.

V. THE ROW/COLUMN DETECT BLOCK

This block is where the status of each phototransistor is transformed into a level that can be interpreted by the computer. With this signal, the computer can determine whether the beam is obstructed or not (corresponding to a touch or no touch).

As mentioned previously, the collectors of all of the phototransistors are tied together and pulled high through a single pull-up resistor (100k ohms). When any one of the LED/phototransistor pairs is selected, an infrared light beam from the selected LED should turn the phototransistor on, bringing the collector voltage somewhere near ground. If while one pair is selected, the beam is blocked, the phototransistor will remain turned off. In this case, the collector voltage approaches +5 volts because of the pull-up resistor.

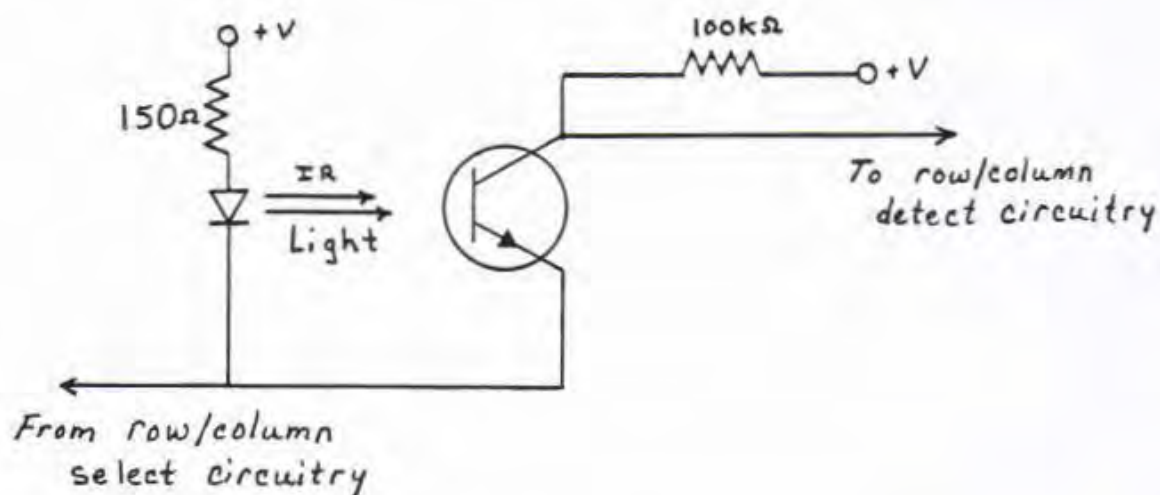


Figure 2.3 Sample LED/Transistor Circuitry

Because of the change in collector voltage from when a beam is blocked to when one is not blocked, the collectors are used as the the input to the row/column detect circuitry. This circuitry begins with two comparators which have adjustable references.

The first comparator is set up in an inverting fashion, so that when any collector voltage is below the reference (no beam blocked), the output of the comparator is at positive saturation. However, if any collector voltage swings above the reference, the

output goes to negative saturation (close to ground). This output is then used as the input to the second comparator.

This second comparator uses the same reference voltage as the first one, however, it is set up in a non-inverting fashion. The main purpose of the second comparator is to clean-up the signal.

When the selected beam is *not* broken, the output of the first comparator (which is the input to the second) is high. This also sends the second comparator into positive saturation. The output of the second comparator, is then sent through an OR gate which has one input tied low, to further clean it up.

This signal is then further conditioned by the status of the row or column selects, to become the RCRET (row/column return) signal. This RCRET signal is then combined with the MSRET signal (menu-select return) to provide one single RET (return) signal for the computer. This signal is polled by the software as a single bit input to a port. By polling in this fashion, the computer can continuously look for a touch, and process one accordingly if it is encountered.

VI. THE MENU-SELECT DETECT BLOCK

The circuitry in the menu-select detect block is almost the same as the row/column detect block. The only real differences are first of all the size of the pull-up resistor for the phototransistor, and secondly the extra select signal is used instead of the row/column selects (for conditioning).

It is appropriate at this time to note the reason for combining the three different chip selects (row select, column select, extra select) with the RCRET and the MSRET signals (see also the Touch-Pad Schematic). Normally if neither the row or column chip is selected, then the RCRET signal is high, falsely signaling a beam being broken. The same problem is encountered when the menu-select chip is not selected, the MSRET signal is high, falsely signaling a beam being broken.

To alleviate this problem, the row and column chip selects are AND'ed with the RCRET signal, and the extra chip select is AND'ed with the MSRET signal. With this conditioning, RCRET can *only* go high when either the 'row' or 'column' chips are selected. Also, MSRET can *only* go high when the 'extra' chip is selected.

The resulting signals are OR'd together to form a single RET line which is high whenever any selected beam is broken. This leaves the computer free to select either a row, column or menu-select (extra) beam, and then determine by polling one line (RET) whether or not that beam is being broken.

Shown below is a photograph of the printed circuit board used inside the touch pad. Several of the main integrated circuits are labeled to help the user locate any components on the board.



Figure 2.5 Printed Circuit Board and Components

GENERAL DISCUSSION

As was mentioned earlier in the scope of the project, the original thought had been that a total hardware system would be best. With such a system, the computer would only have to respond to some sort of interrupt from the touch-pad. During its service request, the computer could then simply read which location had been touched. This would tend to leave the computer more free to do other tasks.

Very briefly, all of this could have been provided by using a hardware clock to run several counters. These counters could in turn select each row pair, then each column pair, and finally each menu-select pair (a process now handled by the computer).

The major disadvantage to this method was that the scan process would be set in one certain fashion, unable to change if a better process was discovered. With the present method, the computer supplies the count to the pad. With this system, the count can be supplied in any order, able to change with only minor software changes.

The current method of using infrared light beams (instead of some other form of detection) was decided upon for various reasons.

- 1) Other touch-pad schemes such as capacitive touch sensing, and pressure sensitive membrane type keypads, are all affected by water, or saliva in this case.
- 2) Most important, breaking a light beam requires the least amount of pressure of any method studied.

The decision to use identical circuits for the RCRET and the MSRET may at first seem redundant. However, because of the limited amount of physical space between the column LEDs and phototransistors, smaller optical components had to be used. These smaller components required the same type of detection circuitry, with only the change of the pull-up resistor.

So, because the two blocks need to be electrically isolated, and because the needed gates and comparators (for duplicate circuitry) were in fact available, it was decided to duplicate the row/column detection for the menu-select detection.

THE ULTRASONIC RANGING SYSTEM

SPECIFICATIONS

The ultrasonic ranging system is considered a protective device. Its major function is to prevent damage to the chair or injury to its operator. It is also necessary to protect other young children who might be in the general area of the chair (innocent bystanders).

When designing the ultrasonic ranging system, the following specifications were used as guidelines.

- 1) The system should be able to sense any object within approximately four feet of the chair, from any of four different directions.
- 2) It should audibly warn the user of these obstructions, so as to allow time to take corrective actions.
- 3) It should also be possible to turn this audible feedback off.
- 4) If corrective actions are not taken in time to avoid a collision, the chair should stop automatically.
- 5) It should be possible to place the ultrasonic units in any desired location on the wheelchair, and should not deface it in any manner.
- 6) If a major failure should occur, it should be possible to remove and retire the complete system without effecting normal operation of the wheelchair.
- 7) Without a major failure, it should be possible to turn the ranging system off.
- 8) Other than stopping the chair in an emergency, the system is not to take offensive control at anytime as this would deter the user from learning to be in complete control of the wheelchair. (It is anticipated that after some practice, the user will be able to control the wheelchair without the use of the ranging system.)

With these specifications in mind, the ultrasonic system generally performs two main functions: It provides feedback to the user as to the approach of obstacles, and it provides a failsafe method of stopping the chair should the child fail to respond to the system's warning.

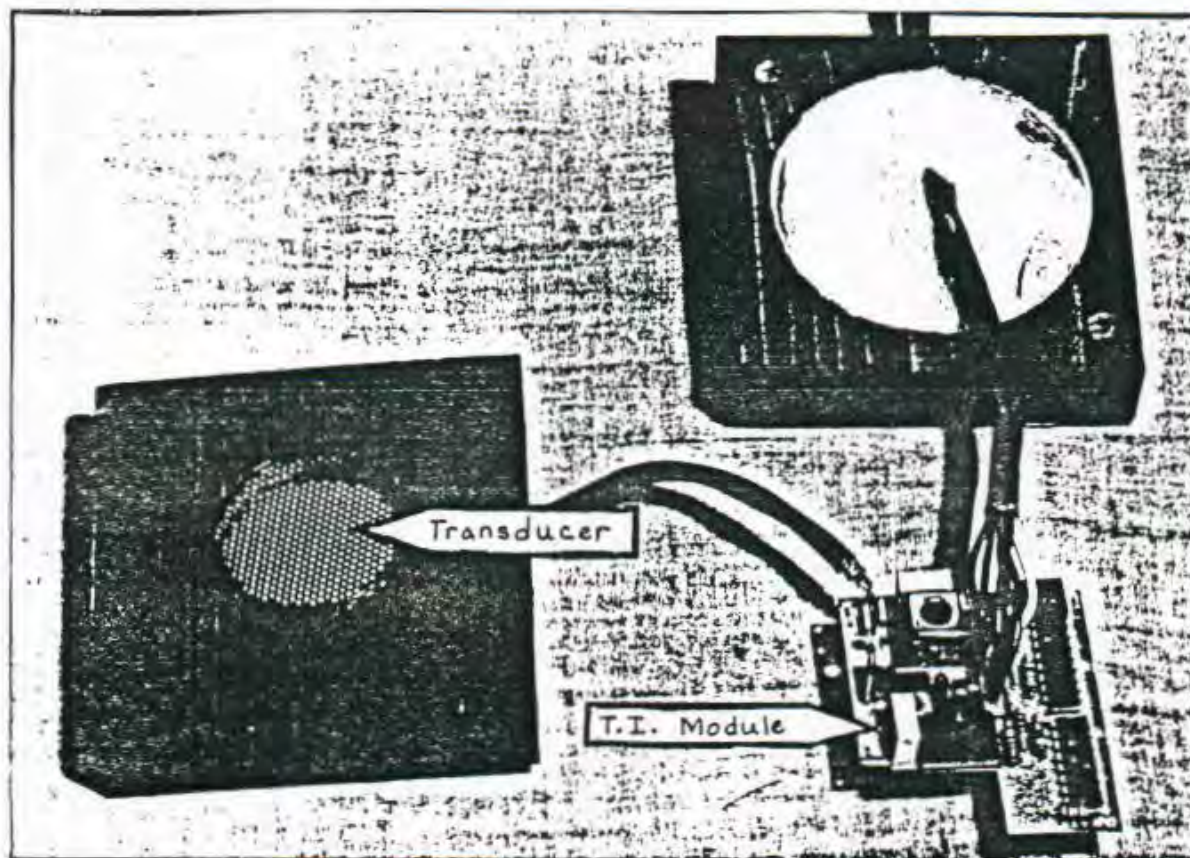


Figure 3.0 Ranging Module

BLOCK DIAGRAM

The block diagram for the ultrasonic system (shown below) consists of four principal parts. These include four directional transducers, the tone generator, a timer to aid in distance calculations, and the additional I/O board which is the system's interface to the computer. Each of these blocks will be discussed in greater detail in the following sections.

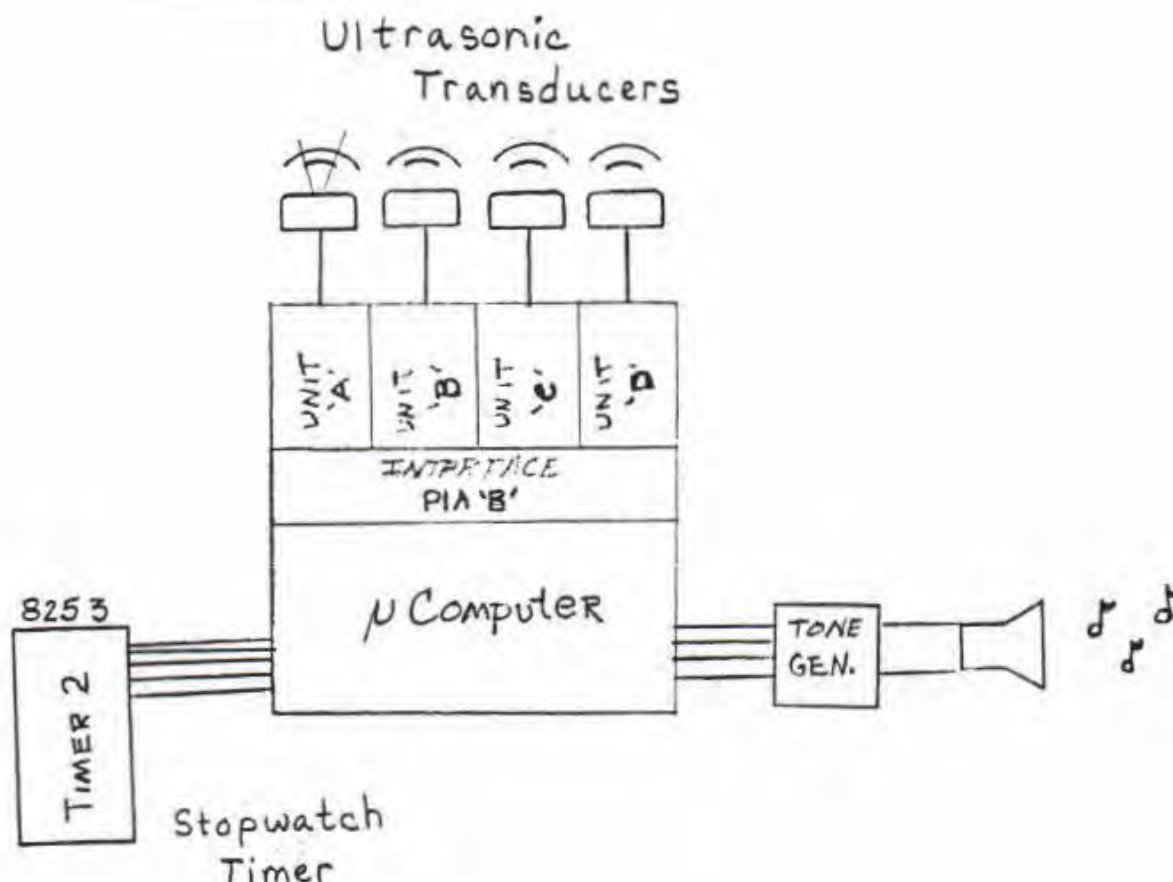


Figure 3.1 Ultrasonic System Block Diagram

I. THE DIRECTIONAL TRANSDUCER BLOCK

The directional transducer block is the heart of the ranging system. It consists of four complete and separate ranging modules. Each module contains a 50 kHz, 300 volt electrostatic transducer, and a small amount of drive circuitry. Each module is capable of ranging from four inches to approximately 35 feet with less than two percent maximum error. (See also Figure 3.2 Ranging Module Schematic)

Each ranging module contains a Texas Instruments SN28B27 sonar ranging module. This T.I. module provides the 150 volt

bias for the transducer and pulses the transducer with 16 cycles of a 50 kHz, 300 volt waveform. This pulse can actually be heard with the naked ear, as it sounds like a short click. This ultrasonic waveform travels at the speed of sound (0.9 ms/foot) until it strikes an obstacle and its echo returns to the transducer at the same speed.

The module provides a controllable blanking period to allow transducer vibration to disappate before it is enabled to wait for a returning echo. All control signals are TTL compatible, but the echo output is of open collector type and needs a pull-up resistor in order to get a reliable TTL signal.

There are three main control signals. The INIT* input starts the ranging process by sending out the click. The BLNK* input defeats the internal echo blanking. And the ECHO* output signals when the click is returned. All three signals are active low, and their relationships to eachother are demonstrated below in Figure 3.3 Timing Diagram.

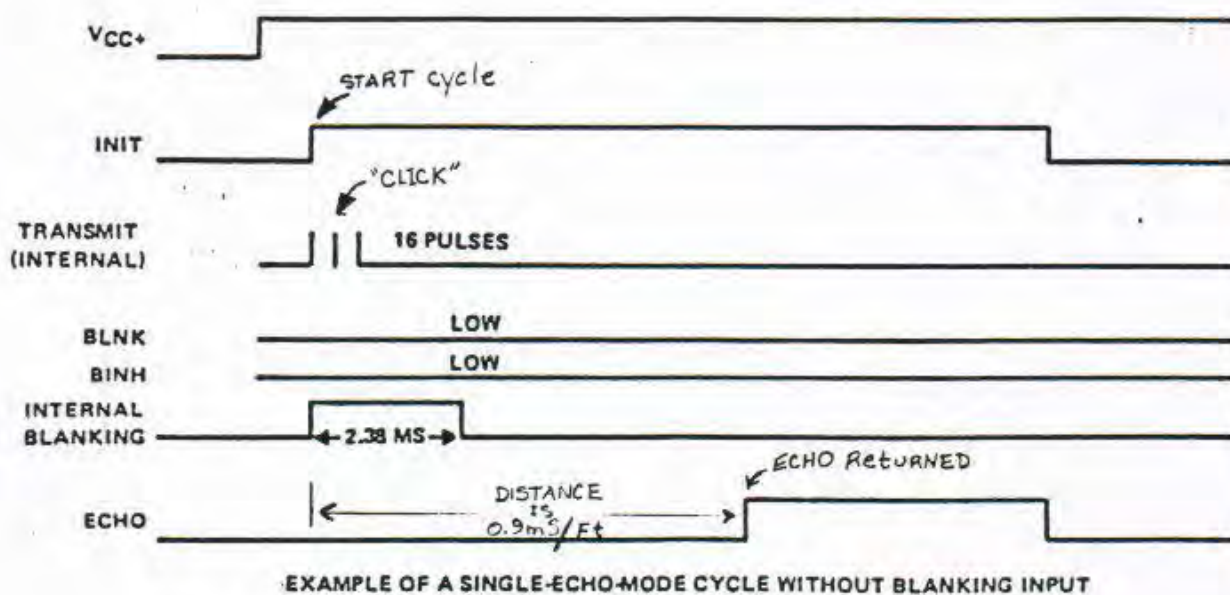


Figure 3.3 Timing Diagram

The only deviation from Texas Instruments design was that a large capacitor was added in parallel with the power connections as they enter each transducer's driver. This was done in order to supply the rated 2000 mA each transducer needs during the 326 μ s transmit period. This is such a rapid drain that the power supply could not source it through six feet of cabling.

II. THE TONE GENERATOR BLOCK

The tone generator block consists mainly of the XR2206 function generator chip (capable of switching between two selected tones) and an LM2002, eight watt audio power amplifier chip. (See also Figure 3.4 Tone Generator Schematic)

The XR2206 has the ability to output two selectable tones. These tones are selected by switching the TTL level at the FSK input. This allows several types of warnings to be generated. The two tones are separately adjustable and independent. These adjustments are made to R4 and R6. See Figure 3.4 Tone Generator Schematic. The potentiometer R7 is a volume adjustment.

Turning the tone off is done with the Amplitude Modulation input. If the AM input is held at half the supply voltage, the output will be turned off. Control was accomplished by switching a voltage divider in and out. This voltage divider has two equal resistances (in series) to ground, creating a reference of one half that of the supply. The junction between the two resistors is connected to the AM input to the chip. An NPN transistor is used to shunt the bottom resistor of the divider when it is turned on, thus turning the output on (or off). This transistor is controlled by a TTL level sent from the computer, allowing the sound to be turned on and off.

III. THE ADDITIONAL PIA AND TIMER BLOCKS

To supply the needed output for the tone circuit and the ultrasonic units, a second 8255 programmable port had to be added. It is configured to have 20 bits of output and 4 bits of input. Ports A, B, and the lower four bits of C are defined as output. The higher four bits of port C defined as input.

Port A controls the ultrasonics INIT* and BLNK* of each transducer. Port B outputs a digital word to be used by the motor control circuits for direction and speed control. Port C controls the tone generator with its upper half and receives the ECHO* from the transducers on the lower half. (See also Figure 3.5 Additional Parallel Group)

The timer block consists of three programmable timers within an 8253 timer chip. The 8253 is part of the SCCS-85 computer. (See also Figure 4.4 Timer Group, I/O Addressing Group).

The first timer is configured to count down from 65,535 (FFFF Hex) and is used as a stop watch during the ranging cycle. The second is used for the generation of the 16 times baud clock needed by the 8251 for RS-232C communication. The last of the three timers on the chip is used for what is termed a 'heartbeat' timer. With the help of a relay, if the timer counts out it will return the chair to the joystick configuration. So if the computer should fail, within 80 milliseconds the timer will count out, and control will return to the joystick.

GENERAL DISCUSSION

The ranging system seems to perform very well. The transducer modules are fairly simple to use, and they are both accurate and reliable. The only noticeable drawback to the ultrasonic units would be the audible click when the transducer fires. This sound could become annoying after time, but one should remember that they can be shut off after they are no longer needed.

From a designers standpoint, using a prebuilt module for the units was definitely better than trying to design the modules themselves. Because they were not familiar, this made troubleshooting harder in the few instances they failed to work. After time, however, that was no longer a problem because of more familiarity.

For reasons of flexibility and pleasing tones, the decision was made to design our own tone circuit. This was chosen instead of buying small tone transducers such as piezo buzzers.

The main problem encountered here was in attempting to drive the eight ohm load of the speaker. After trying to use several voltage amps, current amps, transformers and push-pull amps, it was decided to use an LM2002. This is a self contained amplifier chip which is specifically made for such a purpose.

The additional I/O board was constructed using a point-to-point soldering technique. This method was chosen because it took less time than to create a printed circuit board, and it is a more reliable method than wire wrapping.

The I/O board contains the circuits for the tone generator, the motor control, the power supply (conditioning), the additional parallel port, and the status LED circuitry.

COMPUTER AND MOTOR CONTROL

SPECIFICATIONS

The computer and motor control systems are possibly the most important parts of the Easy Chair system. A failure in either of these two systems could render the entire system inoperative.

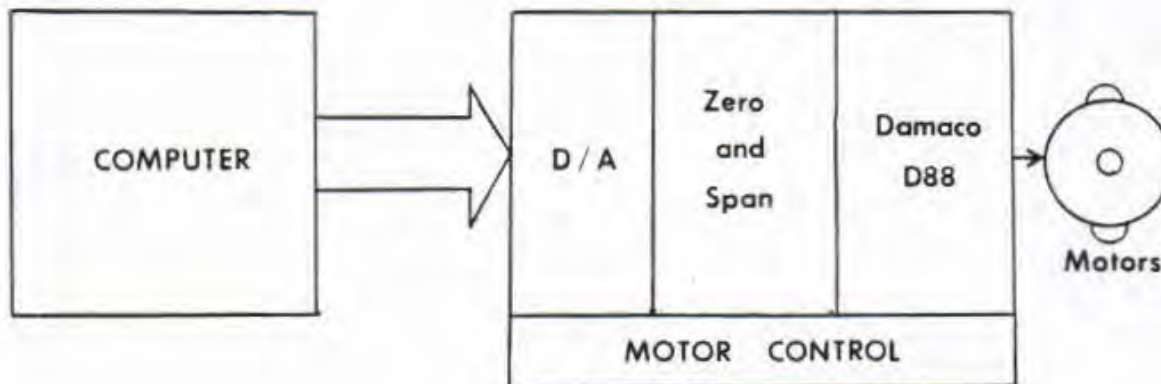
The following guidelines were used when choosing the computer for The Easy Chair:

- 1) As is the case with all of the components, the computer must be extremely rugged.
- 2) Also, it must be usable in the sense that it is user friendly, allowing anyone to alter several different characteristics of the chair.
- 3) It should have an RS-232C serial interface to allow it to communicate with other devices.
- 4) It should be designed so that should the computer fail, the chair would revert back to control by joystick.
- 5) It should have the capability to "remember" several settings, even after the power has been removed.
- 6) It should have the capability to perform some limited self-diagnostics, to identify possible problems.

The computer decided upon was the SCCS-85 single board computer, available at Purdue University. This is an 8085 based computer with many options for memory and I/O.

It was chosen because of its flexibility, the ease of use, and the fact that several faculty members in the Electrical Engineering Technology department at Purdue University are very familiar with it.

BLOCK DIAGRAM



DETAILED BLOCK DESCRIPTIONS

I. THE COMPUTER BLOCK

The computer was initially built according to the manual provided. After operation was verified, the following changes were made.

The clock speed was increased to speed execution time. Memory configurations were altered to accommodate eight kilobytes of EPROM (for startup sequence and monitor), and eight kilobytes of NOVRAM (non-volatile battery backup RAM) for variable storage, program development, and touch pad menu information. The NOVRAM will allow the system to be reconfigured by anyone, at any time. (See also Figures 4.2-4.7)

With the computer in normal operation, a major consideration is the software. This software includes routines which process input from the touch pad, monitor perimeters with the ranging system, control the motors through digital-to-analog circuitry.

Aside from those "real-time" responsibilities of the computer, it will also allow the user to alter such settings as the ranging distances, the audible feedback, the speed settings, durations, and menus which contain the settings mentioned.

Under normal operation, the user would first select a menu to be used for the operation. Once that menu was inserted into the touch pad, the computer would recognize it and alter settings to match those of the menu. With a menu in place, the user can select any defined area on the menu, and the computer will move the chair in the direction defined for that area. While the

chair is moving, the computer will use the ultrasonic ranging system to alert the user of any obstructions.

All of this is accomplished through very complex assembly language software. Outlines for this software can be found in Appendix B: Software Outlines. These outlines will offer an overall view of how the chair is controlled. For further detail, one can consult the actual source code found in Appendix C: Software Listing. This code is effectively commented to offer the most possible insight into the different routines.

II. THE MOTOR CONTROL BLOCK

The motor control block contains all the necessary circuitry to switch control between joystick and the computer, and then to allow the computer to replace the joystick electronically.

The motor control circuit uses a single hexadecimal byte to control both motors. With four bits per motor this gives 16 different speeds; eight speeds forward and eight speeds in reverse. Although eight speeds may not at first seem like much, when compared to the resolution of the joystick it allows for many different speed options.

Operation of the controller is fairly straight forward. Two AD558 digital to analog converters are used to create a digitally controlled voltage which is variable from 0-2 volts. This output is then used as the input to a zero and span circuit, which allows the computer generated voltage to be adjusted so that it can effectively replace the joystick.

Using test equipment, the voltage potentials of the joystick pots were measured. The zero and span circuitry is adjusted to match not only this precise reference, but also the range available with the joysticks. (See also Figure 4.8 Motor Control Schematic)

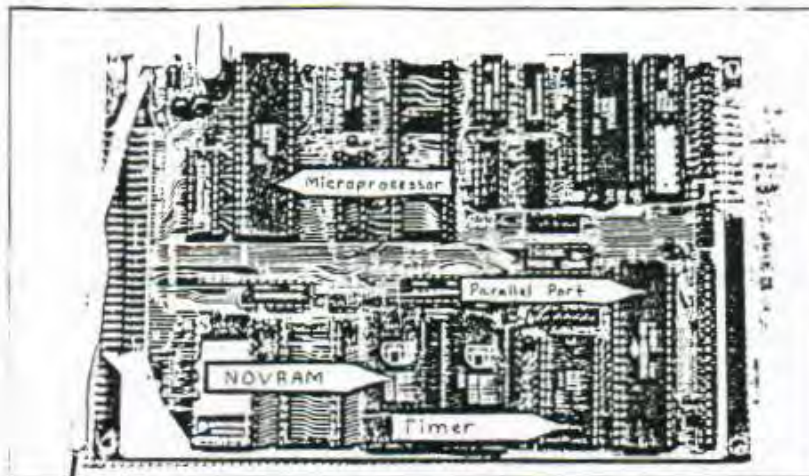


Figure 4.0 SCCS-85 Single Board Computer

GENERAL DISCUSSION

When experimenting with different types of memory, EEPROM's were used for a short time. However, because of timing problems, the RAM was changed to NOVRAM's. This is not to say that the NOVRAM's are without fault, but operation is faster and more reliable than that of the EEPROM's.

As is mentioned in the software outlines, the software can detect several different types of errors with the system. These errors can include something as complex as a bad chip, to something as simple as a menu not fully in place.

This error checking software also uses some special voltage loops in the different connectors. These loops are used to determine whether or not the connections are intact. If they are not (for whatever reason) the software will signal an error by lighting the correct LED, and ignore the corresponding device. (See below, Figure 4.10 Status LED's and Control Switches)

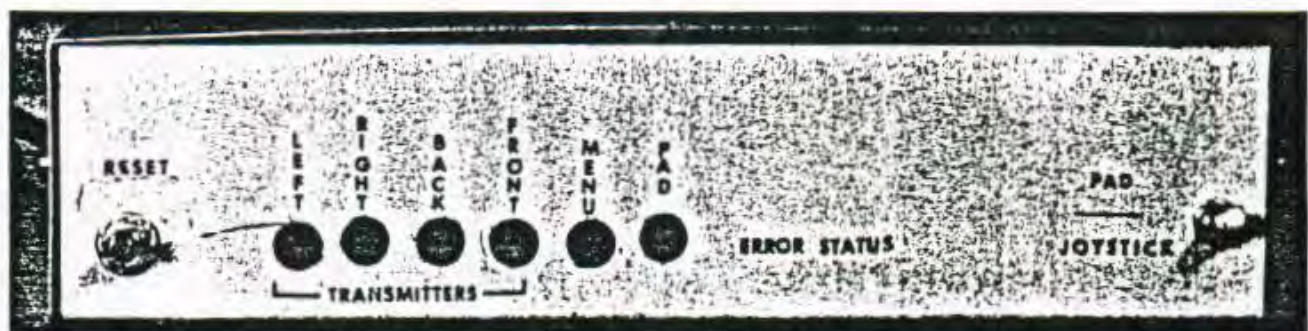


Figure 4.10 Status LED's and Control Switches

The motor control circuitry was modified from previous designs, to give added control to the signals. As mentioned previously, there are now adjustments for both the zero and span of the outputs from the D/A converters. This is in contrast to the original design which afforded only zero (offset) adjustment, but no adjustments for the span or range of the signals.

CONCLUSION

The project as a whole ran very smoothly. All of the design criteria was met, and in some cases surpassed. The work was completed at least on time, with much of it completed ahead of schedule.

As far as software is concerned, the original monitor program used in the SCCS-85 computer has been modified to reduce unneeded code. Then all of the routines to control the overall system were added, and also several small test routines. These test routines exercise each of the separate components of the system to assure that they are working correctly.

As mentioned earlier, discussion of the software in a text form, would be very difficult for the reader to understand. For this reason, the software is explained in the software outlines found in Appendix B Software Outlines. These software outlines use a general 'English language' format, rather than flow charts or diagrams. Actual subroutine and variable names are used in the outlines so that the reader can refer to the code with less difficulty.

In the future, a major recommendation would be to check thoroughly for 'second source' vendors. For instance, after checking with Polaroid for the ultrasonic transducers and ranging modules, they were later found for almost one third the original cost at a second vendor. Also, the cost of LEDs and phototransistors could be lowered by purchasing from a large wholesale distributor, (due to the quantity).

Another thought would be that if the touch-pad were constructed just slightly larger, the same LEDs and transistors could be used for all of the detection. This would eliminate the need for special menu-select detect circuitry, and the special smaller LEDs and transistors.

Overall for the project, having two people working together seemed to greatly enhance not only productivity and problem solving, but also enthusiasm. It always helps to have 'fresh' ideas to solve a problem. With two people working together, it seemed that one problem could usually be solved with the help of another person's 'fresh' outlook.

Because of the durability, ease of use, safety and flexibility, The Easy Chair does provide an effective mode of transportation for handicapped children. With the assistance of the Easy Chair User's Manual, the system can be used by virtually anyone.

