THE DESIGN OF AN ULTRASOUND PHASED ARRAY CONTROLLER FOR USE IN HYPERTERMIA

BY

STEVEN GARY SILVERMAN

B. S., University of Illinois, 1983

THESIS

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering in the Graduate College of the University of Illinois at Urbana-Champaign, 1984

Urbana, Illinois
ACKNOWLEDGMENT

This thesis was a result of the ongoing research at the Bioacoustics Laboratory of the University of Illinois. The author wishes to thank Professor Charles A. Cain, Professor Richard L. Magin, and Professor Leon Frizzel for their guidance of the project. Grateful acknowledgement is given to those who helped with the preparation of this thesis. Thanks also to the other graduate students and department staff members for their help on the project. The author would also like to thank his parents for encouragement and assistance.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2. CONTROLLED HEATING WITH A PHASED ARRAY OF TIPOF</td>
<td>4</td>
</tr>
<tr>
<td>TAPERED TRANSDUCERS</td>
<td></td>
</tr>
<tr>
<td>3. SERIAL COMMUNICATION PROTOCOL</td>
<td>7</td>
</tr>
<tr>
<td>4. CIRCUIT HARDWARE</td>
<td>13</td>
</tr>
<tr>
<td>4.1 BOARD LEVEL OVERVIEW</td>
<td>13</td>
</tr>
<tr>
<td>4.2 MICROCONTROLLER BOARD</td>
<td>17</td>
</tr>
<tr>
<td>4.3 CLOCK BOARD</td>
<td>19</td>
</tr>
<tr>
<td>4.4 ELEMENT DRIVER BOARD</td>
<td>21</td>
</tr>
<tr>
<td>4.5 LED DISPLAY DRIVER BOARD</td>
<td>23</td>
</tr>
<tr>
<td>5. SOFTWARE</td>
<td>24</td>
</tr>
<tr>
<td>5.1 SOFTWARE OVERVIEW</td>
<td>24</td>
</tr>
<tr>
<td>5.2 CONFIG - HARDWARE CONFIGURATION ROUTINE</td>
<td>24</td>
</tr>
<tr>
<td>5.3 SINT - SERIAL INTERRUPT ROUTINE</td>
<td>26</td>
</tr>
<tr>
<td>5.4 TINT - TIMER INTERRUPT ROUTINE</td>
<td>28</td>
</tr>
<tr>
<td>6. AREAS FOR FURTHER DEVELOPMENT</td>
<td>30</td>
</tr>
<tr>
<td>6.1 TRADE-OFFS</td>
<td>30</td>
</tr>
<tr>
<td>6.2 ENHANCEMENTS</td>
<td>31</td>
</tr>
<tr>
<td>7. SUMMARY</td>
<td>34</td>
</tr>
<tr>
<td>APPENDIX A : 8031 PROGRAM CODE</td>
<td>50</td>
</tr>
<tr>
<td>APPENDIX B : GENERATING EPROM LOOK-UP TABLES</td>
<td>64</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>68</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. COMMANDS SENT FROM CENTRAL COMPUTER TO ARRAY CONTROLLER</td>
<td>9</td>
</tr>
<tr>
<td>2. REPLIES SENT FROM ARRAY CONTROLLER TO CENTRAL COMPUTER</td>
<td>9</td>
</tr>
<tr>
<td>3. PSUEDO-HEX DATA CODING</td>
<td>10</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. SYSTEM OVERVIEW</td>
<td>34</td>
</tr>
<tr>
<td>2. PHASED ARRAY OF TAPERED ULTRASOUND TRANSDUCERS</td>
<td>35</td>
</tr>
<tr>
<td>3. LED DISPLAY</td>
<td>36</td>
</tr>
<tr>
<td>4. SHARED BUS ARCHITECTURE</td>
<td>37</td>
</tr>
<tr>
<td>5. MICROCONTROLLER BOARD SCHEMATIC</td>
<td>38</td>
</tr>
<tr>
<td>6. MICROCONTROLLER BOARD COMPONENT LAYOUT</td>
<td>39</td>
</tr>
<tr>
<td>7. CLOCK BOARD SCHEMATIC</td>
<td>40</td>
</tr>
<tr>
<td>8. CLOCK BOARD COMPONENT LAYOUT</td>
<td>41</td>
</tr>
<tr>
<td>9. SYNCHRONIZED GATING OF A CLOCK</td>
<td>42</td>
</tr>
<tr>
<td>10. ELEMENT DRIVER BOARD SCHEMATIC</td>
<td>43</td>
</tr>
<tr>
<td>11. ELEMENT DRIVER BOARD COMPONENT LAYOUT</td>
<td>44</td>
</tr>
<tr>
<td>12. LED DISPLAY DRIVER BOARD SCHEMATIC</td>
<td>45</td>
</tr>
<tr>
<td>13. LED DISPLAY DRIVER BOARD COMPONENT LAYOUT</td>
<td>46</td>
</tr>
<tr>
<td>14. CONFIG FLOWCHART</td>
<td>47</td>
</tr>
<tr>
<td>15. SINT FLOWCHART</td>
<td>48</td>
</tr>
<tr>
<td>16. TINT FLOWCHART</td>
<td>49</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION

Hyperthermia is currently undergoing evaluation as a treatment for tumors, often in conjunction with other forms of treatment such as chemotherapy or ionizing radiation. The goal of a hyperthermia system is to attain a specified temperature distribution within the tumor volume which is sufficiently higher than the normal body temperature to have therapeutic effect. Different parts of the tumor and surrounding healthy tissue may have nonhomogeneous thermal conductivity. Bloodflow in the treatment area may be another variable which affects the rate of heating and cooling. To handle these varying parameters, a closed loop feedback control heating system may be necessary to achieve the desired temperature distribution essential for maximum therapeutic value.

An ultrasound phased transducer array can be used to create a relatively small heating spot which can be steered around within the tumor [Benkeser, 1983; Ocheltree, 1984]. Specific parameters of the transducer array including number, size, and spacing of the elements affect the actual size and shape of the heating focus, but are not part of the control scheme.

Figure 1 illustrates a proposed hyperthermia system with five major components (all figures appear following the closing summary). The central computer handles all operator interaction with the system and performs the computations for any complex thermal modelling necessary for control of the heating. It is
linked with two 8031 microcontroller based subsystems by RS-232C Standard serial connections.

The central computer designates a series of small volumes within a predefined three-dimensional coordinate axis system which are to be heated, along with information about the amount of heating at each point. The array controller uses the information to generate signals for excitation of the individual elements of the array. These outputs are square waves of a specified frequency, with specific phasing relative to one another. They are amplified and transmitted to the individual elements of the transducer array in order to create a point of focus.

