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 - Droid project
  - Ultrasonics
  - Computer - New (re-written) monitor
  - Tone generator
  - Motor control - Ref (4 volt) zero span
  - 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)
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

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

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>DESCRIPTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>The Easy Chair</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>The Original Electric Wheelchair</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Added Devices</td>
<td>3</td>
</tr>
<tr>
<td>1.3</td>
<td>The Easy Chair Block Diagram</td>
<td>4</td>
</tr>
<tr>
<td>2.0</td>
<td>The Infrared Touch Pad</td>
<td>6</td>
</tr>
<tr>
<td>2.1</td>
<td>Touch Pad Block Diagram</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Touch Pad Control Word Diagram</td>
<td>8</td>
</tr>
<tr>
<td>2.3</td>
<td>Sample LED/Transistor Circuitry</td>
<td>10</td>
</tr>
<tr>
<td>2.4.0</td>
<td>Touch Pad Schematic</td>
<td>25</td>
</tr>
<tr>
<td>2.4.1</td>
<td>(Touch Pad Schematic)</td>
<td>26</td>
</tr>
<tr>
<td>2.5</td>
<td>Printed Circuit Board and Components</td>
<td>12</td>
</tr>
<tr>
<td>2.6</td>
<td>Pictorial with cut-away</td>
<td>27</td>
</tr>
<tr>
<td>3.0</td>
<td>Ranging Module</td>
<td>15</td>
</tr>
<tr>
<td>3.1</td>
<td>Ultrasonic System Block Diagram</td>
<td>16</td>
</tr>
<tr>
<td>3.2</td>
<td>Ranging Module Schematic</td>
<td>28</td>
</tr>
<tr>
<td>3.3</td>
<td>Timing Diagram</td>
<td>17</td>
</tr>
<tr>
<td>3.4</td>
<td>Tone Generator Schematic</td>
<td>29</td>
</tr>
<tr>
<td>3.5</td>
<td>Additional Parallel Group Schematic</td>
<td>30</td>
</tr>
<tr>
<td>4.0</td>
<td>SCCS-85 Single Board Computer</td>
<td>22</td>
</tr>
<tr>
<td>4.1</td>
<td>Block Diagram</td>
<td>21</td>
</tr>
<tr>
<td>4.2</td>
<td>CPU Schematic</td>
<td>31</td>
</tr>
<tr>
<td>4.3</td>
<td>Memory Schematic</td>
<td>32</td>
</tr>
<tr>
<td>4.4</td>
<td>Timer Group, I/O Addressing Group Schematic</td>
<td>33</td>
</tr>
<tr>
<td>4.5</td>
<td>Serial Group Schematic</td>
<td>34</td>
</tr>
<tr>
<td>4.6</td>
<td>Parallel Group, Interrupt Group Schematic</td>
<td>35</td>
</tr>
<tr>
<td>4.7</td>
<td>Bus Connector Schematic</td>
<td>36</td>
</tr>
<tr>
<td>4.8</td>
<td>Power Supply Schematic</td>
<td>37</td>
</tr>
<tr>
<td>4.9</td>
<td>Motor Control Schematic</td>
<td>38</td>
</tr>
<tr>
<td>4.10</td>
<td>Status LED's and Control Switches</td>
<td>23</td>
</tr>
<tr>
<td>4.11</td>
<td>Connectors and Jacks</td>
<td>39</td>
</tr>
</tbody>
</table>
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 D88 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.

![Block Diagram of The Easy Chair](image)

**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.
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.
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 received 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 separate 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](image)

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 elevate 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 RCET 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 detector 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
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

1. 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 SN28827 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 dissipate 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 each other are demonstrated below in 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 uS 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 programable 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 SCS-85 computer. (See also Figure 4.4 Timer Group, I/O Addressing Group).

The first timer is configured to count down from 65,535 (0FFF 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.
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.
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 straightforward. 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 4.9 Motor Control Schematic)
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](image)

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
Figure 4.6 Parallel Group, Interrupt Group Schematic
Figure 4.7 Bus Connector Schematic
Figure 4.9 Motor Control Schematic
Figure 4.9 Power Supply Schematic
Figure 4.11 Connectors and Jacks.
MAIN PROGRAM LOOP

INITIAL:
Set stack pointer
Initialize parts
Initialize counters
Send stop values to motors (no 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:
HRBHEAT

HRBHEAT: Refresh heartbeat counter to maximum value
Return
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

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 CHKTABLE routine
IF valid location and DURATION not 0
THEN IF a new motion
THEN call INITMTR

Return
UPDATMTR

RAMP1: IF CNTRAMP 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 RGTFWD (determine RMOTOR)
    IF RMTS<RMCS THEN call RGTRREV (determine RMOTOR)

LSPD1CHK: Store LMCS in LMOTOR (in case LMTS=LMCS)
    IF LMTS=LMCS THEN go to LBL3:
    IF LMTS>LMCS THEN call LFTFWD (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
    Output motor speed byte
    Return

LBL4: Decrement CNTRAMP
    Return
STOP

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

STOP2:
Set CNTRAMP to STOPRAMP
Set RAMPCONT 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:

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 OFFH (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 OFFH
Then return
Else go to LOOP2:

PNTDAT:
Put row/column data in (L)
Return
Appendix B: SOFTWARE OUTLINES

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
SNOCCHK: IF sound ON/Off selected
        THEN toggle sound setting
        Go to PRO2MEN:
RANCCHK: IF ranging ON/Off selected
        THEN toggle range 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:
LBD: IF left (A) ranging distance selected
        THEN get input from range bar on pad
        Store value in correct table
        Go to PRO2MEN:
RBD: 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
        Store value as current entry number
        Get upper left corner of area
        Get lower right corner of area
        Locate area address in memory
        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
        Store value as current entry number
        Select area address in memory
        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

**SNCHK:** 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 USSTOP 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 USSTOP 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 USSTOP 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 USSTOP and store in correct table
  Go to **PRO2MEN**:

**DEFAREA:** IF define area selected
  THEN get input from range bar on pad
  Store value as current entry number
  Set upper left corner of area
  Set lower right corner of area

**DEFOK1:** Locate area address in memory
  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)
Appendix B: SOFTWARE OUTLINES

(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

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\n
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: 
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 RAMP_CNT value
Store in CNTRAMP
Return
Appendix B: SOFTWARE OUTLINES

(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:
Appendix B: SOFTWARE OUTLINES

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:

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, RG Trev, LTFWD, LFTREV

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

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

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

LFTREV: IF CNTRAMP is 0
THEN decrement LMCS
Store in LMOTOR
Return
<table>
<thead>
<tr>
<th>DECIMAL</th>
<th>LOCATION</th>
<th>NAME</th>
<th>ASCII</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>4004</td>
<td>BASE</td>
<td>0</td>
<td>BASE ADDRESS OF MONITOR</td>
</tr>
<tr>
<td>0000</td>
<td>5A00</td>
<td>MONRAM</td>
<td>0</td>
<td>ADDRESS OF RAM FOR MONITOR</td>
</tr>
<tr>
<td>0FFF</td>
<td>5FFFH</td>
<td>ENCRAM</td>
<td>F</td>
<td>END OF RAM MEMORY</td>
</tr>
<tr>
<td>0100</td>
<td>0100H</td>
<td>MCHR</td>
<td>0</td>
<td>MONITOR RAM SIZE</td>
</tr>
<tr>
<td>2900</td>
<td>2900H</td>
<td>USRAM</td>
<td>0</td>
<td>USER RAM: MONRAM-1000H; FIRST BYTE OF USER RAM</td>
</tr>
<tr>
<td>00FF</td>
<td>0FFH</td>
<td>EDL</td>
<td>F</td>
<td>END OF STRING (LINE) CHARACTER</td>
</tr>
<tr>
<td>007F</td>
<td>07H</td>
<td>BEL</td>
<td>0</td>
<td>LINE FEED</td>
</tr>
<tr>
<td>0000</td>
<td>0DH</td>
<td>CR</td>
<td>0</td>
<td>CARriage RETURN</td>
</tr>
<tr>
<td>0006</td>
<td>0AH</td>
<td>LF</td>
<td>0</td>
<td>LINE FEED</td>
</tr>
<tr>
<td>0040</td>
<td>0CH</td>
<td>HOME</td>
<td>0</td>
<td>CURSOR UP AND LEFT</td>
</tr>
<tr>
<td>0018</td>
<td>01H</td>
<td>ESC</td>
<td>0</td>
<td>ESCAPE</td>
</tr>
<tr>
<td>007F</td>
<td>07H</td>
<td>RUB</td>
<td>0</td>
<td>RUBOUT</td>
</tr>
<tr>
<td>7913</td>
<td>013H</td>
<td>XOFF</td>
<td>0</td>
<td>XOFF (Y-OFF)</td>
</tr>
<tr>
<td>0011</td>
<td>011H</td>
<td>XON</td>
<td>0</td>
<td>XON (Y-ON)</td>
</tr>
<tr>
<td>0006</td>
<td>0FH</td>
<td>MWIDTH</td>
<td>0</td>
<td>CONTROLS THE WIDTH OF &quot;DUMP&quot; &quot;PUNCH&quot; COMMANDS</td>
</tr>
<tr>
<td>0020</td>
<td>20H</td>
<td>TIME0</td>
<td>0</td>
<td>2553 TIMER ZERO</td>
</tr>
<tr>
<td>0021</td>
<td>21H</td>
<td>TIME1</td>
<td>0</td>
<td>TIMER ONE</td>
</tr>
<tr>
<td>0022</td>
<td>22H</td>
<td>TIME2</td>
<td>0</td>
<td>TIMER TWO</td>
</tr>
<tr>
<td>0025</td>
<td>25H</td>
<td>TIMCTL</td>
<td>0</td>
<td>CONTROL REGISTER</td>
</tr>
<tr>
<td>0010</td>
<td>01H</td>
<td>PIAB</td>
<td>0</td>
<td>PIA B DATA REGISTER</td>
</tr>
<tr>
<td>0011</td>
<td>011H</td>
<td>PIAB</td>
<td>0</td>
<td>PIA B DATA REGISTER</td>
</tr>
<tr>
<td>0012</td>
<td>012H</td>
<td>PIAD</td>
<td>0</td>
<td>PIA C DATA REGISTER</td>
</tr>
<tr>
<td>0041</td>
<td>041H</td>
<td>PIAD</td>
<td>0</td>
<td>PIA C DATA REGISTER</td>
</tr>
<tr>
<td>0042</td>
<td>042H</td>
<td>PIAD</td>
<td>0</td>
<td>PIA C DATA REGISTER</td>
</tr>
<tr>
<td>0043</td>
<td>043H</td>
<td>PICTRL</td>
<td>0</td>
<td>#2 PIA CONTROL REGISTER</td>
</tr>
<tr>
<td>0013</td>
<td>013H</td>
<td>PICTRL</td>
<td>0</td>
<td>#1 PIA CONTROL REGISTER</td>
</tr>
<tr>
<td>0015</td>
<td>015H</td>
<td>SERCON</td>
<td>0</td>
<td>ADIA CONTROL REGISTER</td>
</tr>
<tr>
<td>0000</td>
<td>00H</td>
<td>SERDAT</td>
<td>0</td>
<td>ADIA DATA REGISTER</td>
</tr>
<tr>
<td>0001</td>
<td>000000013</td>
<td>PROMSK</td>
<td>0</td>
<td>PROGRAM MENU DETECT</td>
</tr>
<tr>
<td>00EF</td>
<td>0FFH</td>
<td>TRUESW</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0000</td>
<td>00H</td>
<td>FALESW</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td>000000019</td>
<td>BEAMSK</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td>000000019</td>
<td>RONMSK</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0020</td>
<td>00000000B</td>
<td>COLMSK</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0040</td>
<td>01000000B</td>
<td>EXTRMSK</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0090</td>
<td>10000000B</td>
<td>TOUCH</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>001E</td>
<td>00000001B</td>
<td>VOLUMEN</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0040</td>
<td>01000000B</td>
<td>MENERA</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0020</td>
<td>00000003B</td>
<td>FADER</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0001</td>
<td>00000001B</td>
<td>PADLED</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0002</td>
<td>00000001B</td>
<td>MENLED</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0004</td>
<td>00000001B</td>
<td>LUELED</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0008</td>
<td>00000003B</td>
<td>RUBLED</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Appendix D: SOFTWARE LISTING

0010 := EUSLED EQU 00010000B ; (SAME) BACK U.S.
0020 := EUSLED EQU 00100000B ; (SAME) FRONT U.S.
0030 := PADLOOP EQU 10000000B ; PAD +5V LOOP (CONNECTED?)
0040 := LUSLED EQU 00000000B ; LEFT U.S. ........
0050 := RUSLOOP EQU 00000000B ; RIGHT U.S. ........
0060 := EUSLED EQU 00010000B ; BACK U.S. ........
0070 := FUSLOOP EQU 00001000B ; FRONT U.S. ........
0080 := PJMASK EQU 00000100B ; MASK FOR INPUT FROM PAD/JOYSTICK
0090 := INTSTOP EQU 08H ; 1/2 OF LED VALUE (USED TO STORE
00A0 := ONOFF EQU 00001000B ; FLAG TO SIGNAL RANGING ON
00B0 := SONOFF EQU 00000003B ; FLAG TO SIGNAL SOUND ON
00C0 := EMPTEN EQU 00000013H ; FLAG TO SIGNAL EMPTY MENU
                 ; (NOT PROGRAMMED)
00D0 := USESTOP EQU 00H ; ULTRASONIC STOPPING DIST
                 ; (2 INCHES)
00E0 := STMRAF EQU 00H ; STOPPING RAMP RATE

; BEGIN EQUATIES FOR PROGRAMMING MENU (SEE FROMEN)
; NUMBERS REPRESENT LOCATIONS OF MENU CHOICES ON THE
; PROGRAMMING MENU. MS NIBBLE: ROW, LS NIBBLE: COLUMN:
00F0 := SOUNDE EQU 21H ; SOUND ON/OFF
0100 := RANGE EQU 31H ; RANGING ON/OFF
0110 := RAMPS EQU 41H ; CENTER RAMP RATE
0120 := BACKR EQU 51H ; BACK...
0130 := FRONTR EQU 61H ; FRONT...
0140 := LEFTR EQU 71H ; LEFT RANGING DISTANCE
0150 := RIGHTR EQU 81H ; RIGHT...
0160 := DEFINE EQU 0A1H ; DEFINE AREA
0170 := SELECT EQU 0B1H ; SELECT AREA FOR EDITING
0180 := LEFTM EQU 0C1H ; LEFT MOTOR SPEED
0190 := RIGHTM EQU 0D1H ; RIGHT...
01A0 := TIME EQU 0E1H ; DURATION TIME
01B0 := RESSET EQU 0F1H ; RESET MENU SELECTION
01C0 := BARBAR EQU 065H
01D0 := BARDIF EQU 066H