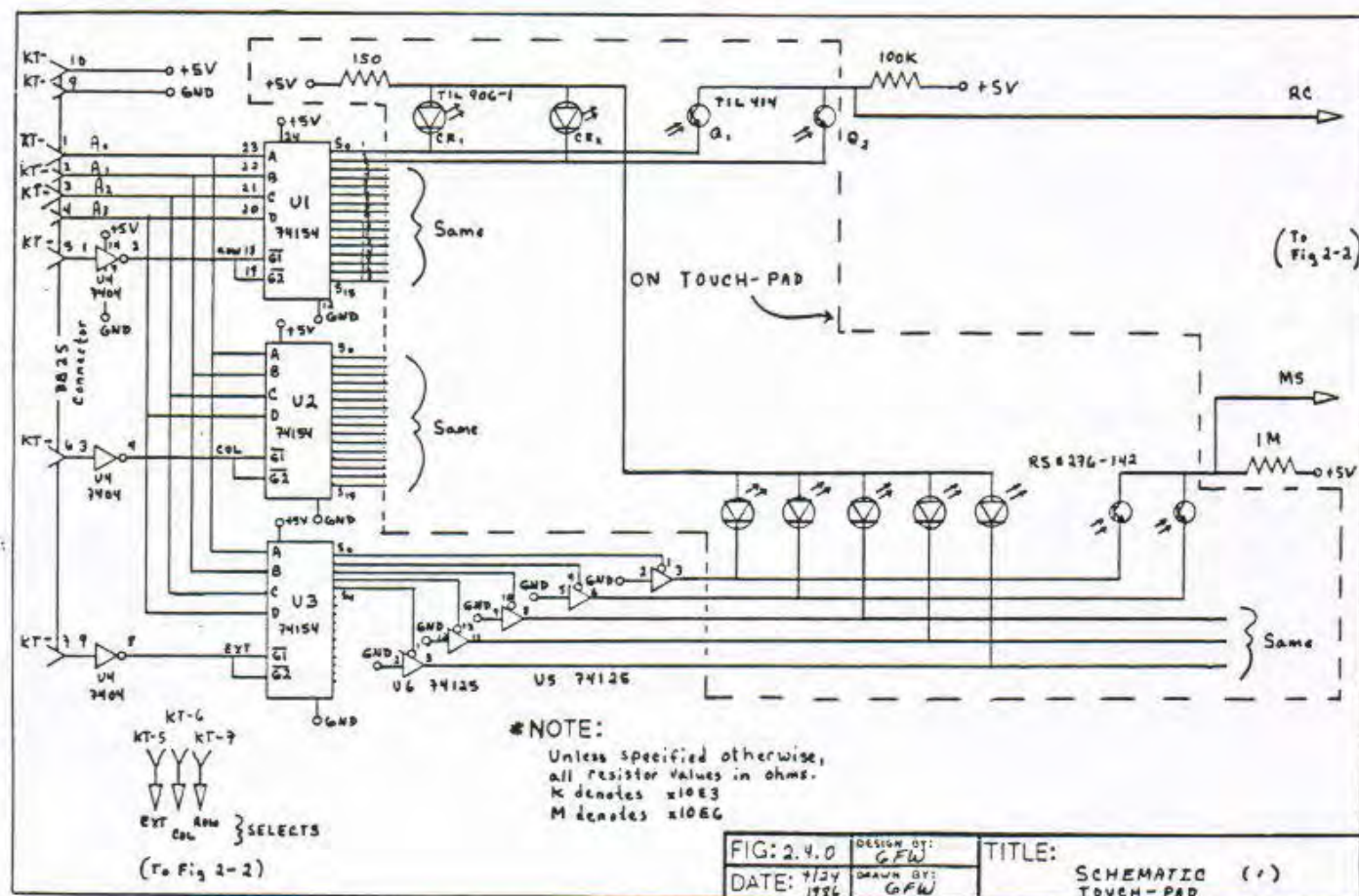


Figure 2.4.0 Touch Pad Schematics

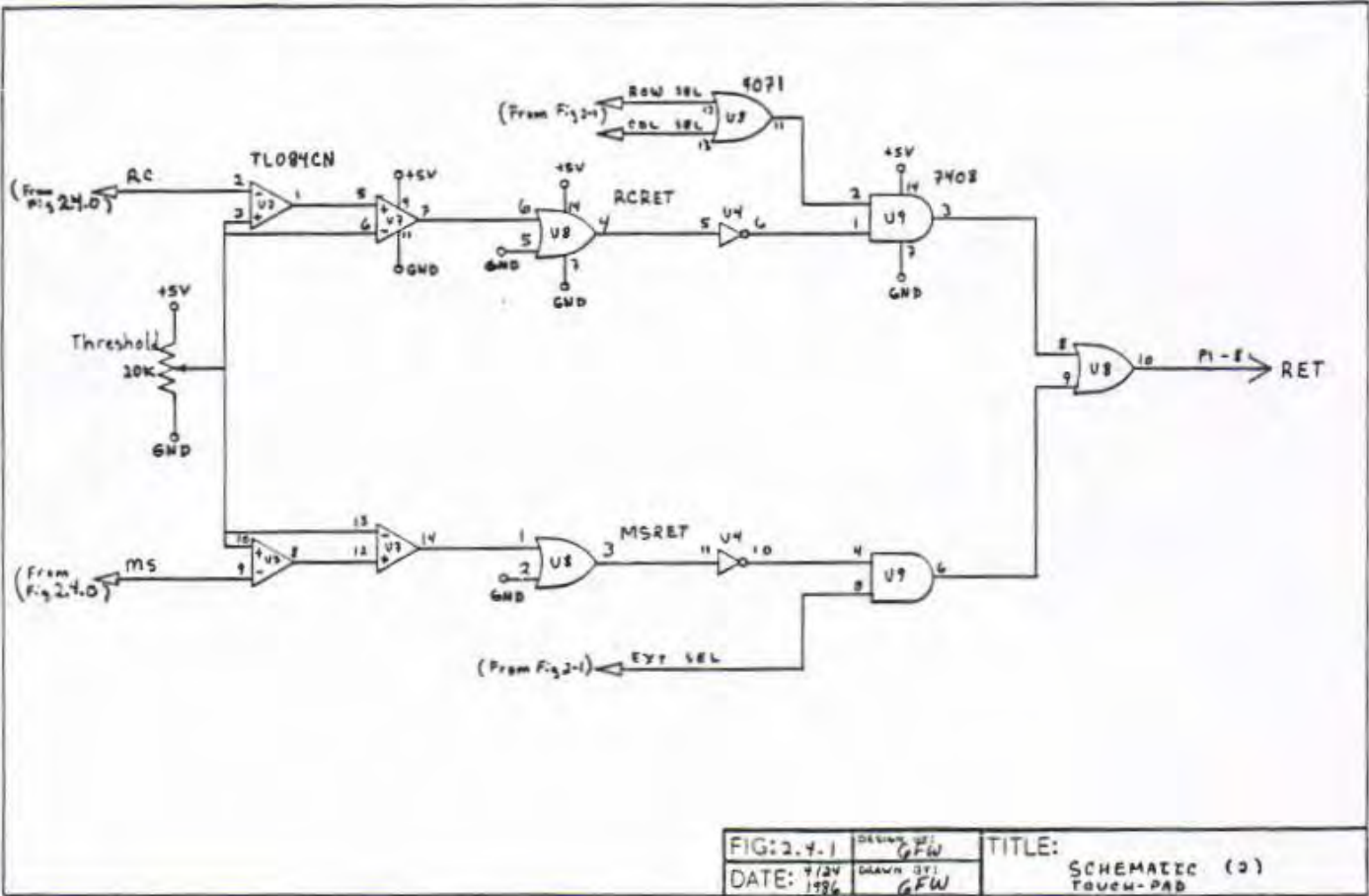


Figure 2.4.1 Touch Pad Schematics

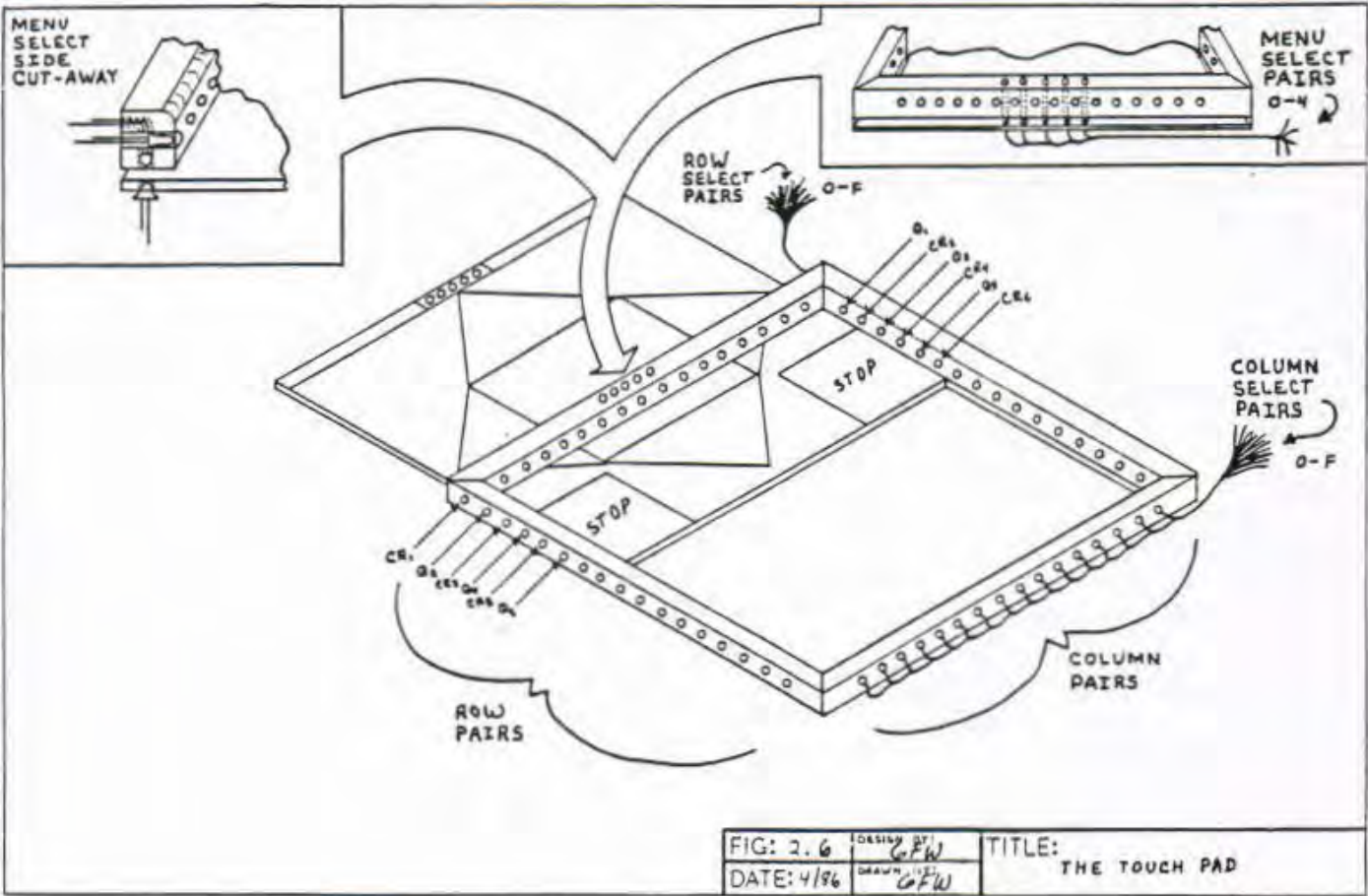


Figure 2.6 Pictorial with Cut-away

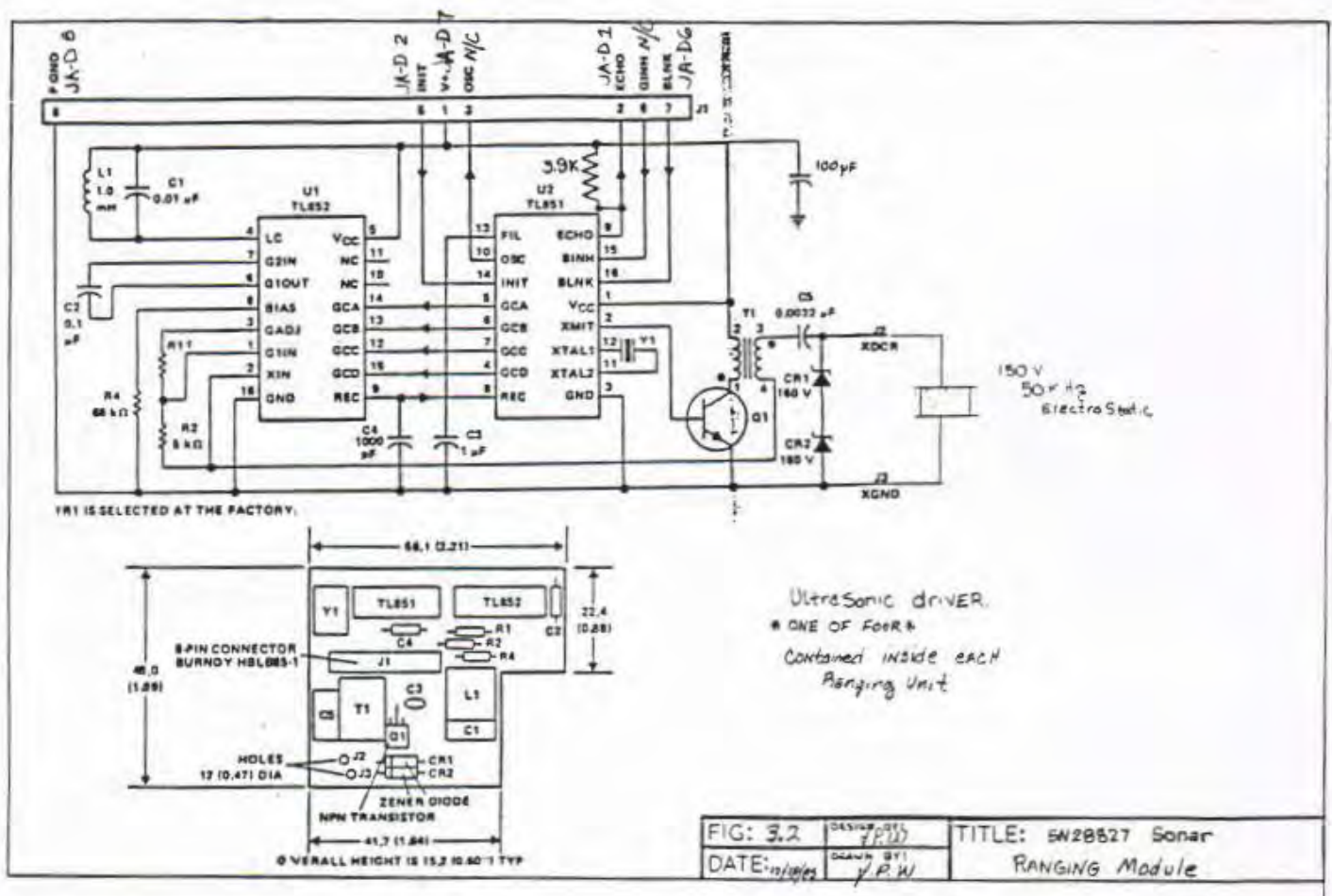


Figure 3.2 Ranging Module Schematic

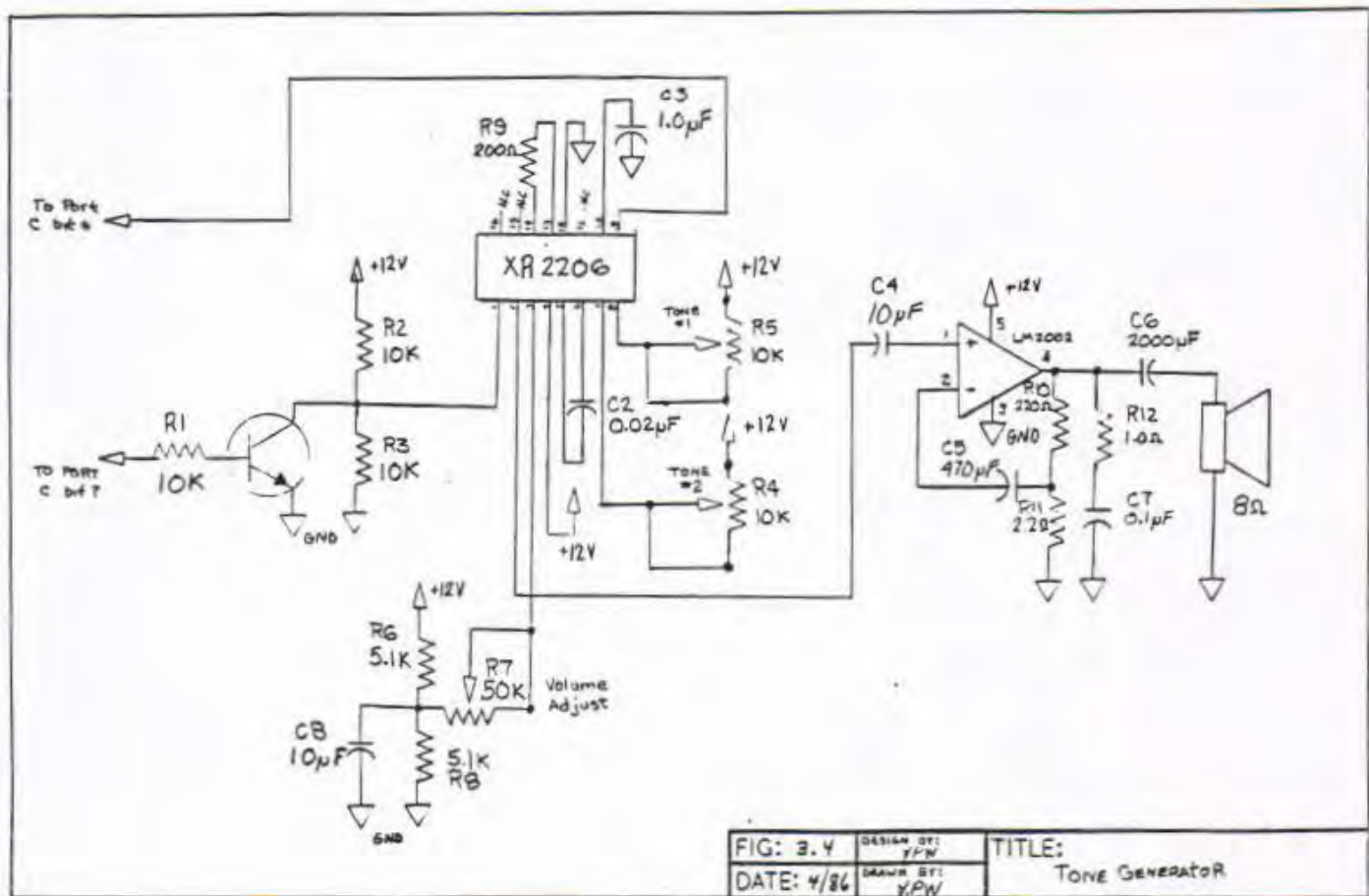


Figure 3.4 Tone Generator Schematic

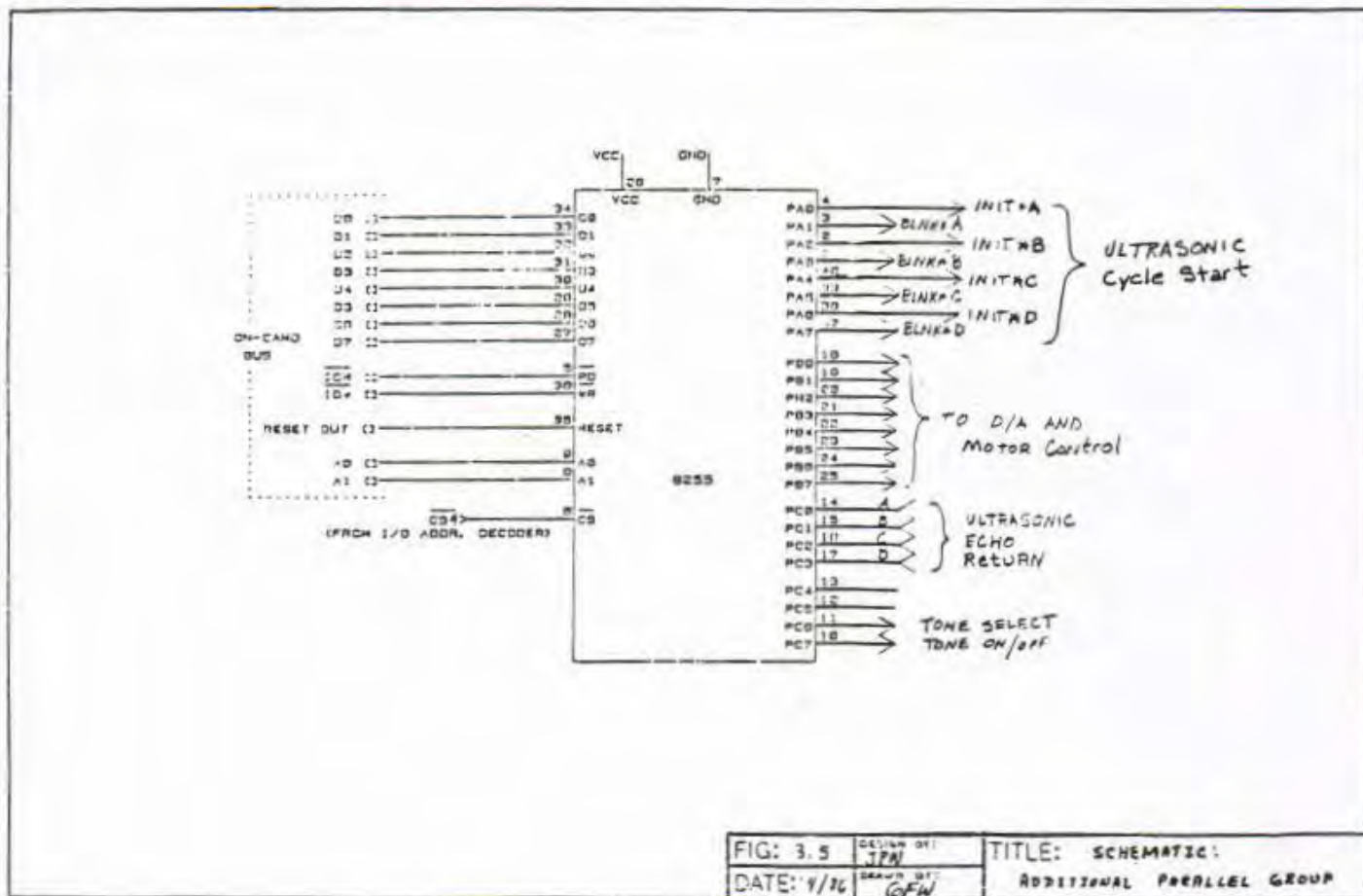


Figure 3.5 Additional Parallel Group Schematic

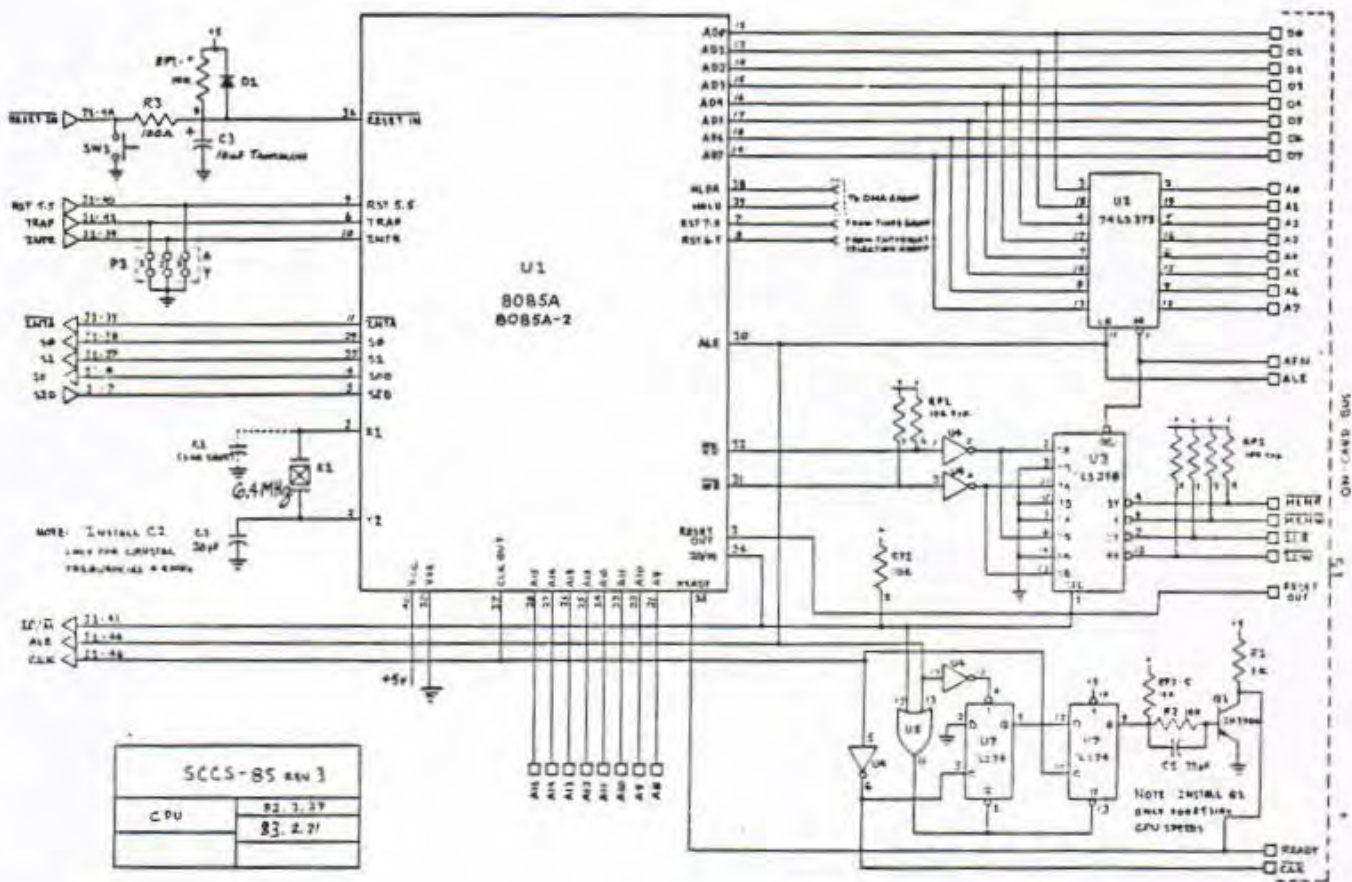


Figure 4.2 CPU Schematic

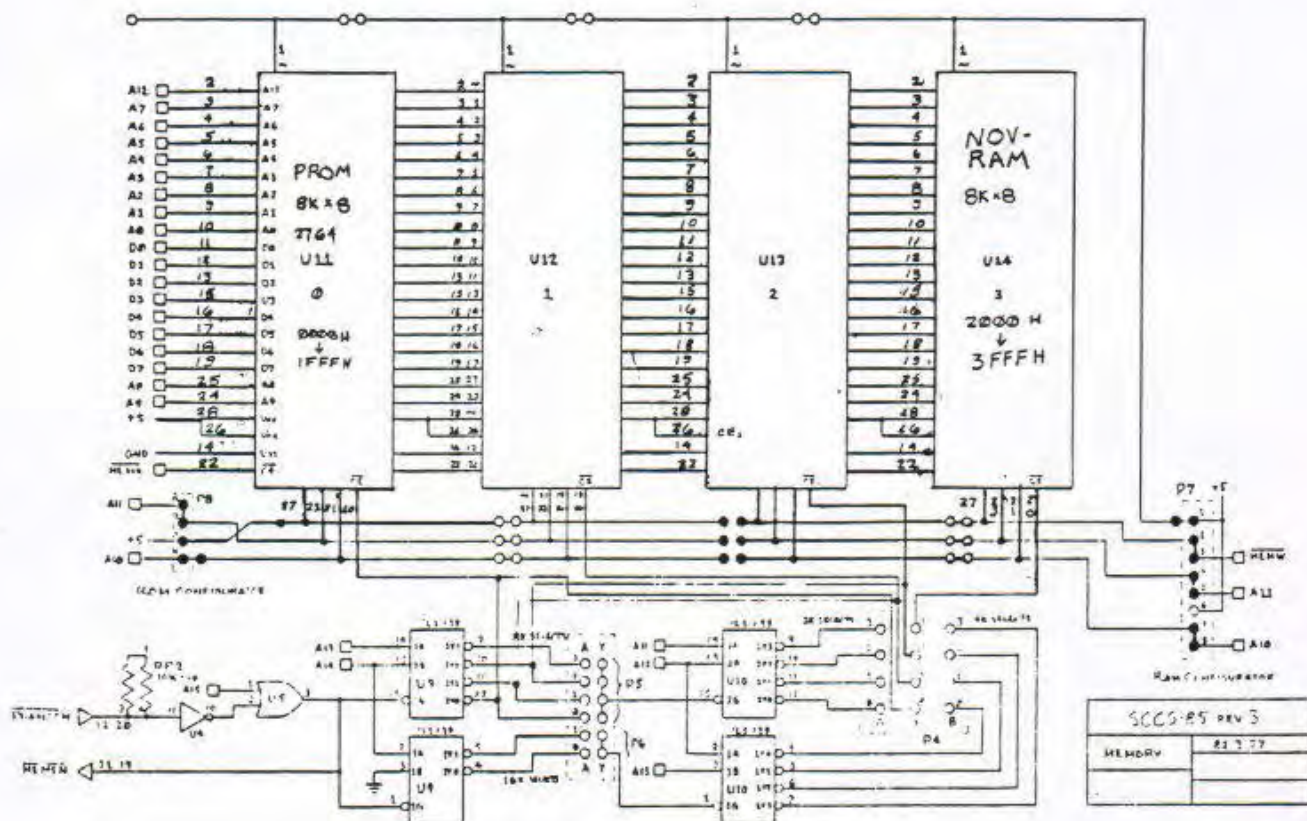


Figure 4.3 Memory Schematic

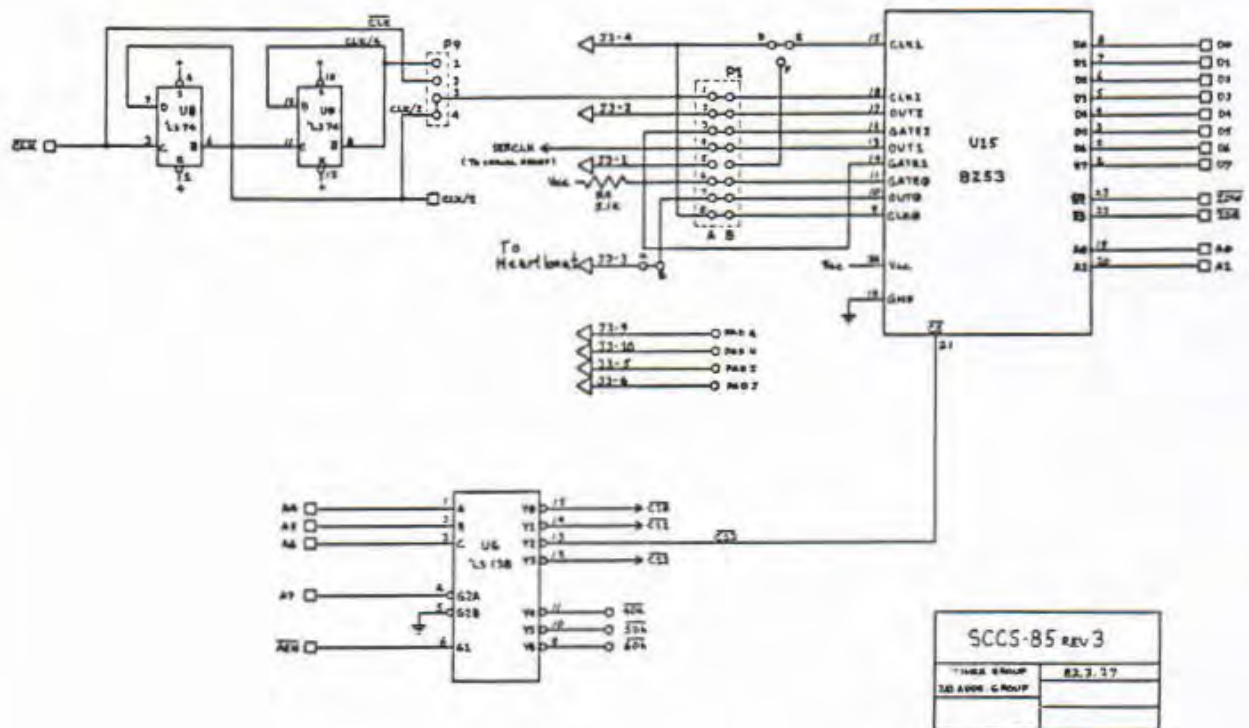


Figure 4.4 Timer Group, I/O Addressing Group Schematic

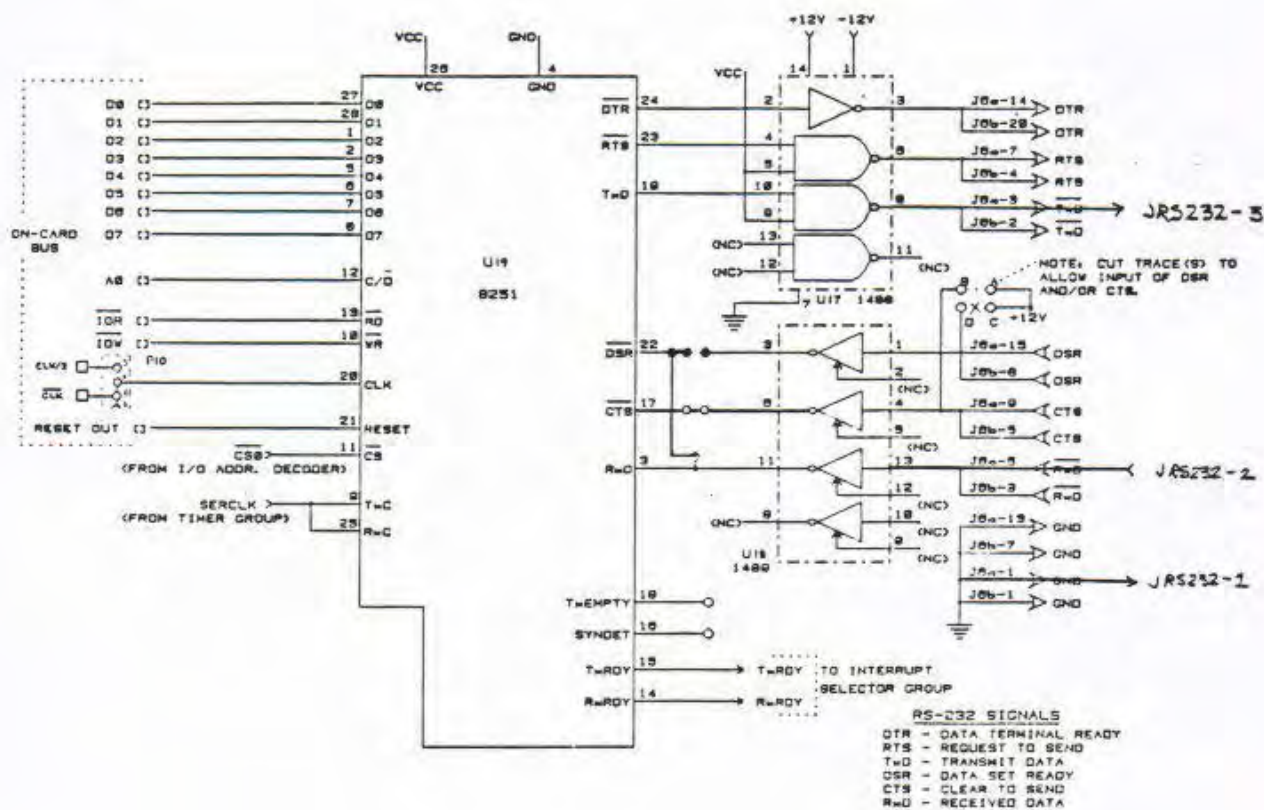
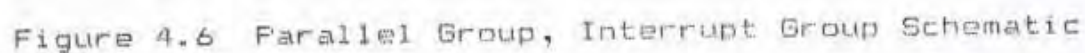
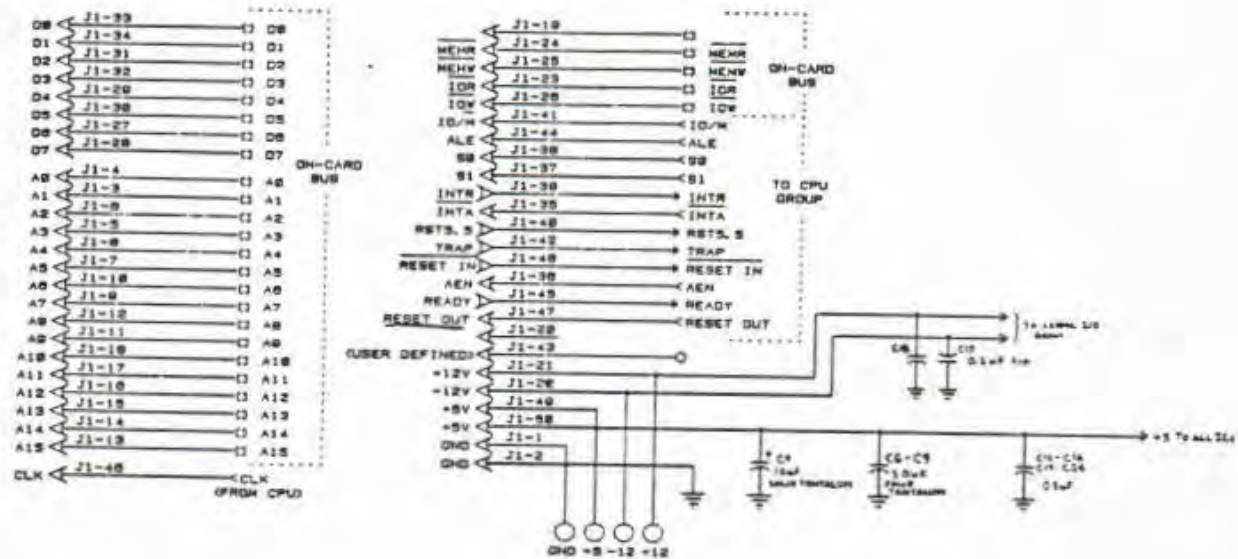


Figure 4.5 Serial Group Schematic





SCCS-85 Rev 3	
BUS CONNECTOR, POWER SUPPLY	REV. 2.1
	DESIGNED BY: M.S.P.
	APPROVED BY: M.S.P.