This thesis describes the design of an array controller which has been constructed for use in a prototype phased array applicator system. The term system will hereafter refer to the hardware and software of the array controller unless specified otherwise. A small portion of the hardware remains incomplete (see Section 4.3), but most of the project has been constructed and the correct operation verified with a logic analyzer. The organization of material in this thesis was intended for the benefit of those who use the array controller for further development of a hyperthermia system, or for implementations of future improvements in the array controller.

The thesis is divided into seven major chapters. Chapter 2 describes the control scheme for heating with a phased array of ultrasound transducer elements. Chapter 3 discusses the communication protocol used over the serial link between the central computer and the array controller. Chapter 4 illustrates and describes the operation of the circuit hardware within the
array controller. Chapter 5 describes the assembly level software of the Intel 8031 microcontroller, a single chip processor located within the array controller. Chapter 6 presents some possible areas for future development and Chapter 7 contains a closing summary. Appendix A contains a full listing of the 8031 software, complete with hexadecimal object code and comment field. Appendix B describes the EPROM look-up tables and includes a program for generating the tables with an Apple IIe computer.
CHAPTER 2

CONTROLLED HEATING WITH A PHASED ARRAY OF TAPERED TRANSDUCERS

Controlled heating within a volume of tumor tissue is the basic goal of the hyperthermia system. The solution is often complicated by non-uniform thermal conductivity in the tumor and surrounding normal tissue. Bloodflow in the treatment area may change during heating, requiring dynamic control to maintain stable temperatures during treatment.

The ultrasound transducer array used in the project has a relatively small focus which can be steered electronically within a larger field of treatment. The amplifiers for the excitation signals of each element operate at a single power level so the array can operate at only two intensity levels, full-on or full-off. The power delivered to points in the treatment area can be controlled by turning the array on and off (creating duty cycle modulation of power).

The treatment field is viewed in a three-dimensional coordinate-axis grid. The origin of the imaginary grid is at the center of the array, as illustrated in Figure 2. The focus in one dimension is created by exciting all of the tapered elements at the same frequency. The other two dimensions of focus can be controlled by proper phasing using presettable counters [Benkeser, 1983]. Sixteen coordinates are defined along each axis and the array can be focused at any of the 4096 nodes of the resulting imaginary grid. The size and shape of the focus are determined by the physical parameters of the array [Ocheltree, 1984].
In order to heat a volume larger than the focus point, a string of \((X,Y,Z)\) coordinate points is heated one after the other in a continuous scan path. The heating at each point is controlled by the percentage of time the array is focused at that point. The net heating within the entire treatment area is controlled by leaving the array unexcited for an adjustable period of time between the heating of each consecutive point. However, if individual points are not heated frequently enough the temperature at some points may begin to exhibit thermal ripple. To avoid this undesirable effect, the repetitive scanning of all points must be done rapidly.

The heating level at each point is controlled by two factors. The first is a relative intensity factor for each point, called \(R\), which specifies an integer intensity level from 0 to 8 for each point in the scan path. The second is an overall path intensity factor, called \(\Theta\), which scales the amount of heating at all points proportionately. The \(\Theta\) term can have any integer value from 0 to 8.

A "data point" is defined as the combination of \(X\), \(Y\), and \(Z\) coordinates of a heated point, along with the relative intensity factor \(R\) for that point (the \(X\), \(Y\), and \(Z\) coordinates have integer values from 0 to 15). A "scan path" is a string of data points used to specify a pattern for heating with the transducer array (it may contain up to 255 data points). A time interval, referred to as "timeslice," is also a part of the control scheme.
The control scheme operates in the following way:

STEP 1 - Examine X, Y, Z, and R of the first data point in the path.

STEP 2 - Examine the current value of θ.

STEP 3 - Excite the array, focusing at the axes coordinates for $\theta \times R$ timeslice periods ($\theta \times R$ will have an integer value between 0 and 64).

STEP 4 - No transducer excitation for the remaining $64 - (\theta \times R)$ timeslice periods.

STEP 5 - Examine the next data point in the scan path $(X,Y,Z,R)$.

STEP 6 - Return to STEP 2.

For a given scan path, some factors can be adjusted while scanning. If temperature throughout the treatment field becomes too high or low, the overall intensity factor can be adjusted. If the scan rate is too fast or slow, the timeslice period can be adjusted. However, a whole new scan path must be initiated if the distribution of heating is not correct.
CHAPTER 3

SERIAL COMMUNICATIONS PROTOCOL

The central computer determines a pattern of heating to be applied by the transducer array. This information must be transmitted to the array controller. An RS-232C Standard interface allows the two devices to communicate with each other. RS-232C is the most common method of interfacing computers and peripherals today [Nichols, Nichols, Musson, 1982].

A minimum implementation of the standard is used in the hyperthermia system, consisting of only three lines between the two devices (RS-232C defines the function of hardware handshaking lines, but they are optional). The three lines are: pin 2 - TxD, pin 3 - RxD, and pin 7 - Signal Ground. The array controller is configured as Data Terminal Equipment (DTE), so the port of the central computer should be configured as Data Communication Equipment (DCE) [Henry, 1980].

American National Standard Code for Information Interchange (ASCII) was chosen for the serial format of information transfer. A single start bit designates the start of a 10-bit transmission frame followed by seven data bits which specify a character or number, a single parity bit for error detection, and a single stop bit. The hyperthermia system uses even parity, which specifies that the parity bit is chosen so that each transmission contains an even number of logical 1s. An advantage of the RS-232C/ASCII protocol is that a standard computer terminal can be used to emulate the central computer for testing and simple operation of the array controller. However, special care must be taken when
connecting the controller directly to a terminal since most terminals are in the DTE configuration. A special cable which allows the connection of two DTE devices must be used.

The central computer sends commands and data to the array controller, and the array controller sends replies back to the central computer when it has completed execution of commands. The command-reply format was chosen to ensure synchronization and error detection within the hyperthermia system. A reply is sent for each command received, indicating either successful execution of the specified operation or a type of error encountered which prevented execution. The central computer must wait for a reply after sending a command and its data to the array controller in order to prevent overwriting the controller's serial input buffer.

Tables 1, 2, and 3 are a summary of the commands, replies, and data used in the serial communications.