; VECTORS FOR HARDWARE INTERRUPTS
01E0 := RST0 EQU USRAM+ 000H ; NOT USED - 'MONITOR' RESET
01F0 := RST1 EQU USRAM+ 008H
0200 := RST2 EQU USRAM+ 010H
0210 := RST3 EQU USRAM+ 018H
0220 := RST4 EQU USRAM+ 016H
0230 := TRAP EQU USRAM+ 024H
0240 := RST5 EQU USRAM+ 028H
0250 := RST6 EQU USRAM+ 020H
0260 := RST7 EQU USRAM+ 030H
0270 := RST8 EQU USRAM+ 034H
0280 := RST9 EQU USRAM+ 038H

; RST 0 ENTRY POINT: POWER UP RESET ; RST 0
4000 := ING ENSE40
6000 := 510650 ; LIX : SP : USRAM+1
4095 := 0387AB ; IXP INITIAL
Assembly 2: SOFTWARE LISTING

<table>
<thead>
<tr>
<th>Offset</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>4000</td>
<td>NOP</td>
</tr>
<tr>
<td>4001</td>
<td>NOP</td>
</tr>
<tr>
<td>4002</td>
<td>ORG BASE+0BH ; RET 1</td>
</tr>
<tr>
<td>4003</td>
<td>JMP RST1</td>
</tr>
<tr>
<td>4004</td>
<td>DB 0,0,0,0,0,0</td>
</tr>
<tr>
<td>4006</td>
<td>NOP</td>
</tr>
<tr>
<td>4007</td>
<td>NOP</td>
</tr>
<tr>
<td>4008</td>
<td>ORG BASE+10H ; RET 2</td>
</tr>
<tr>
<td>4009</td>
<td>JMP RST2</td>
</tr>
<tr>
<td>4010</td>
<td>DB 0,0,0,0,0,0</td>
</tr>
<tr>
<td>4011</td>
<td>NOP</td>
</tr>
<tr>
<td>4012</td>
<td>ORG BASE+18H ; RET 3</td>
</tr>
<tr>
<td>4013</td>
<td>JMP RST3</td>
</tr>
<tr>
<td>4014</td>
<td>DB 0,0,0,0,0,0</td>
</tr>
<tr>
<td>4016</td>
<td>NOP</td>
</tr>
<tr>
<td>4017</td>
<td>ORG BASE+20H ; RET 4</td>
</tr>
<tr>
<td>4018</td>
<td>JMP RST4</td>
</tr>
<tr>
<td>4019</td>
<td>NOP</td>
</tr>
<tr>
<td>4020</td>
<td>ORG BASE+24H ; TRAP</td>
</tr>
<tr>
<td>4021</td>
<td>JMP TRAP</td>
</tr>
<tr>
<td>4022</td>
<td>NOP</td>
</tr>
<tr>
<td>4023</td>
<td>ORG BASE+29H ; RET 5</td>
</tr>
<tr>
<td>4024</td>
<td>JMP RST5</td>
</tr>
<tr>
<td>4025</td>
<td>NOP</td>
</tr>
<tr>
<td>4026</td>
<td>ORG BASE+2CH ; RET 5.5</td>
</tr>
<tr>
<td>4027</td>
<td>JMP RST5.5</td>
</tr>
<tr>
<td>4028</td>
<td>NOP</td>
</tr>
<tr>
<td>4029</td>
<td>ORG BASE+30H ; RET 6</td>
</tr>
<tr>
<td>4030</td>
<td>JMP RST6</td>
</tr>
<tr>
<td>4031</td>
<td>NOP</td>
</tr>
<tr>
<td>4032</td>
<td>ORG BASE+34H ; RET 6.5</td>
</tr>
<tr>
<td>4033</td>
<td>JMP RST6.5</td>
</tr>
<tr>
<td>4034</td>
<td>NOP</td>
</tr>
<tr>
<td>4035</td>
<td>ORG BASE+39H ; RET 7</td>
</tr>
<tr>
<td>4036</td>
<td>JMP RST7</td>
</tr>
<tr>
<td>4037</td>
<td>NOP</td>
</tr>
<tr>
<td>4038</td>
<td>ORG BASE+3CH ; RET 7.5</td>
</tr>
<tr>
<td>4039</td>
<td>JMP RST7.5</td>
</tr>
</tbody>
</table>
Appendix C: SOFTWARE LISTING

Page 56

403F 00 NOP
4040 DB 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,04H ;NOW MAKE USART TO EXPECT MODE WORD
4049 D301 OUT SERCON
404B 7ECE MVI A,0CEH ;MODE BYTE -
404D D301 OUT SERCON ;11 00 11 10
404F 3E77 MVI A,037H ;COMMAND BYTE -
4051 D301 OUT SERCON ;0 0 1 1 0 1 1 1

; INITIALIZE TIMER CHIP TO GENERATE 16K BAUD RATE FOR
4053 210040 LXI H,009EH ; 7200 BAUD
4056 3E76 MVI A,76H ;DIVIDE TIMER I TO DIVIDE BY 64
4058 D323 OUT TIMER1 ;
405A 7D MOV A,L ;
405B D321 OUT TIMER1 ;
405D 7C MOV A,H ;
405F D321 OUT TIMER1 ;

; INITIALIZE MONITOR RAM PERTAINING TO CONSOLE I/O
4061 4F XRA A ;MAKE A ZERO
4064 3E0A STA ECHORL ;O=ECHO 1=NO ECHO
4066 3E0F MVI A,000 ;INITIALIZE WIDTH
4068 32015A STA WIDTH ;

; PRINT STARTUP MESSAGE - ALSO EFFECTIVE WAY TO WAIT A FEW
; CHAR PERIODS WHILE DOUBLE BUFFERED
; INPUT SETTLES.
4069 111A55 LXI D,CLS ;CLEAR SCREEN
406C 3D88 CALL MSG ;
406E 111452 LXI D,STMSG ;PRINT STARTUP MESSAGE
4070 3D89 CALL MSG ;
4072 DB00 IN SERDAT ;EAT POSSIBLE GARbage CHARACTER

; INITIALIZE REMAINDER OF MONITOR RAM
4074 3E7F MVI A,EDL ; ON POWER UP NO ANSWER
4077 32353A STA MISCBF ;
407A 210032 LXI H,3200H ;INITIALIZE
407D 220C5A SHLD CLKBCD ;CLOCK FREQUENCY IN SCD
4082 DA247 CALL BCDTRN ;
4085 220E5A SHLD CLKBIN ; AND BINARY
4088 DB1F4F CALL M5012B ;MULT BY 50/128 (0.4)
408B 22045A SHLD D5001V ;CASE SOMETHING GOES WRONG
408F E1H ENDRAM ;UNCOMMENT FOR MEM TEST
Appendix C: SOFTWARE LISTING

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

40BE 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 CDD24A 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 CASE40 JZ COMND ; CLEAR THE ANSWER
40A2 11BE40 LXI D,COMND ; ADDR FOR PSEUDO CALL COMPLETED BY PCHL
40A5 05 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 CDD24A CALL UCASE ;
40B2 6F MOV L,A ; PUT SECOND CHAR INTO L
40B3 CD8E49 CALL SPACE ; GOD KNOWS WHAT FOR...
40B6 011550 LXI B,CMDN ; 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 CDFF47 CALL CMP16 ; COMPARE TO COMND TYPED
40C2 CADO40 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 ;
40CA C2B940 JNZ CMDNXT ; NOT END... TRY NEXT ENTRY
40CC CDBA49 ERERR: CALL PRBAD ; PRINT ERRER MESSAGE AND RETURN. "COMND" ; IS ON STACK AS RETURN ADDR FOR COMMAND
; NOTE ALL THE COMMANDS USE ERERR LABEL.
;
40DD 3EFF CMDFND: MVI A,EDL ; CLEAR ANSWER
40DE 32365A STA MISCBF ;
40DE 0A LDAX B ; GET LOWER BYTE OF ADDRESS
40D6 5F MOV E,A ;
40D7 03 INX B ; POINT TO LOWER BYTE
40DB 0A LDAX B ; GET UPPER BYTE
40D9 57 MOV D,A ;
Appendix C: SOFTWARE LISTING

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*******************************
;
ASK:  CALL SPACE
40DD CDEE49
LXI  D,MISCBF
40E3 CD3F49
CALL MSG
40E6 C9  RET

;*******************************END OF ASK*******************************

;*******************************BEGINNING OF HELP*******************************
;
HELP:  LXI  D,PHELP
40E7 118051
CALL MSG
40EA CD3F49
RET

;*******************************END OF HELP*******************************

;*******************************BEGINNING OF GOTO*******************************
;
GOTO ROUTINE - STARTS EXECUTION IN MEMORY LOCATION
40EE CD9448  GOTO:  CALL GHWN ;GET HEX WORD
40F1 DACC40  JC  ERRER
40F4 CD5B49  CALL OKCK
40F7 DE  RC
40FB E5  PUSH H
40F9 CD0C40  CALL CRLF
40FC AF  XRA A
40FD CDF247  CALL CO
4100 CDF247  CALL CO
4103 E1  POP H
4104 E9  PCHL

;*******************************END OF GOTO*******************************

;*******************************BEGINNING OF MEMTST*******************************
;
MEMTST:  CALL FROMTO ;GET FROM AND TO ADDRESSES
410B DACC40  JC  ERRER
410B EE  XCHG
410C CD5B49  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 4C  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  MTL:  MOV H,D
411A 6B  MOV L,E
Appendix C: SOFTWARE LISTING

4118 7D  MTFILL: MOV  A,L ;LOW BYTE TO ACCUM.
411C AC  XRA  H ;XOR WITH HIGH BYTE
411D 40  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 C21A41  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  MTFST: MOV  A,L ;GET LOW BYTE
4128 AC  XRA  H ;XOR WITH HIGH BYTE
4129 A8  XRA  B ;XOR WITH MODIFIER
412A C5  PUSH  B
412B 47  MOV  B,A
412C 7E  MOV  A,M ;COMPARE WITH MEMORY LOCATION
412D B8  CMP  B ;COMPARE WITH MEMORY LOCATION
412E C25A41  JNZ  MTFXIT ;ERROR EXIT
4131 C1  POP  B ;UPDATE MEMORY ADDRESS
4132 23  INX  H ;GET HIGH BYTE
4133 7C  MOV  A,H ;COMPARE WITH STOP ADDR
4134 B9  CMP  C ;LOOP BACK
4135 C22741  JNZ  MTFST ;LOAD starting ADDR
4138 3A005A  LOD  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 CD0C48  CZ  CRLF ;CRLF IF RUNNING OUT OF LINE
4143 04  INR  B ;UPDATE MODIFIER
4144 EB  XCH6
4145 3E21  MVI  A,'!' ;PRINT PASS DONE MESSAGE
4147 C6F247  CALL  CO
414A EB  XCH6
414B C1  POP  B
414C 05  DCR  B
414D C5  PUSH  B
414E C21A41  JNZ  MT1 ;RESTART WITH NEW MODIFIER
4151 C1  POP  B
4152 112351 LXI  D,MTGOOD
4155 CD3F49  CALL  MSG
4158 C9  RET
4159 113A51  MTFXIT: LXI  D,MTERR ;FOR 255 TIMES THEN TO CMOS
415C CD3F49  CALL  MSG
415F CD7849  CALL  PHW
4162 115051 LXI  D,MTREAD
4165 CD3F49  CALL  MSG
4168 CD8349  CALL  PHB
416B 115351 LXI  D,MTWROT
416E CD3F49  CALL  MSG
4171 76  MOV  A,B
4172 CD8349  CALL  PHB
END OF MEMTST

BEGINNING OF TEST BOARD

**This routine allows the user to do a hardware check of the PIA and Timer chips.**

```
LOOPA: OUT PIAC
OUT PIAC
; Should appear as stairstep on logic analyzer
JMP LOOPA
; Loop forever
```

**BEGINNING OF MEMED**

```
MEMED: LXI D, EDM2 ; Print "CR, LF, "
CALL MSG
```

```
GHW JNC OK ; SET HEX WORD INTO HL, jump if valid
```

```
FE2F CPI / ; Bad char received - was it "/"
```

```
C0 RZ ; Go back to command level if so
```

```
CDAB CALL PRBAD ; Print "WHAT?"
```

```
C3AB CALL MEMED ; Then try again
```

```
115350 MEMED: LXI D, EDM2 ; Print "CR, LF, 
103F49 CALL MSG
```

```
CALL GHW
JNC OK
```

```
CALL PRBAD ; Print "WHAT?"
JMP MEMED ; Then try again
```
Appendix C: SOFTWARE LISTING

41C0 C03642  OK: CALL DISCON ; DISPLAY CONTENTS OF LOCATION
41C3 C0C941  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 C0F048  EDIT: CALL 6HB ; 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 '^' ; 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 DAE441  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 C0EE49  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 FE0D    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  JJ PR ; PRINT CURRENT LOCATION
4214 FE2D    E3: CPI '-' ; DASH?
4216 C21D42  JNZ E4
4219 28     DCX H
421A C32342  JMP PR ; YES - PRINT PREVIOUS LOCATION
421D FE2F    E4: CPI '/' ; SLASH?
421F C8     R2 ; EDIT ALL DONE IF SO
4220 CD8649  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, "("
Apoendil C: SOFTWARE LISTING Page 62

422C CD3F49 CALL MSG
422F CD7849 CALL PHW
4232 CD3642 CALL DISCON
4233 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 CD6349 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 CD247 CALL CO ;
424E 7D MOV A,L ;PRINT CHARACTER
4250 CD247 CALL CO ;
4252 E1 POP H ;
4253 CD6E49 CALL SPACE ;
4256 C9 RET

; **************** BEGINNING OF LOADER ****************

; HEX-FORMAT LOADER

4257 CDA049 LOADER: CALL GBIAS ;GET BIAS
425A DACC40 JC ERRER ;BAD CHAR - QUIT
425D 22025A SHLD BIAS ;STORE BIAS
4260 CD5449 CALL OKCK ;CHECK WITH USER BEFORE JUMPING
4263 D9 ;
4264 3A005A LDA ECHOFL ;SAVE ECHO FLAG
4267 3205A STA MISCBF+2;MISCBF & MISCBF+1 USED BY ANSWER
426A 3E11 MVI A,XON ;START DATA COMING
426D 32005A STA ECHOFL ;NON-ZERO VALUE (XON) TURNS OFF ECHO
426F CD247 CALL CO ;
4272 CD9442 LOAD1: CALL GETREC ;READ IN ONE REC, (A) = RECORD LENGTH
4275 B7 ORA A ;SET I-FLAG ON RECORD LENGTH
4276 3E47 MVI A,'G' ;ANSWER TO QUESTION = GOOD
4278 CA8242 JI DONE ;IF LENGTH = 0 THEN DONE
427B 7A MOV A,D ;(D) = ERRER FLAG ON GETREC RETURN
427C B7 ORA A ;SEE IF THE "ERRER" FLAG IS NON-ZERO.
427D CA7242 JI LOAD1 ;IF NOT, SO DO NEXT RECORD
4280 3E42 MVI A,'B' ;STORE "BAD" FLAG IN ANSWER TO QUESTION
4283 32365A DONE: STA MISCBF ;STORE GOOD/BAD STRING
4285 3EFF MVI A,EOL ;
4288 32375A STA MISCBF+1 ;
428B 3A3B5A LDA MISCBF+2;RESTORE ECHO FLAG
428D 32005A STA ECHOFL ;
4290 3E13 MVI A,XOFF ;STOP FURTHER OUTPUT
4292 CD247 CALL CO ;
Appendix C: SOFTWARE LISTING