Figure 4.7 Bus Connector Schematic

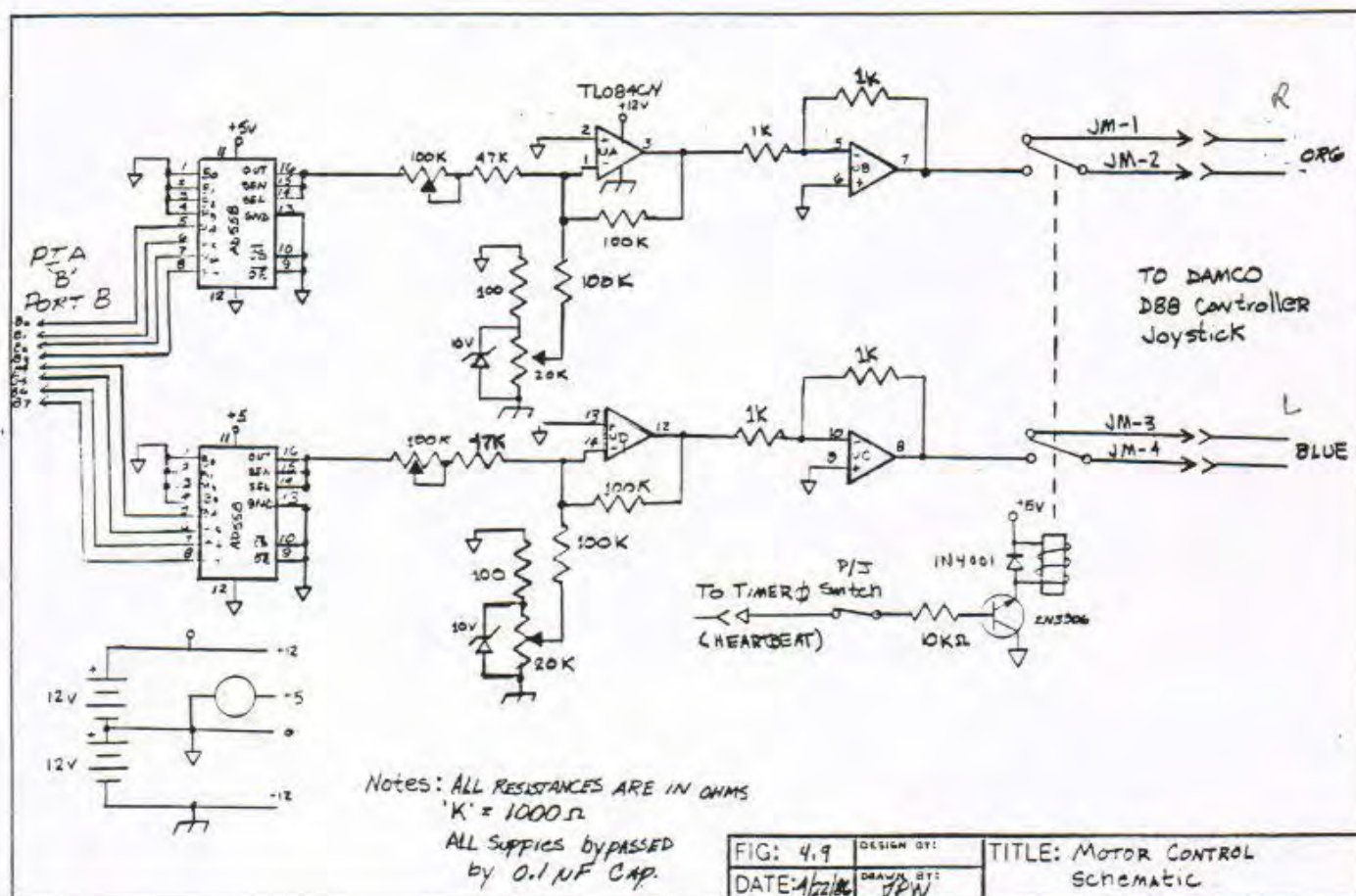


Figure 4.9 Motor Control Schematic

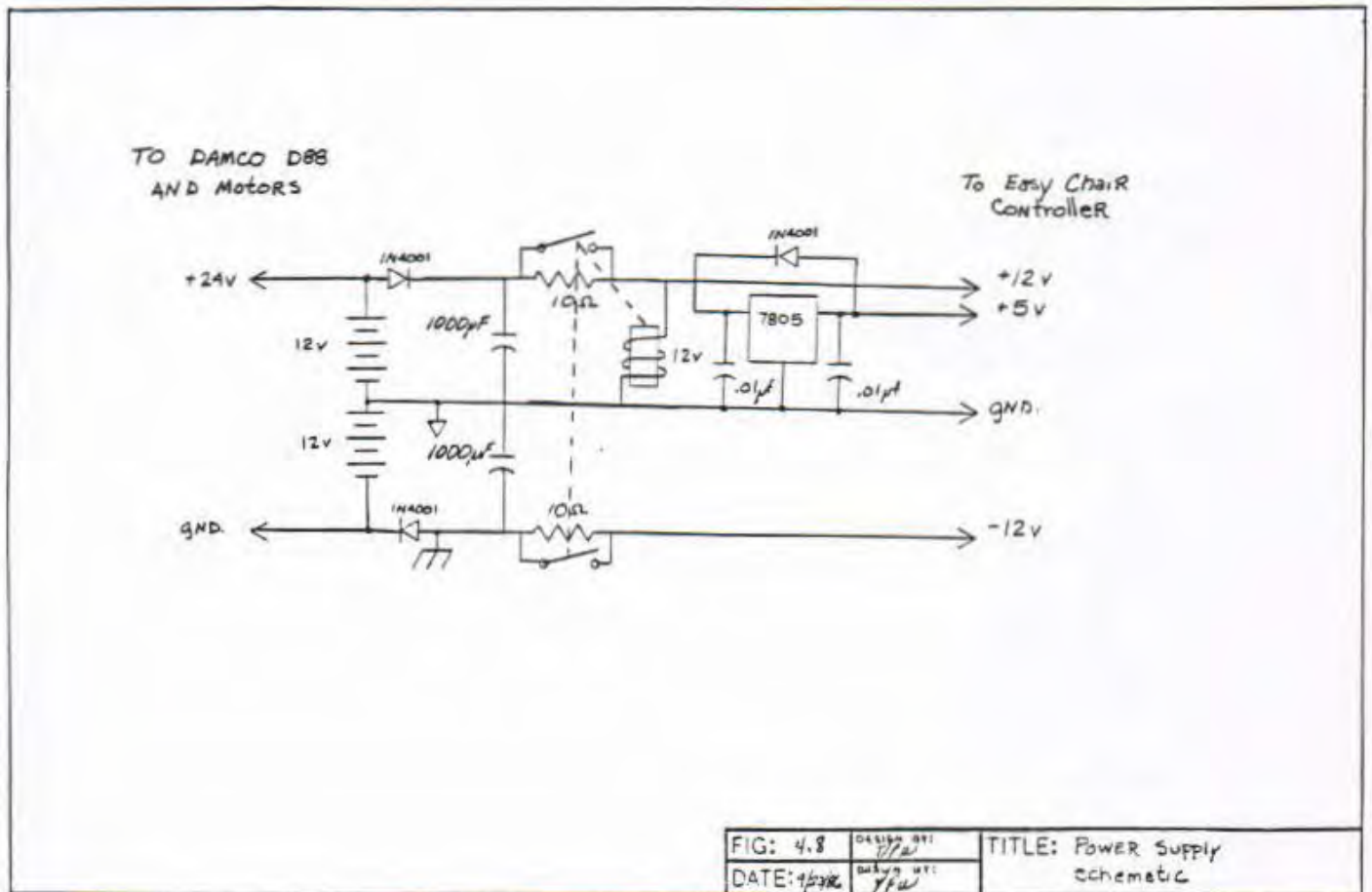


Figure 4.8 Power Supply Schematic



Figure 4.11 Connectors and Jacks

MAIN PROGRAM LOOP

```
INITIAL:  Set stack pointer
          Initialize ports
          Initialize counters
          Send stop valves to motors (to ramp)

RUNCHR:   Set flag for heartbeat on
          Call HRTBEAT to refresh the heartbeat timer
          Call MENCHK routine to act on current menu
          Call ULTRA to check ultrasonic units
          Call PADCHK to act on current pad touch
          Call EXTCHK to see if user has chosen to exit
          IF menu not valid (if empty)
            THEN call STOP
            Go to RUNCHR:

AOK1:     IF DURATION zero
          THEN call STOP
            Go to RUNCHR:

AOK1:     Call UPDATMTR
          Go to RUNCHR:
```

HRTBEAT

HRTBEAT: Refresh heartbeat counter to maximum value
Return

MENCHK

```
MENCHK:  Check to see that the pad is connected
         IF the pad is not connected
           THEN light pad error LED
           Call STOP
           Return
         ELSE clear pad error LED
           Call PADRD (determine menu number)
           IF menu error
             THEN light menu error LED
             ELSE clear menu error LED
               IF menu number 1
                 THEN call PROMEN
                 ELSE read menu variables

Return
```

ULTRA

```
ULTRA:    Check to see if the front unit is connected
          IF unit not connected
            THEN light correct U.S. error L.E.D.
            Return
          ELSE read the object distance
            IF object within critical range
              THEN call STOP
            ELSE IF object within warning range
              THEN sound warning
            ELSE return
```

PADCHK

```
PADCHK:  Check to see that the pad is connected
        IF the pad is not connected
            THEN light the pad error LED
            Call STOP
            Return
        ELSE clear the pad error LED
            Call PADRD routine to scan the pad
            IF pad touched
                THEN call the CHKTBL routine
                    IF valid location and DURATION not 0
                        THEN IF a new motion
                            THEN call INITMTR

Return
```

UPDATMTR

UPDATMTR:

RAMP1: IF ENTRAMP not zero
THEN go to LBL4:

RSPD1CHK: Store RMCS in RMOTOR (in case RMTS=RMCS)
IF RMTS=RMCS THEN go to LSPD2CHK:
IF RMTS>RMCS THEN call RGTFRD (determine RMOTOR)
IF RMTS<RMCS THEN call RGTREV (determine RMOTOR)

LSPD1CHK: Store LMCS in LMOTOR (in case LMTS=LMCS)
IF LMTS=LMCS THEN go to LBL3:
IF LMTS>LMCS THEN call LFTFRD (determine LMOTOR)
IF LMTS<LMCS THEN call LFTREV (determine LMOTOR)

LBL3: Decrement DURATION
Set RMCS to RMOTOR
Set LMCS to LMOTOR
Combine LMCS & RMCS into one byte
Transmit motor speed byte
Return

LBL4: Increment ENTRAMP
Return

STOP

```
STOP:      Set RMTS to MTRSTOP
           Set RMTS to MTRSTOP
           IF both motors already stopped
             THEN return

STOP2:      Set CNTRAMP to STOPRAMP
           Set RAMPONT to STOPRAMP

STOP1:      Short delay
           Call UPDATMTR
           IF both motors not stopped
             THEN go to STOP1:
           Store zero in DURATION
           Store zero in LASTI
           Return
```

PADRD

```

PADRD:
MENU:      Initialize loop counters
           Set menu select counter to 0
LOOP3:     Decrement menu select counter
           Mask counter to select menu select LEDs
           Output count to light IR LED
           Short Delay
           Input from touch pad return to see if beam broken
           Combine into menu number byte
           Rotate menu number byte left one bit
           IF counter at 0
             THEN rotate menu number byte right one bit
             ELSE go to LOOP3:
ERR1:      IF menu number byte is 0 (none of the 5 beams broken)
           THEN signal menu error in status word (H)
           Go to PNTDAT:
ERR2:      IF menu number byte is 1FH (all 5 beams broken)
           THEN signal menu error in status word (H)
           Go to PNTDAT:
SCAN:      Clear row/column data register
ROW:       Set row counter to 17 (row 16 plus 1)
LOOP4:     Decrement row counter
           Mask counter to select row LEDs
           Output count to light IR LED
           Short delay
           Input from touch pad return to see if beam broken
           IF beam being broken
             THEN rotate row count 4 bits
             Store row number in row/column data register
             Go to COL:
           ELSE IF row counter at 0 (all 16 rows scanned)
             THEN return
           ELSE go to LOOP4:
COL:       Set column counter to 0FFH (column 0 minus 1)
LOOP2:     Increment row counter
           Mask counter to select row LEDs
           Output count to light IR LED
           Short delay
           Input from touch pad return to see if beam broken
           IF beam being broken
             THEN combine row number with row/column register
             Mask status word (H) to show a touch
           ELSE IF column counter at 0FH
             THEN return
           ELSE go to LOOP2:
PNTDAT:    Put row/column data in (L)
           Return

```

PROMEN

PROERR: Sound error horn if executed here
 PROMEN: Clear flag to verify there is a current entry
 PRO2MEN: Wait for pad being touched
 IF invalid touch
 THEN go to PROERR:
 IF sample menu in pad
 THEN go to PROERR:
 Get location of correct table
 SNDCHK: IF sound ON/Off selected
 THEN toggle sound setting
 Go to PRO2MEN:
 RANCHK: IF ranging ON/OFF selected
 THEN toggle ranging setting
 Go to PRO2MEN:
 RRCHK: IF ramp rate selected
 THEN get input from ramp bar on pad
 Store value in correct table
 Go to PRO2MEN:
 LRD: IF left (A) ranging distance selected
 THEN get input from range bar on pad
 Store value in correct table
 Go to PRO2MEN:
 RRD: IF right (B) ranging distance selected
 THEN get input from range bar on pad
 Store value in correct table
 Go to PRO2MEN:
 FRD: IF front (C) ranging distance selected
 THEN get input from range bar on pad
 Store value in correct table
 Go to PRO2MEN:
 BRD: IF back (D) ranging distance selected
 THEN get input from range bar on pad
 Store value in correct table
 Go to PRO2MEN:
 DEFAREA: IF define area selected
 THEN get input from range bar on pad
 DEFOK1: Store value as current entry number
 Get upper left corner of area
 Get lower right corner of area
 DEFOK2: Locate area address in memory
 DEFOK3: Store row/col min/max values in entry
 Mark menu control word as NOT empty
 Go to PRO2MEN:
 SELAREA: IF select area was chosen
 THEN get input from range bar on pad
 SELOK1: Store value as current entry number
 SELOK2: Locate area address in memory
 SELOK3: Store pointer to area address
 Go to PRO2MEN:

(Continued on next page)

PROMEN

```

PROERR:    Sound error horn if executed here
PROMEN:    Clear flag to verify there is a current entry
PRO2MEN:   Wait for pad being touched
            IF invalid touch
                THEN go to PROERR:
            IF sample menu in pad
                THEN go to PROERR:
            Get location of correct table
SNDCHK:    IF sound ON/Off selected
                THEN toggle sound setting
                Go to PRO2MEN:
RANCHK:    IF ranging ON/OFF selected
                THEN toggle ranging setting
                Go to PRO2MEN:
RRCHK:     IF ramp rate selected
                THEN get input from ramp bar on pad
                Add 1 and store value in correct table
                Go to PRO2MEN:
LRD:       IF left (A) ranging distance selected
                THEN get input from range bar on pad
                Double, add USBTOP and store in correct table
                Go to PRO2MEN:
RRD:       IF right (B) ranging distance selected
                THEN get input from range bar on pad
                Double, add USBTOP and store in correct table
                Go to PRO2MEN:
FRD:       IF front (C) ranging distance selected
                THEN get input from range bar on pad
                Double, add USBTOP and store in correct table
                Go to PRO2MEN:
BRD:       IF back (D) ranging distance selected
                THEN get input from range bar on pad
                Double, add USBTOP and store in correct table
                Go to PRO2MEN:
DEFAREA:   IF define area selected
                THEN get input from range bar on pad
DEFOK1:    Store value as current entry number
            Get upper left corner of area
            Get lower right corner of area
DEFOK2:    Locate area address in memory
DEFOK3:    Store row/col min/max values in entry
            Mark menu control word as NOT empty
            Go to PRO2MEN:
SELAREA:   IF select area was chosen
                THEN get input from range bar on pad
SELOK1:    Store value as current entry number
SELOK2:    Locate area address in memory
SELOK3:    Store pointer to area address
            Go to PRO2MEN:

```

(Continued on next page)

(Continuation of PROMEN:)

```

LMTR:      IF left motor speed selected
            THEN set flag for left motor data
            Go to MOK1:
RMTR:      IF right motor speed selected
            THEN clear flag for left motor data
            Go to MOK1:
            ELSE go to DUR:
MOK1:      IF flag set for left motor data
            THEN set left motor speed
            ELSE go to RSET:
RSET:      Set right motor speed
            Go to PRO2MEN:
DUR:      IF duration selected
            THEN IF no entry selected
                THEN go to PROERR:
DOK1:      Set input from duration bar on pad
            Store duration with current entry
            Go to PRO2MEN:
RESMEN:    IF reset menu selected
            THEN wait for another input for verification
            IF reset selected again
                THEN Set global parameters to defaults
                Set all ten areas to defaults
                Mark control word for empty menu
                Go to PRO2MEN:

```

CHKTBL

```

CHKTBL:    Set ENTRY to 0
            Set memory pointer at the first entry of the table
ROWMIN:    IF touched row < minimum row
            THEN go to NXTENT1:
COLMIN:    IF touched column < minimum column
            THEN go to NXTENT1:
            Increment memory pointer
ROWMAX:    IF touched row > maximum row
            THEN go to NXTENT2:
COLMAX:    IF touched column > maximum column
            THEN go to NXTENT2:
VALID:     Increment memory pointer
            Store pointer value in MTRADDR
            Increment ENTRY
            Return
NXTENT1:   Increment memory pointer
NXTENT2:   Increment memory pointer to next motion data
            Increment ENTRY
            IF ENTRY is 10 (all 10 entries checked)
                THEN return
            ELSE go to ROWMIN:

```

INITMTR

INITMTR: Set memory pointer to MTRADDR
Read left/right motor target speed from table
Mask for right target speed only (low nibble)
Store in RMTS
Mask for left target speed only (high nibble)
Rotate left data to low nibble
Store in LMTE
Increment memory pointer
Read DURATION from table
Get RAMPENT value
Store in CNTRAMP
Return

(Continuation of PROMEN:)

```
LMTR:      IF left motor speed selected
            THEN set flag for left motor data
              Go to MOK1:
RMTR:      IF right motor speed selected
            THEN clear flag for left motor data
              Go to MOK1:
            ELSE go to DUR:
MOK1:      IF flag set for left motor data
            THEN set left motor speed
            ELSE go to RSET:
RSET:      Set right motor speed
            Go to PRO2MEN:
DUR:      IF duration selected
            THEN IF no entry selected
                  THEN go to PROERR:
DOK1:      Get input from duration bar on pad
            Store duration with current entry
            Go to PRO2MEN:
RESMEN:    IF reset menu selected
            THEN wait for another input for verification
              IF reset selected again
                THEN Set global parameters to defaults
                  Set all ten areas to defaults
                  Mark control word for empty menu
                  Go to PRO2MEN:
```

CHKTBL

CHKTBL: Set ENTRY to 0
Set memory pointer at the first entry of the table

ROWMIN: IF touched row < minimum row
THEN go to NXTENT1:

COLMIN: IF touched column < minimum column
THEN go to NXTENT1:
Increment memory pointer

ROWMAX: IF touched row > maximum row
THEN go to NXTENT2:

COLMAX: IF touched column > maximum column
THEN go to NXTENT2:

VALID: Increment memory pointer
Store pointer value in MTRADDR
Increment ENTRY
Return

NXTENT1: Increment memory pointer

NXTENT2: Increment memory pointer to next motion data
Increment ENTRY
IF ENTRY is 10 (all 10 entries checked)
THEN return
ELSE go to ROWMIN:

INITMTR

INITMTR: Set memory pointer to MTRADDR
Read left/right motor target speed from table
Mask for right target speed only (low nibble)
Store in RMTS
Mask for left target speed only (high nibble)
Rotate left data to low nibble
Store in LMTS
Increment memory pointer
Read DURATION from table
Get RAMPCNT value
Store in CNTRAMP
Return

RGTFWD, RGTREV, LFTFWD, LFTREV

RGTFWD: IF CNTRAMP is 0
 THEN increment RMCS
 Store in RMOTOR
 Return

RGTREV: IF CNTRAMP is 0
 THEN decrement RMCS
 Store in RMOTOR
 Return

LFTFWD: IF CNTRAMP is 0
 THEN increment LMCS
 Store in LMOTOR
 Return

LFTREV: IF CNTRAMP is 0
 THEN decrement LMCS
 Store in LMOTOR
 Return

```

;*****
;*          EASY CHAIR THE BEST IN CHAIRS          *
;*****
;
;200 = BASE EQU 4000H ;BASE ADDRESS OF MONITOR
;400 = MONRAM EQU 5A00H ;BASE ADDRESS OF RAM FOR MONITOR
;5FFF = ENDRAM EQU 5FFFFH ;END OF RAM MEMORY
;0100 = MRSIZ EQU 0100H ;MONITOR RAM SIZE
;5B00 = USRAM EQU MONRAM+100H ;FIRST BYTE OF USER RAM
;00FF = EOL EQU 0FFH ;END OF STRING (LINE) CHARACTER
;0007 = BEL EQU 07H ;BBBBBBBBBBBBBBBB
;000D = CR EQU 0DH ;CARRIAGE RETURN
;000A = LF EQU 0AH ;LINE FEED
;0016 = HOME EQU 01CH ;CURSOR UP AND LEFT
;001B = ESC EQU 01BH ;ESCAPE
;007F = RUB EQU 07FH ;RUBOUT
;0013 = XOFF EQU 013H ;DC3 (X-OFF)
;0011 = XON EQU 011H ;DC1 (X-ON)
;00AF = NWIDTH EQU 0FH ;CONTROLS THE WIDTH OF "DUMP" "PUNCH"
;
;          ;COMMANDS:
;          ;          0FH = 16 BYTES, 32 COLUMNS
;          ;          07H = 8 BYTES, 26 COLUMNS
;
;0020 = TIME0 EQU 20H ;8253 TIMER ZERO
;0021 = TIME1 EQU 21H ;TIMER ONE
;0022 = TIME2 EQU 22H ;TIMER TWO
;0023 = TIMCTL EQU 23H ;8253 CONTROL REGISTER
;0018 = PIAA EQU 010H ;PIA A DATA REGISTER
;0019 = PIAB EQU 011H ;PIA B DATA REGISTER
;001A = PIAC EQU 012H ;PIA C DATA REGISTER
;001B = PIAD EQU 013H ;PIA D DATA REGISTER
;001C = PIAE EQU 014H ;PIA E DATA REGISTER
;001D = PIAF EQU 015H ;PIA F DATA REGISTER
;001E = PISCNTL EQU 016H ;#2 PIA CONTROL REGISTER
;001F = PIACNTL EQU 017H ;#1 PIA CONTROL REGISTER
;0001 = SERCON EQU 01H ;ADIA CONTROL REGISTER
;0000 = SERDAT EQU 00H ;ADIA DATA REGISTER
;0001 = PROMSK EQU 00000001B ;PROGRAM MENU DETECT
;00FF = TRUE EQU 0FFH
;0000 = FALSE EQU 00H
;0001 = BEAMSK EQU 00000001B ;MASK FOR DETECT (PIAB B0)
;0010 = ROWMSK EQU 00010000B ;MASK FOR ROW SELECT (PIAA)
;0020 = COLMSK EQU 00100000B ;COLUMN SELECT (PIAA)
;0040 = EXMSK EQU 01000000B ;EXTRA SELECT (PIAA)
;          ;EXTRA SELECT INCLUDES:
;          ;MENU SELECT LEDS/TRANS.
;          ;ULTRASONIC DIRECTION LEDS
;          ;MASK FOR A TOUCH (HI)
;00B0 = TOUCH EQU 10000000B
;001E = VALMEN EQU 00011110B ;VALID MENU NUMBER MASK
;          ;NOTE THAT BIT 0 IS USED TO SIGNAL
;          ;THE PROGRAM MENU.
;0040 = MENERR EQU 01000000B ;MASK MENU ERROR
;0020 = PADERR EQU 00100000B ;MASK LED/TRANS. ERROR
;0001 = PADLED EQU 00000001B ;MASK FOR PAD ERROR LED (OUTPUT)
;0002 = XENLED EQU 00000010B ;" " MENU " "
;0004 = LUSLED EQU 00000100B ;(SAME) LEFT U.S.
;0008 = RUSLED EQU 00001000B ;(SAME) RIGHT U.S.

```

```

0010 = BUSLED EQU 00010000B ;(SAME) BACK U.S.
0020 = FUSLED EQU 00100000B ;(SAME) FRONT U.S.
0080 = PADLOOP EQU 10000000B ;PAD +5V LOOP (CONNECTED?)
0040 = LUSLOOP EQU 01000000B ;LEFT U.S. ....
0020 = RUSLOOP EQU 00100000B ;RIGHT U.S. ....
0010 = BUSLOOP EQU 00010000B ;BACK U.S. ....
0008 = FUSLOOP EQU 00001000B ;FRONT U.S. ....
0004 = PJMASK EQU 00000100B ;MASK FOR INPUT FROM PAD/JOYSTICK
0008 = MTRSTOP EQU 08H ;1/2 OF 160 VALUE (USED TO STOP)
0004 = RSONOFF EQU 00000100B ;FLAG TO SIGNAL RANGING ON
0002 = SONOFF EQU 00000010B ;FLAG TO SIGNAL SOUND ON
0001 = EMPTMEN EQU 00000001B ;FLAG TO SIGNAL EMPTY MENU
;NOT PROGRAMMED
000C = UESTOP EQU 0CH ;ULTRASONIC STOPPING DIST
;0.6 INCHES
0004 = STCRAMP EQU 04H ;STOPPING RAMP RATE

;BEGIN EQUATES FOR PROGRAMMING MENU (SEE PROMEN)
;NUMBERS REPRESENT LOCATIONS OF MENU CHOICES ON THE
;PROGRAMMING MENU. MS NIBBLE: ROW, LS NIBBLE: COLUMN
0021 = SOUND EQU 21H ;SOUND ON/OFF
0021 = RANGE EQU 31H ;RANGING ON/OFF
0041 = RAMP EQU 41H ;ENTER RAMP RATE
0051 = BACKR EQU 51H ;BACK...
0061 = FRONTR EQU 61H ;FRONT...
0071 = LEFTR EQU 71H ;LEFT RANGING DISTANCE
0081 = RIGHTR EQU 81H ;RIGHT...
00A1 = DEFINE EQU 0A1H ;DEFINE AREA
00B1 = SELECT EQU 0B1H ;SELECT AREA FOR EDITING
00C1 = LEFTM EQU 0C1H ;LEFT MOTOR SPEED
00D1 = RIGHTM EQU 0D1H ;RIGHT...
00E1 = TIME EQU 0E1H ;DURATION TIME
000F = RESET EQU 0FH ;RESET MENU SELECTION
00EB = BAR1BEG EQU 0EBH
00EF = BAR2BEG EQU 0EFH
;
;=====
; VECTORS FOR HARDWARE INTERRUPTS

3800 = RST0 EQU USRRAM+ 000H ; NOT USED - MONITOR RESET
3808 = RST1 EQU USRRAM+ 008H ;
3810 = RST2 EQU USRRAM+ 010H ;
3818 = RST3 EQU USRRAM+ 018H ;
3820 = RST4 EQU USRRAM+ 020H ;
3824 = TRAP EQU USRRAM+ 024H ;
3828 = RST5 EQU USRRAM+ 028H ;
382C = RST5B EQU USRRAM+ 02CH ;
3830 = RST6 EQU USRRAM+ 030H ;
3834 = RST6B EQU USRRAM+ 034H ;
3838 = RST7 EQU USRRAM+ 038H ;
383C = RST7B EQU USRRAM+ 03CH ;
;=====
; RST 0 ENTRY POINT - POWER UP RESET ;RST 0
4000 ORG BASE+0
4000 JNZ BP,ENDRAM+1
4003 JWF INITIAL

```

```

4006 00      NOP
4007 00      NOP
;=====
; RST 1 ENTRY POINT
4008      ORG      BASE+08H      ; RST 1
4008 C3085B      JMP      RST1
400B 0000000000      DB      0,0,0,0,0
;=====
; RST 2 ENTRY POINT
4010      ORG      BASE+10H      ; RST 2
4010 C3105B      JMP      RST2
4013 0000000000      DB      0,0,0,0,0
;=====
; RST 3 ENTRY POINT
4018      ORG      BASE+18H      ; RST 3
4018 C3185B      JMP      RST3
401B 0000000000      DB      0,0,0,0,0
;=====
; RST 4 ENTRY POINT
4020      ORG      BASE+20H      ; RST 4
4020 C3205B      JMP      RST4
4023 00      NOP
;=====
; TRAP ENTRY POINT
4024      ORG      BASE+24H      ; TRAP
4024 C3245B      JMP      TRAP
4027 00      NOP
;=====
; RST 5 ENTRY POINT
4028      ORG      BASE+28H      ; RST 5
4028 C3285B      JMP      RST5
402B 00      NOP
;=====
; RST 5.5 ENTRY POINT
402D      ORG      BASE+2CH      ; RST 5.5
402D C32C5B      JMP      RST55
402F 00      NOP
;=====
; RST 6 ENTRY POINT
4030      ORG      BASE+30H      ; RST 6
4030 C3305B      JMP      RST6
4033 00      NOP
;=====
; RST 6.5 ENTRY POINT
4034      ORG      BASE+34H      ; RST 6.5
4034 C3345B      JMP      RST65
4037 00      NOP
;=====
; RST 7 ENTRY POINT
4038      ORG      BASE+38H      ; RST 7
4038 C3385B      JMP      RST7
403B 00      NOP
;=====
; RST 7.5 ENTRY POINT      ; RST 7.5
403D      ORG      BASE+3CH      ; RST 7.5
403D C33C5B      JMP      RST75

```

```

403F 00          NOP
;=====
4040          ORG      BASE+40H
;
; POWER-UP AND RESET INITIALIZATION
; NOW INITIALIZE USART CHIP
;
4040 310060      START: LXI      SP,ENDRAM+1      ;INIT. SP FOR MONITOR
4043 3E82          MVI      A,82H      ;FORCE USART TO EXPECT CMND WORD
4045 D301          OUT      SERCON
4047 3E40          MVI      A,040H      ;NOW MAKE USART TO EXPECT MODE WORD
4049 D301          OUT      SERCON
404B DECE          MVI      A,0CEH      ;MODE BYTE -
404D D301          OUT      SERCON      ; 11 00 11 10
404F 3E17          MVI      A,037H      ;COMMAND BYTE -
4051 D301          OUT      SERCON      ; 0 0 1 1 0 1 1 1
;
; INITIALIZE TIMER CHIP TO GENERATE 16X BAUDRATE FOR
;
4053 210E00          LXI      H,000EH      ; 7200 BAUD
;                                     ;(1/(16*7200))/0.1/3,2 MHz
4056 3E76          MVI      A,76H      ;INIT TIMER 1 TO DIVIDE BY N
4059 D323          OUT      TIMCTL
405B 7D          MOV      A,L
405D D321          OUT      TIME1
405F 7D          MOV      A,H
4061 D321          OUT      TIME1
;
; INITIALIZE MONITOR RAM PERTAINING TO CONSOLE I/O
;
4060 AF          XRA      A      ;MAKE A ZERO
4061 32005A          STA      ECHOFL      ; 0=ECHO 1=NO ECHO
4064 3E0F          MVI      A,MWIDTH
4066 32015A          STA      WIDTH
;
; PRINT STARTUP MESSAGE - ALSO EFFECTIVE WAY TO WAIT A FEW
;                               CHAR PERIODS WHILE DOUBLE BUFFERED
;                               INPUT SETTLES.
;
4069 111A55          LXI      D,CLE      ;CLEAR SCREEN
406C CD3F49          CALL     MSG
406F 11A652          LXI      D,STMSG
4072 CD3F49          CALL     MSG
4075 DB00          IN      SERDAT      ;EAT POSSIBLE GARBAGE CHARACTER
;
; INITIALIZE REMAINDER OF MONITOR RAM
;
4077 3EFP          MVI      A,EOL      ; ON POWER UP NO ANSWER
4079 32365A          STA      MISCBF
407D 210032          LXI      H,3200H
407F 220C5A          SHLD     CLKBCD
4082 CD4247          CALL     BCDTBIN
4085 220E5A          SHLD     CLKBIN
4089 CD1F47          CALL     M50128
408B 220A5A          SHLD     D50DIV
;
408D 220A5A          LXI      H,ENDRAM      ;UNCOMMENT FOR MEM TEST