Two of the commands in Table 1 require data to be sent after the initial command character. The data are sent from the central computer in a predefined format for each command, and no reply for the command is sent from the array controller until all data are received. The array controller can accept data at the maximum transmission rate of the 2400 BAUD configuration (the BAUD rate can be increased up to 19200 BAUD if an 11.069 MHz crystal is used with the 8031 microcontroller and the array controller will still be able to accept data at the maximum transmission rate). After all of the data are received by the array controller and it has executed the command, a reply is sent to the central computer.
## TABLE 1. COMMANDS SENT FROM CENTRAL COMPUTER TO ARRAY CONTROLLER

<table>
<thead>
<tr>
<th>Command</th>
<th>ASCII Character</th>
<th>Binary Form (with parity) of ASCII Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restart</td>
<td>R</td>
<td>11010010</td>
</tr>
<tr>
<td>No-operation</td>
<td>N</td>
<td>01001110</td>
</tr>
<tr>
<td>Begin scanning</td>
<td>B</td>
<td>01000010</td>
</tr>
<tr>
<td>Stop scanning</td>
<td>S</td>
<td>01010011</td>
</tr>
<tr>
<td>Timeslice (requires data)</td>
<td>T</td>
<td>11010100</td>
</tr>
<tr>
<td>Path (requires data)</td>
<td>P</td>
<td>01010000</td>
</tr>
<tr>
<td>θ intensity factor = 0</td>
<td>0</td>
<td>00110000</td>
</tr>
<tr>
<td>θ intensity factor = 1</td>
<td>1</td>
<td>10110001</td>
</tr>
<tr>
<td>θ intensity factor = 2</td>
<td>2</td>
<td>10110010</td>
</tr>
<tr>
<td>θ intensity factor = 3</td>
<td>3</td>
<td>00110011</td>
</tr>
<tr>
<td>θ intensity factor = 4</td>
<td>4</td>
<td>10110100</td>
</tr>
<tr>
<td>θ intensity factor = 5</td>
<td>5</td>
<td>00110101</td>
</tr>
<tr>
<td>θ intensity factor = 6</td>
<td>6</td>
<td>00110110</td>
</tr>
<tr>
<td>θ intensity factor = 7</td>
<td>7</td>
<td>10110111</td>
</tr>
<tr>
<td>θ intensity factor = 8</td>
<td>8</td>
<td>10111000</td>
</tr>
</tbody>
</table>

## TABLE 2. REPLIES SENT FROM ARRAY CONTROLLER TO CENTRAL COMPUTER

<table>
<thead>
<tr>
<th>Reply</th>
<th>ASCII Character</th>
<th>Binary Form (with parity) of ASCII Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>command executed</td>
<td>C</td>
<td>11000011</td>
</tr>
<tr>
<td>error- cannot begin scan</td>
<td>V</td>
<td>01010110</td>
</tr>
<tr>
<td>error- parity error in</td>
<td>W</td>
<td>11010111</td>
</tr>
<tr>
<td>received frame</td>
<td></td>
<td></td>
</tr>
<tr>
<td>error- data out of bounds</td>
<td>X</td>
<td>11011000</td>
</tr>
<tr>
<td>error- unrecognized command</td>
<td>Y</td>
<td>01011001</td>
</tr>
<tr>
<td>error- error detected in</td>
<td>Z</td>
<td>01011010</td>
</tr>
<tr>
<td>hardware test</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**TABLE 3. Pseudo-HEX Data Coding**

<table>
<thead>
<tr>
<th>Binary value of Data</th>
<th>ASCII Character</th>
<th>Binary Form (with parity) of ASCII Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
<td>00110000</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>10110001</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
<td>10110010</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
<td>00110011</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
<td>10110100</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
<td>00110101</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
<td>00110110</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
<td>10110111</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
<td>10111000</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
<td>00111001</td>
</tr>
<tr>
<td>1010</td>
<td>;</td>
<td>00111010</td>
</tr>
<tr>
<td>1011</td>
<td>;</td>
<td>10111011</td>
</tr>
<tr>
<td>1100</td>
<td>&lt;</td>
<td>00111100</td>
</tr>
<tr>
<td>1101</td>
<td>=</td>
<td>10111101</td>
</tr>
<tr>
<td>1110</td>
<td>&gt;</td>
<td>10111110</td>
</tr>
<tr>
<td>1111</td>
<td>?</td>
<td>00111111</td>
</tr>
</tbody>
</table>

* - These columns contain the actual binary value being represented

The commands operate in the following way:

**Restart (R)** - Instructs the array controller to perform a reset operation. This software reset causes a program jump to address 0000, the same as a hardware reset. The reset command is the only command recognized by the array controller when data are expected. Software performs some simple tests of the hardware and initializes registers inside the microcontroller which are used by the interrupt routines.

**No-operation (N)** - The array controller just sends back a reply.
Begin scanning (B) - The controller begins heating the treatment area by exciting the array. The most recent path since the last reset is used.

Stop scanning (S) - Halts excitation of the array. Scanning can be re-initiated by the Begin command.

Timeslice (T) - Adjusts the speed of scanning as described in Chapter 2. The actual period of a single time slice is determined by the equation:

\[
\text{period} = \left[\frac{12}{\text{microprocessor crystal freq.}}\right] \times \left[2^7 \times (16-\text{HEX value of data})\right]
\]

Path (P) - Creates a new scan path. After receiving the command character, the array controller automatically inhibits any further heating of the old scan path. A reply is not sent until all the data for a path are received or an error is detected in the data.

Overall intensity factors 0,1,2,3,4,5,6,7,8 -

These 9 commands adjust the 9 intensity factor of the scan control.

Table 2 describes the replies which are sent back to the central computer by the array controller. If an error is detected, the array controller sets an internal error flag in addition to sending one of the special error replies. When the
flag is set, all excitation of the transducer array is stopped and all characters received from the central computer are ignored except R (which functions normally and may clear the error flag).

Table 3 defines ASCII characters which are used to represent the 16 values of a Hex digit. The ASCII characters chosen contain the 4-bit binary value they represent within their own 8-bit binary representation. The Timeslice command requires a single psuedo-hex character for data. The Path command requires a variable length string of data. The first two characters in the data string specify (in HEX) the number of data points in the scan path. Four characters of data are then expected for each data point \((X,Y,Z,R)\). \(X\), \(Y\), and \(Z\) can be any psuedo-hex characters, while \(R\) must be a character representing a value from 0 to 8.

EXAMPLES

0,2,1,1,1,4,>,>,>,6   (0 is the first data character)

specifies a data path with two points. The first is point \((1,1,1)\) at relative intensity 4, and the second is point \((14,14,14)\) with relative intensity 6.