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 LHLH 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 CDD442 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 ERROR 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 ;'z'; |
| 42BE C2B742 JNZ FNDMRK ;NOT RECORD MARK - GET NEXT CHAR |
| 42C1 1600 MVI D,0 ;CLEAR D REGISTER (ERROR ACCUMULATOR) |
| 42C3 C9 RET |
| END FNDMRK |
| ;*** DATA *** - INPUT ALL DATA BYTES |
| | (C) = NUMBER OF BYTES TO READ IN |
| | (D) = ERROR 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 |
| 42CB 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 BOTTOM TO ERR
; ACCUMULATOR IN D REGISTER

42D1 CDF048 LGHB: CALL GHB ; GET BYTE
42D4 F5 PUSH PSW ; SAVE BYTE
42D5 B2 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 CD7848 DUMP: CALL FROMTO ; GET BEGINNING ADDRESS AND BYTE COUNT
42DC DAC440 JC ERRER ; NON HEX CHAR TYPED - WHAT ?? ?? ?? ?
42DF CD5449 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 80 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 C0048 CALL CRLF ; GO TO NEW LINE
42F7 CD7849 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
Appendix C: SOFTWARE LISTING

Appendix C: SOFTWARE LISTING

4308 OA1343 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 CR/LF
431C C2FF42 JNZ DI1 'KEEP GOING IF NOT AT END OF LINE
431F E3 DMPLEN: 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 CD849 CALL SPACE 'SPACE OVER A COUPLE
4329 CD849 CALL SPACE ';
432C 05 PUSH D ';
432D 11365A LXY 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 CD6C48 CALL CR/LF ';
433C CD7849 CALL PHW 'PRINT MEMORY ADDRESS
433F C2FF42 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 CD048 IOPORT: CALL GHB 'SET PORT NUMBER
434C DACC40 JC ERROR ';
434F 32365A STA MISCBF+2 'DON'T TRAMP ON EOL
4352 3EC9 MVI A,0C9H 'STORE RETURN
4354 32395A STA MISCBF+3 ';
4357 CD049 CALL SPACE ';
435A CD247 CALL UCASE 'STRIP PARITY
435D CD04A CALL SPACE ';
4360 CD849 CALL CI 'SET IOPORT COMMAND
4363 F652 CPI 'R' 'IF NOT R, CHECK OTHERS
4365 C2643 JNZ IOP1 ';
4368 CD7E43 CALL IOPR 'IOPORT READ ROUTINE
436B C9 RET ';
436C F657 IOP1: CPI 'W' 'IF NOT W, CHECK M
436F C27443 JNZ IOP2 ';
4371 CD9E43 CALL IOPW 'IOPORT WRITE ROUTINE
4374 C9 RET ';

;
Appendix C: SOFTWARE LISTING

4375 FE4D IOP2: CPI 'M' ; IF NOT M, THEN WHAT DO
4377 C2CC40 JNZ ERROR ; YOU WANT ?
437A CDB943 CALL IOPM ; IOPORT MONITOR ROUTINE
4370 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 11EB50 LXI D,IOPDA ; PRINT 'DATA = '
4389 CD3F49 CALL MSG ;
438C CD8349 CALL PHE ; PRINT BYTE IN HEX
438F CDEE49 CALL SPACE ;
4392 CD4548 CALL DIGASC ; 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 11EB50 IOPW: LXI D,IOPDA ; PRINT 'DATA = '
43A0 CD3F49 CALL MSG ;
43A4 CDF048 CALL GH8 ;
43A7 DAC400 JC ERJER ; BAD CHAR TYPED...
43AA CD5B49 CALL OKCK ; CHECK TO BE SURE
43A0 D8 RC ; MUST HAVE GOOFED...
43AE F5 PUSI PSW ; SAVE DATA
43AF 3ED3 MVI A,0D3H ; STORE "OUT" INST
43B1 CD375A STA MISCBF+1 ;
43B4 F1 POP PSW ; SET DATA BACK
43B5 CD375A CALL MISCBF+1 ; WRITE DATA
43BB C9 RET ;

; IOPM - IOPORT MONITOR COMMAND

43B9 11EF50 IOPM: LXI D,1OPMM ; PRINT '@ 50MS * '
43BC CD3F49 CALL MSG ;
43BF CDF048 CALL GH8 ;
43C2 DAC400 JC ERJER ; BAD CHAR...
43C5 CD5B49 CALL OKCK ; GIVE ESCAPE A CHANCE...
43CB D8 RC ; WOULD YOU BELEIVE G FOR COUNTER?
43C9 4F MOV C,A ;
43CA CD0C48 CALL CRJF ;
43CD 3ED8 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 DIGASC ; PRINT BYTE IN ASCII
43E0 7C MOV A,H ;
43E1 CDF247 CALL CO ;
Appendix C: SOFTWARE LISTING

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 IDF M2: DCR B
43F1 DAFA43 JZ IOPM3
43F4 CD15A CALL D5OMS
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 87 ORA A CLEAR CARRY
43FF IF RAR CUT DOWN ONE
4400 A2 ANA D
4401 CD0C48 CJ CRLF
4404 C3D443 JMP IOPM1

;**************************************************************END OF IO PORT COMMAND**************
;**************************************************************

;**************************************************************
;BEGINNING OF ULTRASONIC ROUTINE
;**************************************************************

4407 CD0E4A USFNT: CALL HRTEAT 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 MILISEC.
4419 D8B747 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 SET MAX FRONT DIST.
4423 47 MOV B,A
4424 CD4445 CALL CNTCK FIND OUT HOW LONG
4427 7C MOV A,H
4428 80 CMP B BOOM HAS BEEN GONE
4429 D37B44 JC NEXTA IF SO FORGET IT
442C 3E00 MVI A,00H RESET EVERYTHING
442E D340 OUT PIAD
4430 210000 LXI H,0000H CLEAR DIST.
4433 22105A SHLD FNTDST
4436 99 RET
4437 DB42 NEXTA: IN PIAF TEST FOR BOOM
4439 E601 ANI 01H MASK OFF DIRECTION
443F FE01 CPI 01H TEST FOR DIRECTION
4443 C22044 JNZ LOOPD IF NOT BOOM THEN WAIT
4446 3E00 MVI A,00H
4448 D340 OUT PIAD RESET INIT LINE
444A CD4445 CALL CNTCK SET COUNTER IN HL
444C CD5345 CALL BEEP
444E CD5345 CALL BEEP
Appendix C: SOFTWARE LISTING

MOV B,A
CALL CNTCK
;FIND OUT HOW LONG
MOV A, H
CMP B
; BOOM HAS BEEN GONE
JC NEXTC
; IF SO FORGET IT
OUT PIAD
;RESET EVERYTHING

44D9 DB42
NEXTC: IN PIAF
;TEST FOR BOOM
ANI 04H
;MASK OFF DIRECTION
CPI 04H
;TEST FOR DIRECTION
JNZ LOOPH
;IF NOT BOOM THEN WAIT
MVI A, 00H
;RESET INIT LINE
OUT PIAD
;GET COUNTER IN HL
CALL CNTCK
;SET COUNTER IN HL
CALL BEEP
;STORE AS RIGHT
SHLD RTDST
CALL CNTCK
;SET COUNTER IN HL
CALL BEEP
;RELOAD HEARTBEAT
MVI A, 00H
;INITIALIZE 8253 COUNTER
OUT PIAD
;SEND OUT SONIC BOOM
OUT PIAD
;SEND OUT SONIC BOOM
OUT PIAD
;SEND OUT BLANK INHIBIT
OUT PIAD
; BUT KEEP BOOM HIGH
OUT PIAD
;SET MAX LEFT DIST.
LDA MAXLFT
CALL HRTBEAT
;RELOAD HEARTBEAT
MOV B,A
;FIND OUT HOW LONG
CALL CNTCK
;IF SEATER
MOV A, H
; IF SO FORGET IT
CMP B
;RESET EVERYTHING
OUT PIAD
;RESET INIT LINE
OUT PIAD
;SET COUNTER IN HL
CALL BEEP
CALL FNDDT ; GET DISTANCE
PB HLD LFTDST ; STORE AS LEFT

4544 F5 CNTCK: PUSH PSW
4545 3E80 MVI A,60H
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
4558 FE02 CPI SONOFF ; IS IT ON?
4559 C0 RNZ ; IF NOT FORGET 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 DB747 CALL DELAYD ; WAIT FOR IT
4564 3EC0 MVI A,0CH ; CHANGE TONE
4566 D342 OUT PIAF
4568 54 MOV D,H ; DELAY FOR
4569 5D MOV E,L ; DIST COUNT
456A DB747 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 01FFFC LXI B,OFHHFH ; ZERO BC
4578 03 LOOPM: INX B
4579 DB649 CALL SUB16 ; HL=HL-DE
457C DB745 JNC LOOPM ; DONE YET?
457F 69 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 ; RETURN IF RANGING OFF
458A DB11 IN PIAF ; GET +5 VOLT DATA (LOOP) FROM CONN.
458C E608 ANI FUSLOOP ; MASK TO GET FRONT CONNECTOR STATUS
458E FE08 CPI FUSLOOP ; AFTER COMPARE, I SET = CONNECTED
4590 CA9B45 JZ FUSOK ; CONTINUE IF CONNECTED
4593 3E20 MVI A,FUSLED ; GET DATA TO LIGHT FRONT US ERROR LED
4595 DB04A CALL SETERR ; LIGHT THE FRONT US ERROR LED
4598 C3C945 JMP ULTRA
459B 3E20 FUSOK: MVI A,FUSLED ; CLEAR LED ERROR
CALL CLRERR ; ULTRASONIC RANGING FRONT
CALL USFNT ; DELAY FOR SCAN
LHLD TIMDL ; MAX FRONT DIST.
LHLD FNTDST ; IF GREATER THAN MAX
CALL DELAYD ; THEN FORGET IT
MOV A, H ; PRINT RANGE DIST.
ORA CPI ; MESSAGE
CPI 00H ; AND FRONT DIST. VALUE
JZ UTR1 ; IF GREATER THEN MAX
LXI D, FNTMSG ; THEN CONT.
CALL MSG ; CHECK FOR UNSAFE DIST
PRINT ; IF LESS STOP
; GET +5 VOLT DATA (LOOP) FROM CONN.
MOV A, L ; MASK TO GET BACK CONNECTOR STATUS
CPI BUSLOOP ; AFTER COMPARE, Z SET = CONNECTED
ANL PB ; CONTINUE IF CONNECTED
BUSLOP ; LIGHT BACK US ERROR LED
CAL SETERR ; SET DATA TO LIGHT BACK US ERROR LED
LXI OOH ; ULTRA1
CALL STOP ; SET DATA TO CLEAR BACK US ERROR LED
LXI BUSLOOP ; MAX BACK DIST.
CALL DELAYD ; IF GREATER THAN MAX
MOV A, H ; THEN FORGET IT
CAL CLRERR ; PRINT RANGE DIST.
CALL USBACK ; MESSAGE
LXI ULTRA2 ; AND FRONT DIST. VALUE
LXI BUSOK ; IF GREATER THEN ULTRA STOP
LXI BUSLED ; THEN CONT.
CALL STOP ; GET +5 VOLT DATA (LOOP) FROM CONN.
LXI BUSLOOP ; MASK TO GET BACK CONNECTOR STATUS
CAL SETERR ; AFTER COMPARE, Z SET = CONNECTED
LXI BUSLED ; CONTINUE IF CONNECTED
CALL CLRERR ; LIGHT THE BACK US ERROR LED
LXI BUSLED ; GET DATA TO CLEAR BACK US ERROR LED
LXI BUSLED ; DELAY FOR SCAN
CALL DELAYD ; MAX BACK DIST.
MOV A, H ; CHECK FOR UNSAFE DIST
CALL CLRERR ; IF GREATER THEN ULTRA STOP
CALL USBACK ; THEN CONT.
LXI ULTRA2 ; IF LESS STOP
LXI BUSOK ; GET +5 VOLT DATA (LOOP) FROM CONN.
LXI BUSLOOP ; MASK TO GET RIGHT CONNECTOR STATUS
CALL SETERR ; AFTER COMPARE, Z SET = CONNECTED
LXI BUSLED ; CONTINUE IF CONNECTED
LXI BUSLED ; GET DATA TO LIGHT RIGHT US ERROR LED
LXI BUSLED ; LIGHT THE RIGHT US ERROR LED
LXI BUSLED ; DELAY FOR SCAN
LXI BUSLED ; LIGHT THE RIGHT US ERROR LED
LXI BUSLED ; GET DATA TO CLEAR RIGHT US ERROR LED
Appendix C: SOFTWARE LISTING Page 72

CALL DELAYD ; MAX RIGHT DIST.
462B 2A165A LHLD RTDST ; IF GREATER THAN MAX
462B 7C MOV A,H ; THEN FORGET IT
462C B5 ORA L ; PRINT RANGE DIST.
462D FE00 CPI 00H ; MESSAGE
462F CA4746 JZ ULTRA3 ; AND FRONT DIST. VALUE
4632 114655 CALL MSG
4635 CD3F49 CALL PHW
4638 CD7B49 CALL CRLF
463B CD0C48 CALL STOP
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 CDEB48 CALL STOP ; IF LESS STOP
4647 DB11 ULTRA3: IN PIAB ; SET +5 VOLT DATA (LOOP) FROM CONN.
4649 E640 ANI LUSLOOP ; MASK TO GET LEFT CONNECTOR STATUS
464B FE40 CPI LUSLOOP ; AFTER COMPARE, I SET = CONNECTED
464D CASB46 JZ LUSOK ; CONTINUE IF CONNECTED
4650 3E04 MVI A,LUSLED ; SET DATA TO LIGHT LEFT US ERROR LED
4652 CDF04A CALL SETERR ; LIGHT THE LEFT US ERROR LED
4655 C3B464A JMP ULTRA4 ; MAX LEFT DIST.
4658 3E04 LUSOK: MVI A,LUSLED ; SET DATA TO CLEAR RIGHT US ERROR LED
465A CDFC4A CALL CLRERR ; ULTRASONIC RANGE LEFT
465B CDF344 CALL USLFT ; DELAY FOR SCAN DELAY
4660 2A1C5A LHLD TIMDLY
4663 EB XCHG
4664 CD8747 CALL DELAYD
4667 2A195A LHLD LFTDST
466A 7C MOV A,H
466B B5 ORA L
466C FE00 CPI 00H ; IF GREATER THAN MAX
466E CA846 JA ULTRA4 ; THEN FORGET IT
4671 114F55 LXI D,LFTMSG ; PRINT RANGE DIST.
4674 CD3F49 CALL MSG ; MESSAGE
4677 CD7B49 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 CDEB48 CALL STOP ; IF LESS STOP
4686 C9 ULTRA4: RET ; GO BACK TO CALLING