```

```

;      LXI      D,USRRAM      ; ON RESET/POWER UP
;      CALL     MTO
;
; COMMAND LEVEL - GET CHARACTER; JUMP TO APPROPRIATE ROUTINE

408E CDDF49  COMND:  CALL     SETJMP      ; RUBOUT ABORTED COMNDS COME HERE

4091 119A52      LXI      D,PRMPT      ; PRINT COMMAND PROMPT
4094 CD3F49      CALL     MSG
4097 CDD247      CALL     CI
;      ANI      7FH      ; PUT IN IF UCASE TAKEN OUT
409A CD024A      CALL     UCASE      ; CONVERT LOW TO UP CASE & STRIPS PARITY
;
; SEQUENCE BELOW IS KLUDGE TO ALLOW CR AND ? AS ONE CHAR COMNDS
;
409D FE0D      CPI      CR      ; SPECIAL CASE, (CR) IS NOP THAT DOES NOT
409F C8E40      JZ       COMND      ; CLEAR THE ANSWER
40A2 118E40      LXI      D,COMND      ; ADDR FOR PSEUDO CALL COMPLETED BY PCHL
40A5 D5      PUSH     D
40A6 FE3F      CPI      '?'      ; SPECIAL CASE '?', MUST NOT CLEAR
40AB CADD40      JZ       ASK      ; ANSWER FIRST.
;
; NOW FOR THE REAL COMMANDS...
;
40AB 67      MOV      H,A      ; PUT FIRST CHAR INTO H
40AC CDD247      CALL     CI      ; GET SECOND CHAR
;      ANI      07FH      ; UNCOMMENT IF CALL UCASE REMOVED
40AF CD024A      CALL     UCASE
40B2 6F      MOV      L,A      ; PUT SECOND CHAR INTO L
40B3 CDEE49      CALL     SPACE      ; GOD KNOWS WHAT FOR...
40B6 011550      LXI      B,CMDS      ; SCAN COMMAND TABLE...COMND IN H&L
40B9 0A      CMDNXT: LDAX     B      ; GET COMMAND FROM TABLE
40BA 57      MOV      D,A      ; GET FIRST LETTER
40BB 03      INX      B      ; POINT TO SECOND LETTER
40BC 0A      LDAX     B      ; GET SECOND LETTER
40BD 5F      MOV      E,A      ;
40BE 03      INX      B      ; POINT TO LOWER BYTE OF ADDRESS
40BF C0FF47      CALL     CMP16      ; COMPARE TO COMND TYPED
40C2 CAD040      JZ       CMDFND      ; FOUND IT
40C5 03      INX      B      ; SKIP OVER ADDR OF COMMAND JUST CHECKED
40C6 03      INX      B      ; POINT TO UPPER BYTE OF ADDR THEN NXT CMD
40C7 7A      MOV      A,D      ; CHECK FOR END OF TABLE
40C8 B3      ORA      E      ;
40C9 C2B940      JNZ      CMDNXT      ; NOT END...TRY NEXT ENTRY
40CC CDAB49      ERROR: CALL     PRBAD      ; PRINT ERROR MESSAGE AND RETURN. "COMND"
40CF C9      RET      ; IS ON STACK AS RETURN ADDR FOR COMMAND
;      ; NOTE ALL THE COMMANDS USE ERROR LABEL.
;
40D0 3EFF      CMDFND: MVI      A,EOL      ; CLEAR ANSWER
40D2 32365A      STA      MISCBF
40D5 0A      LDAX     B      ; GET LOWER BYTE OF ADDRESS
40D6 5F      MOV      E,A      ;
40D7 03      INX      B      ; POINT TO LOWER BYTE
40D8 0A      LDAX     B      ; GET UPPER BYTE
40D9 57      MOV      D,A      ;

```

```

40DA 7C      MOV      A,H      ;COMMAND EXPECTS FIRST LETTER IN A REG
40DB EB      XCHG          ;
40DC E9      PCHL          ;
;
;***** END OF COMMAND LEVEL *****
;*****BEGINNING OF ASK*****
;
;      PRINT      ONE BYTE NOTE LEFT BY LAST COMMAND
;
40DD CDEE49  ASK:      CALL      SPACE
40E0 11365A      LXI      D,MISCBF
40E3 CD3F49      CALL      MSG
40E6 C9      RET
;*****END OF ASK*****
;*****BEGINNING OF HELP*****
;
;      HELP
40E7 118051  HELP:      LXI      D,PHELP
40EA CD3F49      CALL      MSG
40ED C9      RET
;*****END OF HELP*****
;*****BEGINNING OF GOTO*****
;
;      GOTO ROUTINE - STARTS EXECUTION IN MEMORY LOCATION
;
40EE CDD948  GOTO:      CALL      GHW      ;GET HEX WORD
40F1 DACC40      JC      ERRER      ;
40F4 CD5849      CALL      OKCK      ;
40F7 D8      RC      ;
40F8 E5      PUSH      H
40F9 CD0C48      CALL      CRLF
40FC AF      XRA      A
40FD CDF247      CALL      CD
4100 CDF247      CALL      CD
4103 E1      POP      H
4104 E9      PCHL          ;          AND GO
;*****END OF GOTO*****
;*****BEGINNING OF MEMTST*****
;
4105 CD7848  MEMTST:  CALL      FROMTO      ;GET FROM AND TO ADDRESSES
4108 DACC40      JC      ERRER      ;
410B EB      XCHG          ;
410C CD5849      CALL      OKCK      ;CHECK WITH USER BEFORE STARTING
410F DA7741      JC      MTEND      ;
4112 4C      MTO:      MOV      C,H      ;STOP AT XX?? WHERE XX-1 IS THE
4113 0C      INR      C      ;UPPER BYTE OF THE USERS TO ADDR
4114 0600      MVI      B,00H      ; ALSO USE OF COUNTER
4116 C5      PUSH      B
4117 0600      MVI      B,0      ;CLEAR B PATTERN MODIFIER
4119 62      MT1:      MOV      H,D      ;
411A 6B      MOV      L,E      ;

```

```

411B 7D      MTFILL: MOV    A,L      ;LOW BYTE TO ACCUM.
411C AC      XRA      H      ;XOR WITH HIGH BYTE
411D AB      XRA      B      ;XOR WITH PATTERN
411E 77      MOV      M,A      ;STORE IN ADDR
411F 23      INX      H      ;INCREMENT ADDR
4120 7C      MOV      A,H      ;LOAD HIGH BYTE OF ADDR
4121 B9      CMP      C      ;COMPARE WITH STOP ADDR
4122 C21B41  JNZ      MTFILL      ;LOOP IF NOT DONE
;
; READ AND CHECK TEST DATA
;
4125 62      MOV      H,D
4126 6B      MOV      L,E      ;GET STARTING ADDR
4127 7D      MTTST: MOV    A,L      ;GET LOW BYTE
4128 AC      XRA      H      ;XOR WITH HIGH BYTE
4129 AB      XRA      B      ;XOR WITH MODIFIER
412A C5      PUSH     B
412B 47      MOV      B,A
412C 7E      MOV      A,M
412D B8      CMP      B      ;COMPARE WITH MEMORY LOCATION
412E C25941  JNZ      MTFXIT      ;ERRER EXIT
4131 C1      POP      B
4132 23      INX      H      ;UPDATE MEMORY ADDRESS
4133 7C      MOV      A,H      ;GET HIGH BYTE
4134 B9      CMP      C      ;COMPARE WITH STOP ADDR
4135 C22741  JNZ      MTTST      ;LOOP BACK
4138 3A015A  LDA      WIDTH      ;GENERATE ((WIDTH+1)*4)-1
413B 37      STC
413C 17      RAL
413D 37      STC
413E 17      RAL
413F A0      ANA      B      ;CHECK FOR TIME FOR CRLF
4140 C0C4B  CZ      CRLF      ;CRLF IF RUNNING OUT OF LINE
4143 04      INR      B      ;UPDATE MODIFIER
4144 EB      XCHG
4145 3E21  MVI      A,'!'      ;PRINT PASS DONE MESSAGE
4147 CDF247  CALL     CO
414A EB      XCHG
414B C1      POP      B
414C 05      DCR      B
414D C5      PUSH     B
414E C21941  JNZ      MT1      ;RESTART WITH NEW MODIFIER
4151 C1      POP      B
4152 112351  LXI      D,MTGOOD
4155 CD3F49  CALL     MSG
4158 C9      RET
; FOR 255 TIMES THEN TO CMDS
4159 113A51  MTFXIT: LXI    D,MTERR      ;PRINT ERRER ADDRESS
415C CD3F49  CALL     MSG
415F CD7849  CALL     PHW
4162 115C51  LXI      D,MTREAD
4165 CD3F49  CALL     MSG
4168 CD8349  CALL     PHB
416B 115351  LXI      D,MTWROT
416E CD3F49  CALL     MSG
4171 78      MOV      A,B
4172 CD8349  CALL     PHB

```

```

4175 C1          POP      B
4176 C1          POP      B

;
4177 C9          MTEND:   RET                ;RETURN TO COMMAND LOOP
;
;
;*****END OF MEMTST*****
;*****BEGINNING OF TEST BOARD*****

4178 CD0C48      TSTBRD:  CALL    CRLF          ;THIS ROUTINE ALLOWS THE
4178 115750      LXI      D,MTSBRD          ;USER TO DO A HARDWARE
417E CD3F49      CALL    MSG                ;CHECK OF THE PIAS AND
                                           ;TIMER CHIPS

4181 3E77        MVI      A,77H
4183 D323        OUT      TIMCTL           ;TIMER 1
4185 3EB7        MVI      A,0B7H
4187 D323        OUT      TIMCTL           ;TIMER 2
4189 97          SUB      A                ;TO DIVIDE BY
418A D321        OUT      TIME1
418C D322        OUT      TIME2
418E 3E20        MVI      A,20H
4190 D321        OUT      TIME1
4192 D322        OUT      TIME2

4194 3EB0        MVI      A,80H
4196 D313        OUT      PIACNTL          ;SET PIAA
4198 D343        OUT      PIBCNTL          ;AND PIAB TO
419A 3E30        MVI      A,30H
419C D323        OUT      TIMCTL
419E D310        LOOPA:  OUT      PIAA      ;LOOP THROUGH
41A0 D312        OUT      PIAC            ; SHOULD APPEAR
                                           ;AS STAIRSTEP ON
41A2 D341        OUT      PIAE            ;LOGIC ANALIZER
41A4 3C          INR      A
41A5 CD0E4A      CALL    HRTBEAT
41A8 C39E41      JMP      LOOPA           ;LOOP FOREVER

;*****END OF TEST BOARD *****
;*****BEGINNING OF MEMED*****
;
; MEMED - HEXADECIMAL MEMORY EDITOR
;
41AB 115350      MEMED:  LXI      D,EDM2    ;PRINT "CR, LF, ("
41AE CD3F49      CALL    MSG

41B1 CDD948      CALL    GHW
41B4 D2C041      JNC     OK                ;GET HEX WORD INTO HL, JUMP IF VALID

41B7 FE2F        CPI      '/'            ;BAD CHAR RECEIVED - WAS IT "/"
41B9 C8          RZ                      ;GO BACK TO COMMAND LEVEL IF SO

41BA CDAB49      CALL    PRBAD            ;PRINT "WHAT ?"
41BD C3AB41      JMP      MEMED           ;THEN TRY AGAIN

```

```

41C0 CD3642    OK:    CALL    DISCON    ;DISPLAY CONTENTS OF LOCATION
41C3 CDC941          CALL    EDIT      ;THEN BEGIN EDITING
41C6 C3AB41          JMP     MEMED     ;LOOP IF EDIT RETURNS
;
;          END      MEMED
;
;
; GET EITHER A NEW HEX BYTE TO BE WRITTEN WHERE HL POINTS,
; FOLLOWED BY ANOTHER COMMAND, OR JUST ANOTHER COMMAND.
;
41C9 CDF048    EDIT:   CALL    GH8      ;GET THE NEW HEX BYTE IF TYPED
41CC D2F441          JNC     EDBYTE    ;GOOD BYTE TYPED - PUT IN MEMORY
41CF FE27          CPI     027H      ;DOES USER WANT LITERAL CHARACTER ?
41D1 CAEF41          JZ      EDLIT     ; YEP...
41D4 FE5E          CPI     'A'      ;DOES USER WANT CONTROL CHARACTER ?
41D6 C2FD41          JNZ     NEXT      ;NOPE...MUST BE COMMAND OR ERROR...
41D9 CDD247          CALL    CI       ;GET CHAR
41DC E67F          ANI     07FH      ;STRIP PARITY
41DE FE40          CPI     040H      ;SEE IF MAKES SENSE...
41E0 DA2042          JC      EDBAD     ;DUMMY
41E3 FE60          CPI     060H      ;FIGURE OUT WHAT TO SUBTRACT...
41E5 DAEA41          JC      EDUC      ;IS UPPER CASE...OK AS IS
41E8 D620          SUI     020H      ;LOWER CASE...MUST BE MOVED DOWN
41EA D640          EDUC:   SUI     040H ;CONVERT TO CONTROL CHAR
41EC C3F441          JMP     EDBYTE    ;
41EF CDD247    EDLIT:   CALL    CI       ;GET CHAR
41F2 E67F          ANI     07FH      ;BETTER STRIP PARITY
41F4 77          EDBYTE:  MOV     M,A    ;ELSE STORE IT IN MEMORY
41F5 CDEE49          CALL    SPACE     ;SPACE TO REINFORCE THAT ONCE TWO DIGITS
;                                ; ARE ENTERED, LOCATION IS CHANGED.
41F8 CDD247          CALL    CI       ;AND GET ANOTHER CHAR & ECHO IT
41FB E67F          ANI     7FH       ;KILL TOP BIT
41FD FE00          NEXT:   CPI     CR    ;CARRIAGE RETURN?
41FF C20642          JNZ     E1
4202 23          INX     H
4203 C32342          JMP     PR        ;YES- PRINT NEXT LOCATION
4206 FE20          E1:     CPI     ' '   ;OR BLANK
4208 C20F42          JNZ     E2
420B 23          INX     H
420C C32342          JMP     PR        ;YES- DO THE SAME
420F FE2E          E2:     CPI     '.'   ; PERIOD?
4211 CA2342          JZ      PR        ;PRINT CURRENT LOCATION
4214 FE2D          E3:     CPI     '-'   ; DASH?
4216 C21D42          JNZ     E4
4219 2B          DCX     H
421A C32342          JMP     PR        ;YES - PRINT PREVIOUS LOCATION
421D FE2F          E4:     CPI     '/'   ;SLASH?
421F C8          RZ          ;EDIT ALL DONE IF SO
4220 CDA849    EDBAD:   CALL    PRBAD   ;IF NONE OF THE ABOVE, PRINT "WHAT ?"
4223 CD2942    PR:      CALL    DISMEM  ;DISPLAY THE NEW CURRENT MEMORY LOCATION
4226 C3C941          JMP     EDIT      ;AND LOOP

; PRINT CR, LF THEN AN ( FOLLOWED BY THE CONTENTS OF HL IN HEX.

4229 115350    DISMEM:  LXI     D,EDM2 ;DO CR,LF, "("

```

```

422C CD3F49      CALL    MSG
422F CD7849      CALL    PHW
4232 CD3642      CALL    DISCON
4235 C9          RET
; **** DISCON ****
;
; PRINT ')' = ' FOLLOWED BY THE CONTENTS OF THE MEMORY LOC.
; POINTED TO BY HL
;
4236 114E50      DISCON: LXI    D,EDM1 ;
4239 CD3F49      CALL    MSG ;
423C 7E          MOV     A,M ;GET CONTENTS OF MEM LOC.
423D CD8349      CALL    PHB ;PRINT IT
4240 114F50      LXI     D,EDM3 ;
4243 CD3F49      CALL    MSG ;
4246 E5          PUSH    H ;SAVE ADDRESS
4247 CD4548      CALL    DISASC ;CONVERT TO PRINTABLE
424A 7C          MOV     A,H ;PRINT ' ' OR '^'
424B CDF247      CALL    CO ;
424E 7D          MOV     A,L ;PRINT CHARACTER
424F CDF247      CALL    CO ;
4252 E1          POP     H
4253 CDEE49      CALL    SPACE ;
4256 C9          RET
;
; *****END OF MEMED*****
;
; *****BEGINNING OF LOADER*****
;
; HEX-FORMAT LOADER
;
; NOTE: RECORD LENGTH = 00 TAKEN AS EOF
4257 CDA048      LOADER: CALL    GBIAS ;GET BIAS
425A DACC40      JC      ERRER ;BAD CHAR - QUIT
425D 22025A      SHLD    BIAS ;STORE BIAS
4260 CD5849      CALL    OKCK ;CHECK WITH USER BEFORE JUMPING
4263 D8          RC      ;
4264 3A005A      LDA     ECHOFL ;SAVE ECHO FLAG
4267 32385A      STA     MISCBF+2; MISCBF & MISCBF+1 USED BY ANSWER
426A 3E11        MVI     A,XON ;START DATA COMING
426C 32005A      STA     ECHOFL ;NON-ZERO VALUE (XON) TURNS OFF ECHO
426F CDF247      CALL    CO ;
4272 CD9642      LOAD1: CALL    GETREC ;READ IN ONE REC, (A) = RECORD LENGTH
4275 B7          ORA     A ;SET Z-FLAG ON RECORD LENGTH
4276 3E47        MVI     A,'G' ;ANSWER TO QUESTION = GOOD
4278 CA8242      JZ      DONE ;IF LENGTH = 0 THEN DONE
427B 7A          MOV     A,D ;(D) = ERROR FLAG ON GETREC RETURN
427C B7          ORA     A ;SEE IF THE "ERROR" FLAG IS NON-ZERO.
427D CA7242      JZ      LOAD1 ;IF NOT, GO DO NEXT RECORD
4280 3E42        MVI     A,'B' ;STORE "BAD" FLAG IN ANSWER TO QUESTION
4282 32365A      DONE:  STA     MISCBF ;STORE GOOD/BAD STRING
4285 3EFF        MVI     A,EOL ;
4287 32375A      STA     MISCBF+1;
428A 3A385A      LDA     MISCBF+2;RESTORE ECHO FLAG
428D 32005A      STA     ECHOFL ;
4290 3E13        MVI     A,XOFF ;STOP FURTHER OUTPUT
4292 CDF247      CALL    CO ;

```

```

4295 C9          RET          ;RETURN TO COMMAND LEVEL
;
;          END      LOADER
;
; *** GETREC *** READ IN ONE RECORD
;
4296 CDB742      GETREC: CALL   FNDMRK  ;SKIP TO RECORD MARK
;
4299 CDD142      CALL   LGHB   ;GET THE RECORD LENGTH
429C 4F          MOV    C,A      ; INTO THE C REG.
429D CDD142      CALL   LGHB   ;GET LOAD ADDRESS FIELD INTO H & L
42A0 67          MOV    H,A      ;
42A1 CDD142      CALL   LGHB   ;
42A4 6F          MOV    L,A      ;
42A5 D5          PUSH   D      ;SAVE D&E
42A6 EB          XCHG      ;
42A7 2A025A      LHLD   BIAS    ;ADD BIAS
42AA 19          DAD     D      ;
42AB D1          POP     D      ;RESTORE D&E
42AC CDD142      CALL   LGHB   ;GET THE RECORD-TYPE BYTE AND IGNORE
42AF CDC442      CALL   DATA   ;PUT THE NEXT (C) BYTES INTO MEMORY
;                  ;STARTING WHERE HL POINTS
42B2 CDD142      CALL   LGHB   ;READ THE CHECKSUM BYTE
42B5 79          MOV    A,C      ;PUT THE RECORD LENGTH BACK INTO A REG.
42B6 C9          RET          ;RETURN FROM GETREC. (D) CONTAINS THE
;                  ; SUM OFF ALL HEX BYTES READ, AND SO
;                  ; IS EFFECTIVELY AN ERRER FLAG
;
;          END      GETREC
;
; *** FNDMRK *** - FIND RECORD MARK
;                  IGNORES ALL TEXT UNTIL ":" FOUND, THEN RET
;
42B7 CDD247      FNDMRK: CALL   CI      ;GET CHARACTER
42BA E67F        ANI     07FH    ;STRIP OFF 8TH BIT
42BC FE3A        CPI     ':'      ;
42BE C2B742      JNZ     FNDMRK   ;NOT RECORD MARK - GET NEXT CHAR
42C1 1600        MVI     D,0      ;CLEAR D REGISTER (ERRER ACCUMULATOR)
42C3 C9          RET          ;
;
;          END      FNDMRK
;
; *** DATA *** - INPUT ALL DATA BYTES
;                  (C) = NUMBER OF BYTES TO READ IN
;                  (D) = ERRER FLAG ACCUMULATOR MAINTAINED BY LGHB
;
42C4 41          DATA: MOV    B,C  ;COPY C REG. TO B
42C5 78          LOOP:  MOV    A,B  ;GET REMAINING BYTE COUNT
42C6 B7          ORA     A      ;GET FLAGS
42C7 C8          RZ      ;RETURN FROM SUBR. IF NONE LEFT
42C8 05          DCR     B      ;ELSE DECREMENT B REG.
42C9 CDD142      CALL   LGHB   ;GET BYTE FROM DATA FIELD
42CC 77          MOV    M,A      ;STORE IN MEMORY
42CD 23          DATA1: INX     H  ;BUMP POINTER
42CE C3C542      JMP     LOOP    ;GO BACK FOR NEXT CHAR.

```

```

;
;      END      DATA
;
;
; *** LGHB *** - LOADER GET HEX BYTE
;                SAME AS GHB EXCEPT ADDS BYTE GOTTEN TO ERROR
;                ACCUMULATOR IN D REGISTER
;
42D1 CDF048  LGHB:  CALL    GHB      ;GET BYTE
42D4 F5      PUSH    PSW      ;SAVE BYTE
42D5 82      ADD     D        ;ADD TO (D)
42D6 57      MOV     D,A      ;PUT SUM IN D-REG
42D7 F1      POP     PSW      ;RESTORE BYTE
42D8 C9      RET           ;
;
;      END      LGHB
;
;*****END OF LOADER*****
;*****BEGINNING OF DUMP*****
;
; DUMP1 IS AN ENTRY POINT FOR EXTERNAL USE OF ROUTINE
;
42D9 CD7B48  DUMP:  CALL    FROMTO ;GET BEGINNING ADDRESS AND BYTE COUNT
42DC DACC40  JC      ERROR    ;NON HEX CHAR TYPED - WHAT ?? ? ?? ?
42DF CD5849  CALL    OKCK     ;CHECK WITH USER BEFORE CONTINUING
42E2 DB      RC           ;
42E3 3A015A  DUMP1:  LDA     WIDTH ;GET WIDTH
42E6 47      MOV     B,A      ;
42E7 2F      CMA        ;ROUND DOWN STARTING ADDRESS
42E8 A5      ANA     L        ;
42E9 6F      MOV     L,A      ;
42EA 7B      MOV     A,E      ;ROUND UP ENDING ADDRESS
42EB B0      ORA     B        ;
42EC 5F      MOV     E,A      ;
42ED E5      PUSH    H        ;D&E=START-ENDING-1
42EE CDF649  CALL    SUB16    ;
42F1 2B      DCX     H        ;
42F2 D1      POP     D        ;
42F3 EB      XCHG      ;
42F4 CD0C48  CALL    CRLF     ;GO TO NEW LINE
42F7 CD7B49  CALL    PHW      ;PRINT MEMORY ADDRESS
42FA E5      PUSH    H        ;PUT RAM ADDRESS ON STACK
42FB 21365A  LXI     H,MISCBF ;GET BUFFER ADDRESS
42FE E3      XTHL       ;PUT BUFFER ADDRESS ON STACK
;                ;GET RAM ADDRESS OFF
;
;
; AT THIS POINT TOP OF STACK HAS BUFFER ADDRESS
;                H&L HAS RAM ADDRESS
;
42FF 7E      DI1:  MOV     A,M      ;GET BYTE
4300 23      INX     H        ;POINT TO NEXT BYTE IN RAM
4301 CDEE49  CALL    SPACE    ;
4304 CD8349  CALL    PHB      ;PRINT BYTE IN HEX
4307 E67F    ANI     07FH     ;STRIP PARITY
4309 FE20    CPI     020H     ;CHECK FOR PRINTABLE

```

```

430B DA1343      JC      DI3      ;NOT PRINTABLE - PRINT '.'
430E FE7F      DI2:  CPI      07FH      ;MAY BE PRINTABLE - CHECK FOR RUBOUT
4310 C21543      JNZ      DI4      ;NOPE..OK
4313 3E2E      DI3:  MVI      A,'.'      ;NOT PRINTABLE - REPLACE WITH SPACE
4315 E3      DI4:  XTHL      ;GET BUFFER ADDRESS
4316 77      MOV      M,A      ;PUT CHAR OR SPACE IN BUFFER
4317 23      INX      H      ;
4318 E3      XTHL      ;PUT BUFFER ADDRESS BACK
4319 13      INX      D      ;DECREMENT COUNT OF NUMBER OF BYTES LEFT
431A 7D      MOV      A,L      ;
431B A0      ANA      B      ;END OF LINE - PRINT ASCII AND CRLF
431C C2FF42      JNZ      DI1      ;KEEP GOING IF NOT AT END OF LINE
431F E3      DMPLIN: XTHL      ;GET BUFFER ADDRESS
4320 36FF      MVI      M,EOL      ;TERMINATE STRING
4322 21365A      LXI      H,MISCBF;POINT BACK TO START OF BUFFER
4325 E3      XTHL      ;PUT BUFFER ADDRESS BACK ON STACK
4326 CDEE49      CALL     SPACE      ;SPACE OVER A COUPLE
4329 CDEE49      CALL     SPACE      ;
432C 05      PUSH     D      ;
432D 11365A      LXI      D,MISCBF;POINT TO BEGINNING OF ASCII BUFFER
4330 CD3F49      CALL     MSG      ;PRINT ASCII BUFFER
4333 D1      POP      D      ;
4334 7B      MOV      A,E      ;
4335 B2      ORA      D      ;
4336 CA4243      JZ      DMPEND      ;DONE
4339 CD0C48      CALL     CRLF      ;
433C CD7849      CALL     PHW      ;PRINT MEMORY ADDRESS
433F C3FF42      JMP      DI1      ;
4342 E1      DMPEND: POP     H      ;CLEAN OFF STACK
4343 3EFF      MVI      A,EOL      ;CLEAR ANSWER...
4345 32365A      STA      MISCBF      ;
4348 C9      RET      ;

;
;*****END OF DUMP*****
;
;*****BEGINNING OF IOPORT*****
;
; IO - I/O PORT MANIPULATION
;
4349 CDF048      IOPORT: CALL     GH8      ;GET PORT NUMBER
434C DACC40      JC      ERROR      ;
434F 32385A      STA      MISCBF+2      ;DON'T TROMP ON EOL
4352 3EC9      MVI      A,0C9H      ;STORE RETURN
4354 32395A      STA      MISCBF+3      ;
4357 CDEE49      CALL     SPACE      ;
435A CDD247      CALL     CI      ;GET IOPORT COMMAND
435D CD024A      CALL     UCASE      ;STRIP PARITY
4360 CDEE49      CALL     SPACE      ;
4363 FE52      CPI      'R'      ;IF NOT R, CHECK OTHERS
4365 C26C43      JNZ      IOP1      ;
4368 CD7E43      CALL     IOPR      ;IOPORT READ ROUTINE
436B C9      RET      ;
436C FE57      IOP1:  CPI      'W'      ;IF NOT W, CHECK M
436E C27543      JNZ      IOP2      ;
4371 CD9E43      CALL     IOPW      ;IOPORT WRITE ROUTINE
4374 C9      RET      ;

```

```

4375 FE4D      IOP2:  CPI      'M'          ; IF NOT M, THEN WHAT DO
4377 C2CC40          JNZ      ERROR        ; YOU WANT ?
437A CDB943          CALL     IOPM         ; IOPORT MONITOR ROUTINE
437D C9          RET                      ;
;          END      IOPORT          ; MAIN PROGRAM
;
; IOPR - IOPORT READ SUBCOMMAND
;
437E 3EDB      IOPR:  MVI      A,0DBH        ; STORE "IN" INST
4380 32375A          STA      MISCBF+1      ;
4383 CD375A          CALL     MISCBF+1      ; GET BYTE FROM PORT
4386 11E850          LXI      D,IOPDA      ; PRINT 'DATA= '
4389 CD3F49          CALL     MSG          ;
438C CD8349          CALL     PHB          ; PRINT BYTE IN HEX
438F CDEE49          CALL     SPACE        ;
4392 CD4548          CALL     DISASC       ; PRINT BYTE IN ASCII
4395 7C          MOV      A,H            ;
4396 CDF247          CALL     CO          ;
4399 7D          MOV      A,L            ;
439A CDF247          CALL     CO          ;
439D C9          RET                      ;
;
; IOPW - IOPORT WRITE COMMAND
;
439E 11E850      IOPW:  LXI      D,IOPDA      ; PRINT 'DATA= '
43A1 CD3F49          CALL     MSG          ;
43A4 CDF048          CALL     GH8          ;
43A7 DACC40          JC       ERROR        ; BAD CHAR TYPED...
43AA CD5849          CALL     OKCK        ; CHECK TO BE SURE
43AD DB          RC          ; MUST HAVE GOOFED...
43AE F5          PUSH     PSW            ; SAVE DATA
43AF 3ED3          MVI      A,0D3H        ; STORE "OUT" INST
43B1 32375A          STA      MISCBF+1      ;
43B4 F1          POP      PSW            ; GET DATA BACK
43B5 CD375A          CALL     MISCBF+1      ; WRITE DATA
43B8 C9          RET                      ;
;
; IOPM - IOPORT MONITOR COMMAND
;
43B9 11EF50      IOPM:  LXI      D,IOPMM      ; PRINT '@ 50MS * '
43BC CD3F49          CALL     MSG          ;
43BF CDF048          CALL     GH8          ;
43C2 DACC40          JC       ERROR        ; BAD CHAR...
43C5 CD5849          CALL     OKCK        ; GIVE ESCAPE A CHANCE...
43C8 DB          RC          ;
43C9 4F          MOV      C,A            ; WOULD YOU BELEIVE C FOR COUNTER?
43CA CD0C48          CALL     CRLF        ;
43CD 3EDB          MVI      A,0DBH        ; STORE "IN" INST
43CF 32375A          STA      MISCBF+1      ;
43D2 1600          MVI      D,0          ;
43D4 CD375A      IOPM1: CALL     MISCBF+1      ; GET BYTE FROM PORT
43D7 CD8349          CALL     PHB          ; PRINT BYTE IN HEX
43DA CDEE49          CALL     SPACE        ;
43DD CD4548          CALL     DISASC       ; PRINT BYTE IN ASCII
43E0 7C          MOV      A,H            ;
43E1 CDF247          CALL     CO          ;

```

```

43E4 7D          MOV     A,L          ;
43E5 CDF247      CALL    CO          ;
43E8 11F950      LXI     D,IOPSM      ;PRINT ' '
43EB CD3F49      CALL    MSG         ;
43EE 41          MOV     B,C         ;WAIT (C)*50MS
43EF 04          INR     B           ;CHECK FOR ZERO
43F0 05          IOPM2: DCR     B           ;
43F1 CAFA43      JZ      IOPM3        ;
43F4 CD1548      CALL    D50MS        ;
43F7 C3F043      JMP     IOPM2        ;
43FA 14          IOPM3: INR     D           ;CHECK TO SEE IF IT IS TIME
43FB 3A015A      LDA     WIDTH        ; FOR A ROUSING ROUND OF CRLF
43FE B7          ORA     A           ;CLEAR CARRY
43FF 1F          RAR          ;CUT DOWN ONE
4400 A2          ANA     D           ;
4401 CC0C48      CZ      CRLF        ;
4404 C3D443      JMP     IOPM1        ;

;
;*****END OF IO PORT COMMAND*****
;
;*****
;*          BEGINNING OF ULTRASONIC ROUTINE          *
;*****

4407 CD0E4A      USFNT: CALL    HRTBEAT      ;RELOAD HEARTBEAT
440A 3E00          MVI     A,00H
440C D340          OUT     PIAD             ;RESET INIT LINE ON SONICS
440E D322          OUT     TIME2           ;ZERO MSB OF COUNT
4410 D322          OUT     TIME2           ;      LSB OF COUNT
4412 3E01          MVI     A,01H
4414 D340          OUT     PIAD             ;SEND OUT SONIC BOOM
4416 115000        LXI     D,0050H         ;DELAY FOR < 1 MILLISEC.
4419 CDB747        CALL    DELAYD         ; OFF TO DELAY
441C 3E03          MVI     A,03H         ;SEND OUT BLANK INHIBIT
441E D340          OUT     PIAD             ; BUT KEEP BOOM HIGH
4420 3A125A        LOOPD: LDA     MAXFNT    ;GET MAX FRONT DIST.
4423 47          MOV     B,A
4424 CD4445        CALL    CNTCK           ;FIND OUT HOW LONG
4427 7C          MOV     A,H
4428 B8          CMP     B           ; BOOM HAS BEEN GONE
4429 DA3744        JC      NEXTA          ; IF SO FORGET IT
442C 3E00          MVI     A,00H
442E D340          OUT     PIAD             ;RESET EVERYTHING
4430 210000        LXI     H,0000H         ; CLEAR DIST.
4433 22105A        SHLD    FNTDST
4436 C9          RET

4437 DB42          NEXTA: IN      PIAF      ;TEST FOR BOOM
4439 E601          ANI     01H           ;MASK OFF DIRECTION
443B FE01          CPI     01H           ;TEST FOR DIRECTION
443D C22044        JNZ     LOOPD         ;IF NOT BOOM THEN WAIT
4440 3E00          MVI     A,00H
4442 D340          OUT     PIAD             ;RESET INIT LINE
4444 CD4445        CALL    CNTCK         ;GET COUNTER IN HL
4447 CD5345        CALL    BEEP
444A CD5345        CALL    BEEP

```

```

444D CD7245      CALL    FNDDT      ;FIND DISTANCE
4450 22105A      SHLD    FNTDST     ; STORE AS FRONT
4453 C9          RET

4454 CD0E4A      USBACK: CALL    HRTBEAT      ;RELOAD HEARTBEAT
4457 3E80        MVI      A,080H      ;INITIALIZE 8253 COUNTER
4459 D323        OUT      TIMCTL      ;TIMER2 BINARY COUNT MODE 0
445B 3E00        MVI      A,00H
445D D340        OUT      PIAD        ;RESET INIT LINE ON SONICS
445F D322        OUT      TIME2      ;ZERO MSB OF COUNT
4461 D322        OUT      TIME2      ;      LSB OF COUNT
4463 3E04        MVI      A,04H
4465 D340        OUT      PIAD        ;SEND OUT SONIC BOOM
4467 115000      LXI      D,0050H     ;DELAY FOR < 1 MILLISEC.
446A CD8747      CALL    DELAYD      ; OFF TO DELAY
446D 3E0C        MVI      A,0CH      ;SEND OUT BLANK INHIBIT
446F D340        OUT      PIAD        ; BUT KEEP BOOM HIGH
4471 3A155A      LOOPF: LDA      MAXBAK     ;GET MAX BACK DIST.
4474 47          MOV      B,A
4475 CD4445      CALL    CNTCK      ;FIND OUT HOW LONG
4478 7C          MOV      A,H
4479 B8          CMP      B          ; BOOM HAS BEEN GONE
447A DA8844      JC      NEXTB      ; IF SO FORGET IT
447D 3E00        MVI      A,00H
447F D340        OUT      PIAD        ;RESET EVERYTHING
4481 210000      LXI      H,0000H
4484 22135A      SHLD    BAKDST
4487 C9          RET

4488 DB42        NEXTB: IN      PIAF      ;TEST FOR BOOM
448A E602        ANI      02H        ;MASK OFF DIRECTION
448C FE02        CPI      02H        ;TEST FOR DIRECTION
448E C27144      JNZ      LOOPF      ;IF NOT BOOM THEN WAIT
4491 3E00        MVI      A,00H
4493 D340        OUT      PIAD        ;RESET INIT LINE
4495 CD4445      CALL    CNTCK      ;GET COUNTER IN HL
4498 CD5345      CALL    BEEP
449B CD5345      CALL    BEEP
449E CD7245      CALL    FNDDT      ;FIND DISTANCE
44A1 22135A      SHLD    BAKDST      ;AND STORE AS
44A4 C9          RET                ;BACK DIST.

44A5 CD0E4A      USRT:  CALL    HRTBEAT      ;RELOAD HEARTBEAT
44A8 3E80        MVI      A,080H      ;INITIALIZE 8253 COUNTER
44AA D323        OUT      TIMCTL      ;TIMER2 BINARY COUNT MODE 0
44AC 3E00        MVI      A,00H
44AE D340        OUT      PIAD        ;RESET INIT LINE ON SONICS
44B0 D322        OUT      TIME2      ;ZERO MSB OF COUNT
44B2 D322        OUT      TIME2      ;      LSB OF COUNT
44B4 3E10        MVI      A,10H
44B6 D340        OUT      PIAD        ;SEND OUT SONIC BOOM
44B8 115000      LXI      D,0050H     ;DELAY FOR < 1 MILLISEC.
44BB CD8747      CALL    DELAYD      ; OFF TO DELAY
44BE 3E30        MVI      A,30H      ;SEND OUT BLANK INHIBIT
44C0 D340        OUT      PIAD        ; BUT KEEP BOOM HIGH
44C2 3A185A      LOOPH: LDA      MAXRT     ;GET MAX RIGHT DIST.