1,4, . . . . . .

specifies a data path with twenty data points \((1 4 \text{ hex} = 20 \text{ decimal})\).
CHAPTER 4
CIRCUIT HARDWARE

4.1 BOARD LEVEL OVERVIEW

The array controller is constructed on a series of 4.5 by 6.5 inch wire-wrap circuit cards. Each card includes a 56 pin edge connector which plugs into a common bus, often called a backplane. The physical dimensions of the circuit cards meet the STD BUS STANDARD specifications, but the 56 lines of the backplane are not used according to the standard. In the array controller the lines of the edge connector bus are defined in the following way:

pin 3 - Logic Ground
pin 4 - Logic Ground
pin 1 - Logic Power (+5 V)
pin 2 - Logic Power (+5 V)
pin 55 - Auxiliary +12 V
pin 56 - Auxiliary -12 V
pin 5 - Serial Transmit (TxD)
pin 6 - Serial Receive (RxD)
pin 37 - "EXCITE" flag
pin 38 - "ERROR" flag
pin 50 - Auxiliary Clock
pin 32 - Data Strobe* (active low)
pin 33 - I/O Select* (active low)
pin 48 - Reset* (active low)

pin 15 - X(3) most significant bit of X coordinate
pin 17 - X(2)
pin 19 - X(1)
pin 21 - X(0) least significant bit of X coordinate
pin 23 - Y(3)
pin 25 - Y(2)
pin 27 - Y(1)
pin 29 - Y(0)
pin 24 - Z(3)
pin 26 - Z(2)
pin 28 - Z(1)
pin 30 - Z(0)

All other pins are unused.
The hardware of the array controller is divided into several functional blocks, and each block is contained on a circuit card. The cards communicate with one another over the edge connector bus. In this way, each card is a semi-independent unit and all interaction between the individual blocks of the system can be observed by monitoring the signal lines of the backplane.

The division of hardware onto many small cards offers several advantages. Once the communication between the cards has been determined, each card can be designed and debugged independently. The functional division of hardware onto small circuit cards should also simplify repair of the system. The failure of an integrated circuit chip or a wire will usually cause the failure of a single board which can be identified and repaired. The modular design also adds versatility and flexibility. Future modifications can be implemented by adding additional cards or by modifying old ones. Multiple cards of the element driver board design can be used to expand the system and allow for different array configurations.

There are also some disadvantages to implementing the system on many small cards instead of a single large circuit board. Each of the small cards requires signal buffers and/or drivers for connection to the backplane bus. These buffers require additional ICs which increases the system cost and power supply requirements. Some functional overlap between the small circuit cards may lead to redundant hardware which also increases the number of ICs. The edge contacts on the circuit cards and the connectors of the backplane also add overhead cost and complexity to the system. The mechanical contacts of the edge connectors are a possible
source of trouble, since they can provide poor electrical contact and introduce electrical noise to the signals on the bus.

An overview of the hardware at the board function level will outline the design. The array controller is divided into four functional blocks, each implemented on a single card with the exception of the element driver function which may be distributed over several identical cards. The four card designs are microcontroller board, clock board, LED display driver board, and element driver board(s). Details of the circuitry are included in Sections 4.2 through 4.5, while the remainder of this chapter presents an overview of each card design.

The microcontroller card coordinates the operation of the entire array controller. It performs the serial communication with the central computer and regulates the operation of the other boards within the array controller. A microcomputer chip resides on this board, along with an Erasable Programmable Read Only Memory (EPROM) chip which contains the array controller software code described in Appendix A. Special hardware on the microcontroller board interface with the RS-232C serial lines whose voltages are not directly compatible with the TTL voltage levels of the 8031 serial port.

The clock board generates a clock signal which is used by the element driver board(s). The presence of this clock on the backplane bus causes excitation of the transducer elements. The frequency of the clock determines the point of focus along the Y-axis. The microcontroller board controls the selection of one frequency from 16 possible values, and then uses a flag on the backplane to enable the clock board output onto the backplane.
The frequency of the clock board output must be eight times the desired excitation frequency for the transducers, since the element driver board(s) divide the clock down in order to synthesize phase differences necessary to focus the array in the X and Z directions.

The element driver board(s) creates the relative phase differences between the signals to the individual elements of the transducer array. The microcontroller board sends X, Y, and Z coordinates to the element driver board(s), where the coordinates are used to select initial count values stored in EPROM look-up tables. These count values are loaded into counters which control the focus of the array [Benkser, 1983]. Each element driver board has eight outputs which can excite up to eight transducers. Several identical boards (with different EPROM look-up tables) can be used simultaneously to excite arrays with more than eight elements.

The LED display driver board connects to the LED display shown in Figure 3. The display simulates the coordinate point focusing of the transducer array in the imaginary threedimensional axis system. The X and Y axes are represented with a 16 by 16 matrix of LEDs and the Z axis is represented by a column of 16 LEDs. Whenever the transducers are being excited, one LED will be illuminated on each display to indicate the coordinate focus. The brightness of each LED during the rapid heating of points in a scan path should reflect the percentage of time spent heating that spot, and therefore, the relative intensity. The brightness indication of duty cycle at each point is very crude, but provides some simple visual feedback of the heating pattern.
4.2 MICROCONTROLLER BOARD

The microcontroller board regulates the operation of all the other boards. An Intel 8031 microcontroller and its support hardware provide the "intelligence" of the array controller.

The following discussion assumes a basic knowledge of microprocessors or computer hardware design and operation. This background can be found in many college level textbooks (e.g., Microprocessors/Microcomputers: An Introduction by Donald G. Givone and Robert P. Roesser, McGraw Hill). A comprehensive discussion of the Intel 8031/8051 microcontroller family can be found in the Intel publications listed as references for this thesis.

There is one special characteristic of the 8031 microcontroller which deserves special attention. The 8031 was designed as a processor for dedicated control applications. It performs better than a general purpose microprocessor in control applications, but cannot function well in a general purpose architecture due to differences in I/O structure. In a general architecture, all communication with the processor takes place over a data bus and an address bus as illustrated in Figure 4. The processor is usually able to wait for data from slow memory, and is able to relinquish control of the busses for block data moves. The 8031 does not have the capability to easily support these features, but instead provides an I/O structure which is well suited to single binary bit control signals which are more common in control applications.
A schematic and board layout of the microcontroller are shown in Figures 5 and 6, respectively. The major components are the 8031 microcontroller chip (IC 3), a 4K-byte EPROM (IC 7), a 2K-byte RAM (IC 6), and special buffer gates for the RS-232C connections and reset signal (ICs 1 and 2). There are also some buffers and a latch (ICs 4 and 5). The offboard connections, indicated by the letter E, go to the edge connector of the circuit card which plugs into the backplane.