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

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

; INITIAL - ROUTINE TO INITIALIZE THE CHAIR UPON STARTUP

INITIAL: LXI SP,ENDRAM+1 ; RESET STACK POINTER
46B0 3EB2 MVI A,10000010B ; PORT A: OUTPUT
46B6 D313 OUT PIACNTL ; PORT B: INPUT
46BF 3EB2 MVI A,0810H ; PORT C (UPPER): OUTPUT
46C1 3EB0 MVI A,0801H ; PORT C (LOWER): OUTPUT
46C6 3E60 MVI A,0800H ; INITIALIZE 6253 COUNTER
Appendix C: SOFTWARE LISTING

4690 D323 OUT TIMCTL ; TIMER2 BINARY COUNT MODE 0
4692 3E01 MVI A, B1H ; 0255 FIA D=IN E=IN
4694 D343 OUT PBCTRL ; F=OUT
4696 3E08 MVI A, MTRSTOP ; SET VALUE TO STOP MOTORS
4698 3225A STA RMCS ; SET MOTOR OUTPUT
469B 3225A STA LMCS ; TO STOP AT FIRST
469E 3225A STA RMTS
46A1 3225A STA LMTS
46A4 07 RLC
46A5 07 RLC
46A6 07 RLC
46A7 07 RLC
46AB F608 ORI ORI MTRSTOP ; ROTATE TO MS NIBBLE
46AA D341 OUT PIAG ; COMBINE FOR BOTH L & R MOTORS
46AC 3E30 MVI A, 30H ; SET UP TIMER 0 FOR
46AE D323 OUT TIMCTL ; HEARTBEAT PROTECTION
46B0 210008 LXI H, 0600H ; SET SCAN DELAY
46BC 221C5A SHLD TIMDLY ; FOR DELAYING ULTRA SAMPLE
46B6 111A55 LXI D, CLS ; CLEAR SCREEN
46B9 CD3F49 CALL MSB
46BC IF254 CALL MSB ; PRINT A MESSAGE SO
46BF CD3F49 CALL MSB ; 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 3235A STA HEARTON
46CA CD0E4A CALL HRTBEAT ; CHECK PAD/JOYSTICK SWITCH, UPDATE
46CD CD154A CALL MENCHK ; HEARTBEAT IF SWITCHED TO PAD
46D0 CD8245 CALL ULTRA ; DETERMINE MENU NUMBER, READ GLOBAL
46D3 CD44A CALL PADCHK ; MENU PARAMETERS
46D6 CD5D48 CALL EXCHK ; CALL U.S. RANGING ROUTINE
46D9 3A205A LDA MENCTRL ; IF CHECKED, READ DATA NEEDED
46DC F601 ANI EMPTMEN ; FOR TOUCH, READ DATA NEEDED
46DE FE01 CPI EMPTMEN ; IF A VALID TOUCH, PROGRAM MENU IF
46EF C2E946 JNZ AOK1 ; APPROPRIATE.
46E3 CDEB48 CALL STOP ; CALL PADCHK
46E6 C3C246 JMP RUNCHR
46E9 3A315A AOK1: LDA DURATION
46EC FE00 CPI 00H
46EE C2F746 JNZ AOK2
Appendix C: SOFTWARE LISTING

46F1 C0EB48 CALL STOP
46F4 C3C246 JMP RUNCHR
46F7 CDB548 ADK2: CALL UPDATMTR ;UPDAT MOTO RS IF APPROPRIATE (AT CORRECT TIME SO AS TO RAMP).
46FA C3C246 JMP RUNCHR

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

; PADRO - ROUTINE TO CHECK THE PAD, DETERMINE ERROR STATUS, RETURN;
; TOUCH LOCATION OF THERE IS ONE.

; PADRO:

46FD 210000 MENU: LXI B,00H ;RESET HL FOR NEW DATA/STATUS INFO
4700 0E00 MVI C,00H ;RESET (C) FOR NEW TOUCH LOCATION
4702 0605 MVI B,05H ;LOAD MENU SELECT COUNTER+1
4704 05 LOOP3: DCR B ;DECREMENT COUNTER OF MENU SELECT BITS
4705 78 MOV A,B ;TRANSFER (B) TO (A) FOR OUTPUT
4706 F640 ORI EXTMSK ;MASK FOR EXTRA DEMUX SELECT
4708 D310 OUT PIAA ;OUTPUT COUNT TO SELECT MENU SELECT BIT
470A 110000 LXI D,0A0H ;SET UP DELAY COUNT
470D CDB747 CALL DELAYD ;SHORT DELAY
4710 D911 IN PIAB ;INPUT TRANSISTOR STATUS
4712 E601 ANI BEAMSKe ;PREPARE INPUT DATA (MASK)
4714 B4 ORA H ;OR CURRENT (H) DATA WITH LED STATUSL
4715 17 RAL H ;ROTATE THE (A) LEFT TO MOVE BITS ONE
4716 67 MOV H,A ;TRANSFER RESULT TO (H) AGAIN
4717 78 MOV A,B ;CHECK COUNT TO SEE IF = 0
4718 FE00 CPI 00H ;
471A C20447 JNZ LOOP3 ;REPEAT PROCESS IF 5 PAIRS NOT YET SCANNED
471D 7C MOV A,H ;VALIDATE MENU DATA
471E 1F RAR ;REPOSITION THE MENU DATA (ROTATED)
471F 67 MOV H,A ;
4720 FE00 ERR1: CPI 00H ;CHECK FOR NO BEAMS BLOCKED (NO MENU)
4722 C22A47 JNZ ERR2 ;CHECK FOR NEXT ERROR IF NOT ERROR 1
4725 2640 MVI H,MENERR ;SIGNAL MENU ERROR
4727 C3B547 JMP PNTDAT ;FINISH AND PRINT MSGS
472A FE1F ERR2: CPI 1FH ;CHECK FOR ALL BEAMS BROKEN (FALSE MENU)
472C C23447 JNZ SCAN ;CONTINUE SCAN IF NO MENU ERRORS
472F 2640 MVI H,MENERR ;SIGNAL MENU ERROR
4731 C3B547 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 RMMSK ;PREPARE FOR ROW SELECT (MASK)
473C D310 OUT PIAA ;OUTPUT ROW LED/TRANSISTOR SELECT
473E 110000 LXI D,0A0H ;LOAD DELAY COUNTER
4741 CDB747 CALL DELAYD ;SHORT DELAY
4744 DB11 IN PIAB ;SET TRANSISTOR STATUS
4746 E601 ANI BEAMSKe ;PREPARE INPUT FROM TRANSISTOR (MASK)
4749 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
Appendix C: SOFTWARE LISTING

Page 75

4740 17
474F 17
4750 17
4751 17
4752 4F
4753 C35D47
4756 78
4757 FE00
4759 C23B47
475C C9
475D 06FF
475F 04
4760 78
4761 F620
4763 0310
4765 1A000
4768 CD8747
476B E811
476D E601
476F FE00
4771 CA7E47
4774 78
4775 B1
4776 4F
4777 7C
4778 F680
477A 67
477B C38547
477E 7B
477F FE0F
4780 1B
4781 CB
4782 C25F47
4785 69
4786 C9
4787 3A355A
478A FE00
478C CA9247
478F CD0E4A
4792 1B
4793 7A
4794 83
4795 C29247
4798 3A355A
479B FE00
479D CB
479E CD0E4A
47A1 C9

; ROTATE COUNT VALUE TO MS NIBBLE
; SAVE ROW IN ROW/COL REGISTER
; JUMP TO COL SCAN BECAUSE ROW TOUCHED
; MOVE COUNT TO 'A' TO DO ZERO CHECK
; REPEAT UNLESS CURRENTLY ZERO
; CONTINUE LOOP IF NOT COUNTED OUT
; RETURN IF LOOP COMPLETED W/NO TOUCH
; LOAD COLUMN COUNTER - 1
; INCREMENT COLUMN COUNTER
; TRANSFER COL COUNT TO ACCUM
; PREPARE CN LED/TRANSISTOR SELECT (MASK)
; SELECT LED/TRANSISTOR
; LOAD DELAY COUNTER
; CALL SHORT DELAY
; INPUT TRANSISTOR STATUS
; PREPARE INPUT FOR USE (MASK)
; SET ZERO FLAG
; REPEAT LOOP IF NO TOUCH ('1' = TOUCH)
; COMPLETE ROW/COL DATA IN ACCUM
; SAVE ROW/COL DATA IN 'C'
; HIGH NIBBLE: ROW
; LOW NIBBLE: COLUMN
; MASK (H) TO SHOW A VALID TOUCH
; DECREMENT DELAY COUNT
; COMPARE D AND E
; CHECK TO SEE IF DE=0
; REPEAT IF <>0

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

4787 3A355A
478A FE00
478C CA9247
478F CD0E4A
4792 1B
4793 7A
4794 83
4795 C29247
4798 3A355A
479B FE00
479D CB
479E CD0E4A
47A1 C9

; HEARTON
; FALSE
; DELAYE
; CALL HRTBEAT
; DCX D
; MOV A,D
; ORA E
; CALL HRTBEAT
;******************************************************
;* UTILITY ROUTINES -                                  *
;******************************************************
;******************************************************
; BCDTBIN - CONVERT BCD IN H&L TO BINARY IN H&L
; ONLY H&L CHANGED

BCDTBIN: PUSH B
47A2 C5
BCDTBIN: PUSH D
47A3 05
BCDTBIN: MOV D,H
47A4 5A
BCDTBIN: ;COPY ORIGINAL
47A5 5D
BCDTBIN: MOV E,L
47A6 2600
BCDTBIN: ;INITIALIZE UPPER PART OF RESULT
47A7 0600
BCDTBIN: MOV B,O
47A8 7A
BCDTBIN: ;INITIALUPPER PART OF B&C
47A9 7A
BCDTBIN: MOV A,D
47AA E0F
BCDTBIN: ;GET UPPER DIGIT
47AB 0F
BCDTBIN: RRC
47AC 0F
BCDTBIN: RRC
47AD 0F
BCDTBIN: RRC
47AE 0F
BCDTBIN: RRC
47AF E60F
BCDTBIN: ANI OFH
47B1 6F
BCDTBIN: MOV L,A
47B2 CD4F49
BCDTBIN: ;START RESULT
47B3 7A
BCDTBIN: CALL MULT10
47B4 E60F
BCDTBIN: ;SHIFT UP ONE DIGIT IN BASE 10
47B5 7A
BCDTBIN: MOV A,D
47B6 E60F
BCDTBIN: ;GET NEXT TO TOP DIGIT
47B7 4F
BCDTBIN: MOV C,A
47B8 09
BCDTBIN: DAD B
47B9 0F
BCDTBIN: ;COMBINE WITH TOP DIGIT
47BA CD4F49
BCDTBIN: CALL MULT10
47BB 7B
BCDTBIN: ;SHIFT UP ONE DIGIT IN BASE 10
47BC 0F
BCDTBIN: MOV A,E
47BD 0F
BCDTBIN: RRC
47BE 0F
BCDTBIN: RRC
47BF 0F
BCDTBIN: RRC
47C0 0F
BCDTBIN: RRC
47C1 0F
BCDTBIN: RRC
47C2 E60F
BCDTBIN: ANI OFH
47C3 4F
BCDTBIN: MOV C,A
47C4 09
BCDTBIN: DAD B
47C5 CD4F49
BCDTBIN: ;COMBINE WITH TOP TWO DIGITS
47C6 7B
BCDTBIN: CALL MULT10
47C7 E60F
BCDTBIN: ;SHIFT UP ONE DIGIT IN BASE 10
47C8 7B
BCDTBIN: MOV A,E
47C9 4F
BCDTBIN: GET BOTTOM DIGIT
47CA E60F
BCDTBIN: ANI OFH
47CB 09
BCDTBIN: DAD B
47CC 0F
BCDTBIN: ;COMBINE WITH TOP THREE DIGITS
47CD 0F
BCDTBIN: POP D
47CE D1
BCDTBIN: POP B
47CF C1
BCDTBIN: RET
47D0 C9
BCDTBIN: END BCDTBIN

CALLIN - INDIRECT CALL TO (H&L)

CALLIN: PCHL
47D1 E9
CALLIN: ;WAIT FOR DATA READY
47D2 DB01
I/O ROUTINES

CALLIN: IN SERCON
47D4 E602
CALLIN: ANI 2
47D6 CAD247
CALLIN: JZ CI
47D9 DB00
CALLIN: IN SERDAT
CALLIN: 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

47EB 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  CD:  PUSH  PSW  
47F3 DB01  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
Appendix C: SOFTWARE LISTING

; D50MS: PUSH PSW ;SAVE PSW
4815 F5 D50MS: PUSH PSW ;SAVE PSW
4816 E5 PUSH PSW ;SAVE PSW
4817 2A0A5A LHLD D50DIV ;
4818 E3 D50MSL: XTHL ;18
4819 E3 XTHL ;18
481A E3 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 C21A48 JNZ D50MSL ;11
4826 E1 POP H ;
4827 F1 POP PSW ;
4828 C9 RET ;

; D10MS - DELAY 10 MS

4829 E5 D10MS: PUSH H ;
482A F5 PUSH PSW ;
482B 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 C23048 JNZ DTWIDL ; 8085/8080 7/10
4834 F1 POP PSW ; TOTAL 26/29
4835 E1 POP H ;
4836 C9 RET ;
END D10MS ;

; D5SEC - DELAY 5 SECONDS

4837 C5 D5SEC: PUSH B ;
4838 0664 MVI B,064H ;WAIT 5 SECOND FOR +25 SWITCHING
4839 CD1548 ON16W1: CALL D50MS ;REGULATOR TO TURN ON OR OFF.
483A F5 DCR B ;
483B C23C40 JNZ ON16W1 ;
483C C1 POP B ;
483D C9 RET ;
END D5SEC ;

; DISASC - DISPLAY ASCII A-REG INTO H&L

483E F5 DISASC: PUSH PSW ;SAVE PSW
483F E67F ANI 07FH ;STRIP PARITY
4840 2620 MVI H,020H ;PUT SPACE IN H-RES
4841 FE20 CPI 020H ;CHAR < 020H ?
4842 D2534B JNC DA1 ;NO-IS PRINTABLE
4843 265E MVI H,05EH ;NOT PRINTABLE - C = '
4844 C640 ADI 040H ;MAKE PRINTABLE
4845 FE7F DA1: CPI 07FH ;IS RUBOUT ?
4846 C25A4B JNZ DA2 ;NOPE...ACK
Appendix C: SOFTWARE LISTING

Page 79

4858 3E20  MVI A,020H ;YEP-MAKE SPACE
4859 6F  DA2:  MOV L,A ;
485A F1  POP PSW ;RESTORE PSW
485B C9  RET ;

;***********************************************************************
;  **                    EXIT TO MONITOR ROUTINE                           
;***********************************************************************
485D DB01  EXTCHK: IN SERCON ;SEE IF THERE IS A CHAR
485E E602  ANI 02H ; IF ZERO THEN NONE
485F C9  RI ; IF NO CHAR. THEN FORGET IT
4860 DB00  IN SERDAT ;GET CHAR. FROM CONSOL
4861 FE45  CPI 45H ; IS IT 'E' FOR EXIT
4862 CA040  J1 START ;OFF TO MONITOR THEN
4863 C9  RET