```

```

44C5 47      MOV      B,A
44C6 CD4445   CALL     CNTCK      ;FIND OUT HOW LONG
44C9 7C      MOV      A,H
44CA B8      CMP      B           ; BOOM HAS BEEN GONE
44CB DAD944   JC       NEXTC      ; IF SO FORGET IT
44CE 3E00     MVI      A,00H
44D0 D340     OUT      PIAD       ;RESET EVERYTHING
44D2 210000   LXI      H,0000H
44D5 22165A   SHLD     RTDST
44D8 C9      RET

44D9 DB42     NEXTC: IN      PIAF      ;TEST FOR BOOM
44DB E604     ANI      04H          ;MASK OFF DIRECTION
44DD FE04     CPI      04H          ;TEST FOR DIRECTION
44DF C2C244   JNZ      LOOPH       ;IF NOT BOOM THEN WAIT
44E2 3E00     MVI      A,00H
44E4 D340     OUT      PIAD       ;RESET INIT LINE
44E6 CD4445   CALL     CNTCK       ;GET COUNTER IN HL
44E9 CD5345   CALL     BEEP
44EC CD7245   CALL     FNDDT       ;GET DISTANCE
44EF 22165A   SHLD     RTDST       ;STORE AS RIGHT
44F2 C9      RET

44F3 CD0E4A   USLFT: CALL     HRTBEAT ;RELOAD HEARTBEAT
44F6 3EB0     MVI      A,0B0H      ;INITIALIZE 8253 COUNTER
44F8 D323     OUT      TIMCTL      ;TIMER2 BINARY COUNT MODE 0
44FA 3E00     MVI      A,00H
44FC D340     OUT      PIAD       ;RESET INIT LINE ON SONICS
44FE D322     OUT      TIME2      ;ZERO MSB OF COUNT
4500 D322     OUT      TIME2      ;      LSB OF COUNT
4502 3E40     MVI      A,40H
4504 D340     OUT      PIAD       ;SEND OUT SONIC BOOM
4506 115000   LXI      D,0050H    ;DELAY FOR < 1 MILLISEC.
4509 CD8747   CALL     DELAYD      ; OFF TO DELAY
450C 3EC0     MVI      A,0C0H      ;SEND OUT BLANK INHIBIT
450E D340     OUT      PIAD       ; BUT KEEP BOOM HIGH
4510 3A1B5A   LOOPJ: LDA      MAXLFT ;GET MAX LEFT DIST.
4513 47      MOV      B,A
4514 CD4445   CALL     CNTCK      ;FIND OUT HOW LONG
4517 7C      MOV      A,H
4518 B8      CMP      B           ; IF GEATER
4519 DA2745   JC       NEXTD      ; IF SO FORGET IT
451C 3E00     MVI      A,00H
451E D340     OUT      PIAD       ;RESET EVERYTHING
4520 210000   LXI      H,0000H
4523 22195A   SHLD     LFTDST
4526 C9      RET

4527 DB42     NEXTD: IN      PIAF      ;TEST FOR BOOM
4529 E608     ANI      08H          ;MASK OFF DIRECTION
452B FE08     CPI      08H          ;TEST FOR DIRECTION
452D C21045   JNZ      LOOPJ       ;IF NOT BOOM THEN WAIT
4530 3E00     MVI      A,00H
4532 D340     OUT      PIAD       ;RESET INIT LINE
4534 CD4445   CALL     CNTCK       ;GET COUNTER IN HL
4537 CD5345   CALL     BEEP

```

```

453A CD5345      CALL    BEEP
453D CD7245      CALL    FNDDT      ;GET DISTANCE
4540 22195A      SHLD    LFTDST    ;STORE AS LEFT
4543 C9          RET

4544 F5          CNTCK:  PUSH    PSW
4545 3E80          MVI      A,80H
4547 D323          OUT      TIMCTL    ;LATCH CURRENT COUNT
4549 DB22          IN       TIME2     ;GET LSB
454B 2F           CMA          ; FLIP IT TO REAL TIME
454C 6F           MOV      L,A
454D DB22          IN       TIME2     ;GET MSB
454F 2F           CMA          ; FLIP TO REAL TIME
4550 67           MOV      H,A
4551 F1           POP      PSW
4552 C9          RET

4553 3A205A      BEEP:   LDA      MENCTRL    ;GET SOUND FLAG
4556 E602          ANI      SONOFF    ;MASK TO SEE IF SOUND ON
455B FE02          CPI      SONOFF    ; IS IT ON ?
455A C0          RNZ          ; IF NOT FOGET IT
455B 3E40          MVI      A,40H
455D D342          OUT      PIAF      ;TURN ON TONE
455F 54          MOV      D,H        ;DELAY FOR
4560 5D          MOV      E,L        ; DIST.COUNT
4561 CD8747       CALL    DELAYD    ;WAIT FOR IT
4564 3EC0          MVI      A,0COH    ;CHANGE TONE
4566 D342          OUT      PIAF
4568 54          MOV      D,H        ;DELAY FOR
4569 5D          MOV      E,L        ;DIST COUNT
456A CD8747       CALL    DELAYD
456D 3E00          MVI      A,00H    ;NOW TURN EVERYTHING OFF
456F D342          OUT      PIAF
4571 C9          RET

4572 11F000      FNDDT:  LXI      D,00FOH    ;COUNT TO DIST. RATIO
4575 01FFFF       LXI      B,0FFFFH    ; ZERO BC
4578 03          LOOPM:  INX      B
4579 CDF649       CALL    SUB16    ;HL=HL-DE
457C D27B45       JNC      LOOPM    ;DONE YET?
457F 67          MOV      L,C        ;MOVE BC TO HL
4580 60          MOV      H,B
4581 C9          RET

4582 3A205A      ULTRA:  LDA      MENCTRL    ;GET MENU CONTROL WORD
4585 E604          ANI      RONOFF    ;MASK TO SEE IF RANGING ON
4587 FE04          CPI      RONOFF
4589 C0          RNZ          ;
458A DB11          IN       PIAB      ;RETURN IF RANGING OFF
458C E608          ANI      FUSLOOP    ;GET +5 VOLT DATA (LOOP) FROM CONN.
458E FE08          CPI      FUSLOOP    ;MASK TO GET FRONT CONNECTOR STATUS
4590 CA9B45       JZ       FUSOK      ;AFTER COMPARE, Z SET = CONNECTED
4593 3E20          MVI      A,FUSLED    ;CONTINUE IF CONNECTED
4595 CDF04A       CALL    SETERR    ;GET DATA TO LIGHT FRONT US ERROR LED
4598 C3C945       JMP      ULTRA1    ;LIGHT THE FRONT US ERROR LED
459B 3E20          FUSOK: MVI      A,FUSLED    ;CLEAR LED ERROR

```

```

459D CDFC4A      CALL    CLRERR      ;
45A0 CD0744      CALL    USFNT      ;ULTRASONIC RANGING FRONT
45A3 2A1C5A      LHL    TIMDLY      ; DELAY FOR SCAN
45A6 EB          XCHG
45A7 CDB747      CALL    DELAYD
45AA 2A105A      LHL    FNTDST      ;MAX FRONT DIST.
45AD 7C          MOV    A,H
45AE B5          ORA    L
45AF FE00        CPI    00H      ; IF GREATER THAN MAX
45B1 CAC945      JZ     ULTRA1     ; THEN FORGET IT
45B4 113555      LXI    D,FNTMSG  ; PRINT RANGE DIST.
45B7 CD3F49      CALL    MSG      ; MESSAGE
45BA CD7849      CALL    PHW      ; AND FRONT DIST. VALUE
45BD CD0C48      CALL    CRLF
45C0 7D          MOV    A,L      ;CHECK FOR UNSAFE DIST
45C1 FE0C        CPI    USSTOP    ; IF GREATER THEN ULTRA STOP
45C3 D2C945      JNC    ULTRA1     ; THEN CONT.
45C6 CDEB48      CALL    STOP      ; IF LESS STOP
45C9 DB11        ULTRA1: IN    PIAB ;GET +5 VOLT DATA (LOOP) FROM CONN.
45CB E610        ANI    BUSLOOP    ;MASK TO GET BACK CONNECTOR STATUS
45CD FE10        CPI    BUSLOOP    ;AFTER COMPARE, Z SET = CONNECTED
45CF CADA45      JZ     BUSOK      ;CONTINUE IF CONNECTED
45D2 3E10        MVI    A,BUSLED   ;GET DATA TO LIGHT BACK US ERROR LED
45D4 CDF04A      CALL    SETERR    ;LIGHT THE BACK US ERROR LED
45D7 C30846      JMP     ULTRA2
45DA 3E10        BUSOK: MVI    A,BUSLED ;GET DATA TO CLEAR BACK US ERROR LED
45DC CDFC4A      CALL    CLRERR
45DF CD5444      CALL    USBACK    ;ULTRASONIC RANGE BACK
45E2 2A1C5A      LHL    TIMDLY    ; DELAY FOR SCAN
45E5 EB          XCHG
45E6 CDB747      CALL    DELAYD
45E9 2A135A      LHL    BAKDST     ;MAX BACK DIST.
45EC 7C          MOV    A,H
45ED B5          ORA    L
45EE FE00        CPI    00H      ; IF GREATER THAN MAX
45F0 CA0846      JZ     ULTRA2     ; THEN FORGET IT
45F3 113E55      LXI    D,BAKMSG   ; PRINT RANGE DIST.
45F6 CD3F49      CALL    MSG      ; MESSAGE
45F9 CD7849      CALL    PHW      ; AND FRONT DIST. VALUE
45FC CD0C48      CALL    CRLF
45FF 7D          MOV    A,L      ;CHECK FOR UNSAFE DIST
4600 FE0C        CPI    USSTOP    ; IF GREATER THEN ULTRA STOP
4602 D20846      JNC    ULTRA2     ; THEN CONT.
4605 CDEB48      CALL    STOP      ; IF LESS STOP
4608 DB11        ULTRA2: IN    PIAB ;GET +5 VOLT DATA (LOOP) FROM CONN.
460A E620        ANI    RUSLOOP    ;MASK TO GET RIGHT CONNECTOR STATUS
460C FE20        CPI    RUSLOOP    ;AFTER COMPARE, Z SET = CONNECTED
460E CA1946      JZ     RUSOK      ;CONTINUE IF CONNECTED
4611 3E08        MVI    A,RUSLED   ;GET DATA TO LIGHT RIGHT US ERROR LED
4613 CDF04A      CALL    SETERR    ;LIGHT THE RIGHT US ERROR LED
4616 C34746      JMP     ULTRA3
4619 3E08        RUSOK: MVI    A,RUSLED ;GET DATA TO CLEAR RIGHT US ERROR LED
461B CDFC4A      CALL    CLRERR
461E CDA544      CALL    USRT
4621 2A1C5A      LHL    TIMDLY    ; DELAY FOR SCAN
4624 EB          XCHG

```

```

4625 CD8747      CALL    DELAYD
4628 2A165A      LHLD    RTDST      ;MAX RIGHT DIST.
462B 7C          MOV     A,H
462C B5          ORA     L
462D FE00        CPI     00H      ; IF GREATER THAN MAX
462F CA4746      JZ      ULTRA3   ; THEN FORGET IT
4632 114655      LXI     D,RTMSG   ; PRINT RANGE DIST.
4635 CD3F49      CALL    MSG      ; MESSAGE
4638 CD7849      CALL    PHW      ; AND FRONT DIST. VALUE
463B CD0C48      CALL    CRLF
463E 7D          MOV     A,L      ;CHECK FOR UNSAFE DIST
463F FE0C        CPI     USSTOP   ; IF GREATER THEN ULTRA STOP
4641 D24746      JNC     ULTRA3   ; THEN CONT.
4644 CDEB4B      CALL    STOP     ; IF LESS STOP
4647 DB11        ULTRA3: IN      PIAB ;GET +5 VOLT DATA (LOOP) FROM CONN.
4649 E640        ANI     LUSLOOP  ;MASK TO GET LEFT CONNECTOR STATUS
464B FE40        CPI     LUSLOOP  ;AFTER COMPARE, Z SET = CONNECTED
464D CA5846      JZ      LUSOK    ;CONTINUE IF CONNECTED
4650 3E04        MVI     A,LUSLED ;GET DATA TO LIGHT LEFT US ERROR LED
4652 CDF04A      CALL    SETERR   ;LIGHT THE LEFT US ERROR LED
4655 C38646      JMP     ULTRA4
4658 3E04        LUSOK: MVI     A,LUSLED ;GET DATA TO CLEAR RIGHT US ERROR LED
465A CDFC4A      CALL    CLRERR
465D CDF344      CALL    USLFT    ;ULTRASONIC RANGE LEFT
4660 2A1C5A      LHLD    TIMDLY   ; DELAY FOR SCAN DELAY
4663 E9          XCHG
4664 CD8747      CALL    DELAYD
4667 2A195A      LHLD    LFTDST   ;MAX LEFT DIST.
466A 7C          MOV     A,H
466B B5          ORA     L
466C FE00        CPI     00H      ; IF GREATER THAN MAX
466E CAB646      JZ      ULTRA4   ; THEN FORGET IT
4671 114F55      LXI     D,LFTMSG  ; PRINT RANGE DIST.
4674 CD3F49      CALL    MSG      ; MESSAGE
4677 CD7849      CALL    PHW      ; AND FRONT DIST. VALUE
467A CD0C48      CALL    CRLF
467D 7D          MOV     A,L      ;CHECK FOR UNSAFE DIST
467E FE0C        CPI     USSTOP   ; IF GREATER THEN ULTRA STOP
4680 D28646      JNC     ULTRA4   ; THEN CONT.
4683 CDEB4B      CALL    STOP     ; IF LESS STOP
4686 C9          ULTRA4: RET      ;GO BACK TO CALLING
                                   ; ROUTINE

```

***** END OF SONICS ROUTINE *****

***** BEGINNING OF CHAIR PROGRAMS *****

INITIAL - ROUTINE TO INITIALIZE THE CHAIR UPON STARTUP

```

4687 310060      INITIAL: LXI     SP,ENDRAM+1 ;RESET STACK POINTER
468A 3E82        MVI     A,10000010B ;PORT A: OUTPUT
468C D313        OUT     PIACNTL  ;PORT B: INPUT
                                   ;PORT C (UPPER): OUTPUT
                                   ;PORT C (LOWER): OUTPUT
468E 3E80        MVI     A,0B0H   ;INITIALIZE 8253 COUNTER

```

```

4690 D323      OUT      TIMCTL      ;TIMER2 BINARY COUNT MODE 0
4692 3E81      MVI      A,81H      ;8255 PIA D=IN E=IN
4694 D343      OUT      PIBCNTL     ;F=OUT

4696 3E08      MVI      A,MTRSTOP   ;SET VALUE TO STOP MOTORS
4698 322E5A    STA      RMCS        ; SET MOTOR OUTPUT
469B 322D5A    STA      LMCS        ; TO STOP AT FIRST
469E 322C5A    STA      RMTS
46A1 322B5A    STA      LMTS
46A4 07        RLC                  ;ROTATE TO MS NIBBLE
46A5 07        RLC
46A6 07        RLC
46A7 07        RLC
46A8 F608      ORI      MTRSTOP     ;COMBINE FOR BOTH L & R MOTORS
46AA D341      OUT      PIAE        ;SEND STOPS TO MOTORS
46AC 3E30      MVI      A,30H      ;SET UP TIMER 0 FOR
46AE D323      OUT      TIMCTL      ; HEARTBEAT PROTECTION
46B0 210008    LXI      H,0800H     ; SET SCAN DELAY
46B3 221C5A    SHLD     TIMDLY      ; FOR DELAYING ULTRA SAMPLE
46B6 111A55    LXI      D,CLS       ; CLEAR SCREEN
46B9 CD3F49    CALL     MSG
46BC 11F254    LXI      D,MSG1      ; PRINT A MESSAGE SO
46BF CD3F49    CALL     MSG         ; WE KNOW WE MADE IT

```

```

;*****

```

```

; RUNCHR - MAIN SOFTWARE LOOP. REPEAT LOOP CONSTANTLY, REGARDLESS
; OF THE PAD/JOYSTICK SETTING, BUT DO NOT UPDATE HEARTBEAT
; IF SWITCHED TO JOYSTICK
;
;*****

```

```

46C2 00      RUNCHR: NOP
46C3 00      NOP
46C4 00      NOP
46C5 3EFF      MVI      A,TRUE
46C7 32355A    STA      HEARTON
46CA CD0E4A    CALL     HRTBEAT ;CHECK PAD/JOYSTICK SWITCH, UPDATE
                                ;HEARTBEAT IF SWITCHED TO PAD
46CD CD154A    CALL     MENCHK  ;DETERMINE MENU NUBER, READ GLOBAL
                                ;MENU PARAMETERS
46D0 CD8245    CALL     ULTRA   ;CALL U.S. RANGING ROUTINE
46D3 CDA44A    CALL     PADCHK  ;CHECK PAD FOR TOUCH, READ DATA NEEDED
                                ;IF A VALID TOUCH. PROGRAM MENU IF
                                ;APPROPRIATE.
46D6 CD5D48    CALL     EXTCHK  ; SEE IF USER HAS RS-232 CONNECTED
                                ; AND IF THEY WANT THE MONITOR PRG.
46D9 3A205A    LDA      MENCTRL ;GET MENU CONTROL WORD
46DC E601      ANI      EMPTMEN ;MASK TO SEE IF EMPTY MENU
46DE FE01      CPI      EMPTMEN ;
46E0 C2E946    JNZ      AOK1    ;UPDATE IF MENU NOT EMPTY
46E3 CDEB4B    CALL     STOP    ;OTHERWISE, STOP MOTORS
46E6 C3C246    JMP      RUNCHR
46E9 3A315A    AOK1:  LDA      DURATION
46EC FE00      CPI      00H
46EE C2F746    JNZ      AOK2

```

```

46F1 C0E84B      CALL    STOP
46F4 C3C246      JMP     RUNCHR
46F7 C0B54B      ADK2:   CALL    UPDATMTR ;UPDAT MOTORS IF APPROPRIATE (AT CORRECT
                        ;TIME SO AS TO RAMP).
46FA C3C246      JMP     RUNCHR

;*****
; END OF MAIN LOOP
;*****

;
; PADRD - ROUTINE TO CHECK THE PAD, DETERMINE ERROR STATUS, RETURN
; TOUCH LOCATION OF THERE IS ONE.
;
PADRD:
MENU:   LXI      H,00H      ;RESET HL FOR NEW DATA/STATUS INFO
        MVI      C,00H      ;RESET (C) FOR NEW TOUCH LOCATION
        MVI      B,05H      ;LOAD MENU SELECT COUNTER+1
LOOP3:  DCR      B          ;DECREMENT COUNTER OF MENU SELECT BITS
        MOV      A,B        ;TRANSFER (B) TO (A) FOR OUTPUT
        ORI      EXTMSK     ;MASK FOR EXTRA DEMUX SELECT
        OUT      PIAA       ;OUTPUT COUNT TO SELECT MENU SELECT BIT
        LXI      D,0A0H     ;SET UP DELAY COUNT
        CALL     DELAYD     ;SHORT DELAY
        IN       PIAB       ;INPUT TRANSISTOR STATUS
        ANI      BEAMSK     ;PREPARE INPUT DATA (MASK)
        ORA      H          ;OR CURRENT (H) DATA WITH LED STATUSL
        RAL      ;ROTATE THE (A) LEFT TO MOVE BITS ONE
        MOV      H,A        ;TRANSFER RESULT TO (H) AGAIN
        MOV      A,B        ;CHECK COUNT TO SEE IF = 0
        CPI      00H        ;
        JNZ      LOOP3      ;REPEAT PROCESS IF 5 PAIRS NOT YET SCANNED
        MOV      A,H        ;VALIDATE MENU DATA
        RAR      ;REPOSITION THE MENU DATA (ROTATED)
        MOV      H,A        ;
ERR1:   CPI      00H        ;CHECK FOR NO BEAMS BLOCKED (NO MENU)
        JNZ      ERR2      ;CHECK FOR NEXT ERROR IF NOT ERROR 1
        MVI      H,MENERR   ;SIGNAL MENU ERROR
        JMP      PNTDAT     ;FINISH AND PRINT MSGS
ERR2:   CPI      1FH        ;CHECK FOR ALL BEAMS BROKEN (FALSE MENU)
        JNZ      SCAN      ;CONTINUE SCAN IF NO MENU ERRORS
        MVI      H,MENERR   ;SIGNAL MENU ERROR
        JMP      PNTDAT     ;FINISH AND PRINT MSGS

4734 0E00      SCAN:   MVI      C,00H      ;CLEAR ROW/COL REGISTER
4736 0610      ROW:    MVI      B,10H      ;INITIAL COUNTER VALUE OF 16 LEDS + 1
4738 05        LOOP4:  DCR      B          ;DECREMENT COUNTER
4739 7B        MOV      A,B        ;TRANSFER COUNT TO ACCUM
473A F610      ORI      ROWMSK     ;PREPARE FOR ROW SELECT (MASK)
473C D310      OUT      PIAA       ;OUTPUT ROW LED/TRANSISTOR SELECT
473E 11A000     LXI      D,0A0H     ;LOAD DELAY COUNTER
4741 C0B747     CALL     DELAYD     ;SHORT DELAY
4744 DB11      IN       PIAB       ;GET TRANSISTOR STATUS
4746 E601      ANI      BEAMSK     ;PREPARE INPUT FROM TRANSISTOR (MASK)
4748 FE00      CPI      00H        ;SET ZERO FLAG
474A CA5647     JZ       COUNT3     ;CONTINUE LOOP IF NO TOUCH ('1'=TOUCH)
474D 7B        MOV      A,B        ;TRANSFER COUNT TO ACCUM

```

```

474E 17          RAL          ;ROTATE COUNT VALUE TO MS NIBBLE
474F 17          RAL
4750 17          RAL
4751 17          RAL
4752 4F          MOV          C,A          ;SAVE ROW IN ROW/COL REGISTER
4753 C35D47      JMP          COL          ;JUMP TO COL SCAN BECAUSE ROW TOUCHED

4756 78          COUNT3: MOV    A,B          ;MOVE COUNT TO 'A' TO DO ZERO CHECK
4757 FE00        CPI          00H          ;REPEAT UNLESS CURRENTLY ZERO
4759 C23B47      JNZ          LODP4         ;CONTINUE LOOP IF NOT COUNTED OUT
475C C9          RET          ;RETURN IF LOOP COMPLETED W/NO TOUCH

475D 06FF      COL:  MVI      B,0FFH        ;LOAD COLUMN COUNTER - 1
475F 04          LOOP2: INR      B          ;INCREMENT COLUMN COUNTER
4760 78          MOV      A,B          ;TRANSFER COL COUNT TO ACCUM
4761 F620        ORI      COLMSK         ;PREPARE CN LED/TRANSISTOR SELECT (MASK)
4763 D310        OUT      PIAA          ;SELECT LED/TRANSISTOR
4765 11A000      LXI      D,0A0H         ;LOAD DELAY COUNTER
4768 C0B747      CALL     DELAYD         ;CALL SHORT DELAY
476B DB11        IN       PIAB          ;INPUT TRANSISTOR STATUS
476D E601        ANI      BEAMSK         ;PREPARE INPUT FOR USE (MASK)
476F FE00        CPI      00H          ;SET ZERO FLAG
4771 CA7E47      JZ       COUNT4         ;REPEAT LOOP IF NO TOUCH ('1'=TOUCH)
4774 78          MOV      A,B          ;TRANSFER COUNT TO ACCUM
4775 B1          ORA      C            ;COMPLETE ROW/COL DATA IN ACCUM
4776 4F          MOV      C,A          ;SAVE ROW/COL DATA IN 'C'
                                   ;HIGH NIBBLE: ROW
                                   ;LOW NIBBLE: COLUMN
4777 7C          MOV      A,H          ;MASK (H) TO SHOW A VALID TOUCH
4778 F6B0        ORI      TOUCH
477A 67          MOV      H,A
477B C3B547      JMP      PNTDAT         ;PRINT MESSAGE

477E 78          COUNT4: MOV    A,B          ;CHECK TO SEE IF COUNT=16 DECIMAL
477F FE0F        CPI      0FH          ;SCANNED ALL 16 LEDS?
4781 CB          RZ
4782 C25F47      JNZ      LOOP2         ;CONTINUE LOOP 2 TO CHECK FOR COL TOUCH
4785 69          PNTDAT: MOV    L,C
4786 C9          RET

;
;***** END OF PAD READ ROUTINE *****
;

4787 3A355A      DELAYD: LDA      HEARTON
478A FE00        CPI      FALSE
478C CA9247      JZ       DELAYE
478F CD0E4A      CALL     HRTBEAT         ;DON'T LET HEARTBEAT DIE !!
4792 1B          DELAYE: DCX      D          ;DECREMENT DELAY COUNT
4793 7A          MOV      A,D          ;COMPARE D AND E
4794 B3          ORA      E            ;CHECK TO SEE IF DE=0
4795 C29247      JNZ      DELAYE         ;REPEAT IF <>0
4798 3A355A      LDA      HEARTON
479B FE00        CPI      FALSE
479D CB          RZ
479E CD0E4A      CALL     HRTBEAT
47A1 C9          RET

```

```

;*****
;* UTILITY ROUTINES -
;*****
; BCDTBIN - CONVERT BCD IN H&L TO BINARY IN H&L
;          ONLY H&L CHANGED
47A2 C5      BCDTBIN:PUSH      B          ;
47A3 D5          PUSH      D          ;
47A4 54          MOV       D,H        ; COPY ORIGINAL
47A5 5D          MOV       E,L        ;
47A6 2600        MVI       H,0        ; INITIALIZE UPPER PART OF RESULT
47A8 0600        MVI       B,0        ; INITIAL UPPER PART OF B&C
47AA 7A          MOV       A,D        ; GET UPPER DIGIT
47AB 0F          RRC              ;
47AC 0F          RRC              ;
47AD 0F          RRC              ;
47AE 0F          RRC              ;
47AF E60F        ANI       0FH        ;
47B1 6F          MOV       L,A        ; START RESULT
47B2 CD4F49      CALL      MULT10     ; SHIFT UP ONE DIGIT IN BASE 10
47B5 7A          MOV       A,D        ; GET NEXT TO TOP DIGIT
47B6 E60F        ANI       0FH        ;
47B8 4F          MOV       C,A        ;
47B9 09          DAD       B          ; COMBINE WITH TOP DIGIT
47BA CD4F49      CALL      MULT10     ; SHIFT UP ONE DIGIT IN BASE 10
47BD 7B          MOV       A,E        ; GET NEXT TO BOTTOM DIGIT
47BE 0F          RRC              ;
47BF 0F          RRC              ;
47C0 0F          RRC              ;
47C1 0F          RRC              ;
47C2 E60F        ANI       0FH        ;
47C4 4F          MOV       C,A        ;
47C5 09          DAD       B          ; COMBINE WITH TOP TWO DIGITS
47C6 CD4F49      CALL      MULT10     ; SHIFT UP ONE DIGIT IN BASE 10
47C9 7B          MOV       A,E        ; GET BOTTOM DIGIT
47CA E60F        ANI       0FH        ;
47CC 4F          MOV       C,A        ;
47CD 09          DAD       B          ; COMBINE WITH TOP THREE DIGITS
47CE D1          POP       D          ;
47CF C1          POP       B          ;
47D0 C9          RET              ;
;          END      BCDTBIN ;
;
;
; CALLIN - INDIRECT CALL TO (H&L)
;
47D1 E9      CALLIN: PCHL          ;
;          ;
;          END      CALLIN ;
;
; I/O ROUTINES
;
47D2 DB01      CI:      IN          SERCON          ; WAIT FOR DATA READY
47D4 E602      ANI       2          ;
47D6 CAD247     JZ       CI          ;
47D9 DB00      IN          SERDAT          ; GET BYTE

```

```

47DB F5          PUSH    PSW          ;SAVE PSW
47DC 3A005A      LDA     ECHOFL       ;CHECK ECHO FLAG
47DF B7          ORA     A            ;
47E0 C2FD47      JNZ     COEND        ;IF NOT ZERO ECHO-RET ON CO
47E3 F1          POP     PSW         ;ECHO CHARACTER
47E4 F5          PUSH    PSW         ;
47E5 C3F347      JMP     C1          ;GO ECHO CHARACTER
;
;
; CISTAT - RETURNS NON-ZERO IN A IF RECIEVER BUFFER HAS A CHAR
;
47E8 C5          CISTAT: PUSH    B            ;
47E9 F5          PUSH    PSW          ;
47EA DB01        IN      SERCON       ;
47EC E602        ANI     2            ;
47EE C1          POP     B            ;
47EF 78          MOV     A,B          ;
47F0 C1          POP     B            ;
47F1 C9          RET                 ;
;
;**** CO CONSOLE OUTPUT - DESTROYS ONLY FLAGS...
;
47F2 F5          CO:    PUSH    PSW          ;
47F3 DB01        C1:    IN      SERCON       ;
47F5 0F          RRC                 ;
47F6 D2F347      JNC     C1            ;
47F9 F1          POP     PSW          ;
47FA F5          PUSH    PSW          ;
47FB D300        OUT     SERDAT        ;
47FD F1          COEND: POP     PSW        ;
47FE C9          RET                 ;
;***** CMP16 ** 16 BIT COMPARE H&L AND D&E *****
;
;      IF( H&L = D&E ) Z=1, CY=0
;      IF( H&L > D&E ) Z=0, CY=0
;      IF( H&L < D&E ) Z=0, CY=1
;
47FF E5          CMP16: PUSH    H            ;SAVE PSW & H&L
4800 F5          PUSH    PSW          ;
4801 7C          MOV     A,H          ;IF H = D ENOUGH INFO FOUND
4802 92          SUB     D            ;
4803 C20848      JNZ     CMP16E        ;
4806 7D          MOV     A,L          ;IF H=D THEN COMPARE LOWER BYTES
4807 93          SUB     E            ;
4808 E1          CMP16E: POP     H            ;
4809 7C          MOV     A,H          ;
480A E1          POP     H            ;
480B C9          RET                 ;
;
;      END      CMD16
;
480C D5          CRLF:  PUSH    D            ;
480D 11EF54      LXI     D,MCRLF        ;
4810 CD3F49      CALL    MSG           ;
4813 D1          POP     D            ;
4814 C9          RET                 ;
;
; D50MS - DELAY FOR 50 MILLI-SECONDS

```

```

;
4815 F5      D50MS:  PUSH    PSW          ;SAVE PSW
4816 E5              PUSH    H            ;SAVE H&L
4817 2A0A5A              LHLD    D50DIV    ;
481A E3      D50MSL:  XTHL              ;18
481B E3              XTHL              ;18
481C E3              XTHL              ;18
481D E3              XTHL              ;18
481E E5              PUSH    H            ;11
481F E1              POP     H            ;10
4820 2B              DCX     H            ; 5
4821 23              INX     H            ; 5
4822 2B              DCX     H            ; 5
4823 7C              MOV     A,H          ; 5
4824 B5              ORA     L            ; 4
4825 C21A4B          JNZ     D50MSL       ;11
4828 E1              POP     H            ;
4829 F1              POP     PSW          ;
482A C9              RET                  ;

;
; D10MS - DELAY 10 MS
;
482B E5      D10MS:  PUSH    H            ;
482C F5              PUSH    PSW          ;
482D 210103          LXI     H,769        ;
4830 7D      DTWIDL:  MOV     A,L          ; ~0.01 SECONDS ON A 2 MHZ 8085      5
4831 B4              ORA     H            ; (CPU CLOCK FREQ)                4
4832 2B              DCX     H            ; ;                                10
4833 C2304B          JNZ     DTWIDL        ; ;                                8085/8080      7/10
4836 F1              POP     PSW          ; ;                                TOTAL          26/29
4837 E1              POP     H            ;
4838 C9              RET                  ;
;
; END      D10MS      ;

;
; D5SEC - DELAY 5 SECONDS
;
4839 C5      D5SEC:  PUSH    B            ;
483A 0664          MVI     B,064H          ;WAIT 5 SECOND FOR +25 SWITCHING
483C CD154B      ON16W1:  CALL    D50MS      ;REGULATOR TO TURN ON OR OFF.
483F 05              DCR     B            ;
4840 C23C4B          JNZ     ON16W1        ;
4843 C1              POP     B            ;
4844 C9              RET                  ;
;
; END      D5SEC      ;

;
; DISASC - DISPLAY ASCII A-REG INTO H&L
;
4845 F5      DISASC:  PUSH    PSW          ;SAVE PSW
4846 E67F          ANI     07FH          ;STRIP PARITY
4848 2620          MVI     H,020H        ;PUT SPACE IN H-REG
484A FE20          CPI     020H          ;CHAR < 020H ?
484C D2534B        JNC     DA1           ;NO-IS PRINTABLE
484F 265E          MVI     H,05EH        ;NOT PRINTABLE - C = '^'
4851 C640          ADI     040H          ;MAKE PRINTABLE
4853 FE7F      DA1:   CPI     07FH          ;IS RUBOUT ?
4855 C25A4B        JNZ     DA2           ;NOPE...ADK