Most pins of the 8031 are used for standard support of the 4K EPROM and 2K RAM. These connections are described in the Intel data books. However, two pins are used for special functions in the array controller. The "EXCITE" flag comes from P3.4 of the 8031 and goes to line 37 on the bus. It is activated when the transducer elements of the array are to be excited. The "ERROR" flag comes from P3.5 and goes to line 38 of the bus. It can be activated by the processor if an error is detected. The high order address bit of the 8031 is also sent out on the bus (line 33).

The microcontroller board sends (X,Y,Z) coordinates to other boards in the system for focusing of the array during scanning. Each of the three coordinates requires a 4-bit field for a total width of 12 bits. The 8031 has an 8-bit data field and a 16-bit address field. The 12 bits of coordinate data are more easily transferred from the 8031 as an address. When the processor sends coordinates to the rest of the system, it performs an external memory READ with the most significant address bit equal to "1" and the low order 12 address bits specifying the coordinates. The high address bit becomes an I/O select signal on the backplane.
bus. The RD* data strobe of the 8031 is also connected to the backplane bus (line 32 - Data Strobe*) in order to provide a strobe signal for the transfer. The data RAM (IC 6) will send a byte of data to the 8031 during this process, but the result is simply ignored by the 8031.

4.3 CLOCK BOARD

The clock board provides a selectable frequency source for the clock inputs of the counters on the element driver board(s). The focus of the array along the Y-axis is a direct function of this signal on the backplane bus (line 50). Figures 7 and 8 are a schematic and a component layout of the clock board. The major components of the circuit are a latch/buffer (IC 3), a D-type flip flop (IC 1), some buffers (IC 5), and a digitally controlled clock source (indicated by the dashed line box in Figure 7). The exact requirements of the digital clock source were not fully determined at the time of this writing, so the specific implementation of the selectable frequency clock source was left for later development.

There are three steps in the operation of the clock board. These are

STEP 1: The processor strobes an X, Y, Z coordinate point out on the bus, and the 4-bit Y-axis coordinate is latched into IC 3. The latch input is enabled when both the I/O Select* and Data Strobe* signals from the bus are simultaneously low.
STEP 2: The digitally controlled clock source output stabilizes at a frequency which corresponds to the 4-bit binary value received at its input. This clock can be free running and unsynchronized with the rest of the system.

STEP 3: The frequency source is gated by the "EXCITE" flag from the bus (line 37) and the resulting signal goes to the Auxiliary Clock (line 50) of the bus.

A flip flop (IC 1) is used to synchronize the "EXCITE" flag with the free running clock source before taking the logical OR of the two signals. The synchronization is necessary to prevent glitches in the Auxiliary Clock signal on the bus, as illustrated in Figure 9. If the very short pulses caused by a nonsynchronized gating were to appear on the bus, they might cause some counters on the element driver board(s) to advance their count while not being recognized by other counters. This would modify the relative phasing of the individual elements and would likely destroy the focus.

A few notes of caution are in order. First, the TTL and LS-TTL type components used in the array controller will not operate at frequencies above 20 or 30 MHz. If the maximum clock frequency is 20 MHz, the maximum excitation frequency for the transducer elements is eight times less (roughly 2.5 MHz). Second, the output from the controlled clock source should stabilize within one instruction cycle time of the 8031 to ensure that the clock will be stable before the "EXCITE" flag can be reactivated by the microcontroller board.
4.4 ELEMENT DRIVER BOARD

The element driver board(s) generates the outputs to the transducer elements. The outputs are all square waves of the same frequency, but the relative phase of the various outputs determines the focus along the X and Z axes. The schematic and component layout for an element driver card are shown in Figures 10 and 11. Each board contains four 4K-byte Electrically Programable Read Only Memories (EPROMs), eight 4-bit presetable counters (ICs 2,3,5,6,10,11,13,14), eight NOR logic 50-ohm line driver gates (ICs 7 and 8), and buffers for the inputs from the bus (ICs 16 and 18). There are also some additional logic gates.

Each EPROM contains a look-up table which converts all of the possible X, Y, Z coordinate combinations into initial count load values for the counters of two elements in the array. The counters, which are loaded with values from the EPROMs, receive a clock signal from the bus (line 50) when the array is to be excited. The most significant bit of each counter becomes the excitation signal for an element of the array.

The contents of the EPROM look-up tables must be generated ahead of time. They take into account the location of the specific pair of elements which that EPROM controls, the Y-coordinate frequencies, and the location of the X and Z coordinates relative to the origin at the center of the array. A program to generate these tables for a variety of array configurations is presented in Appendix B.

The EPROM tables are used to load the counters with initial count values whenever the microcontroller board sends a new X, Y,
Z coordinate point out on the backplane bus. The "EXCITE" flag should be deactivated by the microcontroller card when this operation takes place. The actual binary coordinate values from the bus are used to form an address for the EPROMs. The Z coordinate forms the high order address bits, X the middle, and Y the low order bits. The outputs of the EPROMS are enabled by the I/O Select\* signal from the bus (line 33). The Data Strobe\* (line 32) then strobes counter load values from the EPROM outputs into the counters. The counters are now ready to receive a square wave from the Auxiliary Clock line of the bus which will cause excitation of the phased elements.

One output of each counter is connected to one input of a NOR logic line driver gate. The other input of each NOR gate is connected by a jumper to a logic function of the "ERROR" flag (line 38) and the "EXCITE" flag (line 37). This provides a safety feature, whereby the elements cannot be excited if the "ERROR" flag is set or if the "EXCITE" flag is not set. The jumper is used in place of a simple wired connection to allow for possible future array designs where it is necessary to excite only a subset of the elements in order to focus at some points. In this case, the jumper could be removed from each driver board in the system and replaced by a cable running to some additional hardware which would perform the element selection.
4.5 LED DISPLAY DRIVER BOARD

The LED display driver board controls the LED display shown previously in Figure 4. The schematic and component layout of the board are shown in Figures 12 and 13. The display simulates the focusing of the transducer array at nodes of the imaginary axis grid. The LED display is connected to the LED driver board by a 50 pin connector on the board denoted by H1 in the figures.

The circuit card contains latches for the focus coordinates (ICs 8 and 9), a 4-to-16 decoder (IC 7), two open collector output 4-to-16 decoders (ICs 3 and 4), inverter drivers for the LED matrix display (ICs 5 and 6), buffers for connections with the bus (ICs 11, 12, and 13), and some additional logic gates. There are also two Dual In Line (DIP) packages of resistors, used to limit current for the LED matrix. A single LED on the circuit card provides a visual display of the "ERROR" flag status.