;***********************************************************************
;  * FRMCNT - ASKS " FROM "XXX" TO "Yyyy "                             
;***********************************************************************
4864 D5  FRMCNT: PUSH D ;
4865 E5  PUSH H ;
4866 CD7E48  CALL FROMTO ;
4867 DA9D48  JC FRTOE ;
4868 E5  PUSH H ;
4869 CDF649  CALL SUB16 ;CALC NUMBER OF BYTES TO BE PROCESSED
486A D1  POP D ;
486B 2B  DCX H ;H&L = NEGATIVE OF NUMBER OF BYTES
486C C39648  JMP FRCLN ;THIS DOES XCHG & CLEANS OFF STACK...
486D C9  RET ;

; FROMTO - " FROM "XXX" TO "Yyyy ";
486E D5  FROMTO: PUSH D ;
486F E5  PUSH H ;
4870 116951  LXI D,PLO ;PROMPT FOR LO LIMIT
4871 CD3F49  CALL MSG ;
4872 CDD948  CALL GHW ;
4873 DA9D48  JC FRTOE ;RETURN IF ERROR
4874 116451  LXI D PHI ;PROMPT FOR HI LIMIT
4875 DA9D48  CALL MSG ;
4876 CDD948  CALL GHW ;
4877 DA9D48  JC FRTOE ;
4878 EB  FRCLN: XCHS ;
4879 E3  XTXL ;GET CRAP OFF OF STACK
487A E1  POP H ;
487B E3  XTXL ;
487C E1  POP H ;
487D B7  ORA A ;BETTER BE SURE CARRY IS CLEAR
487E C9  RET ;RETURN...
487F E1  FRTOE: POP H ;
4880 D1  POP D ;
4881 C9  RET ;
Appendix C: SOFTWARE LISTING

; GBIA S - GET 16 BIT BIAS

4B00 F5  GBIAS: PUSH PSW ; SAVE PSW
4B01 E3  PUSH H ; AND H\&L
4B02 D5  PUSH D ; AND D\&E
4B03 17051 LXI D,\#BIAS ; PRINT BIAS MESSAGE
4B06 CD3F49 CALL MSG ;
4B09 CD9A48 CALL GHW ; GET BIAS
4BAC D2CC48 JNC GBIA S2 ; IF NO CARRY GOOD BIAS ENTERED
4BAF FE2D CPI '-'; CHECK FOR NEGATIVE BIAS
4BB1 C8F48 JI GBIA S1 ; OHHH- WANT NEGATIVE NUMBER...
4BB4 FE0D CPI CR ; CARRIAGE RETURN?
4BB6 C2D448 JNI GBI ASE ; NOPE ERRRER
4BB9 210000 LXI H,0 ; AHHHH - NO BIAS
4BBC C3CC48 JMP GBIA S2 ;
4BBD CD9A48 GBIA S1: CALL GHW ; SET NEGATIVE BIAS
4BCE 2AD448 JC GBI ASE ; BAD CHAR...BYE
4BDF 110000 LXI D,O ;
4BB0 EB  XCHG ; SET UP SUBTRACTION FROM ZERO
4BBD CD6449 CALL GHL6 ; NEGATE BIAS
4BBF CD0C48 GBIA S2: CALL CRLF ; PREVENT A MESS
4BC0 D1  POP D ; RESTORE D
4BD0 F1  POP PSW ; RESTORE ORIGINAL H\&L
4BD1 F1  POP PSW ; RESTORE PSW
4BD2 B7  ORA A ; CLEAR CARRY
4BD3 C9  RET ; RESTORE D\&E
4BD4 D1  GBI ASE: POP D ; RESTORE ORIGINAL H\&L
4BD5 E1  POP H ; RESTORE PSW
4BD6 F1  POP PSW ; SET CARRY
4BD7 37  STC ;
4BD8 C9  RET ;

; GHW - GET HEX WORD

4BD9 C5  GHW: PUSH B ; GET FIRST BYTE IN A-REGISTER
4BDA F5  PUSH PSW ; RETURN IF BAD CHAR
4BD8 CD0F48 CALL GHB ; MOVE BYTE TO FINAL DESTINATION
4BDE DAED48 JC GHWEND ; GET SECOND BYTE
4BE1 67  MOV H,A ;
4BE2 CD0F48 CALL GHB ;
4BE5 DAED48 JC GHWEND ;
4BE6 6F  MOV L,A ;
4BE9 C1  POP B ;
4BEA 78  MOV A,B ;
4BEB C1  POP B ;
4BEC C9  RET ;
4BED C1  GHWEND: POP B ; DO NOT RESTORE A
4BEE C1  POP B ;
4BEF C9  RET ;
4BF1 CD0549 CALL GHD ; END GHW

; GHB - GET HEX BYTE

4BF0 C5  GHB: PUSH B ; SAVE B\&C
4BF1 CD0549 CALL GHD ; GET FIRST HEX DIGIT IN A-REG
Appendix C: SOFTWARE LISTING

48F4 DA0349  JC  GHBEND  ; 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  GHD  ; SET SECOND DIGIT
48FF DA0349  JC  GHBEND  ; BAD CHAR READ, RET IT TO CALLER
4902 B0    ORA  B  ; COMBINE FIRST AND SECOND DIGITS
4903 CI    GHBEND: POP  B  ; RESTORE ORIGINAL B&C
4904 C9    RET  ;
4905 CDD247 GHD:  CALL  CI  ; GET HEX DIGIT
4908 CD024A  ATH: CALL UCASE  ; GET CHARACTER & ECHO
490B FE30  CPI  'O'  ; PUT IN IF UCASE TAKEN OUT
490D DB    RC  ; MAP LOWER TO UPPER CASE AND
490E FE3A  CPI  ';'  ; STRIP PARITY.
4910 DA1C49  JC  GHD2  ; NON-HEX CHARACTER
4913 FE41  CPI  'A'  ; IF (A) <= '9'+1
4915 DB    RC  ; 'O' - '9' TYPED - CONVERT
4916 FE47  CPI  'B'  ; IF (A) < 'A'
4918 3F    CMC  ; NON-HEX CHARACTER
4919 DB    RC  ; IF (A) >= 'G'
491A D607  SUI  07H  ; NON-HEX CHARACTER
491C D630  GHD2: SUI  'O'  ; SHIFT 'A'-'F' DOWN
491E C9    RET  ; CONVERT
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 5A    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 40    MOV  C,L  ;
492D 29    DAD  H  ;*8
492E 09    DAD  B  ;*12
492F EB    XCHB  ;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
Appendix C: SOFTWARE LISTING

; MSG -
493F F5       MSG:  PUSH  PSW
4940 1A       LOUPE:  LDAI 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

; MULTI0 - MULTIPLY H&L BY 10
494F D5       MULTI0:  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  ;

; D50E D5       OKCK:  PUSH  D
4958 F5       PUSH  PSW
4959 A111D51   LXI  D,M0K
495D CD3F49   CALL  MSG
4960 CDD247   CALL  CI
4963 E67F     ANI  07FH
4965 FE0D     CPI  CR
4967 CA7149   JZ      OKCKEND
496A 113B50   LXI  D,ABORT
496D CD3F49   CALL  MSG
4970 37       STC
4971 CD0C48   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
497B F5       PHW:  PUSH  PSW  ; SAVE A-REGISTER AND FLAGS
497C 7C       MOV  A,H  ;
497D CD8349   CALL  PHB  ; PRINT HIGH-ORDER BYTE
497E 7D       MOV  A,L  ;
4981 F1       CALL  PHB  ; PRINT LOW-ORDER BYTE
4982 C7       POP  PSW  ; RESTORE A-REGISTER AND FLAGS
4983 F5       RET  ;  END  PHW

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

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 PHD

; 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'-; ; CONVERT 'A'-'F'
49A1 CDF247 PHD1: CALL CO ; 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 ;
49A9 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 RJSAV ;
49B2 2A065A LHLD RJSP ;
49B5 F9 SPHL ;
49B6 2A045A LHLD RJSAV ;
49B9 E5 PUSH H ;
49BA 2A085A LHLD 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 ;
49CB F1 POP PSW ;
Appendix C: SOFTWARE LISTING

; RET

; SHRHL - SHIFT RIGHT H&L - ZERO FILL ON LEFT
; SHIFTS (B) BITS

49C9 C9 RET ;
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 SHRHLL: DCR B ;
49CE CADD49 JI SHRHLE ;
49D1 B7 ORA A ;CLEAR CARRY FLAG
49D2 7C MOV A,H ;GET H
49D3 1F RAR ;SHIFT RIGHT
49D4 67 MOV H,A ;PUT H BACK
49D5 7D MOV A,L ;GET L
49D6 1F RAR ;SHIFT RIGHT
49D7 6F MOV L,A ;PUT L BACK
49D8 C3D049 JMP SHRHLL ;BACK...
49DB C1 SHRHL: POP B ;RESTORE A
49DC 7B 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 3F DAD SP ;
49E4 22065A SHLD RJSP ;
49E7 E1 POP H ;GET H&L BACK
49EB E3 XTHL ;GET RET ADDR
49EC 22085A SHLD RJVECT ;SQUIREL AWAY
49ED E3 XTHL ;PUT RET ADDR BACK
49EE 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 ;
Appendix C: SOFTWARE LISTING

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 FE7B   CPI   7BH
4A0A D0      RNC     ;DON'T CONVERT IF AFTER 'Z'
4A0B D620   SUI   20H   ;CONVERT LOWER TO UPPER
4A0D C9   RET

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

4A15 D811   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  JI   PADOK1   ;CONTINUE IF PAD CONNECTED
4A1D 3E01   MVI   A,PADLED   ;GET DATA TO LIGHT PAD ERROR LED
4A20 CDF04A  CALL   SETERR   ;LIGHT THE PAD ERROR LED
4A22 3E40   MVI   A,MENERR; PAD ERROR WILL GIVE MENU ERROR
4A24 D620   CALL   CLRERR   ;SO CLEAR MENU LED
4A26 D620   CALL   STOP   ;STOP THE CHAIR (RAMP DOWN) IF DISCONN.
4A28 C9   RET
4A2A 3E01   PADOK1:  MVI   A,PADLED   ;GET DATA TO CLEAR PAD LED
4A2E CDF04A  CALL   CLRERR   ;CLEAR THE PAD LED (ALL IS OK)
4A31 CDF064  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
4A40 CDEB4B  CALL   STOP   ;STOP THE CHAIR (RAMP DOWN) IF MENU ERROR
4A42 C9   RET
4A46 3E02   PADOK2:  MVI   A,MENLED   ;GET DATA TO CLEAR THE MENU LED
4A4C CDF04A  CALL   CLRERR   ;CLEAR MENU ERROR LED
4A4A 7C   MOV   A,H   ;PUT MENU NUMBER (STATUS) IN (A)
4A4E E601   ANI   PROMSK   ;MASK PROGRAM MENU NUMBER
4A50 FE01   CPI   PROMSK   ;MASK TO SEE IF PROGRAMMING MENU IN PAD
4A54 C2564A  JNZ   IFTBL1   ;SEE IF MENU 1 IF NOT PROGRAMMING MENU
4A58 CDF04A  CALL   PROMEN   ;CALL ROUTINE TO ALLOW TABLE UPDATES
4A5C C9   RET
Ae2endix C: SOFTWARE LISTING Page 86

```assembly
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 C264A  JNZ OTHERS ;CHECK FOR NUMBER 3 (TABLE 2) IF NOT 2
4A5F 21096  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)
4A69 54  MOV D,H  ;TEMPORARY STORE OF HL IN DE
4A6C 5D  MOV E,L  ;(SAME)
4A6D 21005B  LXI H,USRRA M ;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

GETVARS: SHLD SBLTBL ;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 RAMPCTR ;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 321B5A  STA MAXRT ;RIGHT U.S. DISTANCE
4A9F 23  INX H  ;
```
Appendix C: SOFTWARE LISTING

4AA0 22245A          SHLD    BESTBL  ;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
4AA9 CABB4A            J2      PADK3  ;CONTINUE IF PAD CONNECTED
4AAE 3E01              MVI     A,PADLED ;GET DATA TO LIGHT PAD ERROR LED
4AF0 CDF04A            CALL      SETERR ;LIGHT THE PAD ERROR LED
4AF2 3E40              MVI     A,MENERR; PAD ERROR WILL GIVE MENU ERROR
4AF4 CDFC4A            CALL      CLRERR  ;SO CLEAR MENU LED
4AF7 CDE84B            CALL      STOP   ;STOP THE CHAIR (RAMP DOWN) IF DISCONN.
4AF9 C9                RET

4AB0 3E01              PADOK3: MVI     A,PADLED ;GET DATA TO CLEAR PAD ERROR LED
4AB1 CDFC4A            CALL      CLRERR
4AB6 CDFD46            CALL      PADRD  ;SCAN THE PAD FOR A TOUCH
4AB7 7C                MOV     A,H     ;PUT PAD STATUS WORD IN A
4ABC E680              ANI      TOUCH  ;
4ABE FE80              CPI      TOUCH  ;
4ADF CO                RNI     ;RETURN IF NO TOUCH
4ADB CD094B            CALL      CHKTBL ;DETERMINE IF TOUCH WAS A VALID LOCATION
4ADD 3A265A            LDA     ENTRY  ;FOR THIS CURRENT MENU (CHECK TABLE)
4AE1 FE00              CPI     00H    ;GET THE TABLE ENTRY NUMBER
4AE3 C8                RZ                 ;ENTRY OF 1-10 MEANS A MATCH FOUND
4AE4 3A325A            LDA     LAST1  ;SEE IF SAME AS LAST ENTRY
4AE6 47                MOV     B,A
4AE8 3A265A            LDA     ENTRY
4AEE B5                CMP     B
4AF1 C264A             JNZ     PADOK4  ;IF NOT SAME AS LAST, SET NEW VARS
4AF3 2A275A            LHLD    MTRADDR ;IF SAME AS LAST, REINIT DURATION
4AF6 23                INX     H
4AE1 7E                MOV     A,M
4AE3 32315A            STA     DURATION
4AE5 C9                RET

4AE6 3A265A            PADOK4: LDA     ENTRY
4AE7 32325A            STA     LAST1
4AE8 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.
Apoen d

SOFTWARE

Page
SB

SETERR: MOV B, A ; PUT THE NEW ERROR WORD IN B
LDA ERRWRD ; GET THE CURRENT ERROR WORD
ORA B ; COMBINE THE CURRENT ERRORS WITH THE NEW
STA ERRWRD ; UPDATE ERROR WORD
CMA ; COMPLEMENT FOR NEGATIVE LOGIC
OUT PIAC ; LIGHT THE CORRECT ERROR LEDS
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).

B,A
ERRWRD
PIAC
;

CLRERR: CMA ; COMPLEMENT THE BIT TO BE CLEARED
MOV B, A ; PUT THE ERROR CLEAR WORD IN B
LDA ERRWRD ; GET THE CURRENT ERROR WORD
ANA B ; COMBINE THE CLEAR WORD WITH THE CURRENT
STA ERRWRD ; UPDATE ERROR WORD
CMA ; COMPLEMENT FOR NEGATIVE LOGIC
OUT PIAC ; LIGHT (OR TURN OFF) THE CORRECT LEDS
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.