```

```

4858 3E20      MVI      A,020H ;YEP-MAKE SPACE
485A 6F        DA2:    MOV      L,A      ;
485B F1        POP      PSW      ;RESTORE PSW
485C C9        RET              ;

;*****
;*              EXIT TO MONITOR ROUTINE              *
;*****

485D DB01      EXTCHK: IN      SERCON ;SEE IF THERE IS A CHAR
485F E602      ANI      02H

; IF ZERO THEN NONE
4861 C8        RZ              ; IF NO CHAR. THEN FORGET IT
4862 DB00      IN      SERDAT ;GET CHAR. FROM CONSOL
4864 FE45      CPI      45H      ; IS IT 'E' FOR EXIT
4866 CA4040    JZ      START ;OFF TO MONITOR THEN
4869 C9        RET

;*****
;              FRMCNT - ASKS " FROM "XXXX" TO "YYYY"              *
;*****

486A D5        FRMCNT: PUSH    D      ;
486B E5        PUSH    H      ;
486C CD7B48    CALL    FROMTO ;
486F DA9D48    JC      FRTOE      ;
4872 E5        PUSH    H      ;
4873 CDF649    CALL    SUB16 ;CALC NUMBER OF BYTES TO BE PROCESSED
4876 D1        POP      D      ;
4877 2B        DCX      H      ;H&L = NEGATIVE OF NUMBER OF BYTES
4878 C39648    JMP      FRCLN ;THIS DOES XCHG & CLEANS OFF STACK...
;              END      FROMTO
;
; FROMTO - " FROM "XXXX" TO "YYYY"
;
487B D5        FROMTO: PUSH    D      ;
487C E5        PUSH    H      ;
487D 116951    LXI      D,PLO ;PROMPT FOR LO LIMIT
4880 CD3F49    CALL    MSG      ;
4883 CDD948    CALL    GHW      ;
4886 DA9D48    JC      FRTOE ;RETURN IF ERRER
4889 116451    LXI      D,PHI ;PROMPT FOR HI LIMIT
488C CD3F49    CALL    MSG      ;
488F EB        XCHG      ;
4890 CDD948    CALL    GHW      ;
4893 DA9D48    JC      FRTOE      ;
4896 EB        FRCLN: XCHG      ;
4897 E3        XTHL      ;GET CRAP OFF OF STACK
4898 E1        POP      H      ;
4899 E3        XTHL      ;
489A E1        POP      H      ;
489B B7        ORA      A      ;BETTER BE SURE CARRY IS CLEAR
489C C9        RET      ;RETURN...
489D E1        FRTOE: POP      H      ;
489E D1        POP      D      ;
489F C9        RET      ;

```

```

; GBIAS - GET 16 BIT BIAS
;
48A0 F5      GBIAS:  PUSH    PSW          ;SAVE PSW
48A1 E5              PUSH    H           ; AND H&L
48A2 D5              PUSH    D           ; AND D&E
48A3 117051        LXI     D,PBIAS      ;PRINT BIAS MESSAGE
48A6 CD3F49        CALL    MSG          ;
48A9 CDD948        CALL    GHW          ;GET BIAS
48AC D2CC48        JNC     GBIAS2       ;IF NO CARRY GOOD BIAS ENTERED
48AF FE2D          CPI     '-'          ;CHECK FOR NEGATIVE BIAS
48B1 CABF48        JZ      GBIAS1       ;OHHH- WANT NEGATIVE NUMBER ...
48B4 FE0D          CPI     CR           ;CARRIAGE RETURN ?
48B6 C2D448        JNZ     GBIASE       ;NOPE ERRER
48B9 210000        LXI     H,0          ;AKHHH - NO BIAS
48BC C3CC48        JMP     GBIAS2       ;
48BF CDD948        GBIAS1: CALL    GHW   ;GET NEGATIVE BIAS
48C2 DAD448        JC      GBIASE       ;BAD CHAR...BYE
48C5 110000        LXI     D,0          ;
48C8 EB           XCHG                 ;SET UP SUBTRACTION FROM ZERO
48C9 CDF649        CALL    SUB16        ;NEGATE BIAS
48CC CD0C48        GBIAS2: CALL    CRLF  ;PREVENT A MESS
48CF D1           POP     D            ;RESTORE D
48D0 F1           POP     PSW          ;LOOSE ORIGINAL H&L
48D1 F1           POP     PSW          ;RESTORE PSW
48D2 B7           ORA     A            ;CLEAR CARRY
48D3 C9           RET                  ;
48D4 D1        GBIASE: POP     D        ;RESTORE D&E
48D5 E1          POP     H            ;RESTORE ORIGINAL H&L
48D6 F1          POP     PSW          ;RESTORE PSW
48D7 37          STC                  ;SET CARRY
48D8 C9          RET                  ;

;
; GHW - GET HEX WORD
;
48D9 C5      GHW:    PUSH    B
48DA F5              PUSH    PSW
48DB CDF048        CALL    GHB          ; GET FIRST BYTE IN A-REGISTER
48DE DAED48        JC      GHWEND       ; RETURN IF BAD CHAR
48E1 67           MOV     H,A          ; MOVE BYTE TO FINAL DESTINATION
48E2 CDF048        CALL    GHB          ; GET SECOND BYTE
48E5 DAED48        JC      GHWEND       ;
48E8 6F           MOV     L,A          ;
48E9 C1           POP     B            ;
48EA 78           MOV     A,B          ;
48EB C1           POP     B            ;
48EC C9           RET                  ;
48ED C1        GHWEND: POP     B        ;
48EE C1          POP     B            ; DO NOT RESTORE A
48EF C9          RET                  ;
;          END      GHW              ;

;
; GHB - GET HEX BYTE
;
48F0 C5      GHB:    PUSH    B          ; SAVE B&C
48F1 CD0549        CALL    GHD          ; GET FIRST HEX DIGIT IN A-REG

```

```

48F4 DA0349      JC      GHXBEND      ; IF BAD CHAR QUIT AND PASS BACK
48F7 07          RLC                  ; SHIFT TO UPPER HALF OF BYTE
48F8 07          RLC                  ; .
48F9 07          RLC                  ; .
48FA 07          RLC                  ; .
48FB 47          MOV      B,A          ; SAVE FIRST DIGIT
48FC CD0549      CALL     GHX          ; GET SECOND DIGIT
48FF DA0349      JC      GHXBEND      ; BAD CHAR READ, RET IT TO CALLER
4902 B0          DRA      B            ; COMBINE FIRST AND SECOND DIGITS
4903 C1          GHXBEND: POP      B    ; RESTORE ORIGINAL B&C
4904 C9          RET                  ;
;
;
; GHX - GET HEX DIGIT
;
4905 CD0247      GHX:   CALL     CX          ; GET CHARACTER & ECHO
;
; ANI      07FH          ; PUT IN IF UCASE TAKEN OUT
4908 CD024A      ATH:   CALL     UCASE      ; MAP LOWER TO UPPER CASE AND
;
; STRIP PARITY.
490B FE30        CPI      '0'
490D DB          RC                  ; NON-HEX CHARACTER
490E FE3A        CPI      ':'          ; IF (A) =< '9'+1
4910 DA1C49      JC      GHX2          ; '0'-'9' TYPED - CONVERT
4913 FE41        CPI      'A'          ; IF (A) < 'A'
4915 DB          RC                  ; NON-HEX CHARACTER
4916 FE47        CPI      'B'          ; IF (A) >= 'B'
4918 3F          CMC                  ;
4919 DB          RC                  ; NON-HEX CHARACTER
491A D607        SUI      07H          ; SHIFT 'A'-'F' DOWN
491C D630        GHX2:  SUI      '0'    ; CONVERT
491E C9          RET                  ;
;
;
; M50128 - MULTIPLY BY 50/128
;
491F 0601        M50128: MVI      B,1    ; DIVIDE BY TWO SO * 12.5
4921 CDCA49      CALL     SHRHL          ; WILL FIT IN 16 BITS.
4924 CDBF49      CALL     RNDHL          ; AND ROUND
4927 54          MOV      D,H          ; SAVE *1
4928 5D          MOV      E,L          ; .
4929 29          DAD      H            ; *2
492A 29          DAD      H            ; *4
492B 44          MOV      B,H          ; SAVE *4 IN D&E
492C 4D          MOV      C,L          ;
492D 29          DAD      H            ; *8
492E 09          DAD      B            ; *12
492F EB          XCHG                  ; GENERATE * 0.5
4930 0601        MVI      B,1          ; .
4932 CDCA49      CALL     SHRHL          ; .
4935 19          DAD      D            ; *12 + *0.5
4936 0604        MVI      B,4          ; DIVIDE H&L BY 16
4938 CDCA49      CALL     SHRHL          ;
493B CDBF49      CALL     RNDHL          ; ROUND
493E C9          RET                  ;
;
;
; END      M50128

```

```

; MSG -
493F F5      MSG:  PUSH    PSW
4940 1A      LOUPE: LDAX   D      ;GET CHAR
4941 FEFF      CPI     EOL      ;END OF STRING?
4943 13      INX      D      ;BUMP POINTER
4944 CA4D49    JZ       MDN      ;JUMP IF SO
4947 CDF247    CALL    CO      ;ELSE PRINT IT
494A C34049    JMP      LOUPE   ;DO IT AGAIN
494D F1      MDN:  POP     PSW
494E C9      RET

;
; MULT10 - MULTIPLY H&L BY 10
494F D5      MULT10: PUSH   D      ;
4950 29      DAD      H      ;*2
4951 54      MOV      D,H      ;SAVE *2
4952 5D      MOV      E,L      ;
4953 29      DAD      H      ;*4
4954 29      DAD      H      ;*8
4955 19      DAD      D      ;*10
4956 D1      POP      D      ;
4957 C9      RET           ;

;
4958 D5      OKCK:  PUSH   D
4959 F5      PUSH   PSW
495A 111D51   LXI     D,MOK
495D C03F49   CALL    MSG
4960 CDD247   CALL    CI
4963 E67F     ANI     07FH
4965 FE0D     CPI     CR
4967 CA7149   JZ      OKCKEND
496A 113B50   LXI     D,ABORT
496D C03F49   CALL    MSG
4970 37      STC
4971 C00C48   OKCKEND:CALL CRLF
4974 D1      POP     D
4975 7A      MOV     A,D
4976 D1      POP     D
4977 C9      RET
;      END      OKCK
;
; PHW - PRINT HEX WORD
4978 F5      PHW:  PUSH   PSW      ; SAVE A-REGISTER AND FLAGS
4979 7C      MOV     A,H      ;
497A CD8349   CALL    PHB      ; PRINT HIGH-ORDER BYTE
497D 7D      MOV     A,L      ;
497E CD8349   CALL    PHB      ; PRINT LOW-ORDER BYTE
4981 F1      POP     PSW      ; RESTORE A-REGISTER AND FLAGS
4982 C9      RET           ;
;      END      PHW      ;

;
; PHB - PRINT HEX BYTE
4983 F5      PHB:  PUSH   PSW      ; SAVE PSW
4984 C5      PUSH   B      ; SAVE B&C
4985 47      MOV     B,A      ; SAVE LOWER NIBBLE
4986 0F      RRC      ; SHIFT TO LOWER HALF OF BYTE
4987 0F      RRC
;

```

```

4988 0F          RRC          ;
4989 0F          RRC          ;
498A CD9549      CALL    PHD          ; PRINT UPPER HEX DIGIT
498D 78          MOV     A,B          ; GET LOWER NIBBLE
498E CD9549      CALL    PHD          ; ...AND PRINT
4991 78          MOV     A,B          ; RESTORE ORIGINAL BYTE TO A
4992 C1          POP      B           ; RESTORE B&C
4993 F1          POP      PSW         ; RESTORE PSW
4994 C9          RET              ;
;
;          END      PHB          ;
;
; PHD - PRINT HEX DIGIT
4995 F5          PHD:    PUSH    PSW          ;SAVE PSW
4996 E60F        ANI      0FH          ; MASK OFF LOWER NIBBLE
4998 C630        ADI      '0'          ; CONVERT '0'-'9' TO ASCII
499A FE3A        CPI      '9'+1        ; IF '0'-'9'
499C DAA149      JC       PHD1          ; THEN DONE
499F C607        ADI      'A'-'1'      ; CONVERT 'A'-'F'
49A1 CDF247      PHD1:    CALL    CD          ; PRINT DIGIT
49A4 F1          POP      PSW          ;
49A5 C9          RET              ;
;
;          END      PHD          ;
;
; POPPC - POP THE PC INTO H&L
;          - ON RETURN (H&L) = ADDRESS RETURNED TO
49A6 E1          POPPC:    POP      H           ;
49A7 E9          PCHL          ;
;
; ***** PRBAD - PRINT ' WHAT ?' ***** DESTROYS D&E *****
49A8 114650      PRBAD:    LXI      D,BAD      ;
49AB CD3F49      CALL    MSG          ;
49AE C9          RET              ;
;
;          END      PRBAD          ;
;
; RETJMP - RETURN JUMP
;          SETS STACK POINTER TO (RJSP) AND PC TO (RJVECT)
;          DOES NOT DESTROY ANY REGISTERS
49AF 22045A      RETJMP:    SHLD    RJSV          ;
49B2 2A065A      LHL      RJSP          ;
49B5 F9          SPHL          ;
49B6 2A045A      LHL      RJSV          ;
49B9 E5          PUSH    H           ;
49BA 2A085A      LHL      RJVECT        ;
49BD E3          XTHL          ;
49BE C9          RET              ;
;
; RNDHL - ADD CARRY FLAG TO H&L TO ROUND AFTER USING
;          SHRHL TO DIVIDE BY A POWER OF 2
49BF F5          RNDHL:    PUSH    PSW          ;
49C0 7D          MOV     A,L          ;
49C1 CE00        ACI      0           ;ROUND
49C3 6F          MOV     L,A          ;
49C4 7C          MOV     A,H          ;PROPAGATE POSSIBLE ROUND-UP
49C5 CE00        ACI      0           ; CARRY INTO H.
49C7 67          MOV     H,A          ;
49C8 F1          POP      PSW          ;

```

```

49C9 C9          RET          ;
;
; SHRHL - SHIFT RIGHT H&L - ZERO FILL ON LEFT
;          SHIFTS (B) BITS
49CA C5      SHRHL:  PUSH    B          ;SAVE B
49CB F5          PUSH    PSW          ;SAVE A
49CC 04          INR     B          ;CHECK FOR NO MORE BITS TO SHIFT
49CD 05      SHRHL:  DCR     B          ;
49CE CAD849      JZ      SHRHLE      ;
49D1 B7          ORA     A          ;CLEAR CARRY FLAG
49D2 7C          MOV     A,H          ;GET H
49D3 1F          RAR     A          ;SHIFT RIGHT
49D4 67          MOV     H,A          ;PUT H BACK
49D5 7D          MOV     A,L          ;GET L
49D6 1F          RAR     A          ;SHIFT RIGHT
49D7 6F          MOV     L,A          ;PUT L BACK
49D8 C3CD49      JMP     SHRHLL      ;BACK...
49DB C1      SHRHLE:  POP     B          ;RESTORE A
49DC 78          MOV     A,B          ;
49DD C1          POP     B          ;RESTORE B
49DE C9          RET          ;BYE...
;          END     SHRHL      ;
;
; SETJMP - SET SP AND PC FOR RETJMP
;          DOES NOT DESTROY ANY REGISTERS
49DF E5      SETJMP:  PUSH    H          ;
49E0 210400      LXI     H,04          ;GET SP BEFORE PUSH H AND RET ADDR
49E3 39          DAD     SP          ;
49E4 22065A      SHLD    RJSP          ;
49E7 E1          POP     H          ;GET H&L BACK
49E8 E3          XTHL          ;GET RET ADDR
49E9 22085A      SHLD    RJVECT        ;SQUIREL AWAY
49EC E3          XTHL          ;PUT RET ADDR BACK
49ED C9          RET          ;

; ***** SPACE ***** PRINT SPACE
49EE F5      SPACE:  PUSH    PSW
49EF 3E20      MVI     A,' '
49F1 CDF247      CALL    CO
49F4 F1          POP     PSW
49F5 C9          RET

; ***** SUB16 ***** 16 BIT SUBTRACT (H&L) <- (H&L) - (D&E)
;          IF (D&E) < (H&L)      CY = 1
;          IF (D&E) >= (H&L)     CY = 0
49F6 D5      SUB16:  PUSH    D          ;
49F7 F5          PUSH    PSW          ;
49F8 7D          MOV     A,L          ;
49F9 93          SUB     E          ;
49FA 6F          MOV     L,A          ;
49FB 7C          MOV     A,H          ;
49FC 9A          SBB     D          ;
49FD 67          MOV     H,A          ;
49FE D1          POP     D          ;
49FF 7A          MOV     A,D          ;
4A00 D1          POP     D          ;

```

```

4A01 C9          RET          ;
;
; UCASE - SUBROUTINE WHICH CHECKS THE A REG FOR A LOWER CASE
; ASCII LETTER. IF ONE PRESENT, IT IS CONVERTED TO UPPER CASE.
; IF NOT PRESENT, NOTHING DONE. STRIPS PARITY FIRST.
4A02 E67F      UCASE: ANI      07FH      ;STRIP PARITY
4A04 FE61          CPI      61H
4A06 3F          CMC
4A07 D0          RNC          ;DON'T CONVERT IF BEFORE 'A'
4A08 FE78          CPI      7BH
4A0A D0          RNC          ;DON'T CONVERT IF AFTER 'Z'
4A0B D620          SUI      20H      ;CONVERT LOWER TO UPPER
4A0D C9          RET
;
; HRTBEAT - SUBROUTINE WHICH REFRESHES THE HEARTBEAT COUNTER (TIME0).
;
4A0E 3E00      HRTBEAT: MVI      A,00H      ;LONGEST AVAILABLE COUNTER VALUE
4A10 D320          OUT      TIME0      ;LEAST SIG BYTE OF TIMER
4A12 D320          OUT      TIME0      ;MOST SIG BYTE OF TIMER
4A14 C9          RET
;
; MENCHK - SUBROUTINE TO CHECK THE MENU NUMBER, AND DO ONE OF THE
; FOLLOWING: SIGNAL AN ERROR AND STOP THE CHAIR IF IT IS
; AN INCORRECT MENU (0 OR 32), CALL THE PROGRAM MENU IF
; APPLICABLE, OR READ THE GLOBAL MENU VARIABLES FOR THE
; SELECTED MENU, AND RETURN THE MENU NUMBER.
;
4A15 DB11      MENCHK: IN      PIAB      ;READ THE +5 LOOP FROM THE PAD CONNECTOR
4A17 E680          ANI      PADLOOP      ;MASK TO DETERMINE THE CONNECTOR STATUS
4A19 FE80          CPI      PADLOOP      ;VALUE READ, 1=CONNECTED, 0=DISCONNECTED
4A1B CA2C4A      JZ      PADOK1      ;CONTINUE IF PAD CONNECTED
4A1E 3E01          MVI      A,PADLED      ;GET DATA TO LIGHT PAD ERROR LED
4A20 CDF04A      CALL      SETERR      ;LIGHT THE PAD ERROR LED
4A23 3E40          MVI      A,MENERR      ;PAD ERROR WILL GIVE MENU ERROR
4A25 CDFC4A      CALL      CLRERR      ;SO CLEAR MENU LED
4A28 CDEB4B      CALL      STOP      ;STOP THE CHAIR (RAMP DOWN) IF DISCONN.
4A2B C9          RET
4A2C 3E01      PADOK1: MVI      A,PADLED      ;GET DATA TO CLEAR PAD LED
4A2E CDFC4A      CALL      CLRERR      ;CLEAR THE PAD LED (ALL IS OK)
4A31 CDFD46      CALL      PADRD      ;DETERMINE MENU NUMBER OR STATUS (IF ERROR)
4A34 7C          MOV      A,H      ;PUT MENU NUMBER (STATUS) IN (A)
4A35 E640          ANI      MENERR      ;MASK FOR MENU ERROR
4A37 FE40          CPI      MENERR      ;MASK FOR MENU ERROR
4A39 C2454A      JNZ      PADOK2      ;IF VALID MENU, THEN PROCEED, OTHERWISE...
4A3C 3E02          MVI      A,MENLED      ;GET DATA TO LIGHT THE MENU ERROR LED
4A3E CDF04A      CALL      SETERR      ;LIGHT THE MENU ERROR LED
4A41 CDEB4B      CALL      STOP      ;STOP THE CHAIR (RAMP DOWN) IF MENU ERROR
4A44 C9          RET
4A45 3E02      PADOK2: MVI      A,MENLED      ;GET DATA TO CLEAR THE MENU LED
4A47 CDFC4A      CALL      CLRERR      ;CLEAR MENU ERROR LED
4A4A 7C          MOV      A,H      ;PUT MENU NUMBER (STATUS) IN (A)
4A4B E601          ANI      PROMSK      ;MASK PROGRAM MENU NUMBER
4A4D FE01          CPI      PROMSK      ;MASK TO SEE IF PROGRAMMING MENU IN PAD
4A4F C2564A      JNZ      IFTBL1      ;SEE IF MENU 1 IF NOT PROGRAMMING MENU
4A52 CD504C      CALL      PROMEN      ;CALL ROUTINE TO ALLOW TABLE UPDATES
4A55 C9          RET

```

```

4A56 7C      IFTBL1: MOV    A,H      ;PUT MENU NUMBER IN (A)
4A57 E61E    ANI    VALMEN    ;MASK FOR VALID MENU NUMBERS (0-16)
4A59 1F      RAR      ;ROTATE TO USE ONLY 4 OF THE 5 MENU BITS
4A5A FE01    CPI     01H      ;CHECK TO SEE IF MENU NUMBER 1 (SAMPLE MENU)
4A5C D2664A  JNZ     OTHERS    ;CHECK FOR NUMBER 3 (TABLE 2) IF NOT 2
4A5F 210956  LXI     H,SAMPLE    ;PUT STARTING ADDRESS OF SAMPLE TBL IN (HL)
4A62 CD7F4A  CALL    GETVARS    ;GET GLOBAL VARIABLES FROM TABLE
4A65 C9      RET

      OTHERS:      ;
                  ;BEGIN BY DETERMINING THE STARTING ADDRESS OF TABLE
                  ;FORMULA TO CALCULATE OFFSET IS AS FOLLOWS...
                  ;      (MENU NUMBER - 2)(50) = OFFSET
                  ;NOTE THAT THE MENU TABLES ARE 50 BYTES LONG,
                  ;MENU 0 IS INVALID, AND MENU NUMBER 1 IS THE
                  ;SAMPLE MENU (HARD CODED).
                  ;
4A66 D602    SUI     02H      ;SUBTRACT 2 FROM MENU NUMBER (SEE ABOVE)
4A68 CD754A  CALL    MULT50    ;MULTIPLY A BY 50 (SEE ABOVE)
4A6B 54      MOV     D,H      ;TEMPORARY STORE OF HL IN DE
4A6C 5D      MOV     E,L      ;(SAME)
4A6D 21005B  LXI     H,USRRAM  ;GET STARTING ADDRESS OF FIRST TABLE
4A70 19      DAD     D        ;ADD OFFSET TO TABLE START ADDRESS
4A71 CD7F4A  CALL    GETVARS    ;GET GLOBAL VARIABLES FROM TABLE
4A74 C9      RET

;
; MULT50 - ROUTINE TO MULTIPLY A BY 50D (32H).  RESULT WILL
; BE PLACED IN HL, AND HL ARE THE ONLY REGISTERS DESTROYED.
;
4A75 210000  MULT50: LXI     H,0000H ;CLEAR HL FOR THE RESULT
4A78 6F      MOV     L,A      ;PUT NUMBER IN L FOR INITIAL ADD
4A79 29      DAD     H        ;DAD 5 TIMES TO MULTIPLY BY 32H (50D)
4A7A 29      DAD     H        ;
4A7B 29      DAD     H        ;
4A7C 29      DAD     H        ;
4A7D 29      DAD     H        ;
4A7E C9      RET

;
4A7F 22225A  GETVARS: SHLD   GBLTBL ;PUT STARTING ADDRESS IN GLOBAL POINTER
4A82 7E      MOV     A,M      ;PUT MENU CONTROL WORD IN A
4A83 32205A  STA     MENCTRL ;STORE CONTROL WORD IN VARIABLE
4A86 23      INX     H        ;STEP POINTER UP ONE
4A87 7E      MOV     A,M      ;PUT RAMP RATE IN A
4A88 321F5A  STA     RAMPNT   ;STORE RAMP RATE IN VARIABLE
4A8B 23      INX     H        ;(CONTINUE W/SAME.....)
4A8C 7E      MOV     A,M      ;
4A8D 32155A  STA     MAXBAK   ;BACK U.S. DISTANCE
4A90 23      INX     H        ;
4A91 7E      MOV     A,M      ;
4A92 32125A  STA     MAXFNT   ;FRONT U.S. DISTANCE
4A95 23      INX     H        ;
4A96 7E      MOV     A,M      ;
4A97 32185A  STA     MAXLFT   ;LEFT U.S. DISTANCE
4A9A 23      INX     H        ;
4A9B 7E      MOV     A,M      ;
4A9C 32185A  STA     MAXRT    ;RIGHT U.S. DISTANCE
4A9F 23      INX     H        ;

```

```

4AA0 22245A      SHLD      BEGTBL ;STORE ADDRESS OF ENTRY IN BEGIN POINTER
4AA3 C9          RET

;
; PADCHK - SUBROUTINE TO DETERMINE THE STATUS OF THE PAD. IF THE
; PAD IS NOT BEING TOUCHED, THEN IT WILL SIMPLY RETURN WITH
; NO ACTION. IF THE PAD WAS TOUCHED, IT WILL CALL A ROUTINE
; TO DETERMINE IF THE LOCATION WAS AMONG VALID LOCATIONS FOR
; THE CURRENT MENU. IF SO, IT WILL CALL ANOTHER ROUTINE TO
; INITIALIZE THE MOTOR PARAMETERS.
;
4AA4 DB11      PADCHK: IN      PIAB      ;READ THE +5 LOOP FROM THE PAD CONNECTOR
4AA6 E680      ANI      PADLOOP ;MASK TO DETERMINE THE CONNECTOR STATUS
4AA8 FE80      CPI      PADLOOP ;VALUE READ, 1=CONNECTED, 0=DISCONNECTED
4AAA CABB4A      JZ      PADOK3 ;CONTINUE IF PAD CONNECTED
4AAD 3E01      MVI      A,PADLED ;GET DATA TO LIGHT PAD ERROR LED
4AAF CDF04A      CALL     SETERR ;LIGHT THE PAD ERROR LED
4AB2 3E40      MVI      A,MENERR; PAD ERROR WILL GIVE MENU ERROR
4AB4 CDFC4A      CALL     CLRERR ; SO CLEAR MENU LED
4AB7 CDEB4B      CALL     STOP      ;STOP THE CHAIR (RAMP DOWN) IF DISCONN.
4ABA C9          RET
4AB9 3E01      PADOK3: MVI      A,PADLED ;GET DATA TO CLEAR PAD ERROR LED
4ABD CDFC4A      CALL     CLRERR
4AC0 CDFD46      CALL     PADRD      ;SCAN THE PAD FOR A TOUCH
4AC3 7C          MOV      A,H        ;PUT PAD STATUS WORD IN A
4AC4 E680      ANI      TOUCH      ;
4AC6 FE80      CPI      TOUCH      ;
4AC8 C0          RNZ          ;RETURN IF NO TOUCH
4AC9 CD094B      CALL     CHKTBL     ;DETERMINE IF TOUCH WAS A VALID LOCATION
;FOR THIS CURRENT MENU (CHECK TABLE)
4ACC 3A265A      LDA      ENTRY      ;GET THE TABLE ENTRY NUMBER
;0 SIGNALS NO TABLE ENTRY VALID W/TOUCH
;ENTRY OF 1-10 MEANS A MATCH FOUND
4ACF FE00      CPI      00H        ;SEE IF THE ENTRY IS 0
4AD1 C8          RZ          ;RETURN IF NO TABLE ENTRY MATCH FOUND
4AD2 3A325A      LDA      LAST1      ;SEE IF SAME AS LAST ENTRY
4AD5 47          MOV      B,A
4AD6 3A265A      LDA      ENTRY
4AD9 B8          CMP      B
4ADA C2E64A      JNZ      PADOK4 ;IF NOT SAME AS LAST, GET NEW VARS
4ADD 2A275A      LHLD     MTRADDR ;IF SAME AS LAST, REINIT DURATION
4AE0 23          INX      H
4AE1 7E          MOV      A,M
4AE2 32315A      STA      DURATION
4AE5 C9          RET
4AE6 3A265A      PADOK4: LDA      ENTRY
4AE9 32325A      STA      LAST1
4AEC CD664B      CALL     INITMTR ;IF VALID ENTRY MATCH, INIT, MOTOR VARS.
4AEF C9          RET

;
; SETERR - ROUTINE TO TAKE THE ERROR BIT PASSED IN THE ACC. AND
; UPDATE THE ERROR LEDS TO REFLECT THE NEW ERROR. THIS WILL
; USE THE VARIABLE ERRWRD (ERROR WORD) TO STORE THE CURRENT
; SYSTEM ERRORS. WITH THIS, SEVERAL ERRORS CAN BE DISPLAYED
; DISPLAYED AT THE SAME TIME.
;

```

```

4AF0 47      SETERR: MOV      B,A      ;PUT THE NEW ERROR WORD IN B
4AF1 3A215A   LDA      ERRWRD    ;GET THE CURRENT ERROR WORD
4AF4 B0       ORA      B          ;COMBINE THE CURRENT ERRORS WITH THE NEW
4AF5 32215A   STA      ERRWRD    ;UPDATE ERROR WORD
4AF8 2F       CMA      ;COMPLEMENT FOR NEGATIVE LOGIC
4AF9 D312     OUT      PIAC      ;LIGHT THE CORRECT ERROR LEDS
4AFB C9       RET

;
; CLRERR - ROUTINE TO COMPLEMENT SETERR. THIS ROUTINE WILL TAKE
; THE ERROR BIT (LED) TO BE CLEARED (PASSED IN A) AND CLEAR
; THAT LED WHILE LEAVING THE REST ON (SET).
;
4AFC 2F      CLRERR: CMA      ;COMPLEMENT THE BIT TO BE CLEARED
4AFD 47      MOV      B,A      ;PUT THE ERROR CLEAR WORD IN B
4AFE 3A215A   LDA      ERRWRD    ;GET THE CURRENT ERROR WORD
4B01 A0       ANA      B          ;COMBINE THE CLEAR WORD WITH THE CURRENT
4B02 32215A   STA      ERRWRD    ;UPDATE ERROR WORD
4B05 2F       CMA      ;COMPLEMTN FOR NEGATIVE LOGIC
4B06 D312     OUT      PIAC      ;LIGHT (OR TURN OFF) THE CORRECT LEDS
4B08 C9       RET

;
; CHKTBL - ROUTINE TO DETERMINE WHETHER OR NOT THE LOCATION
; TOUCHED IS AMONG THE VALID LOCATIONS FOR THE CURRENT MENU.
; THE TABLE ENTRY NUMBER (OUT OF 10) WILL BE RETURNED IN
; THE VARIABLE ENTRY. IF NOT VALID, ENTRY WILL BE 0, ELSE
; ENTRY WILL BE 1-10.
;
4B09 0600     CHKTBL: MVI      B,00H  ;CLEAR B TO STORE ENTRY NUMBER
4B0B 3E00     MVI      A,00H  ;CLEAR A
4B0D 32265A   STA      ENTRY    ;CLEAR ENTRY TO BEGIN
4B10 54       MOV      D,H      ;MOVE HL TO DE SO HL CAN BE USED TO POINT
4B11 5D       MOV      E,L      ;
4B12 2A245A   LHLD     BEGTBL    ;GET STARTING ADDRESS OF MENU DATA TABLE

4B15 7E      ROWMIN: MOV      A,M      ;GET ROW/COL FROM TABLE
4B16 E6F0     ANI      0F0H    ;MASK OFF ROW
4B18 4F       MOV      C,A      ;SAVE THIS IN C
4B19 7B       MOV      A,E      ;GET ROW/COL FROM PAD
4B1A E6F0     ANI      0F0H    ;MASK OFF ROW
4B1C B9       CMP      C        ;IS PAD ROW < TABLE ROW ?
4B1D DA5A4B   JC      NXTENT1   ;IF SO FORGET IT

4B20 7E      COLMIN: MOV      A,M      ;GET ROW/COL FROM TABLE
4B21 E60F     ANI      0FH      ;MASK OFF COL
4B23 4F       MOV      C,A      ;SAVE THIS IN C
4B24 7B       MOV      A,E      ;GET ROW/COL FROM PAD
4B25 E60F     ANI      0FH      ;MASK OFF COL
4B27 B9       CMP      C        ;IS PAD COL < TABLE COL ?
4B28 DA5A4B   JC      NXTENT1   ;IF SO FORGET IT

4B2B 23      ROWMAX: INX      H      ;NEXT TABLE LOCATION
4B2C 7E      MOV      A,M      ;GET ROW/COL FROM TABLE
4B2D E6F0     ANI      0F0H    ;MASK OFF ROW
4B2F 4F       MOV      C,A      ;SAVE THIS IN C
4B30 7B       MOV      A,E      ;GET ROW/COL FROM PAD
4B31 E6F0     ANI      0F0H    ;MASK OFF ROW