When the I/O Select* and Data Strobe* lines of the bus are both activated by the microcontroller board, the LED driver board latches the X, Y, and Z coordinates for a new focus from the bus. The Y coordinate causes one output of the 4-to-16 decoder IC 7 to go low, which causes the output of one inverter of IC 5 or IC 6 to become high. When the "EXCITE" flag is activated, one output of each 4-to-16 open collector decoder will become capable of sinking current. The X coordinate is decoded by IC 3, which selects a column of the LED matrix, and IC 4 decodes the Z coordinate, which selects one LED of the Z-axis bar graph display. However, if the "ERROR" flag is ever activated, the open collector drivers are both disabled which prevents the illumination of any LEDs.
CHAPTER 5
SOFTWARE

5.1 SOFTWARE OVERVIEW

The assembly level program code for the Intel 8031 on the microcontroller board is listed in Appendix A. The program is divided into three sections. The CONFIG routine performs some hardware test algorithms and then configures the microcontroller chip internal hardware by loading Special Function Registers (SFRs) within the 8031. The second and third segments are interrupt service routines. The SINT routine is the serial interrupt routine which provides for serial communication with the central computer via the built-in serial port. The TINT routine is the timer interrupt routine which services interrupts generated by an internal counter/timer. The TINT routine directs the array focus and creates on-off intensity control of the heating.

5.2 CONFIG - HARDWARE CONFIGURATION ROUTINE

A flowchart of the CONFIG routine is shown in Figure 14. The program is initiated by a hardware reset of the processor chip, or by the execution of a software reset command received from the central computer.

The program tests the ability of the processor to access external RAM and several internal registers. A stuck-at-fault model is used, which assumes that in the event of a hardware fault the input or output of a device may become fixed at a value of
logic 1 or logic 0. This model may also detect other types of faults. Stuck-at-fault models are often used to test integrated circuits [Muehldorf and Savakar, 1981]. The memory access test will detect stuck-at-0 and stuck-at-1 faults in the 11 address lines and in the eight data lines of RAM [Sohl, 1977]. The test covers the 8031 ports, the address latch and buffers, and the RAM chip itself. Any broken connections will also be detected, since TTL circuitry defines an unconnected input as a fixed logic 1. The implementation details of the test are described in the comment field of the program listing included in Appendix A.

The CONFIG routine also programs the Special Function Registers (SFRs) of the 8031, which in turn control the special internal hardware. A 16-bit counter/timer is configured to generate interrupts at regular intervals (when the interrupt is enabled). A full-duplex serial port is configured for communication with the central computer. The interrupt priorities are assigned so that serial port interrupts receive lower priority than timer interrupts. This is necessary to maintain real-time control of the array when scanning.

A hardware reset is automatically initiated upon power-up of the system. After performing a hardware test and configuration, the program flow enters an infinite loop which performs no further operations. The loop is only exited to service interrupts and resumes execution upon return from an interrupt handling routine. When a software restart is performed, part of CONFIG is executed within the serial interrupt call.
5.3 SINT - SERIAL INTERRUPT ROUTINE

A flowchart of the SINT routine is shown in Figure 15. This interrupt routine is initiated by the microcontroller's internal serial port. An interrupt occurs following the completion of a full frame serial transmission or reception. The software must determine whether the interrupt was due to a reception interrupt flag (RI) or a transmission interrupt flag (TI).

To initiate a transmission, the SINT routine moves a byte of data to the serial port output buffer. The actual serial transmission including start and stop bits is then performed independently by the hardware. The TI flag is set by hardware at the end of a transmission, and an interrupt is generated to convey the information back to the software program. In the event of a reception by the serial port hardware, the RI flag is set after an entire frame has been received and the data portion of the frame placed in a buffer register.

The SINT routine also processes the information received when a reception interrupt occurs. The first step is a check for the restart command character. If this command is detected (along with proper parity) a program jump occurs to perform a portion of the CONFIG routine for fault testing and re-initialization of internal registers. A program jump back to SINT occurs from CONFIG for completion of the interrupt service routine. If the received character does not translate to a restart command and the error flag of the microcontroller is set, the received character is simply ignored by the array controller.
If there is no restart command, if the error flag is not set, and if the parity is correct, then SINT processes the received character as data or a command. When a previously received command requires additional data, the SINT routine assumes this new character is part of that data. If the received character fits within the value boundaries for the type of data expected, it is stored. When data are not expected for a previous command, the new character is assumed to be a command. If it does not match any of the defined character codes, the error flag is set and transmission of an error code character is initiated. When a command is identified which requires no data, or when the final data for a command are received, the command is executed and transmission of the "command executed" reply code character is initiated before returning from the interrupt call. When more data are expected for a command, the routine performs a return from interrupt without initiating any transmissions.

The SINT routine maintains an internal status flag which indicates any uncompleted transmissions. This flag is checked before any new transmission is initiated. If the flag is set, the program performs a small loop not indicated in the flowchart. It waits for the TI flag to be set, clears the TI flag, and begins the next transmission. This bypasses the normal procedure which would involve a separate call of SINT to clear the TI flag.
5.4 TINT - TIMER INTERRUPT ROUTINE

A flowchart of the TINT routine is shown in Figure 16. An internal counter advances once for every instruction cycle of the 8031. A counter overflow occurs when the binary count advances from 111...1 to 000...0. The TINT routine is initiated whenever the timer interrupt is enabled and the timer/counter overflows. By loading the counter with a 16-bit binary value, the interval between interrupts can be regulated.

The first step in the TINT routine is a reload of the 16-bit counter which will determine the time period before the next counter overflow and interrupt call. The binary number is 111111ABCD00000000, where the value of the ABCD field has been defined by timeslice command data. If ABCD = 0000, then the next counter overflow will occur after 2024 instruction cycles of the processor. If ABCD = 1111, then the next overflow occurs after 128 instruction cycles of the processor. To ensure that the present interrupt call will be completed before the possibility of another overflow, TINT must never require more than 128 instruction cycles to execute.

The TINT routine performs the controlled heating scheme discussed in Chapter 2. When scanning the heated points with the transducer array, 64 timer/counter interrupts occur for each point as the path is scanned. If the interrupt interval is increased, the rate of scanning is reduced. When the path includes a large number of points, the period between consecutive heatings of each individual spot will increase. The timer/counter interrupt which initiates TINT is enabled and disabled by SINT.
The TINT routine is executed 64 times for each data point as the array focus is scanned from one point to the next point. The corresponding 64 periods are divided into on-cycles and off-cycles for the array. For each on-cycle, the transducers are excited to create a spot of heating at coordinates specified by the data point. During any off-cycles, the outputs to the transducers do not cause excitation, and no ultrasound heating occurs.