CHKTBL: MVI B, A, 00H ; CLEAR B TO STORE ENTRY NUMBER
MVI A, A, 00H ; CLEAR A
STA ENTRY ; CLEAR ENTRY TO BEGIN
MOV D, H ; MOVE HL TO DE SO HL CAN BE USED TO POINT
MOV E, L ;
LHLD BEGTLBL ; GET STARTING ADDRESS OF MENU DATA TABLE

GET ROW/COL FROM TABLE
GET ROW/COL FROM PAD
GET ROW/COL FROM TABLE
IS PAD ROW < TABLE ROW ?
NXTENT1 ; IF SO FORGET IT

GET ROW/COL FROM TABLE
GET ROW/COL FROM PAD
GET ROW/COL FROM TABLE
IS PAD COL < TABLE COL ?
NXTENT1 ; IF SO FORGET IT

ROWMAX: INX H ; NEXT TABLE LOCATION
MOV A, M ; GET ROW/COL FROM TABLE
ANI OFH ; MASK OFF COL
MOV C, A ; SAVE THIS IN C
MOV A, E ; GET ROW/COL FROM PAD
ANI OFH ; MASK OFF COL
CMP C ; IS PAD COL < TABLE COL ?
JC NXTENT1 ; IF SO FORGET IT

ROWMIN: MOV A, M ; GET ROW/COL FROM TABLE
ANI OFH ; MASK OFF ROW
MOV C, A ; SAVE THIS IN C
MOV A, E ; GET ROW/COL FROM PAD
ANI OFH ; MASK OFF ROW
CMP C ; IS PAD ROW < TABLE ROW ?
JC NXTENT1 ; IF SO FORGET IT
Appendix C: SOFTWARE LISTING

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 OFH ;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 OFH ;MASK OFF COL
4B44 89 CMP C ;IS PAD COL > TABLE COL?
4B45 DA4E4B JC VALID ;IF SO ONWARD
4B48 CA4E4B JZ VALID ;IF EQUAL ONWARD
4B4A C35B4B JMP NXTENT2 ;IF NEITHER THEN FORGET IT
4B4E 23 VALID: INX H
4B4F 22275A SHLD MTRADDR
4B52 0A INR B
4B53 7B MOV A,B
4B54 32265A STA ENTRY ;STORE ENTRY NUMBER
4B57 62 MOV H,D ;RESTORE HL FROM DE
4B58 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 0A INR B ;INCREMENT ENTRY COUNTER
4B5F 7B MOV A,B ;CHECK TO SEE IF COMPLETELY THROUGH 10 ENTRIES
4B60 FE0A CPI OAH ;
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 OFH ;MASK FOR RIGHT MOTOR DATA ONLY
4B6C 32255A STA RMTS ;STORE SPEED IN RIGHT MOTOR TARGET SPEED
4B6F 7E MOV A,M ;GET LEFT/RIGHT MOTOR DATA AGAIN
4B70 E6F0 ANI OFOH ;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 32285A 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 BEEN FULLFILLED.

UPDATMTR:
4BB5 3A335A RAMP1: LDA CNTRAMP ; GET THE CURRENT RAMP COUNTER (FOR RATE)
4BB8 FE00 CPI OOH ; CHECK TO SEE IF IT IS COUNTED OUT
4BB9 C2E648 JNZ LBL4 ; IF NOT, THEN CONTINUE (DCR RAMP COUNT)
4BB0 3A2C5A RS1CHK: LDA RMTS ; GET THE RIGHT MOTOR TARGET SPEED
4BB3 32305A STA RMOtor ; SET DEFAULT RIGHT MOTOR VALUE
4BB4 47 MOV B,A ; TRANSFER TO B
4BB5 3A2E5A LDA RMCS ; SET RIGHT MOTOR CURRENT SPEED
4BB7 B8 CMP B ; COMPARE RMTS TO RMCS
4BB8 CAA74B JI LS1CHK ; IF RMTS = RMCS THEN CHECK LEFT DATA
4BB9 D2A448 JNC LBL1 ; IF RMCS > RMTS THEN TO LBL1
4BBE CD2D4C CALL RGFWD ; RMCS < RMTS SO INCREMENT FWD
4BA1 C3A74B JMP LS1CHK ;
4BA4 CD354C LBL1: CALL RGFREV ; RMCS > RMTS SO INCREMENT REV
4BA7 3A2B5A LS1CHK: LDA LMTS ; GET THE LEFT MOTOR TARGET SPEED
4BAA 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 CAC148 JI LBL3 ; IF LMTS = LMCS THEN CHECK DURATION
4BB5 D2BE48 JNC LBL2 ; IF LMCS > LMTS THEN TO LBL2
4BB8 CD304C CALL LFFWD ; LMCS < LMTS SO INCREMENT FWD
4BBE C3C14B JMP LBL3 ;
4BBF CD454C LBL2: CALL LFFREV ; LMCS > LMTS SO INCREMENT REV
4BC1 3A315A LBL3: LDA DURATION ; GET THE CURRENT DURATION COUNTER
4BC4 5D DCR A ; DECREMENT THE DURATION COUNTER
4BC5 32315A STA DURATION ; STORE THE CHANGE IN VARIABLE
4BC8 3A305A LDA RMOtor ; SET NEW MOTOR VALUE
4BC9 322E5A STA RMCS ; UPDATE THE CURRENT SPEED
4BCE 3A2F5A LDA LMOTOR ;
4BD1 322D5A STA LMCs ; ROTATE LEFT MOTOR DATA TO MS NIBBLE
4BD4 07 RLC
4BD5 07 RLC
4BD6 07 RLC
4BD7 07 RLC
4BD8 47 MOV B,A ; TEMPORARY TRANSFER TO B
4BD9 3A2E5A LDA RMCS ; SET NEW RIGHT MOTOR DATA
4BDC 80 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
4BE5 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
Appendix C: SOFTWARE LISTING

LDA RMCS
MOV B,A
LDA LMCS
CMP B
JNZ STOP2
MVI A,MTRSTOP
STA CNTRAMP
STA RAMPCNT
LXI D,0A00H
CALL DELAYD
CALL UPDATMTR
LDA RMCS
MOV B,A
LDA LMCS
CMP B
JNZ STOP1
MVI A,MTRSTOP
STA CNTRAMP
D,0A00H ;ALLOW THE NEXT PAD INPUT TO BE USED
STA DURATION
STA LAST1
RET

; RGTFWD - ROUTINE TO DETERMINE R Motor WHICH WILL ALTER RMCS TO MOVE FORWARD.

LDA RMCS ;OTHERWISE IF 0, GET NEW SPEED
INR A ;STEP SPEED UP BY 1
STA RMOTOR ;STORE IN RMOTOR TO BE SENT TO MOTOR
RET

; RG Trev - SAME, BUT TO REVERSE.

LDA RMCS ;OTHERWISE IF 0, GET NEW SPEED
DCR A ;STEP SPEED DOWN BY 1
STA RMOTOR ;STORE IN RMOTOR TO BE SENT TO MOTOR
RET

; LFTFWD - SAME, BUT FORWARD MOTION FOR LEFT MOTOR.

LDA LMCS ;OTHERWISE IF 0, GET NEW SPEED
INR A ;STEP SPEED UP BY 1
STA LMOTOR ;STORE IN LMOTOR TO BE SENT TO MOTOR
RET

; LFT Trev - SAME, BUT FOR REVERSE.

LDA LMCS ;OTHERWISE IF 0, GET NEW SPEED
DCR A ;STEP SPEED DOWN BY 1
STA LMOTOR ;STORE IN LMOTOR TO BE SENT TO MOTOR
RET
Appendix C: SOFTWARE LISTING

4C4D CDEB4F PROERR: CALL RASBER ; SOUN ERR 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 3A3451 LDA PADFLG ; GET PAD/MENU STATUS FLAG
4C68 FE00 CPI FALSE ; SEE IF PAD OK
4C6A C6 PUSH RZ ; 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 DA4D4C 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 E5 PUS H 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 D874C JNZ RANCHK ; CHECK RANGING IF NOT
4C7F 3E02 MVI A, ASOONOFF ; GET SOUND ON/OFF MASK
4C81 CD574F CALL ONOFF ; TOGGLE FROM ON TO OFF, VIA 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, VIA VERSA
4C91 C35A4C JMP PRO2MEN
4C94 FE41 RRCHK: CPI RAMP ; RAMP RATE SELECTED?
4C96 D254C JNZ LRD ; CHECK LEFT RANGING DISTANCE IF NOT
4C99 DBFC4F CALL HORN1 ; HIGH BEEP
4C9C 3E0E MVI A, BAR2BEG ; GET BEGINNING ROW/COLUMN OF BAR 2
4C9E CD04E 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 OFOH ; MASK TO SEE IF ANY VALUE IN MS NIBBLE
4CA4 FE00 CPI OOH ;
4CA6 CD44B4C JNZ PROERR ; IF ANYTHING IN MS NIBBLE, THEN THERE WAS ; AN INVALID TOUCH, SO SOUND ERROR AND RETURN
4CA9 7B MOV A,B ; RESTORE A TO B
4CAB CD01 ADI 01H ; ADD ONE TO THE RATE, 'CAUSE 0 RATE NO GOOD
4CAC CD04F CALL RRATE ; UPDATE TABLE VALUE OF RAMP RATE
4CAF CD0250 CALL HORN2 ; LOW BEEP
4CB2 C35A4C JMP PRO2MEN
4CB5 FE71 LRD: CPI LEFTTR ; LEFT RANGING SELECTED
4CB7 CD24AC JNZ RRD ; CHECK MOTOR DURATION IF NOT
4CB8 CDFC4F CALL HORN1 ; HIGH BEEP
4CBD 3EEE MVI A, BAR2BEG ; GET BEGINNING ROW/COLUMN OF BAR 2
Appendix C: SOFTWARE LISTING

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   OFOH  ;MASK TO SEE IF ANY VALUE IN MS NIBBLE
4CC5  FE00   CPI   OOH   ;
4CC7  C24D4C JNZ   PROERR  ;IF ANYTHING IN MS NIBBLE, THEN THERE WAS

4CCA  78     MOV   A,B   ;RESTORE A TO B
4CCB  87     ADD   A     ;DOUBLE VALUE FOR RANGING DISTANCE
4CCC  C60C   ADI   USSTOP  ;ADD O 'CAUSE O NO GOOD
4CCE  2A225A  LHL   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   PRO2MEN
4CDA  FE81   RRD:  CPI   RIGHTR  ;LEFT RANGING SELECTED
4CDA  C2FF4C JNZ   FRD    ;CHECK MOTOR DURATION IF NOT
4CDF  DDFC4F CALL  HORN1  ;HIGH BEEP
4CE2  3EE8   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   OFOH  ;MASK TO SEE IF ANY VALUE IN MS NIBBLE
4CEA  FE00   CPI   OOH   ;
4CEC  C24D4C JNZ   PROERR  ;IF ANYTHING IN MS NIBBLE, THEN THERE WAS

4CEF  78     MOV   A,B   ;RESTORE A TO B
4CF0  87     ADD   A     ;DOUBLE VALUE FOR RANGING DISTANCE
4CF1  C60C   ADI   USSTOP  ;ADD O 'CAUSE O NO GOOD
4CF3  2A225A  LHL   GBLTBL  ;PUT START OF TABLE ADDRESS IN POINTER
4CF6  DDC94F CALL  LRUDATE  ;UPDATE LEFT RANGING VALUE IN TABLE
4CF9  CD0250  CALL  HORN2  ;LOW BEEP
4CFC  C35A4C JMP   PRO2MEN
4CFF  FE61   FRD:  CPI   FRONTR  ;LEFT RANGING SELECTED
4D01  C2244D JNZ   BRD    ;CHECK MOTOR DURATION IF NOT
4D04  DDFC4F CALL  HORN1  ;HIGH BEEP
4D07  3EE8   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   OFOH  ;MASK TO SEE IF ANY VALUE IN MS NIBBLE
4D0F  FE00   CPI   OOH   ;
4D11  C24D4C JNZ   PROERR  ;IF ANYTHING IN MS NIBBLE, THEN THERE WAS

4D14  7B     MOV   A,B   ;RESTORE A TO B
4D15  87     ADD   A     ;DOUBLE VALUE FOR RANGING DISTANCE
4D16  C60C   ADI   USSTOP  ;ADD O 'CAUSE O NO GOOD
4D18  2A225A  LHL   GBLTBL  ;PUT START OF TABLE ADDRESS IN POINTER
4D1B  CDCA4E CALL  LRUDATE  ;UPDATE LEFT RANGING VALUE IN TABLE
4D1E  CD0250  CALL  HORN2  ;LOW BEEP
4D21  C35A4C JMP   PRO2MEN
4D24  FE51   BRD:  CPI   BACKR  ;LEFT RANGING SELECTED
4D26  C2494D JNZ   DEFAREA  ;CHECK MOTOR DURATION IF NOT
4D29  DDFC4F CALL  HORN1  ;HIGH BEEP
4D2C  3EE8   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   OFOH  ;MASK TO SEE IF ANY VALUE IN MS NIBBLE
4D34  FE00   CPI   OOH   ;
4D36  C24D4C JNZ   PROERR  ;IF ANYTHING IN MS NIBBLE, THEN THERE WAS
Appendix C: SOFTWARE LISTING