```

```

4B33 B9          CMP      C          ;IS PAD ROW > TABLE ROW ?
4B34 DA3D4B      JC       COLMAX    ;IF SO ONWARD
4B37 CA3D4B      JZ       COLMAX    ;IF EQUAL ONWARD
4B3A C35B4B      JMP      NXTENT2   ;IF NEITHER THEN FORGET IT

4B3D 7E          COLMAX: MOV      A,M      ;GET ROW/COL FROM TABLE
4B3E E60F        ANI      0FH        ;MASK OFF COL
4B40 4F          MOV      C,A        ;SAVE THIS IN C
4B41 7B          MOV      A,E        ;GET ROW/COL FROM PAD
4B42 E60F        ANI      0FH        ;MASK OFF COL
4B44 B9          CMP      C          ;IS PAD COL > TABLE COL ?
4B45 DA4E4B      JC       VALID     ;IF SO ONWARD
4B48 CA4E4B      JZ       VALID     ;IF EQUAL ONWARD
4B4B C35B4B      JMP      NXTENT2   ;IF NEITHER THEN FORGET IT

4B4E 23          VALID:  INX      H          ;
4B4F 22275A      SHLD     MTRADDR
4B52 04          INR      B          ;
4B53 7B          MOV      A,B        ;
4B54 32265A      STA      ENTRY    ;STORE ENTRY NUMBER
4B57 62          MOV      H,D        ;RESTORE HL FROM DE
4B5B 6B          MOV      L,E        ;
4B59 C9          RET

4B5A 23          NXTENT1: INX      H          ;INCREMENT POINTER TO TABLE DATA FOR NEXT
4B5B 23          NXTENT2: INX      H          ;ENTRY ROW/COL MINIMUM.
4B5C 23          INX      H          ;
4B5D 23          INX      H          ;
4B5E 04          INR      B          ;INCREMENT ENTRY COUNTER
4B5F 7B          MOV      A,B        ;CHECK TO SEE IF COMPLETELY THROUGH 10 ENTRIES
4B60 FE0A        CPI      0AH        ;
4B62 C2154B      JNZ      ROWMIN    ;TRY NEXT ENTRY
4B65 C9          RET                ;NO VALID TOUCH, RETURN

;
; INITMTR - ROUTINE TO INITIALIZE THE MOTOR VARIABLES (GET THE DATA
; FROM MEMORY).
;
4B66 2A275A      INITMTR: LHLD     MTRADDR ;GET STARTING ADDRESS OF MOTOR DATA
4B69 7E          MOV      A,M        ;PUT LEFT/RIGHT MOTOR DATA IN A
4B6A E60F        ANI      0FH        ;MASK FOR RIGHT MOTOR DATA ONLY
4B6C 322C5A      STA      RMTS      ;STORE SPEED IN RIGHT MOTOR TARGET SPEED
4B6F 7E          MOV      A,M        ;GET LEFT/RIGHT MOTOR DATA AGAIN
4B70 E6F0        ANI      0F0H       ;MASK FOR LEFT MOTOR DATA ONLY
4B72 0F          RRC              ;ROTATE LEFT MOTOR DATA TO LS NIBBLE
4B73 0F          RRC              ;
4B74 0F          RRC              ;
4B75 0F          RRC              ;
4B76 322B5A      STA      LMTS      ;STORE SPEED IN LEFT MOTOR TARGET SPEED
4B79 23          INX      H          ;MOVE POINTER TO DURATION DATA
4B7A 7E          MOV      A,M        ;PUT DURATION IN A
4B7B 32315A      STA      DURATION  ;STORE MOTION DURATION
4B7E 3A1F5A      LDA      RAMPCNT
4B81 32335A      STA      CNTRAMP
4B84 C9          RET

;
; UPDATMTR - ROUTINE TO DETERMINE THE NEXT VALUE TO BE SENT TO THE

```

```

;      MOTOR SO THAT IT CAN EFFECTIVELY RAMP UP TO THE TARGET SPEED,
;      THIS WILL THEN SEND THE PROPER CODE TO THE MOTOR D/A CIRCUIT,
;      OR STOP THE MOTORS IF THE DURATION HAS BEN FULFILLED.
;
UPDATMTR:
4B85 3A335A  RAMP1: LDA    CNTRAMP ;GET THE CURRENT RAMP COUNTER (FOR RATE)
4B88 FE00      CPI    00H    ;CHECK TO SEE IF IT IS COUNTED OUT
4B8A C2E64B      JNZ    LBL4  ;IF NOT, THEN CONTINUE (DCR RAMP COUNT)

4B8D 3A2C5A  RS1CHK: LDA    RMTS    ;GET THE RIGHT MOTOR TARGET SPEED
4B90 32305A      STA    RMOTOR    ;SET DEFAULT RIGHT MOTOR VALUE
4B93 47      MOV    B,A        ;TRANSFER TO B
4B94 3A2E5A  LDA    RMCS    ;SET RIGHT MOTOR CURRENT SPEED
4B97 B8      CMP    B        ;COMPARE RMTS TO RMCS
4B98 CAA74B  JZ      LS1CHK    ;IF RMTS = RMCS THEN CHECK LEFT DATA
4B9B D2A44B  JNC      LBL1     ;IF RMCS > RMTS THEN TO LBL1
4B9E CD2D4C  CALL    RGTFRD    ;RMCS < RMTS SO INCREMENT FWD
4BA1 C3A74B  JMP     LS1CHK    ;
4BA4 CD354C  LBL1: CALL    RGTREV ;RMCS > RMTS SO INCREMENT REV
4BA7 3A2B5A  LS1CHK: LDA    LMTS    ;GET THE LEFT MOTOR TARGET SPEED
4BA8 322F5A  STA    LMOTOR    ;SET DEFAULT LEFT MOTOR VALUE
4BAD 47      MOV    B,A        ;TRANSFER TO B
4BAE 3A2D5A  LDA    LMCS    ;GET LEFT MOTOR CURRENT SPEED
4BB1 B8      CMP    B        ;COMPARE LMTS TO LMCS
4BB2 CAC14B  JZ      LBL3     ;IF LMTS = LMCS THEN CHECK DURATION
4BB5 D2BE4B  JNC      LBL2     ;IF LMCS > LMTS THEN TO LBL2
4BB8 CD3D4C  CALL    LFTFRD    ;LMCS < LMTS SO INCREMENT FWD
4BBB C3C14B  JMP     LBL3     ;
4BBE CD454C  LBL2: CALL    LFTREV ;LMCS > LMTS SO INCREMENT REV
4BC1 3A315A  LBL3: LDA    DURATION ;GET THE CURRENT DURATION COUNTER
4BC4 3D      DCR     A        ;DECREMENT THE DURATION COUNTER
4BC5 32315A  STA    DURATION ;STORE THE CHANGE IN VARIABLE
4BC8 3A305A  LDA    RMOTOR    ;GET NEW MOTOR VALUE
4BCB 323E5A  STA    RMCS    ;UPDATE THE CURRENT SPEED
4BCE 3A2F5A  LDA    LMOTOR    ;
4BD1 322D5A  STA    LMCS    ;
4BD4 07      RLC            ;ROTATE LEFT MOTOR DATA TO MS NIBBLE
4BD5 07      RLC
4BD6 07      RLC
4BD7 07      RLC
4BD8 47      MOV    B,A        ;TEMPORARY TRANSFER TO B
4BD9 3A2E5A  LDA    RMCS    ;GET NEW RIGHT MOTOR DATA
4BDC B0      ORA    B        ;COMBINE LEFT AND RIGHT MOTOR DATA
4BDD D341    OUT    PIAE    ;OUTPUT NEW MOTOR DATA TO D/A CKT
4BDF 3A1F5A  LDA    RAMPCNT
4BE2 32335A  STA    CNTRAMP
4BE3 C9      RET
4BE6 3D      LBL4: DCR     A        ;DECREMENT THE RAMP COUNT
4BE7 32335A  STA    CNTRAMP ;UPDATE THE RAMP COUNTER
4BEA C9      RET

;
; STOP - ROUTINE TO STOP THE MOTORS, USING A RAMP TO THE STOP.
;
4BE8 3E08  STOP: MVI     A,MTRSTOP
4BED 322C5A  STA     RMTS
4BF0 322B5A  STA     LMTS

```

```

4BF3 3A2E5A      LDA      RMCS
4BF6 47          MOV      B,A
4BF7 3A2D5A      LDA      LMCS
4BFA B8          CMP      B
4BFB C2024C      JNZ      STOP2
4BFE 3E08        MVI      A,MTRSTOP
4C00 B8          CMP      B
4C01 C8          RZ
4C02 3E04      STOP2: MVI      A,STOPRMP
4C04 32335A      STA      CNTRAMP
4C07 321F5A      STA      RAMPCNT
4C0A 11000A      STOP1: LXI      D,0A00H
4C0D CD8747      CALL     DELAYD
4C10 CD854B      CALL     UPDATMTR
4C13 3A2E5A      LDA      RMCS
4C16 47          MOV      B,A
4C17 3A2D5A      LDA      LMCS
4C1A B8          CMP      B
4C1B C20A4C      JNZ      STOP1
4C1E 3E08        MVI      A,MTRSTOP
4C20 B8          CMP      B
4C21 C20A4C      JNZ      STOP1
4C24 3E00        MVI      A,00H      ;ALLOW THE NEXT PAD INPUT TO BE USED
4C26 32315A      STA      DURATION
4C29 32325A      STA      LAST1
4C2C C9          RET

;
; RGTFRD - ROUTINE TO DETERMINE RMOTOR WHICH WILL ALTER RMCS
;          TO MOVE FORWARD.
;
4C2D 3A2E5A      RGTFRD: LDA      RMCS      ;OTHERWISE IF 0, GET NEW SPEED
4C30 3C          INR      A      ;STEP SPEED UP BY 1
4C31 32305A      STA      RMOTOR ;STORE IN RMOTOR TO BE SENT TO MOTOR
4C34 C9          RET

;
; RGTREV - SAME, BUT TO REVERSE.
;
4C35 3A2E5A      RGTREV: LDA      RMCS      ;OTHERWISE IF 0, GET NEW SPEED
4C38 3D          DCR      A      ;STEP SPEED DOWN BY 1
4C39 32305A      STA      RMOTOR ;STORE IN RMOTOR TO BE SENT TO MOTOR
4C3C C9          RET

;
; LFTFRD - SAME, BUT FORWARD MOTION FOR LEFT MOTOR.
;
4C3D 3A2D5A      LFTFRD: LDA      LMCS      ;OTHERWISE IF 0, GET NEW SPEED
4C40 3C          INR      A      ;STEP SPEED UP BY 1
4C41 322F5A      STA      LMOTOR ;STORE IN LMOTOR TO BE SENT TO MOTOR
4C44 C9          RET

;
; LFTREV - SAME, BUT FOR REVERSE.
;
4C45 3A2D5A      LFTREV: LDA      LMCS      ;OTHERWISE IF 0, GET NEW SPEED
4C48 3D          DCR      A      ;STEP SPEED DOWN BY 1
4C49 322F5A      STA      LMOTOR ;STORE IN LMOTOR TO BE SENT TO MOTOR
4C4C C9          RET

```

```

;
; PROMEN - ROUTINE TO ALLOW THE USER TO CREATE, RESET OR EDIT ANY
; ONE OF THE 14 POSSIBLE MENUS. IT WILL ALLOW THEM TO UPDATE
; ANY OF THE GLOBAL PARAMETERS, OR THE INDIVIDUAL AREA DATA.
;
4C4D CDEB4F PROERR: CALL RASBER ;SOUND ERROR TONE IF NEEDED
4C50 3E00 PROMEN: MVI A,FALSE
4C52 32355A STA HEARTON
4C55 3E00 MVI A,00H ;0 OUT ENTRY VAR
4C57 32265A STA ENTRY
4C5A 3E00 PRO2MEN: MVI A,00H
4C5C 322F5A STA LMOTOR
4C5F 32305A STA RMOTOR
4C62 CD734F CALL POLL ;WAIT FOR PAD TOUCH
4C65 3A345A LDA PADFLG ;GET PAD/MENU STATUS FLAG
4C68 FE00 CPI FALSE ;SEE IF PAD OK
4C6A CB RZ
4C6B E5 PUSH H ;SAVE TOUCH LOCATION INFO FOR LATER
4C6C 7C MOV A,H ;PUT VALID MENU NUMBER IN A
4C6D E61E ANI 00011110B ;MASK TO GET THE MENU NUMBER
4C6F FE02 CPI 00000010B ;TRYING TO CHANGE THE SAMPLE MENU?
4C71 CA4D4C JZ PROERR ;IF SO, SOUND ERROR AND RETURN, OTHERWISE...
4C74 1F RAR ;ROTATE TO GET MENU NUMBER
4C75 CD664A CALL OTHERS ;GET LOCATIONS OF GLOBAL VARIABLES,
; AND START OF MENU TABLE
4C78 E1 POP H ;RESTORE TOUCH LOCATION DATA
4C79 7D MOV A,L ;PUT TOUCH LOC. IN A FOR COMPARISONS
4C7A FE21 SNDCHK: CPI SOUND ;SOUND ON/OFF SELECTED?
4C7C D2B74C JNZ RANCHK ;CHECK RANGING IF NOT
4C7F 3E02 MVI A,SONOFF ;GET SOUND ON/OFF MASK
4C81 CD574F CALL ONOFF ;TOGGLE FROM ON TO OFF, VISA VERSA
4C84 C35A4C JMP PRO2MEN
4C87 FE31 RANCHK: CPI RANGE ;RANGING ON/OFF SELECTED?
4C89 C2944C JNZ RRCHK ;CHECK RAMP-RATE IF NOT
4C8C 3E04 MVI A,RONOFF ;GET RANGING ON/OFF MASK
4C8E CD574F CALL ONOFF ;TOGGLE FROM ON TO OFF, VISA VERSA
4C91 C35A4C JMP PRO2MEN
4C94 FE41 RRCHK: CPI RAMP ;RAMP RATE SELECTED?
4C96 C2B54C JNZ LRD ;CHECK LEFT RANGING DISTANCE IF NOT
4C99 CDFC4F CALL HORN1 ;HIGH BEEP
4C9C 3EEE MVI A,BAR2BEG ;GET BEGINNING ROW/COLUMN OF BAR 2
4C9E CDD04E CALL BARREAD ;WAIT FOR PAD TOUCH, RETURN VALUE OF 0-10 IN A
4CA1 47 MOV B,A ;TEMPORARY XFER TO B
4CA2 E6F0 ANI 0F0H ;MASK TO SEE IF ANY VALUE IN MS NIBBLE
4CA4 FE00 CPI 00H ;
4CA6 D24D4C JNZ PROERR ;IF ANYTHING IN MS NIBBLE, THEN THERE WAS
;AN INVALID TOUCH, SO SOUND ERROR AND RETURN
4CA9 78 MOV A,B ;RESTORE A TO B
4CAA D601 ADI 01H ;ADD ONE TO THE RATE, 'CAUSE 0 RATE NO GOOD
4CAC CD4E4F CALL RRATE ;UPDATE TABLE VALUE OF RAMP RATE
4CAF CD0250 CALL HORN2 ;LOW BEEP
4CB2 C35A4C JMP PRO2MEN
4CB5 FE71 LRD: CPI LEFTR ;LEFT RANGING SELECTED
4CB7 C2DA4C JNZ RRD ;CHECK MOTOR DURATION IF NOT
4CBA CDFC4F CALL HORN1 ;HIGH BEEP
4CBD 3EEE MVI A,BAR2BEG ;GET BEGINNING ROW/COLUMN OF BAR 2

```

```

4CBF CDD04E      CALL    BARREAD ;WAIT FOR PAD TOUCH, RETURN VALUE OF 0-10 IN A
4CC2 47          MOV     B,A      ;TEMPORARY XFER TO B
4CC3 E6F0        ANI     0F0H    ;MASK TO SEE IF ANY VALUE IN MS NIBBLE
4CC5 FE00        CPI     00H     ;
4CC7 C24D4C      JNZ     PRDERR  ;IF ANYTHING IN MS NIBBLE, THEN THERE WAS
                                ;AN INVALID TOUCH, SO SOUND ERROR AND RETURN

4CCA 78          MOV     A,B      ;RESTORE A TO B
4CCB 87          ADD     A        ;DOUBLE VALUE FOR RANGING DISTANCE
4CCC C60C        ADI     USSTOP  ;ADD 0 'CAUSE 0 NO GOOD
4CCE 2A225A      LHLD    GBLTBL  ;PUT START OF TABLE ADDRESS IN POINTER
4CD1 CDCA4E      CALL    LRUDATE ;UPDATE LEFT RANGING VALUE IN TABLE
4CD4 CD0250      CALL    HORN2   ;LOW BEEP
4CD7 C35A4C      JMP     PRD2MEN
4CDA FEB1        RRD:    CPI     RIGHTR ;LEFT RANGING SELECTED
4CDC C2FF4C      JNZ     FRD     ;CHECK MOTOR DURATION IF NOT
4CDF CDFC4F      CALL    HORN1   ;HIGH BEEP
4CE2 3EEE        MVI     A,BAR2BEG ;GET BEGINNING ROW/COLUMN OF BAR 2
4CE4 CDD04E      CALL    BARREAD ;WAIT FOR PAD TOUCH, RETURN VALUE OF 0-10 IN A
4CE7 47          MOV     B,A      ;TEMPORARY XFER TO B
4CE8 E6F0        ANI     0F0H    ;MASK TO SEE IF ANY VALUE IN MS NIBBLE
4CEA FE00        CPI     00H     ;
4CEC C24D4C      JNZ     PRDERR  ;IF ANYTHING IN MS NIBBLE, THEN THERE WAS
                                ;AN INVALID TOUCH, SO SOUND ERROR AND RETURN

4CEF 78          MOV     A,B      ;RESTORE A TO B
4CF0 87          ADD     A        ;DOUBLE VALUE FOR RANGING DISTANCE
4CF1 C60C        ADI     USSTOP  ;ADD 0 'CAUSE 0 NO GOOD
4CF3 2A225A      LHLD    GBLTBL  ;PUT START OF TABLE ADDRESS IN POINTER
4CF6 CDC94E      CALL    RRUDATE ;UPDATE LEFT RANGING VALUE IN TABLE
4CF9 CD0250      CALL    HORN2   ;LOW BEEP
4CFC C35A4C      JMP     PRD2MEN
4CFF FE61        FRD:    CPI     FRONTR ;LEFT RANGING SELECTED
4D01 C2244D      JNZ     BRD     ;CHECK MOTOR DURATION IF NOT
4D04 CDFC4F      CALL    HORN1   ;HIGH BEEP
4D07 3EEE        MVI     A,BAR2BEG ;GET BEGINNING ROW/COLUMN OF BAR 2
4D09 CDD04E      CALL    BARREAD ;WAIT FOR PAD TOUCH, RETURN VALUE OF 0-10 IN A
4D0C 47          MOV     B,A      ;TEMPORARY XFER TO B
4D0D E6F0        ANI     0F0H    ;MASK TO SEE IF ANY VALUE IN MS NIBBLE
4D0F FE00        CPI     00H     ;
4D11 C24D4C      JNZ     PRDERR  ;IF ANYTHING IN MS NIBBLE, THEN THERE WAS
                                ;AN INVALID TOUCH, SO SOUND ERROR AND RETURN

4D14 78          MOV     A,B      ;RESTORE A TO B
4D15 87          ADD     A        ;DOUBLE VALUE FOR RANGING DISTANCE
4D16 C60C        ADI     USSTOP  ;ADD 0 'CAUSE 0 NO GOOD
4D18 2A225A      LHLD    GBLTBL  ;PUT START OF TABLE ADDRESS IN POINTER
4D1B CDCB4E      CALL    FRUDATE ;UPDATE LEFT RANGING VALUE IN TABLE
4D1E CD0250      CALL    HORN2   ;LOW BEEP
4D21 C35A4C      JMP     PRD2MEN
4D24 FE51        BRD:    CPI     BACKR  ;LEFT RANGING SELECTED
4D26 C2494D      JNZ     DEFAREA ;CHECK MOTOR DURATION IF NOT
4D29 CDFC4F      CALL    HORN1   ;HIGH BEEP
4D2C 3EEE        MVI     A,BAR2BEG ;GET BEGINNING ROW/COLUMN OF BAR 2
4D2E CDD04E      CALL    BARREAD ;WAIT FOR PAD TOUCH, RETURN VALUE OF 0-10 IN A
4D31 47          MOV     B,A      ;TEMPORARY XFER TO B
4D32 E6F0        ANI     0F0H    ;MASK TO SEE IF ANY VALUE IN MS NIBBLE
4D34 FE00        CPI     00H     ;
4D36 C24D4C      JNZ     PRDERR  ;IF ANYTHING IN MS NIBBLE, THEN THERE WAS

```

```

;AN INVALID TOUCH, SO SOUND ERROR AND RETURN
4039 78      MOV      A,B      ;RESTORE A TO B
403A 87      ADD      A      ;DOUBLE VALUE FOR RANGING DISTANCE
403B C60C    ADI      USSTOP   ;ADD 0 'CAUSE 0 NO GOOD
403D 2A225A  LHLD     GBTLBL   ;PUT START OF TABLE ADDRESS IN POINTER
4040 CDCC4E  CALL     BRUDATE   ;UPDATE LEFT RANGING VALUE IN TABLE
4043 CD0250  CALL     HORN2    ;LOW BEEP
4046 C35A4C  JMP      PRO2MEN
4049 FE01    DEFAREA: CPI     DEFINE ;DEFINE MENU SELECTED?
404B C2A24D  JNZ      SELAREA ;CHECK SELECT AREA IF NOT
404E CDFC4F  CALL     HORN1
4051 3EEE    MVI      A,BAR2BEG ;GET BEGINNING ROW/COLUMN OF BAR 2
4053 CD004E  CALL     BARREAD  ;WAIT FOR PAD TOUCH, RETURN VALUE OF 0-10 IN A
4056 47      MOV      B,A      ;TEMPORARY XFER TO B
4057 E6F0    ANI      0F0H     ;MASK TO SEE IF ANY VALUE IN MS NIBBLE
4059 FE00    CPI      00H      ;
405B C24D4C  JNZ      PROERR   ;IF ANYTHING IN MS NIBBLE, THEN THERE WAS
;AN INVALID TOUCH, SO SOUND ERROR AND RETURN
405E 78      MOV      A,B      ;RESTORE A TO B
405F FE00    CPI      00H      ;SEE IF 0 SELECTED AS ENTRY NUMBER (1-10 OK)
4061 C2674D  JNZ      DEFOK1   ;IF 0 NOT SELECTED, THEN CONTINUE
4064 C34D4C  JMP      PROERR   ;OTHERWISE CALL RASBERRIES, RETURN TO PROMEN
4067 CDFC4F  DEFOK1: CALL     HORN1
406A 32265A  STA      ENTRY    ;STORE 1-10 NUMBER TEMPORARILY IN ENTRY VAR
406D CD734F  CALL     POLL      ;GET UPPER LEFT CORNER OF BOX (MIN VALUES)
4070 CDFC4F  CALL     HORN1
4073 55      MOV      D,L      ;PUT MIN VALUES IN D
4074 D5      PUSH     D        ;SAVE FIRST TOUCH LOCATION
4075 CD734F  CALL     POLL      ;GET LOWER RIGHT CORNER OF BOX (MAX VALUES)
4078 D1      POP      D        ;GET FIRST TOUCH LOCATION (MIN VALS)
4079 5D      MOV      E,L      ;PUT MAX DATA IN E
407A 3A265A  LDA      ENTRY    ;GET THE CURRENT ENTRY NUMBER FROM MEM
407D 2A245A  LHLD     BEGTBL   ;SET MEM POINTER AT START OF ENTRIES
4080 D601    SUI      01H     ;REDUCE ENTRY NUMBER BY 1, TO GET LOOP CNTR
4082 FE00    DEFOK2: CPI      00H ;IS A DOWN TO 0 YET?
4084 CABF4D  JZ       DEFOK3   ;IF SO, THEN POINTER IS AT CURRENT ENTRY
4087 23      INX      H        ;OTHERWISE INCREMENT POINTER 4 TIMES TO
4088 23      INX      H        ;GET TO NEXT ENTRY IN TABLE
4089 23      INX      H
408A 23      INX      H
408B 3D      DCR      A        ;DECREMENT ENTRY COUNTER (NOW AT NEXT ENTRY)
408C C3B24D  JMP      DEFOK2   ;REPEAT CHECK
408F 22295A  DEFOK3: SHLD    POINTER ;SAVE STARTING ADDRESS OF POINTER
4092 72      MOV      M,D      ;PUT MIN VALS IN TABLE
4093 23      INX      H        ;INCREMENT POINTER TO MAX VAL LOCATION
4094 73      MOV      M,E      ;PUT MAX VALS IN TABLE
4095 CD0250  CALL     HORN2    ;SOUND LOW (FINISH) HORN
4098 2A225A  LHLD     GBTLBL   ;GET START ADDRESS OF TABLE
409B 7E      MOV      A,M      ;GET TABLE CONTROL WORD
409C E6FE    ANI      1111110B ;CLEAR BIT 0 TO SIGNAL MENU NOT EMPTY
409E 77      MOV      M,A      ;RESTORE THE CONTROL WORD
409F C35A4C  JMP      PRO2MEN ;RETURN W/OUT ZEROING OUT ENTRY
40A2 FEB1    SELAREA: CPI     SELECT ;SELECT AREA SELECTED?
40A4 C2E14D  JNZ      LMTR     ;CHECK LEFT MOTOR SPEED IF NOT
40A7 CDFC4F  CALL     HORN1
40AA 3EEE    MVI      A,BAR2BEG ;GET BEGINNING ROW/COLUMN OF BAR 2

```

```

4DAC CDD04E      CALL    BARREAD ;WAIT FOR PAD TOUCH, RETURN VALUE OF 0-10 IN A
4DAF 47          MOV     B,A      ;TEMPORARY XFER TO B
4DB0 E6F0        ANI     0F0H     ;MASK TO SEE IF ANY VALUE IN MS NIBBLE
4DB2 FE00        CPI     00H      ;
4DB4 C24D4C      JNZ     PROERR   ;IF ANYTHING IN MS NIBBLE, THEN THERE WAS
                                ;AN INVALID TOUCH, SO SOUND ERROR AND RETURN
4DB7 78          MOV     A,B      ;RESTORE A TO B
4DB8 FE00        CPI     00H      ;SEE IF 0 SELECTED AS ENTRY NUMBER (1-10 OK)
4DBA C2C04D      JNZ     SELOK1   ;IF 0 NOT SELECTED, THEN CONTINUE
4DBD C34D4C      JMP     PROERR   ;OTHERWISE CALL RASBERRIES, RETURN TO PROMEN
4DC0 CDFC4F      SELOK1: CALL    HORN1
4DC3 32265A      STA     ENTRY    ;STORE IN ENTRY VAR FOR TIME BEING
4DC6 2A245A      LHLD    BEGTBL   ;SET MEM POINTER AT START OF ENTRIES
4DC9 D601        SUI     01H      ;REDUCE ENTRY NUMBER BY 1, TO GET LOOP CNTR
4DCB FE00        SELOK2: CPI     00H ;IS A DOWN TO 0 YET?
4DCD CAD84D      JZ      SELOK3   ;IF SO, THEN POINTER IS AT CURRENT ENTRY
4DD0 23          INX     H        ;OTHERWISE INCREMENT POINTER 4 TIMES TO
4DD1 23          INX     H        ;GET TO NEXT ENTRY IN TABLE
4DD2 23          INX     H
4DD3 23          INX     H
4DD4 3D          DCR     A        ;DECREMENT ENTRY COUNTER (NOW AT NEXT ENTRY)
4DD5 C3CB4D      JMP     SELOK2   ;REPEAT CHECK
4DD8 22295A      SELOK3: SHLD    POINTER ;SAVE STARTING ADDRESS OF POINTER
4DDB CD0250      CALL    HORN2    ;LOW BEEP (FINISH)
4DDE C35A4C      JMP     PRO2MEN  ;RETURN W/OUT ZEROING OUT ENTRY
4DE1 FEC1        LMTR:  CPI     LEFTM ;WAS LEFT MOTOR SETTING SELECTED?
4DE3 C2F64D      JNZ     RMTR     ;IF NOT, CHECK THE RIGHT MOTOR LOCATION
4DE6 3EFF        MVI     A,TRUE   ;SET FLAG FOR LEFT MOTOR DATA
4DEB 322F5A      STA     LMOTOR
4DEB 3A265A      LDA     ENTRY    ;GET THE TABLE ENTRY VALUE
4DEE FE00        CPI     00H      ;SEE IF CURRENT ENTRY IS 0 (SHOULD NOT BE)
4DF0 C20B4E      JNZ     MOK1     ;IF NOT 0, THEN CONTINUE
4DF3 C34D4C      JMP     PROERR   ;OTHERWISE SOUND ALARM AND RETURN
4DF6 FED1        RMTR:  CPI     RIGHTM ;WAS LEFT MOTOR SETTING SELECTED?
4DF8 C24A4E      JNZ     DUR      ;IF NOT, CHECK THE RIGHT MOTOR LOCATION
4DFB 3E00        MVI     A,FALSE  ;CLEAR FLAG FOR LEFT MOTOR DATA
4DFD 322F5A      STA     LMOTOR
4E00 3A265A      LDA     ENTRY    ;GET THE TABLE ENTRY VALUE
4E03 FE00        CPI     00H      ;SEE IF CURRENT ENTRY IS 0 (SHOULD NOT BE)
4E05 C20B4E      JNZ     MOK1     ;IF NOT 0, THEN CONTINUE
4E08 C34D4C      JMP     PROERR   ;OTHERWISE SOUND ALARM AND RETURN
4E0B CDFC4F      MOK1:  CALL    HORN1
4E0E 3EEB        MVI     A,BAR1BEG ;GET BEGINNING ROW/COLUMN OF BAR 1
4E10 CDD04E      CALL    BARREAD ;WAIT FOR PAD TOUCH, RETURN VALUE OF 0-10 IN A
4E13 47          MOV     B,A      ;TEMPORARY XFER TO B
4E14 E6F0        ANI     0F0H     ;MASK TO SEE IF ANY VALUE IN MS NIBBLE
4E16 FE00        CPI     00H      ;
4E18 C24D4C      JNZ     PROERR   ;IF ANYTHING IN MS NIBBLE, THEN THERE WAS
                                ;AN INVALID TOUCH, SO SOUND ERROR AND RETURN
4E1B 2A295A      LHLD    POINTER  ;SET MEM POINTER TO CORRECT ENTRY
4E1E 23          INX     H        ;GET TO LEFT MOTOR DATA
4E1F 23          INX     H
4E20 3A2F5A      LDA     LMOTOR   ;GET LEFT MOTOR SETTING FLAG
4E23 FE00        CPI     FALSE    ;IF FALSE, THEN ENTERING RIGHT MOTOR DATA
4E25 CA3B4E      JZ      RSET     ;GO TO SET RIGHT MOTOR SPEED
4E28 78          MOV     A,B      ;GET BAR INPUT BACK

```

```

4E29 C603      ADI      03H      ;ADD 3 TO GET CORRECT MOTOR DATA
4E2B 07        RLC
4E2C 07        RLC
4E2D 07        RLC
4E2E 07        RLC
4E2F 47        MOV      B,A      ;PUT IN B AGAIN
4E30 7E        MOV      A,M      ;GET CURRENT MOTOR SETTING
4E31 E60F      ANI      0FH      ;CLEAR OUT OLD LEFT MOTOR DATA
4E33 B0        ORA      B        ;COMBINE NEW LEFT DATA W/OLD RIGHT DATA
4E34 77        MOV      M,A      ;RESTORE IN TABLE
4E35 CD0250    CALL     HORN2     ;LOW BEEP (FINISH)
4E3B C35A4C    JMP      PRO2MEN
4E3B 78        RSET:    MOV      A,B      ;GET BAR INPUT BACK
4E3C C603      ADI      03H      ;ADD 3 TO GET CORRECT MOTOR DATA
4E3E 47        MOV      B,A      ;PUT BACK IN B AGAIN
4E3F 7E        MOV      A,M      ;GET OLD MOTOR DATA (CURRENT SETTINGS)
4E40 E6F0      ANI      0F0H     ;CLEAR OUT OLD RIGHT MOTOR DATA
4E42 B0        ORA      B        ;COMBINE OLD LEFT DATA W/NEW RIGHT
4E43 77        MOV      M,A      ;RESTORE IN TABLE
4E44 CD0250    CALL     HORN2     ;LOW BEEP (FINISH)
4E47 C35A4C    JMP      PRO2MEN
4E4A FEE1      DUR:    CPI      TIME    ;DURATION AREA SELECTED?
4E4C D27F4E    JNZ      RESMEN     ;CHECK RESET MENU SELECTION
4E4F 3A265A    LDA      ENTRY     ;GET THE TABLE ENTRY VALUE
4E52 FE00      CPI      00H      ;SEE IF CURRENT ENTRY IS 0 (SHOULD NOT BE)
4E54 C25A4E    JNZ      DOK1      ;IF NOT 0, THEN CONTINUE
4E57 C34D4C    JMP      PROERR     ;OTHERWISE SOUND ALARM AND RETURN
4E5A CDFC4F    DOK1:    CALL     HORN1
4E5D 3EEE      MVI      A,BAR2BEG ;GET BEGINNING ROW/COLUMN OF BAR 2
4E5F CDD04E    CALL     BARREAD    ;WAIT FOR PAD TOUCH, RETURN VALUE OF 0-10 IN A
4E62 47        MOV      B,A      ;TEMPORARY XFER TO B
4E63 E6F0      ANI      0F0H     ;MASK TO SEE IF ANY VALUE IN MS NIBBLE
4E65 FE00      CPI      00H      ;
4E67 C24D4C    JNZ      PROERR     ;IF ANYTHING IN MS NIBBLE, THEN THERE WAS
                                        ;AN INVALID TOUCH, SO SOUND ERROR AND RETURN
4E6A 78        MOV      A,B      ;RESTORE TOUCH BAR VALUE
4E6B 2600      MVI      H,00H     ;0 OUT H FOR MULT
4E6D 6F        MOV      L,A      ;PUT VALUE IN L FOR MULT
4E6E CD4F49    CALL     MULT10    ;MULTIPLY TOUCHED BAR VALUE BY 10
4E71 7D        MOV      A,L      ;GET VALUE*10 FROM L (NEVER > FFH)
4E72 2A295A    LHLD     POINTER   ;SET MEM POINTER AT CORRECT ENTRY IN TABLE
4E75 23        INX      H        ;MOVE POINTER TO DURATION LOCATION IN TABLE
4E76 23        INX      H
4E77 23        INX      H
4E78 77        MOV      M,A      ;PUT NEW DURATION IN TABLE
4E79 CD0250    CALL     HORN2     ;LOW BEEP (FINISH)
4E7C C35A4C    JMP      PRO2MEN
4E7F FE09      RESMEN:  CPI      RESET ;WAS THE RESET MENU FUNCTION SELECTED?
4E81 C24D4C    JNZ      PROERR     ;RETURN AND SOUND ERROR IF NO MENU CHOICE
                                        ;WAS SELECTED (BAD TOUCH)
4E84 CDFC4F    CALL     HORN1     ;BEEP TO ACKNOWLEDGE THE FIRST TOUCH
4E87 CD734F    CALL     POLL      ;WAIT FOR SECOND RESET TO VERIFY
4E8A 7D        MOV      A,L
4E8B FE09      CPI      RESET     ;WAS THE RESET VERIFIED?
4E8D C24D4C    JNZ      PROERR     ;RETURN WITH ERROR SOUND IF NOT.
4E90 CDFC4F    CALL     HORN1