The flowchart of TINT is complicated by the prefetch of stored data points from RAM. As the scan proceeds from point-to-point, a prefetch brings the next data point into the 8031 (from RAM) during the 64th cycle of the previous data point. This is necessary to avoid a period of dead time in the scanning operation when switching to a new data point. The TINT routine not only prefetches the next data point, but also calculates the number of on and off cycles for the next spot based on the point's relative intensity factor (R) and the current overall path intensity factor (Θ). After each prefetch, the new "X, Y, Z" flag is set. It indicates that the coordinates of the focus are to be changed at the beginning of the next execution of TINT.

The flowchart does not indicate the initialization values of ON, OFF, and the "new X, Y, Z" flag. These values are loaded by SINT before enabling the timer/counter interrupt. The scan always begins at the first spot in the defined scan path and performs a single off-cycle to allow prefetch of the first data point from RAM.
CHAPTER 6
AREAS FOR FURTHER DEVELOPMENT

6.1 TRADE-OFFS

Many design trade-offs were encountered during the design of the array controller. While the present design was intended for maximum versatility, future developments may require a re-evaluation of the decisions made.

The first major trade-off involved hardware versus software. In the present design, the array controller converts X, Y, Z coordinates into initial counter load values by the use of look-up tables stored in EPROMs. This requires the initial load values for every possible X, Y, Z combination to be calculated ahead of time and stored in the EPROMs.

Another possibility would be to use the 8031 microcontroller already within the system to perform the necessary calculations for just those points chosen in each specific scanpath. The counter load values could be calculated and stored in RAM for use when focusing the array. This scheme might work reasonably well in a system with only a few elements, but becomes impractical for more than 15 or 20 elements. Scanning speed is the major problem. It would take the 8031 several seconds to calculate the counter load values for even a small number of scan points, so every time a new path was initiated all heating would be stopped while the values were calculated. Even after the initial count values were calculated for all points, the 8031 would require many instruction cycles to retrieve values and load the many counters each time the focus was shifted to a new point in the scan path.
In the present design, the 8031 is isolated from the physical parameters of the actual transducer array. If the array is changed, the contents of the EPROM look-up tables must be recalculated, but no changes are necessary in the 8031 assembly code.

A second trade-off involves the speed of scanning and the number of possible heating intensity levels. The number of timer cycles spent at each point in the scan path is determined by the product of the number of relative intensity factors and the number of possible overall intensity factors (8 * 8 = 64 in the present system). The minimum timer interrupt period is limited by the execution time of the interrupt service routine and cannot be reduced. If the maximum number of points in a scan path could be reduced, or if thermal ripple at the points in a slower scan was found acceptable, the number of cycles spent at each point could be increased and more levels of control could be implemented.

6.2 ENHANCEMENTS

The reliability of the array controller could be improved in several ways. At present, there is no feedback from the EPROM outputs to the 8031. This would allow the 8031 to verify that EPROMs were in place and to perhaps identify them from the contents of some specific address. Hardware could be added to monitor the excitation outputs, since there is no way for the system to detect a malfunctioning counter on the element driver
card(s) in the present system. Software could perform additional tests of the 8031 and other hardware within the array controller.

Another improvement would be the ability to store multiple scan paths within the array controller. The 2K-byte RAM on the microcontroller board is capable of storing data for up to four complete paths of 255 points each. Additional commands could allow the central computer to send several paths to the array controller and switch from one to another for scanning.
In this thesis I have presented the design of an ultrasound phased element array controller which will function within a hyperthermia treatment system. A scheme for controlled heating with a focus point was developed, along with a protocol for communication between the array controller and a central computer. The array controller will operate with a variety of array configurations. It incorporates an imaginary grid of focus points and EPROM look-up tables for loading counters with initial count values which create the element phasing.

The array controller has been designed and documented in a modular fashion. The intention was to simplify future development and modification.
Figure 1. System Overview
Figure 2. Phased Array Of Tapered Ultrasound Transducers
Figure 3. LED Display
Figure 4. Shared Bus Architecture
Figure 5. Microcontroller Board Schematic
Card Edge Connector (E)

Component Side View

Figure 6. Microcontroller Board Component Layout
Digitally Controlled Clock Source

Binary Select
Input

Output

IC 4
74LS240

IC 3
74LS373

IC 2
7432

IC 4

CLK 7474

Figure 7. Clock Board Schematic
Figure 8. Clock Board component Layout
Figure 9. Synchronized Gating Of A Clock

WAVEFORM 1 - "EXCITE" FLAG SIGNAL FROM BUS
WAVEFORM 2 - CLOCK SOURCE OUTPUT
WAVEFORM 3 - LOGICAL OR OF 1 AND 2
WAVEFORM 4 - SYNCHRONIZED GATING OF CLOCK USING FLIP-FLOP
FIGURE 10. Element Driver Board Schematic
Figure 11. Element Driver Board Component Layout
Figure 13. LED Display Driver Board Component Layout
Figure 14. CONFIG Flowchart
Figure 15. SINT Flowchart
FIGURE 16. TINT Flowchart
APPENDIX A

8031 PROGRAM CODE

This appendix includes the program code for the 8031 microcontroller chip on the microcontroller circuit board. The 8031 assembly code was cross-assembled with a commercial software package running on an Apple IIe computer equipped with special CP/M 80 software/hardware. The cross assembler is titled XASMX1 8051 CROSS ASSEMBLER Version 1.07 by Avocet Systems Incorporated. It accepts input of a CP/M text file and creates an output file which is in Intel Hex Format. The Intel Hex file can be converted to a file of pure binary object code by commands in the CP/M operating system.

The assembler requires the origin of the assembled object code to be placed at or above the address 100Hex. Interrupts in the 8031 cause jumps to addresses below 100Hex. To bypass this problem, the interrupt service routines were placed at addresses above 100Hex by the assembler and several bytes of 8031 binary instruction code were appended to the output file of the assembler. The added instruction bytes cause program jumps from the hardware-fixed interrupt addresses to the assembled service routine addresses (which are above 100Hex). The appended code bytes are described in the comment field at the beginning of the cross assembler generated listing included in this appendix.
Definition of some terminology and mnemonics used in the program code may be helpful.

XYBUFF, ZBUFF - Buffers for the three axis coordinates and the relative intensity factor of the data point.

OBUFF - Buffer containing the current overall intensity factor.

TSLICEH, TSLICEL - Contain the high and low bytes of the 16-bit binary value used to reload the interrupt counter/timer.

TEMPH, TEMPL - A temporary buffer for the value of a 16-bit external memory address pointer.