Page 94

4D39 78  MOV  A,B  ;RESTORE A TO B
4D3A 87  ADD  A  ;DOUBLE VALUE FOR RANGING DISTANCE
4D3B C60C  ADI  USSTDP  ;ADD O 'CAUSE O NO GOOD
4D3D 2A225A  LHLDA  GBLTBL  ;PUT START OF TABLE ADDRESS IN POINTER
4D40 CDCC4E  CALL  BRUATE  ;UPDATE LEFT RANGING VALUE IN TABLE
4D43 CD0250  CALL  HORN2  ;LOW BEEP
4D46 C3544C  JMP  PRO2MEN
4D49 FEA1  DEFAREA:  CPI  DEFINE  ;DEFINE MENU SELECTED?
4D4B C2A24D  JNIZ  SELAREA  ;CHECK SELECT AREA IF NOT
4D4E CDFF4F  CALL  HORN1
4D51 3E6F  MVI  A,BAR2BEG  ;GET BEGINNING ROW/COLUMN OF BAR 2
4D53 CD004E  CALL  BARREAD  ;WAIT FOR PAD TOUCH, RETURN VALUE OF 0-10 IN A
4D56 47  MOV  B,A  ;TEMPORARY XFER TO B
4D57 6F00  ANI  0FH  ;MASK TO SEE IF ANY VALUE IN MS NIBBLE
4D59 FE00  CPI  00H  ;
4D5A C2A44C  JNIZ  PROERR  ;IF ANYTHING IN MS NIBBLE, THEN THERE WAS
4D5E 78  MOV  A,B  ;AN INVALID TOUCH, SO SOUND ERROR AND RETURN
4D5F FE00  CPI  OOH  ;SEE IF O SELECTED AS ENTRY NUMBER (1-10 OK)
4D61 C2674D  JNIZ  DEFOK1  ;IF O NOT SELECTED, THEN CONTINUE
4D64 C3404C  JMP  PROERR  ;OTHERWISE CALL RASBERRIES, RETURN TO PROMEN
4D67 CDFF4F  DEFOK1:  CALL  HORN1
4D6A 32265A  STA  ENTRY  ;STORE 1-10 NUMBER TEMPORARILY IN ENTRY VAR
4D6D CD734F  CALL  POLL  ;GET UPPER LEFT CORNER OF BOX (MIN VALUES)
4D70 CDFF4F  CALL  HORN1
4D73 55  MOV  D,L  ;PUT MIN VALUES IN D
4D74 D5  PUSH  D  ;SAVE FIRST TOUCH LOCATION
4D75 CD734F  CALL  POLL  ;GET LOWER RIGHT CORNER OF BOX (MAX VALUES)
4D79 01  PDC  D  ;GET FIRST TOUCH LOCATION (MIN VALS)
4D79 50  MOV  E,L  ;PUT MAX DATA IN E
4D7A 3A265A  LDA  ENTRY  ;GET THE CURRENT ENTRY NUMBER FROM MEM
4D7D 2A245A  LHLDA  BGTBL  ;SET MEM POINTER AT START OF ENTRIES
4D80 501  SUI  O1H  ;REDUCE ENTRY NUMBER BY 1, TO GET LOOP CNTR
4D82 FE00  DEFOK2:  CPI  OOH  ;IS A DOWN TO 0 YET?
4D84 CAF4D  JZ  DEFOK3  ;IF SO, THEN POINTER IS AT CURRENT ENTRY
4D87 23  INX  H  ;OTHERWISE INCREMENT POINTER 4 TIMES TO
4D88 23  INX  H  ;GET TO NEXT ENTRY IN TABLE
4D89 23  INX  H
4D8A 23  INX  H
4D8B 3D  DCR  A  ;DECREMENT ENTRY COUNTER (NOW AT NEXT ENTRY)
4D8C C3824D  JMP  DEFOK2  ;REPEAT CHECK
4D8F 22295A  DEFOK3:  SHLDA  POINTER  ;SAVE STARTING ADDRESS OF POINTER
4D92 72  MOV  M,D  ;PUT MIN VALS IN TABLE
4D93 23  INX  H  ;INCREASE POINTER TO MAX VAL LOCATION
4D94 73  MOV  M,E  ;PUT MAX VALS IN TABLE
4D95 CD0250  CALL  HORN2  ;SOUND LOW (FINISH) HORN
4D98 2A225A  LHLDA  GBLTBL  ;GET START ADDRESS OF TABLE
4D9B 7E  MOV  A,M  ;SET TABLE CONTROL WORD
4D9C E6FD  ANI  11111110B  ;CLEAR BIT 0 TO SIGNAL MENU NOT EMPTY
4D9E 77  MOV  M,A  ;RESTORE THE CONTROL WORD
4D9F C3544C  JMP  PRO2MEN  ;RETURN W/OUT ZEROING OUT ENTRY
4DA2 FEB1  SELAREA:  CPI  SELECT  ;SELECT AREA SELECTED?
4DA4 C2E14D  JNZ  LMTR  ;CHECK LEFT MOTOR SPEED IF NOT
4DA7 CDFF4F  CALL  HORN1
4DA9 3E6E  MVI  A,BAR2BEG  ;SET BEGINNING ROW/COLUMN OF BAR 2

AN INVALID TOUCH, SO SOUND ERROR AND RETURN

A,B TO B

DOUBLE VALUE FOR RANGING DISTANCE

ADD O 'CAUSE O NO GOOD

PUT START OF TABLE ADDRESS IN POINTER

BARREAD; WAIT FOR PAD TOUCH, RETURN VALUE OF 0-10 IN A

TEMPORARY XFER TO B

MASK TO SEE IF ANY VALUE IN MS NIBBLE

AN INVALID TOUCH, SO SOUND ERROR AND RETURN

SEE IF O SELECTED AS ENTRY NUMBER (1-10 OK)

IF O NOT SELECTED, THEN CONTINUE

OTHERWISE CALL RASBERRIES, RETURN TO PROMEN

STORE 1-10 NUMBER TEMPORARILY IN ENTRY VAR

GET UPPER LEFT CORNER OF BOX (MIN VALUES)

PUT MIN VALUES IN D

SAVE FIRST TOUCH LOCATION

GET LOWER RIGHT CORNER OF BOX (MAX VALUES)

GET FIRST TOUCH LOCATION (MIN VALS)

PUT MAX DATA IN E

GET THE CURRENT ENTRY NUMBER FROM MEM

SET MEM POINTER AT START OF ENTRIES

REDUCE ENTRY NUMBER BY 1, TO GET LOOP CNTR

IS A DOWN TO 0 YET?

IF SO, THEN POINTER IS AT CURRENT ENTRY

OTHERWISE INCREMENT POINTER 4 TIMES TO

GET TO NEXT ENTRY IN TABLE

DECREMENT ENTRY COUNTER (NOW AT NEXT ENTRY)

REPEAT CHECK

SAVE STARTING ADDRESS OF POINTER

PUT MIN VALS IN TABLE

INCREASE POINTER TO MAX VAL LOCATION

PUT MAX VALS IN TABLE

SOUND LOW (FINISH) HORN

GET START ADDRESS OF TABLE

SET TABLE CONTROL WORD

CLEAR BIT 0 TO SIGNAL MENU NOT EMPTY

RESTORE THE CONTROL WORD

RETURN W/OUT ZEROING OUT ENTRY

SELECT AREA SELECTED?

CHECK LEFT MOTOR SPEED IF NOT

SET BEGINNING ROW/COLUMN OF BAR 2
Appendix C: SOFTWARE LISTING

40AC CDD04E  CALL  BARREAD  ; WAIT FOR PAD TOUCH, RETURN VALUE OF 0-10 IN A
40AF 47  MOV  B,A  ; TEMPORARY XFER TO B
40B0 E6F0  ANI  OF0H  ; MASK TO SEE IF ANY VALUE IN MS NIBBLE
40B2 FE00  CPI  00H  ;
40B4 C24D4C  JNZ  PROERR  ; IF ANYTHING IN MS NIBBLE, THEN THERE WAS
40B7 78  MOV  A,B  ; AN INVALID TOUCH, SO SOUND ERROR AND RETURN
40BB FE00  CPI  00H  ; SEE IF 0 SELECTED AS ENTRY NUMBER (1-10 OK)
40BA C2C04D  JNZ  SELOK1  ; IF 0 NOT SELECTED, THEN CONTINUE
40BD C34D4C  JMP  PROERR  ; OTHERWISE CALL RASBERRIES, RETURN TO PROMEN
40C0 CDFC4F  SELK1:  CALL  HORN1
40C3 32265A  STA  ENTRY  ; STORE IN ENTRY VAR FOR TIME BEING
40C6 2A245A  LHLD  B6TBL  ; SET MEM POINTER AT START OF ENTRIES
40C9 D601  SUI  01H  ; REDUCE ENTRY NUMBER BY 1, TO GET LOOP CNTR
40CB FE00  SELOK2:  CPI  00H  ; IS A DOWN TO 0 YET?
40CD CADD4D  JZ  SELOK3  ; IF SO, THEN POINTER IS AT CURRENT ENTRY
40D0 23  INX  H  ; OTHERWISE INCREMENT POINTER 4 TIMES TO
40D2 23  INX  H  ; GET TO NEXT ENTRY IN TABLE
40D3 23  INX  H
40D4 3D  DCR  A  ; DECREMENT ENTRY COUNTER (NOW AT NEXT ENTRY)
40D5 C3CB4C  JMP  SELOK2  ; REPEAT CHECK
40D8 22295A  SELOK3:  SHLD  POINTER  ; SAVE STARTING ADDRESS OF POINTER
40DB CDD0250  CALL  HORN2  ; LOW BEEP (FINISH)
40DE C35A4C  JMP  PRO2MEN  ; RETURN W/OUT ZEROING OUT ENTRY
40E1 FECD  LMTR:  CPI  LEFTM  ; WAS LEFT MOTOR SETTING SELECTED?
40E3 C2F64D  JNZ  RMTR  ; IF NOT, CHECK THE RIGHT MOTOR LOCATION
40E6 3EFF  MVI  A,TRUE  ; SET FLAG FOR LEFT MOTOR DATA
40E8 322F5A  STA  LMOTOR  ;
40EB 3A265A  LDA  ENTRY  ; GET THE TABLE ENTRY VALUE
40EE FE00  CPI  00H  ; SEE IF CURRENT ENTRY IS 0 (SHOULD NOT BE)
40F0 C20B4E  JNZ  MOK1  ; IF NOT 0, THEN CONTINUE
40F3 C34D4C  JMP  PROERR  ; OTHERWISE SOUND ALARM AND RETURN
40F6 FE01  RMTR:  CPI  RMTR1  ; WAS LEFT MOTOR SETTING SELECTED?
40FB C24A4E  JNZ  DUR  ; IF NOT, CHECK THE RIGHT MOTOR LOCATION
40FB 3E00  MVI  A,FALSE  ; CLEAR FLAG FOR LEFT MOTOR DATA
40FD 322F5A  STA  LMOTOR  ;
40E0 3A265A  LDA  ENTRY  ; GET THE TABLE ENTRY VALUE
40E3 FE00  CPI  00H  ; SEE IF CURRENT ENTRY IS 0 (SHOULD NOT BE)
40E5 C20B4E  JNZ  MOK1  ; IF NOT 0, THEN CONTINUE
40E8 C34D4C  JMP  PROERR  ; OTHERWISE SOUND ALARM AND RETURN
40EB CDFC4F  MOK1:  CALL  HORN1
40E8 3EEB  MVI  A,BARIBES  ; GET BEGINNING ROW/COLUMN OF BAR
40E9 CDD04E  CALL  BARREAD  ; WAIT FOR PAD TOUCH, RETURN VALUE OF 0-10 IN A
40E7 47  MOV  B,A  ; TEMPORARY XFER TO B
40EE E6F0  ANI  OF0H  ; MASK TO SEE IF ANY VALUE IN MS NIBBLE
40F0 FE00  CPI  00H  ;
40F1 C24D4C  JNZ  PROERR  ; IF ANYTHING IN MS NIBBLE, THEN THERE WAS
40F4 2A295A  LHLD  POINTER  ; SET MEM POINTER TO CORRECT ENTRY
40F8 23  INX  H  ; GET TO LEFT MOTOR DATA
40F1 23  INX  H
40E2 3A265A  LDA  LMOTOR  ; GET LEFT MOTOR SETTING FLAG
40E4 FE00  CPI  FALSE  ; IF FALSE, THEN ENTERING RIGHT MOTOR DATA
40E6 C24D4C  JNZ  PROERR  ; IF ANSWER IS A, SET TARGET TO ZERO TO SET RIGHT MOTOR SPEED
40E8 7B  MOV  A,B  ; GET BAR INPUT BACK
Appendix C: SOFTWARE LISTING

Page 96

4E29 C603 ADI 03H ;ADD 3 TO GET CORRECT MOTOR DATA
4E2B 07 RLC ;SWITCH NIBBLES
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 80 ORA B ;COMBINE NEW LEFT DATA W/OLD RIGHT DATA
4E34 77 MOV M,A ;RESTORE IN TABLE
4E35 CD0250 CALL HORNZ ;LOW BEEP (FINISH)
4E38 C35A4C JMP PROZMEN
4E39 79 RSET: MOV A,B ;SET 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 E60F ANI 0FH ;CLEAR OUT OLD RIGHT MOTOR DATA
4E42 80 ORA B ;COMBINE OLD LEFT DATA W/NEW RIGHT
4E43 77 MOV M,A ;RESTORE IN TABLE
4E44 CD0250 CALL HORNZ ;LOW BEEP (FINISH)
4E47 C35A4C JMP PROZMEN
4E4A FEE1 DUR: CPI TIME ;DURATION AREA SELECTED?
4E4C C27F4E JNZ RESMEN ;CHECK RESET MENU SELECTION
4E4F 3A2A5A 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 CD004E CALL BARREAD ;WAIT FOR PAD TOUCH, RETURN VALUE OF 0-10 IN A
4E62 47 MOV B,A ;TEMPORARY XFER TO B
4E63 E60F ANI 0FH ;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
4E6A 7B MOV A,B ;AN INVALID TOUCH, SO SOUND ERROR AND RETURN
4E6B 2500 MVI H,00H ;O OUT H FOR MULT
4E6D 6F MOV L,A ;PUT VALUE IN L FOR MULT
4E6F CD4F49 CALL MULTIO ;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
4E79 77 MOV M,A ;PUT NEW DURATION IN TABLE
4E79 CD0250 CALL HORNZ ;LOW BEEP (FINISH)
4E7C 035A4C JMP PROZMEN
4E7F FE09 RESMEN: CPI RESET ;WAS THE RESET MENU FUNCTION SELECTED?
4E81 C24D4C JNZ PROERR ;RETURN AND SOUND ERROR IF NO MENU CHOICE
4E84 CDFF4F CALL HORN1 ;BEEP TO ACKNOWLEDGE THE FIRST TOUCH
4E87 CD734F CALL POLL ;WAIT FOR SECOND RESET TO VERIFY
4E89 7D MOV A,L
4E8B FE09 CPI RESET ;WAS THE RESET VERIFIED?
4E8D C24D4C JNZ PROERR ;RETURN WITH ERROR SOUND IF NOT.
Appendix C: SOFTWARE LISTING

4E93 2A225A    LHLD GBLTBL ;OTHERWISE, RESET MENU
4E96 3E07      MVI A,0000011B ;SET CONTROL WORD FOR RESET MENU,
                ;SOUND ON, RANGING ON.
4E99 77         MOV M,A ;STORE MENU CONTROL WORD IN TABLE
4E9A 23         INX H
4E9A 23         MOV M,A
4E9D 23         INX H
4E9E 23         MOV M,A
4EA0 23         INX H
4EA0 23         MOV M,A
4EA0 23         MOV M,A
4EA1 23         MOV M,A
4EA2 77         INX H
4EA4 77         MOV M,A
4EA4 77         MOV M,A
4EA5 77         MOV M,A
4EA6 77         MOV M,A
4EA7 23         MOV M,A