```

```

4E93 2A225A      LHLD      GBLTBL ;OTHERWISE, RESET MENU
4E96 3E07        MVI      A,00000111B ;SET CONTROL WORD FOR RESET MENU,
                                ; SOUND ON, RANGING ON.
4E98 77          MOV      M,A      ;STORE MENU CONTROL WORD IN TABLE
4E99 23          INX      H
4E9A 3E04        MVI      A,04H    ;DEFAULT RAMP RATE
4E9C 77          MOV      M,A
4E9D 23          INX      H
4E9E 3E0A        MVI      A,0AH    ;DEFAULT A, B, C AND D RANGING DISTANCES
4EA0 77          MOV      M,A
4EA1 23          INX      H
4EA2 77          MOV      M,A
4EA3 23          INX      H
4EA4 77          MOV      M,A
4EA5 23          INX      H
4EA6 77          MOV      M,A
4EA7 23          INX      H
                                ;DONE WITH GLOBAL VARS (ALL DEFAULT NOW)
4EA8 060A        MVI      B,0AH    ;LOAD COUNTER FOR TEN MENU AREAS TO DEFAULT
4EAA 3E00        DEFLT: MVI      A,00H ;STORE 0'S IN ROW/COL MIN/MAX FOR DEFAULTS
4EAC 77          MOV      M,A
4EAD 23          INX      H
4EAE 77          MOV      M,A
4EAF 23          INX      H
4EB0 3E08        MVI      A,MTRSTOP ;SET SPEED DEFAULTS (LEFT/RIGHT) AT STOP
4EB2 07          RLC
4EB3 07          RLC
4EB4 07          RLC
4EB5 07          RLC
4EB6 F608        ORI      MTRSTOP
4EB8 77          MOV      M,A
4EB9 23          INX      H
4EBA 3E00        MVI      A,00H    ;DEFAULT DURATION TO 0
4EBC 77          MOV      M,A
4EBD 23          INX      H
4EBE 05          DCR      B        ;DECREMENT TABLE ENTRY COUNTER
4EBF 88          CMP      B        ;IS COUNT DOWN TO ZERO?
4EC0 C2AA4E      JNZ      DEFLT    ;REPEAT FOR NEXT ENTRY IF NOT
4EC3 CD0250      CALL     HORN2    ;CALL LOW TONE TO SIGNAL DONE
4EC6 C35A4C      JMP      PRO2MEN

```

```

;
; RRUDATE, LRUDATE, FRUDATE, BRUDATE - ROUTINES TO MODIFY THE EXISTING
; VALUES FOR RANGING DISTANCE (RRUDATE=RIGHT RANGE UPDATE)
;

```

```

4EC9 23          RRUDATE: INX      H      ;POINTER AT DESIRED RANGE IN TABLE
4ECA 23          LRUDATE: INX      H
4ECB 23          FRUDATE: INX      H
4ECC 23          BRUDATE: INX      H
4ECD 23          INX      H
4ECE 77          MOV      M,A      ;PUT NEW RANGING DISTANCE IN TABLE
4ECF C9          RET

```

```

;
;
; BARREAD - ROUTINE TO TAKE THE ROW/COLUMN START OF THE BAR,
; AND RETURN A NUMBER 1-11 CORRESPONDING TO BAR LOCATIONS
; 0-10. THIS VALUE CAN THEN BE MANIPULATED TO DO WHATEVER.
;
4ED0 F5      BARREAD: PUSH    PSW
4ED1 C0734F      CALL    POLL    ;WAIT FOR PAD TOUCH
4ED4 F1      POP     PSW
4ED5 BD      RR0:    CMP     L      ;TOUCHED WHERE ON THE BAR?
4ED6 C2DC4E      JNZ     RR1      ;IF NOT, CHECK THE NEXT
4ED9 3E00      MVI     A,00H      ;RETURN VALUE TOUCHED ON THE BAR
4EDB C9      RET
4EDC CD444F      RR1:    CALL    MSNDCR ;DECREMENT MS NIBBLE
4EDF BD      CMP     L      ;TOUCHED WHERE ON THE BAR?
4EE0 C2E64E      JNZ     RR2      ;IF NOT, CHECK THE NEXT
4EE3 3E01      MVI     A,01H      ;RETURN VALUE TOUCHED ON THE BAR
4EE5 C9      RET
4EE6 CD444F      RR2:    CALL    MSNDCR ;DECREMENT MS NIBBLE
4EE9 BD      CMP     L      ;TOUCHED WHERE ON THE BAR?
4EEA C2F04E      JNZ     RR3      ;IF NOT, CHECK THE NEXT
4EED 3E02      MVI     A,02H      ;RETURN VALUE TOUCHED ON THE BAR
4EEF C9      RET
4EF0 CD444F      RR3:    CALL    MSNDCR ;DECREMENT MS NIBBLE
4EF3 BD      CMP     L      ;TOUCHED WHERE ON THE BAR?
4EF4 C2FA4E      JNZ     RR4      ;IF NOT, CHECK THE NEXT
4EF7 3E03      MVI     A,03H      ;RETURN VALUE TOUCHED ON THE BAR
4EF9 C9      RET
4EFA CD444F      RR4:    CALL    MSNDCR ;DECREMENT MS NIBBLE
4EFD BD      CMP     L      ;TOUCHED WHERE ON THE BAR?
4EFE C2044F      JNZ     RR5      ;IF NOT, CHECK THE NEXT
4F01 3E04      MVI     A,04H      ;RETURN VALUE TOUCHED ON THE BAR
4F03 C9      RET
4F04 CD444F      RR5:    CALL    MSNDCR ;DECREMENT MS NIBBLE
4F07 BD      CMP     L      ;TOUCHED WHERE ON THE BAR?
4F08 C20E4F      JNZ     RR6      ;IF NOT, CHECK THE NEXT
4F0B 3E05      MVI     A,05H      ;RETURN VALUE TOUCHED ON THE BAR
4F0D C9      RET
4F0E CD444F      RR6:    CALL    MSNDCR ;DECREMENT MS NIBBLE
4F11 BD      CMP     L      ;TOUCHED WHERE ON THE BAR?
4F12 C2184F      JNZ     RR7      ;IF NOT, CHECK THE NEXT
4F15 3E06      MVI     A,06H      ;RETURN VALUE TOUCHED ON THE BAR
4F17 C9      RET
4F18 CD444F      RR7:    CALL    MSNDCR ;DECREMENT MS NIBBLE
4F1B BD      CMP     L      ;TOUCHED WHERE ON THE BAR?
4F1C C2224F      JNZ     RR8      ;IF NOT, CHECK THE NEXT
4F1F 3E07      MVI     A,07H      ;RETURN VALUE TOUCHED ON THE BAR
4F21 C9      RET
4F22 CD444F      RR8:    CALL    MSNDCR ;DECREMENT MS NIBBLE
4F25 BD      CMP     L      ;TOUCHED WHERE ON THE BAR?
4F26 C22C4F      JNZ     RR9      ;IF NOT, CHECK THE NEXT
4F29 3E08      MVI     A,08H      ;RETURN VALUE TOUCHED ON THE BAR
4F2B C9      RET

```

```

4F2C CD444F   RR9:    CALL    MSNDCR ;DECREMENT MS NIBBLE
4F2F BD                CMP     L      ;TOUCHED WHERE ON THE BAR?
4F30 C2364F                JNZ     RR10 ;IF NOT, CHECK THE NEXT
4F33 3E09                MVI     A,09H ;RETURN VALUE TOUCHED ON THE BAR
4F35 C9                RET
4F36 CD444F   RR10:   CALL    MSNDCR ;DECREMENT MS NIBBLE
4F39 BD                CMP     L      ;TOUCHED WHERE ON THE BAR?
4F3A C2404F                JNZ     BADENT1 ;IF NOT, SOUND ALARM AND RETURN
4F3D 3E0A                MVI     A,0AH ;RETURN VALUE TOUCHED ON THE BAR
4F3F C9                RET
4F40 CDEB4F   BADENT1: CALL RASBER ;SOUND RASBERRIES (BAD ENTRY NOISE)
4F43 C9                RET
;
;
; MSNDCR - ROUTINE TO DECREMENT THE MS NIBBLE OF A BYTE
;
4F44 0F   MSNDCR: RRC                ;PUT HIGH NIBBLE IN LOW
4F45 0F                RRC
4F46 0F                RRC
4F47 0F                RRC
4F48 3D                DCR     A      ;DECREMENT HIGH NIBBLE
4F49 07                RLC                ;RETURN HIGH NIBBLE TO HIGH SPOT
4F4A 07                RLC
4F4B 07                RLC
4F4C 07                RLC
4F4D C9                RET
;
;
; RRATE - ROUTINE TO MODIFY THE RAMP RATE VALUE IN THE CURRENT.
;
4F4E 2A225A   RRATE:  LHLD     GBLTBL ;SET STARTING LOCATION OF THE TABLE
4F51 23                INX     H      ;MOVE POINTER TO RAMP RATE
4F52 77                MOV     M,A    ;PUT NEW RAMP RATE IN TABLE
4F53 CD0250                CALL    HORN2 ;LOW BEEP TO SIGNAL DONE
4F56 C9                RET
;
;
; QNOFF - ROUTINE TO TOGGLE THE STATUS OF THE BIT PASSED IN A
;          IN THE MENU CONTROL WORD, TURNING SOUND/RANGING, ON/OFF.
;
4F57 47   QNOFF:  MOV     B,A      ;XFER MASK TO B
4F58 2A225A                LHLD     GBLTBL ;POINT TO FIRST LOCATION OF CURRENT TABLE
4F5B 4E                MOV     C,M    ;PUT CURRENT MENU CONTROL WORD IN C
4F5C 79                MOV     A,C    ;PUT " " " " " " A
4F5D A0                ANA     B      ;AND SELECTED BIT WITH CONTROL WORD
4F5E B8                CMP     B      ;COMPARE WITH SELECTED BIT
4F5F CA694F                JZ      TURNOFF ;IF SELECTED BIT IS A 1, THEN MAKE A 0
4F62 79   TURNOFF: MOV     A,C      ;PUT CURRENT MENU CONTROL WORD BACK IN A
4F63 B0                ORA     B      ;SET SELECTED BIT HIGH.
4F64 77                MOV     M,A    ;RESTORE CONTROL WORD TO TABLE
4F65 CDFC4F                CALL    HORN1 ;HIGH BEEP TO SIGNAL ON
4F68 C9                RET
4F69 7B   TURNOFF: MOV     A,B      ;GET SELECTED BIT MASK
4F6A 2F                CMA                ;COMPLEMENT SELECTED BIT MASK

```

```

4F6B 47      MOV     B,A      ;PUT BACK IN B
4F6C 79      MOV     A,C      ;PUT CURRENT MENU CONTROL WORD IN A
4F6D A0      ANA      B        ;AND COMPLEMENT OF SELECTED BIT MASK WITH A
4F6E 77      MOV     M,A      ;RESTORE CONTROL WORD TO TABLE
4F6F CD0250  CALL    HORN2    ;LOW BEEP TO SIGNAL OFF
4F72 C9      RET

;
;
; POLL - POLL THE TOUCH-PAD TO WAIT FOR A TOUCH, AND THEN WAIT FOR
; A NO TOUCH (THE OBJECT REMOVED FROM THE PAD).
;
4F73 CD5D48  POLL:   CALL    EXTCHK  ;CHECK FOR 'E' FROM CONSOL
4F76 CD9C4F  CALL    PDMNCHK ;CHECK FOR PAD, MENU ERROR AND SEE IF
; STILL THE PROGRAM MENU.
4F79 3A345A  LDA      PADFLG  ;GET PAD/MENU STATUS FLAG
4F7C FE00    CPI     FALSE  ;SEE IF PAD OK
4F7E C8      RZ
4F7F 7C      MOV     A,H      ;GET TOUCH STATUS WORD
4F80 E680    ANI     TOUCH   ;CHECK FOR A TOUCH
4F82 FE80    CPI     TOUCH
4F84 C2734F  JNZ     POLL
4F87 D5      NTCH:   PUSH    D
4F8B 110002  LXI     D,0200H
4F8B 1B      DELAY:  DCX     D          ;DECREMENT DELAY COUNT
4F8C 7A      MOV     A,D          ;COMPARE D AND E
4F8D B3      ORA     E          ;CHECK TO SEE IF DE=0
4F8E C28B4F  JNZ     DELAY  ;REPEAT IF <0
4F91 D1      POP     D
4F92 DB11    IN      PIAB      ;WAIT FOR NO TOUCH
4F94 E601    ANI     BEAMSK
4F96 FE01    CPI     BEAMSK
4F98 CA874F  JZ      NTCH      ;IF TOUCH, CONTINUE TO LOOP
4F9B C9      RET

;
;
; PDMNCHK - ROUTINE TO CHEK THE PAD AND MENU, SIGNAL ERRORS AND LOOP
; IF THERE ARE ANY, OR CONTINUE IF NOT. ALSO, THIS WILL
; CHECK TO SEE THAT THE PROMEN MENU IS STILL IN PLACE, AND
; RETURN IF NOT.
;
4F9C 3EFF    PDMNCHK: MVI     A,TRUE  ;SET PAD/MENU OK FLAG TO TRUE (BEGINNING)
4F9E 32345A  STA     PADFLG  ;
4FA1 DB11    IN      PIAB      ;READ THE +5 LOOP FROM THE PAD CONNECTOR
4FA3 E680    ANI     PADLOOP ;MASK TO DETERMINE THE CONNECTOR STATUS
4FA5 FE80    CPI     PADLOOP ;VALUE READ, 1=CONNECTED, 0=DISCONNECTED
4FA7 CABA4F  JZ      PADOK5  ;CONTINUE IF PAD CONNECTED
4FAA 3E01    MVI     A,PADLED ;GET DATA TO LIGHT PAD ERROR LED
4FAC CDF04A  CALL    SETERR  ;LIGHT THE PAD ERROR LED
4FAF 3E40    MVI     A,MENERR ;PAD ERROR WILL GIVE MENU ERROR
4FB1 CDFC4A  CALL    CLRERR ;SO CLEAR MENU LED
4FB4 3E00    MVI     A,FALSE  ;SIGNAL PAD/MENU OK FLAG AS NOT OK
4FB6 32345A  STA     PADFLG  ;
4FB9 C9      RET
4FBA 3E01    PADOK5: MVI     A,PADLED ;GET DATA TO CLEAR PAD LED
4FBC CDFC4A  CALL    CLRERR ;CLEAR THE PAD LED (ALL IS OK)

```

```

4FBF CDFD46      CALL    PADRD    ; DETERMINE MENU NUMBER OR STATUS (IF ERROR)
4FC2 7C          MOV     A,H      ; PUT MENU NUMBER (STATUS) IN (A)
4FC3 E640        ANI     MENERR   ; MASK FOR MENU ERROR
4FC5 FE40        CPI     MENERR   ; MASK FOR MENU ERROR
4FC7 C2D54F      JNZ     PADOK6   ; IF VALID MENU, THEN PROCEED, OTHERWISE...
4FCA 3E02        MVI     A,MENLED ; GET DATA TO LIGHT THE MENU ERROR LED
4FCC CDF04A      CALL    SETERR   ; LIGHT THE MENU ERROR LED
4FCF 3E00        MVI     A,FALSE  ; SIGNAL PAD/MENU OK FLAG AS NOT OK
4FD1 32345A      STA     PADFLG   ;
4FD4 C9          RET
4FD5 3E02        PADOK6: MVI     A,MENLED ; GET DATA TO CLEAR THE MENU LED
4FD7 CDFC4A      CALL    CLRERR   ; CLEAR MENU ERROR LED
4FDA 7C          MOV     A,H      ; PUT MENU NUMBER (STATUS) IN (A)
4FDB E601        ANI     PROMSK   ; MASK PROGRAM MENU NUMBER
4FDD FE01        CPI     PROMSK   ; MASK TO SEE IF PROGRAMMING MENU IN PAD
4FDF C8          RZ             ; RETURN IF ALL OK, AND STILL PROMEN MENU
4FE0 3E00        MVI     A,FALSE  ; SIGNAL PAD/MENU OK FLAG AS NOT OK
4FE2 32345A      STA     PADFLG   ;
4FE5 3E02        MVI     A,MENLED
4FE7 CDF04A      CALL    SETERR
4FEA C9          RET

```

;

;

```

; RASBER - SUBROUTINE TO SOUND A 'RASBERIES' TONE TO ALERT THE USER
; OF AN ERROR IN MENU ENTRY (FROM PROMEN)
;

```

```

4FEB F5          RASBER: PUSH    PSW
4FEC 3E04        MVI     A,04H   ; SOUND RASBERRY COUNTER
4FEE CDFC4F      RAS1:  CALL    HORN1 ; SOUND LOW BEEP ONCE
4FF1 C00250      CALL    HORN2   ; SOUND HIGH BEEP ONCE
4FF4 3D          DCR     A
4FF5 FE00        CPI     00H     ; IS RASBERRY COUNTER ZERO?
4FF7 C2EE4F      JNZ     RAS1    ; REPEAT IF NOT COUNTED OUT
4FFA F1          POP     PSW     ; OTHERWISE RETURN
4FFB C9          RET

```

;

;

```

; HORN1 - ROUTINE TO SOUND THE FIRST OF TWO TONES
;

```

```

4FFC F5          HORN1: PUSH    PSW
4FFD 3E40        MVI     A,40H   ; LOAD THE VALUE FOR FIRST TONE
4FFF C30550      JMP     HORNA
5002 F5          HORN2: PUSH    PSW
5003 3EC0        MVI     A,0C0H  ; LOAD VALUE FOR SECOND TONE
5005 D342        HORNA: OUT     PIAF ; SEND VALUE TO SOUND CIRCUIT
5007 D5          PUSH    D
5008 110030      LXI     D,3000H ; SHORT DELAY
500B CDB747      CALL    DELAYD
500E 3E00        MVI     A,00H
5010 D342        OUT     PIAF
5012 D1          POP     D
5013 F1          POP     PSW
5014 C9          RET

```

;

;

;

```

; ROM CONSTANT ALLOCATION - ALPHABETICAL ORDER (SORTOF)
;                               - FUNCTIONAL ORDER TOO
; COMMAND TABLE
5015 444D  CMDS:  DB      'DM'          ; DUMP MEMORY
5017 D942          DW      DUMP          ;
5019 444C          DB      'DL'          ; DOWN LOAD
501B 5742          DW      LOADER         ;
501D 454D          DB      'EM'          ; EDIT MEMORY
501F AB41          DW      MEMED          ;
5021 474F          DB      'GO'          ; GO
5023 EE40          DW      GOTO          ;
5025 4845          DB      'HE'          ; HELP COMMAND
5027 E740          DW      HELP          ;
5029 494F          DB      'IO'          ; IO PORT R/W/M
502B 4943          DW      IOPORT        ;
502D 5442          DB      'TB'          ; TEST TIMERS AND PORTS ON BOARD
502F 7841          DW      TSTBRD        ;
5031 544D          DB      'TM'          ; TEST MEMORY
5033 0541          DW      MEMTST        ;
5035 5243          DB      'RC'          ; RUN WHEELCHAIR
5037 B746          DW      INITIAL        ;
5039 0000          DB      0,0           ; END OF TABLE MARK

;
; MESSAGES...
503B 2041424F52ABORT: DB      ' ABORTED '
5045 FF          DB      EOL
5046 2057484154BAD:  DB      ' WHAT ? '
504D FF          DB      EOL
504E 29          EDM1: DB      ')'
504F 203D20      EDM3: DB      ' = '
5052 FF          DB      EOL
5053 0D0A        EDM2: DB      CR,LF
5055 28          DB      '('
5056 FF          DB      EOL
5057 0D0A        MTSBRD DB      CR,LF
5059 5445535449    DB      'TESTING TIMERS AND PIA PORTS',CR,LF
5077 4C4F4F4B20    DB      'LOOK FOR 1000 HZ SQUAREWAVE ON TIMER OUTPUTS',CR,LF
50A5 4441544120    DB      'DATA ANALIZER SHOULD SHOW PORTS COUNTING',CR,LF
50CF 494E204120    DB      'IN A STAIRSTEP FASTION',CR,LF
50E7 FF          DB      EOL
50EB 444154413D10PDA: DB      'DATA= '
50EE FF          DB      EOL
50EF 402035306D10PMM: DB      '@ 50ms * '
50FB FF          DB      EOL
50F9 2C2020      10PSM: DB      ' '
50FC FF          DB      EOL
50FD 0D0A        GCLKM: DB      CR,LF
50FF 454E544552    DB      'ENTER CPU CLK FREQ XXXX KHZ: '
511C FF          DB      EOL
511D 204F4B203FMQK: DB      ' OK ? '
5122 FF          DB      EOL
5123 0D0A        MTGOOD: DB      CR,LF
5125 4D454D4F52    DB      'MEMORY TEST PASSED'
5137 0D0AFF        DB      CR,LF,EOL
513A 0D0A        MTERR: DB      CR,LF
513C 4D454D4F52    DB      'MEMORY TEST FAILED AT '

```

```

5152 FF DB EOL
5153 3A2057524FMTWROT: DB ': WROTE '
515B FF DB EOL
515C 2C20524541MTREAD: DB ', READ '
5163 FF DB EOL
5164 20544F20 PHI: DB ' TO '
5168 FF DB EOL
5169 2046524F4DPLQ: DB ' FROM '
516F FF DB EOL
5170 4F46465345PBIAS: DB 'OFFSET VALUE ? '
517F FF DB EOL
5180 0D0A0A0A0APHELP: DB CR,LF,LF,LF,LF,LF,LF,LF
5188 2020202057 DB ' WELCOME TO THE EASYCHAIR MONITOR'
51AC 0D0A DB CR,LF
51AE 2020544845 DB ' THE FOLLOWING TWO CHARACTER COMMANDS'
51D4 0D0A DB CR,LF
51D6 2020202020 DB ' ARE AVAILIBLE : '
51F0 0D0A0D0A DB CR,LF,CR,LF
51F4 444D202044 DB 'DM Dump Memory'
5203 0D0A DB CR,LF
5205 444C202044 DB 'DL Down Load from dev. system'
5223 0D0A DB CR,LF
5225 454D202045 DB 'EM Edit Memory'
5234 0D0A DB CR,LF
5236 474F202047 DB 'GO Goto'
523E 0D0A DB CR,LF
5240 494F202049 DB 'IO I/O port r/w/m'
5252 0D0A DB CR,LF
5254 5442202054 DB 'TB Test Board utility'
526A 0D0A DB CR,LF
526C 544D202054 DB 'TM Test Memory'
527B 0D0A DB CR,LF
527D 5243202052 DB 'RC Run Chair program'
5292 0D0A DB CR,LF
5294 0D0A0A0A0A DB CR,LF,LF,LF,LF
5299 FF DB EOL
529A 0D0A PRMPT: DB CR,LF
529C 4541535936 DB 'EASY6'
52A1 0D0A DB CR,LF
52A3 203E DB ' >'
52A5 FF DB EOL
52A6 2A2A2A2A2A5TMSG: DB '*****'
52CF 0D0A DB CR,LF
52D1 2A2A2A2020 DB '*** EASYCHAIR CONTROLER V 6.0 ***'
52FA 0D0A DB CR,LF
52FC 2A2A2A2A2A DB '*****'
5325 0D0A DB CR,LF
5327 0D0A DB CR,LF
5329 2020544849 DB ' THIS SYSTEM WAS CREATED BY : '
5347 0D0A0A DB CR,LF,LF
534A 2020202020 DB ' JAMES WILLIAMS '
5361 0D0A DB CR,LF
5363 2020202020 DB ' AND '
5372 0D0A DB CR,LF
5374 2020202020 DB ' GREGORY WELCH '
538A 0D0A0A DB CR,LF,LF

```

```

538D 2020495420      DB      ' IT IS THE CONTROLLER PROGRAM THAT'
53AF 0D0A             DB      CR,LF
53B1 204F504552      DB      ' OPERATES THE ULTRASONICS, LIGHT BOARD,'
53D8 0D0A             DB      CR,LF
53DA 414E44204D      DB      'AND MOTORS OF THE EASYCHAIR WHEELCHAIR.'
5401 0D0A             DB      CR,LF
5403 2054484953      DB      ' THIS PROGRAM ALSO ALLOWS MENUS FOR'
5426 0D0A             DB      CR,LF
5428 2054484520      DB      ' THE LIGHT BOARD TO BE CREATED FOR '
544B 0D0A             DB      CR,LF
544D 2045414348      DB      ' EACH CHILD AND ADDED TO AND CHANGED '
5472 0D0A             DB      CR,LF
5474 204153204E      DB      ' AS NEEDED.'
547F 0D0A             DB      CR,LF
5481 20414C4C20      DB      ' ALL ATTEMPTS WERE MADE TO FORESEE ALL'
54A7 0D0A             DB      CR,LF
54A9 2054484520      DB      ' THE POSSIBLE PROBLEMS THAT MAY ARISE.'
54CF 0D0A             DB      CR,LF
54D1 2020202020      DB      '          HOWEVER, -NO- PROMISES.'
54EF 0D0AFF          MCRLF: DB      CR,LF,EOL
54F2 574B45454CMSG1: DB      'WHEELCHAIR NOW UNDER COMPUTER CONTROL'
5517 0D0AFF          DB      CR,LF,EOL
551A 0A0A0A0A0A0A0A: DB      LF,LF,LF,LF,LF,LF,LF,LF,LF,LF,LF,LF,LF,LF,LF,LF
552B 0A0A0A0A0A      DB      LF,LF,LF,LF,LF,LF,LF,LF,LF,LF,LF,LF,LF,HOME,EOL
5535 46524F4E54FNTHSG: DB      'FRONT = ',EOL
553E 4241434B20BAKMSG: DB      'BACK = ',EOL
5546 5249474B54RTMSG: DB      'RIGHT = ',EOL
554F 4C45465420LFTMSG: DB      'LEFT = ',EOL
5557 4C45442F54RDWERR: DB      'LED/TRANSISTOR ERROR IN ROW (0-F): ',EOL
557B 4C45442F54COLERR: DB      'LED/TRANSISTOR ERROR IN COLUMN (0-F): ',EOL
55A2 5041442054TCHMSG: DB      'PAD TOUCHED AT LOCATION: ',EOL
559C 424547494EINTMSG: DB      'BEGIN INFRA-RED TOUCH PAD DIAGNOSTICS',EOL
55E2 454E44204FENDMSG: DB      'END OF INFRA-RED TOUCH PAD DIAGNOSTICS',EOL
5609 06              SAMPLE: DB      00000110B ;MENU CONTROL WORD
560A 02              DB      02H          ;RAMP RATE
560B 10              DB      10H          ;BAK,FNT,LFT,RST RANFGING DIST
560C 10              DB      10H
560D 10              DB      10H
560E 10              DB      10H

                                ;BEGIN ENTRY 1
560F 00              DB      00H          ;ROW/COL MIN
5610 44              DB      44H          ;ROW/COL MAX
5611 AC              DB      0ACH         ;MOTOR SPEEDS (L/R)
5612 10              DB      10H          ;DURATION
                                ;NEXT ENTRIES

5613 05              DB      05H
5614 4A              DB      4AH
5615 CC              DB      0CCH
5616 10              DB      10H

                                ;
5617 0B              DB      0BH
5618 4F              DB      4FH
5619 CA              DB      0CAH
561A 10              DB      10H

                                ;
561B 50              DB      50H

```

```

561C A4      DB      0A4H
561D 8C      DB      8CH
561E 10      DB      10H
;
561F 55      DB      55H
5620 AA      DB      0AAH
5621 88      DB      88H
5622 10      DB      10H
;
5623 5B      DB      5BH
5624 AF      DB      0AFH
5625 C8      DB      0C8H
5626 10      DB      10H
;
5627 80      DB      0B0H
5628 F4      DB      0F4H
5629 46      DB      46H
562A 10      DB      10H
;
562B 85      DB      0B5H
562C FA      DB      0FAH
562D 44      DB      44H
562E 10      DB      10H
;
562F 8B      DB      0BBH
5630 FF      DB      0FFH
5631 64      DB      64H
5632 10      DB      10H
;
5633 00      DB      00H
5634 00      DB      00H
5635 8B      DB      8BH
5636 01      DB      01H

```

```

;
;END OF SAMPLE TABLE DEFINITIONS

```

```

;
; RAM ALLOCATION IN ALPHABETICAL AND FUNCTIONAL ORDER

```

```

5A00      ORG      MONRAM      ;BEGINNING OF MONITOR RAM
;
5A00      ECHOFL: DS      1      ;ECHO FLAG; 0=ECHO 1=NO ECHO
5A01      WIDTH: DS      1      ;WIDTH+1 = NUMBER OF BYTES PER LINE
5A02      BIAS: DS      2      ;BIAS FOR LOAD
5A04      RJSAV: DS      2      ;TEMP SAVE AREA FOR RETJMP
5A06      RJSP: DS      2      ;RETURN JUMP STACK POINTER
5A08      RJVECT: DS      2      ;RETURN JUMP VECTOR (PC)
5A0A      D50DIV: DS      2      ;COUNTER FOR TIMING OF 50MS PULSE
5A0C      CLKBOD: DS      2      ;CLOCK FREQUENCY IN BCD
5A0E      CLKBIN: DS      2      ;CLOCK FREQUENCY IN BINARY
5A10      FNTDST: DS      2      ;ULTRASONIC FNT DIST.
5A12      MAXFNT: DS      1      ; MAX FRONT DIST.
5A13      BAKDST: DS      2      ; BACK DIST.
5A15      MAXBAK: DS      1      ; MAX BACK DIST.
5A16      RTDST: DS      2      ; RIGHT DIST.
5A18      MAXRT: DS      1      ; MAX RIGHT DIST.
5A19      LFTDST: DS      2      ; LEFT DIST.

```

```

5A1B      MAXLFT: DS      1      ; MAX LEFT DIST.
5A1C      TIMDLY: DS      2      ; DELAY TIME
5A1E      HONOFF: DS      1      ; HIGH SPEED FLAG
5A1F      RAMPCNT: DS      1      ; RAMP RATE
5A20      MENCTRL: DS      1      ; MENU CONTROL WORD (FLAGS...)
5A21      ERRWRD: DS      1      ; CURRENT ERROR WORD (SETERR, CLRERR)
5A22      GBLTBL: DS      2      ; STARTING ADDRESS OF GLOBAL MEN VARS
5A24      BEGTBL: DS      2      ; STARTING ADDRESS OF TABLE ENTRIES
5A26      ENTRY: DS      1      ; CURRENT ENTRY NUMBER (IN DATA TABLES)
5A27      MTRADDR: DS      2      ; ADDRESS OF CURRENT ENTRY DATA
5A29      POINTER: DS      2      ; POINTER USED IN PROMEN TO UPDATE TABLE
5A2B      LMTS: DS      1      ; LEFT MOTOR TARGET SPEED
5A2C      RMTS: DS      1      ; RIGHT MOTOR TARGET SPEED
5A2D      LMCS: DS      1      ; LEFT MOTOR CURRENT SPEED
5A2E      RMCS: DS      1      ; RIGHT MOTOR CURRENT SPEED
5A2F      LMOTOR: DS      1      ; VALUES TO BE SENT TO L & R MOTORS
5A30      RMOTOR: DS      1      ;
5A31      DURATION: DS      1      ; DURATION OF MOTOR ACTION
5A32      LAST1: DS      1      ; LAST ENTRY NUMBER
5A33      CNTRAMP: DS      1      ; IMMEDIATE RAMP COUNT
5A34      PADFLG: DS      1      ; PAD/MENU OK FLAG
5A35      HEARTON: DS      1      ; HEARTBEAT ON/OFF FLAG
5A36      MISCBF: DS      17     ; BUFFER FOR USE BY COMMANDS
;          ; PUT LAST SO AN OVERRUN WON'T BOMB
;          ; SYSTEM
;          ; END OF EASYCHAIR MONITOR

; ++++++
5A47      END

```

THE INFRARED TOUCH PAD

40 - Infrared LEDS	\$ 26.00
40 - Infrared phototransistors	22.00
1 - Miscellaneous wood/plastic	60.00
1 - Electronic components	85.00
1 - Electronic cable	27.00
1 - Miscellaneous hardware	75.00
	295.00

ULTRASONIC RANGING

4 - Ultrasonic transducers	375.00
1 - Electronic components	50.00
1 - Electronic cable	32.00
	457.00

COMPUTER AND MOTOR CONTROL

1 - Working 8085 based computer	400.00
1 - Additional 8255 PIA	17.00
1 - 2816A EEPROM	16.00
2 - DS1225 8K NOVRAM	32.00
2 - AD558 D/A Converters	15.00
1 - Electronic components	15.00
1 - Power supply components	27.00
	522.00

MISCELLANEOUS COSTS

1 - Miscellaneous	55.61
-------------------	-------

=====

GRAND TOTAL \$ 1329.61

- 1) Lotto W., Milner M., "Evaluations and Development Of Powered Mobility Aids For Two-To-Five Year Olds With Neuromusculoskeletal Disorders", Ontario Crippled Children's Center, 1984
- 2) Jaffe, David L., "Polaroid Ultrasonic Ranging Sensors In Robotic Applications", Robotics Age, March, 1985
- 3) Jaffe, David L., "A Design/Development Methodolgy For Rehabilitation Devices Using Embedded Microcomputers", Rehabilitation Research and Development Center, Palo Alto Veterans Administration Medical Center, 1983
- 4) Mims, Forrest M., "Making Your Own Pressure-Sensitive Resistors", Computers and Electronics, 1983
- 5) Mims, Forrest M., "Ultrasonic Sound Polaroid Rangefinder, LM3905 Ap Note Lower Supply Voltages Device Developments", Computer and Electronics, June, 1983
- 6) Byers, T.J., "Keyboards: The Power At Your Fingertips", Computers and Electronics, September, 1984
- 7) Welch, Gregory F., Williams, James P., "The Pressure Sensitive Touch-Pad", Purdue Universtiy, school of Electrical Engineering Technology, 1985
- 8) Jaffe, David L., "Ultrasonic Head Control Unit", Rehabilitation Research and Development Center, Palo Alto Veterans Administration Medical Center, 1983
- 9) Ciarcias, "An Ultrasonic Ranging System", Byte Magazine, October, 1984