4EAB 060A       MVI A,0AH ;DEFAULT A, B, C AND D RANGING DISTANCES
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 F60B       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 0A         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 C0D250     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
4ECF 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 CD734F 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 C2OE4F 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
```
Appendix C: SOFTWARE LISTING

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 3EO9  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 3EOA  MVI A,0AH  ; RETURN VALUE TOUCHED ON THE BAR
4F3F C9  RET
4F40 CDEB4F  BADENT1:  CALL RASBER  ; SOUND RASBERRIES (BAD ENTRY NOISE)
4F43 09  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 09  RET

; RRATE - ROUTINE TO MODIFY THE RAMP RATE VALUE IN THE CURRENT. ;

4F4E 2A225A  RRATE:  LHLD  GBLTBL  ; GET STARTING LOCATION OF THE TABLE
4F51 23  INI  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

; ONOFF - ROUTINE TO TOGGLE THE STATUS OF THE BIT PASSED IN A ; IN THE MENU CONTROL WORD, TURNING SOUND/RANGING, ON/OFF.

4F57 47  ONOFF:  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 " " " " " "
4F5D A0  ANA  B  ; AND SELECTED BIT WITH CONTROL WORD
4F5E BB  CMP  B  ; COMPARE WITH SELECTED BIT
4F5F DA694F  JZ TURNOFF ; IF SELECTED BIT IS A 1, THEN MAKE A 0
4F62 79  TURNON:  MOV  A,C  ; PUT CURRENT MENU CONTROL WORD BACK IN A
4F63 80  ORA  B  ; SET SELECTED BIT HIGH.
4F64 77  MOV  M,A  ; RESTORE CONTROL WORD TO TABLE
4F65 DDFC4F  CALL HORN1 ; HIGH BEEP TO SIGNAL ON
4F68 C9  RET
4F69 78  TURNOFF:  MOV  A,B  ; GET SELECTED BIT MASK
4F6A 2F  CMA  ; COMPLEMENT SELECTED BIT MASK
Appendix C: SOFTWARE LISTING

Page 100

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 EXTCMK ;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 JMI POLL
4F87 D5 NTCH: PUSH D
4F89 110002 LXI D,0200H
4F8B 1B DELAYS: 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 02BE4F JNZ DELAYS ;REPEAT IF DE<0
4F91 D1 PDP D
4F92 DB11 IN PIAH ;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 CHECK 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 ;
4F9F DB11 IN PIAH ;READ THE +5 LOOP FROM THE PAD CONNECTOR
4FAC EE80 ANI PADLOOP ;MASK TO DETERMINE THE CONNECTOR STATUS
4FAC F880 CPI PADLOOP ;VALUE READ, 1=CONNECTED, 0=DISCONNECTED
4FAD 00A4JF JZ PADC05 ;CONTINUE IF PAD CONNECTED
4FAB 003F MVI A,PADLED ;SET DATA TO LIGHT PAD ERROR LED
4FAC CDF04A CALL SETERR ;LIGHT THE PAD ERROR LED
4FAD 3E40 MVI A,MENERR ;PAD ERROR WILL GIVE MENU ERROR
4FBE CD04A CALL CLRERR ;CLEAR MENU LED
4FBD 3E00 MVI A,FALSE ;SIGNAL PAD/MENU OK FLAG AS NOT OK
4FBE 32345A STA PADFLG ;
4FBB C9 RET
4FBA 3E01 PADC05: MVI A,PADLED ;GET DATA TO CLEAR PAD LED
4FBC CD04A CALL CLRERR ;CLEAR THE PAD LED (ALL IS OK)
Appendix C: SOFTWARE LISTING

4FBF CDF046 CALL PAORD ; DETERMINE MENU NUMBER OR STATUS (IF ERROR)
4FC2 7C MOV A, H ; PUT MENU NUMBER (STATUS) IN (A)
4FC3 E6A0 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 CLERRR ; 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 CB RJ ; 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 'RASBERRIES' 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 CD04F RAS1: CALL HORN1 ; SOUND LOW BEEP ONCE
4FF1 CD0250 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 ;

; HORN1 - ROUTINE TO SOUND THE FIRST OF TWO TONES

4FFC F5 HORN1: PUSH PSW
4FFD 3E40 MVI A, 10H ; LOAD THE VALUE FOR FIRST TONE
4FFF CD0550 JMP HORNAN
5002 F5 HORN2: PUSH PSW
5003 3E00 MVI A, 0COH ; LOAD VALUE FOR SECOND TONE
5005 D342 HORNAN: OUT PIAF ; SEND VALUE TO SOUND CIRCUIT
5007 D5 PUSH D
5009 110030 LXI D, 3000H ; SHORT DELAY
500B CDD747 CALL DELAYND
500E 3E00 MVI A, 00H
5010 D342 OUT PIAF
5012 D1 POP D
5013 F1 POP PSW
5014 C9 ;

;
Appendix C: SOFTWARE LISTING

; ROM CONSTANT ALLOCATION - ALPHABETICAL ORDER (SORT OF)
; COMMAND TABLE

5015 444D DB 'DM' ; DUMP MEMORY
5017 0942 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 'ID' ; ID PORT R/W/M
502B 4943 DW IPORT
502D 5442 DW 'TB' ; TEST TIMERS AND PORTS ON BOARD
502F 7B41 DW TSTBRD
5031 544D DB 'TM' ; TEST MEMORY
5033 0541 DW MENTST
5035 5243 DB 'RC' ; RUN WHEELCHAIR
5037 B745 DW INITIAL
5039 0C00 DB 0,0 ; END OF TABLE MARK

; MESSAGES...

503B 2041424F52 ABORT: DB 'ABORTED' EOL
5045 FF DB EOL
5046 2057484154 AD: DB 'WHAT?' EOL
504D FF DB EOL
504E 29 EDML: DB ')' EOL
504F 203D20 EDML: DB '=' EOL
5052 FF DB EOL
5053 0D0A EDML: DB CR,LF
5055 28 DB EOL
5056 FF DB EOL
5057 0D0A TSTBRD DB CR,LF
5059 5445534449 DB 'TESTING TIMERS AND PIA PORTS',CR,LF
505F 4C4F4F4B20 DB 'LOOK FOR 1000 Hz SQUAREWAVE ON TIMER OUTPUTS',CR,LF
506A 4441544120 DB 'DATA ANALYZER SHOULD SHOW PORTS COUNTING',CR,LF
506C 494E204120 DB 'IN A STAIRSTEP FASHION',CR,LF
506E FF DB EOL
506F 444154413D0PDA: DB 'DATA=' EOL
506F FF DB EOL
5070 FF DB '@ 50mS +' EOL
5071 2C2D20 IOPSM: DB ' , EOL
5072 FF DB EOL
5073 6C41 LCKM: DB CR,LF
5074 4D4E544552 DB 'ENTER CPU CLK FREQ XXXX KHz:' EOL
5111 CFF DB 'OK?' EOL
5112 FF DB EOL
5113 0D0A MTGDOD: DB CR,LF
5114 4D454D4F52 DB 'MEMORY TEST PASSED'
5115 0D0AFF DB CR,LF
5116 0D0A MERR: DB CR,LF
5117 C4454D4F52 DB 'MEMORY TEST FAILED AT'
WELCOME TO THE EASYCHAIR MONITOR

THE FOLLOWING TWO CHARACTER COMMANDS ARE AVAILABLE:

'DM' Dump Memory
'DL' Down Load from dev. system
'EM' Edit Memory
'GO' Goto
'TB' Test Board utility
'TM' Test Memory
'RC' Run Chair program

THIS SYSTEM WAS CREATED BY :

JAMES WILLIAMS
AND
GREGORY WELCH
IT IS THE CONTROLLER PROGRAM THAT OPERATES THE ULTRASONICS, LIGHT BOARD, AND MOTORS OF THE EASYCHAIR WHEELCHAIR.

THIS PROGRAM ALSO ALLOWS MENUS FOR THE LIGHT BOARD TO BE CREATED FOR EACH CHILD AND ADDED TO AND CHANGED AS NEEDED.

ALL ATTEMPTS WERE MADE TO FORESEE ALL THE POSSIBLE PROBLEMS THAT MAY ARISE.

HOWEVER, NO PROMISES.

WHEELCHAIR NOW UNDER COMPUTER CONTROL.

BEGIN INFRA-RED TOUCH PAD DIAGNOSTICS

END OF INFRA-RED TOUCH PAD DIAGNOSTICS.

SAMPLE:

<table>
<thead>
<tr>
<th>DB</th>
<th>02H</th>
<th>RAMP RATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB</td>
<td>10H</td>
<td>BAK,FNT,LFT,RGT RANFGING DIST</td>
</tr>
<tr>
<td>DB</td>
<td>10H</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB</th>
<th>00H</th>
<th>ROW/COL MIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB</td>
<td>44H</td>
<td>ROW/COL MAX</td>
</tr>
<tr>
<td>DB</td>
<td>0ACH</td>
<td>MOTOR SPEEDS (L/R)</td>
</tr>
<tr>
<td>DB</td>
<td>10H</td>
<td>DURATION</td>
</tr>
<tr>
<td>DB</td>
<td></td>
<td>NEXT ENTRIES</td>
</tr>
</tbody>
</table>

BEGIN ENTRY 1

<table>
<thead>
<tr>
<th>DB</th>
<th>05H</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB</td>
<td>4AH</td>
</tr>
<tr>
<td>DB</td>
<td>0CAH</td>
</tr>
<tr>
<td>DB</td>
<td>10H</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB</th>
<th>0BH</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB</td>
<td>4FH</td>
</tr>
<tr>
<td>DB</td>
<td>0CAH</td>
</tr>
<tr>
<td>DB</td>
<td>10H</td>
</tr>
</tbody>
</table>

| DB | 50H |
Appendix C: SOFTWARE LISTING

561C A4 DB 0A4H
561D 8C DB 8CH
561E 10 DB 10H

561F 55 DB 55H
5620 AA DB 0AAH
5621 BB DB 0BH
5622 10 DB 10H

5623 5B DB 5BH
5624 AF DB 0AFH
5625 CB DB 0CBH
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 BB DB 0BBH
5630 FF DB 0FFH
5631 64 DB 64H
5632 10 DB 10H

5633 00 DB 00H
5634 00 DB 00H
5635 BB DB 0BH
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: DB 1
5A01 WIDTH: DB 1
5A02 BIAS: DS 2
5A04 RJSAV: DS 2
5A06 RJSP: DS 2
5A08 RJVECT: DS 2
5A0A D50DIV: DS 2
5A0C CLKBCD: DS 2
5A0E CLKBIN: DS 2
5A10 FNTDST: DS 2
5A12 MAXFNT: DS 1
5A13 BAKDST: DS 2
5A15 MAXBAK: DS 1
5A16 RTDST: DS 2
5A18 MAXRT: DS 1
5A19 LFTDST: DS 2

; ECHO FLAG: 0=ECHO 1=NO ECHO
; WIDTH+1 = NUMBER OF BYTES PER LINE
; BIAS FOR LOAD
; TEMP SAVE AREA FOR RETJMP
; RETURN JUMP STACK POINTER
; RETURN JUMP VECTOR (PC)
; COUNTER FOR TIMING OF 50MS PULSE
; CLOCK FREQUENCY IN BCD
; CLOCK FREQUENCY IN BINARY
; ULTRASONIC FNT DIST.
; MAX FRONT DIST.
; BACK DIST.
; MAX BACK DIST.
; RIGHT DIST.
; MAX RIGHT DIST.
; LEFT DIST.
Appendix C: SOFTWARE LISTING

MAXLFT: DS 1 ; MAX LEFT DIST.
TIMDLY: DS 2 ; DELAY TIME
HONOFF: DS 1 ; HIGH SPEED FLAG
RAMPCT: DS 1 ; RAMP RATE
MENCTRL: DS 1 ; MENU CONTROL WORD (FLAGS...)
ERRWD: DS 1 ; CURRENT ERROR WORD (SETERR, CLRERR)
GBLTBL: DS 2 ; STARTING ADDRESS OF GLOBAL MEN VARS
BEGTBL: DS 2 ; STARTING ADDRESS OF TABLE ENTRIES
ENTRY: DS 1 ; CURRENT ENTRY NUMBER (IN DATA TABLES)
MTRADDR: DS 2 ; ADDRESS OF CURRENT ENTRY DATA
POINTER: DS 2 ; POINTER USED IN PROMEN TO UPDATE TABLE
LMTS: DS 1 ; LEFT MOTOR TARGET SPEED
RMTS: DS 1 ; RIGHT MOTOR TARGET SPEED
LMCS: DS 1 ; LEFT MOTOR CURRENT SPEED
RMCS: DS 1 ; RIGHT MOTOR CURRENT SPEED
LMOTOR: DS 1 ; VALUES TO BE SENT TO L & R MOTORS
RMOTOR: DS 1 ;
DURATION: DS 1 ; DURATION OF MOTOR ACTION
LAST: DS 1 ; LAST ENTRY NUMBER
CNTRAMP: DS 1 ; IMMEDIATE RAMP COUNT
PADFLG: DS 1 ; PAD/MENU OK FLAG
HEARTON: DS 1 ; HEARTBEAT ON/OFF FLAG
MSCBF: DS 17 ; BUFFER FOR USE BY COMMANDS
; PUT LAST SO AN OVERRUN WON'T BOMB
; SYSTEM
; END OF EASYCHAIR MONITOR

END
## Appendix D: COSTING

### THE INFRARED TOUCH PAD

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Quantity</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>Infrared LEDs</td>
<td></td>
<td>$26.00</td>
</tr>
<tr>
<td>40</td>
<td>Infrared phototransistors</td>
<td></td>
<td>22.00</td>
</tr>
<tr>
<td>1</td>
<td>Miscellaneous wood/plastic</td>
<td></td>
<td>60.00</td>
</tr>
<tr>
<td>1</td>
<td>Electronic components</td>
<td></td>
<td>85.00</td>
</tr>
<tr>
<td>1</td>
<td>Electronic cable</td>
<td></td>
<td>27.00</td>
</tr>
<tr>
<td>1</td>
<td>Miscellaneous hardware</td>
<td></td>
<td>75.00</td>
</tr>
</tbody>
</table>

**Total:** $295.00

### ULTRASONIC RANGING

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Quantity</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Ultrasonic transducers</td>
<td></td>
<td>375.00</td>
</tr>
<tr>
<td>1</td>
<td>Electronic components</td>
<td></td>
<td>50.00</td>
</tr>
<tr>
<td>1</td>
<td>Electronic cable</td>
<td></td>
<td>32.00</td>
</tr>
</tbody>
</table>

**Total:** $457.00

### COMPUTER AND MOTOR CONTROL

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Quantity</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Working 8085 based computer</td>
<td></td>
<td>400.00</td>
</tr>
<tr>
<td>1</td>
<td>Additional 8255 PIA</td>
<td></td>
<td>17.00</td>
</tr>
<tr>
<td>1</td>
<td>2816A EEPROM</td>
<td></td>
<td>16.00</td>
</tr>
<tr>
<td>2</td>
<td>DS1225 8K NOVRAM</td>
<td></td>
<td>32.00</td>
</tr>
<tr>
<td>2</td>
<td>AD558 D/A Converters</td>
<td></td>
<td>15.00</td>
</tr>
<tr>
<td>1</td>
<td>Electronic components</td>
<td></td>
<td>15.00</td>
</tr>
<tr>
<td>1</td>
<td>Power supply components</td>
<td></td>
<td>27.00</td>
</tr>
</tbody>
</table>

**Total:** $522.00

### MISCELLANEOUS COSTS

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Quantity</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Miscellaneous</td>
<td></td>
<td>55.61</td>
</tr>
</tbody>
</table>

**Total:** $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 Methodology For Rehabilitation Devices Using Embedded Microcomputers", Rehabilitation Research and Development Center, Palo Alto Veterans Administration Medical Center, 1983


7) Welch, Gregory F., Williams, James P., "The Pressure Sensitive Touch-Pad", Purdue University, 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