LENGTH - Contains the number of points in the scan path.

COUNT - A pointer to a particular point within the path. (register 02 of the 8031)

ON - Indicates the number of remaining time cycles the array will be excited while focused at the present coordinates. (register 00 of the 8031)

OFF - Indicates the number of remaining time cycles for which the array will not be excited, before moving on to the next data point. (register 01 of the 8031)

DSW - Data Status Word; 8 single-bit flags which indicate data is expected

- DSW.7 - LENGTH (high HEX digit)
- DSW.6 - LENGTH (low HEX digit)
- DSW.5 - X coordinate of a data point
- DSW.4 - Y coordinate of a data point
- DSW.3 - Z coordinate of a data point
- DSW.2 - R intensity factor of a data point
- DSW.1 - Timeslice (single HEX digit)

ISW - Interrupt Status Word; 4 single-bit flags

- ISW.7 - "new X, Y, Z" flag
- ISW.6 - scan path available flag
- ISW.4 - serial transmission in progress
- ISW.3 - hardware reset

"ERROR" flag - P3.5 of the 8031 = line 38 of the backplane

"EXCITE" flag - P3.4 of the 8031 = line 37 of the backplane
;***********************************************************************
;* 5/25/84 *
;* ARRAY CONTROLLER (V2.3) *
;* BY *
;* STEVEN SILVERMAN *
;* UNIVERSITY OF ILLINOIS AT CHAMPAIGN-URBANA *
;* DIBIOACoustICS RESEARCH LABORATORY *
;* THIS SOFTWARE WAS WRITTEN FOR CONTROLLING A TAPERED AND PHASED *
;* ELEMENT ARRAY. IT IS INTENDED FOR AN INTEL 8031 MICRO-CONTROLLER AND *
;* SPECIALLY DESIGNED HARDWARE BOARDS. ADDITIONAL INFORMATION EXPLAINING *
;* THE 8031 MICRO-CONTROLLER IS AVAILABLE FROM INTEL CORPORATION. *
;* THE PROGRAM INCLUDES SOME HARDWARE ERROR DETECTION, NON-ESSENTIAL *
;* TO CONTROLLING THE ANTENNA ARRAY. *
;*
;***********************************************************************

0100 ORG $100 ;also expect 0000 contains: AJMP 105 (= 21 05 Hex)
0020 ISW EQU 32 ; 0008 contains: AJMP 102 (= 21 02 Hex)
0021 DSM EQU 33 ; 0023 contains: AJMP 100 (= 21 00 Hex)
0022 TL SICEL EQU 34 ; These addresses are fixed by the
0023 TSLICEH EQU 35 ; processor interrupt service
0031 X YBUF EQU 49 ; hardware.
0032 IBUF EQU 50
0033 OBIF EQU 51
0034 LENGTH EQU 52
0035 PMC EQU 97H
0036 TEMPL EQU 53
0036 TEMPH EQU 54

0100 615A AJMP SERIAL
0102 8198 AJMP TINT
0104 00 NOP

; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
; ; ; ; ; ;
;
AVOCET SYSTEMS 8051 CROSS-ASSEMBLER - VERSION 1.09

SOURCE FILE NAME: SOURCE.ASM

010E E4  CLR  A       ; test for stuck-at-0 faults in the low
010F 900000  MOV  DPTR, #0000H  ; order 11 bits of the data memory
0110 758201  MOV  DPL, #01H  ; addressing by writing a unique value
0111 04  INC  A  ; to each memory address 2^i (for i = 0, 1, 11) and then reading back from
0112 758202  MOV  DPL, #02H  ; these addresses
0113 04  INC  A
0114 758204  MOV  DPL, #04H
0115 04  INC  A
0116 758208  MOV  DPL, #08H
0117 04  INC  A
0118 758210  MOV  DPL, #10H
0119 04  INC  A
011A 758220  MOV  DPL, #20H
011B 04  INC  A
011C 758240  MOV  DPL, #40H
011D 04  INC  A
011E 758280  MOV  DPL, #80H
011F 04  INC  A
0120 7582B0  MOV  DPL, #B0H
0121 04  INC  A
0122 900100  MOV  DPTR, #0100H
0123 04  INC  A
0124 758302  MOV  DPH, #02H
0125 04  INC  A
0126 758304  MOV  DPH, #04H
0127 04  INC  A
0128 900000  MOV  DPTR, #0000H
0129 75F000  MOV  B, #00H
012A 00  MOVX  A, @DPTR
012B 85F045  CJNE  A, B, MEME
012C 05F0  INC  B
012D 758201  MOV  DPL, #01H
012E 00  MOVX  A, @DPTR
012F 85F05C  CJNE  A, B, MEME
0130 05F0  INC  B
0131 758202  MOV  DPL, #02H
0132 00  MOVX  A, @DPTR
0133 85F053  CJNE  A, B, MEME
0134 05F0  INC  B
0135 758204  MOV  DPL, #04H
0136 00  MOVX  A, @DPTR
AVOCET SYSTEMS 8051 CROSS-ASSEMBLER - VERSION 1.09

SOURCE FILE NAME: SOURCE.ASM

016C B5F04A CJNE A, B, MEME
016F 05F0 INC B
0171 75B208 MOV DPL, #$08H
0174 E0 MOVX A, @DPR
0175 B5F041 CJNE A, B, MEME
0178 05F0 INC B
017A 75B210 MOV DPL, #$10H
017D E0 MOVX A, @DPR
017E B5F038 CJNE A, B, MEME
0181 05F0 INC B
0183 75B220 MOV DPL, #$20H
0186 E0 MOVX A, @DPR
0187 B5F02F CJNE A, B, MEME
018A 05F0 INC B
018C 75B240 MOV DPL, #$40H
018F E0 MOVX A, @DPR
0190 B5F026 CJNE A, B, MEME
0193 05F0 INC B
0195 75B280 MOV DPL, #$80H
0198 E0 MOVX A, @DPR
0199 B5F010 CJNE A, B, MEME
019C 05F0 INC B
019E 900100 MOV DPTR, #$100H
01A1 E0 MOVX A, @DPR
01A2 B5F014 CJNE A, B, MEME
01A5 05F0 INC B
01A7 75B302 MOV DP#H, #$2H
01A9 E0 MOVX A, @DPR
01AA B5F008 CJNE A, B, MEME
01AE 05F0 INC B
01B0 75B304 MOV DP#H, #$4H
01B3 E0 MOVX A, @DPR
01B4 B5F002 CJNE A, B, MEME
01B7 2188 AJMP SAD
01B9 0285 MEME: SETB P3.5
01BA E4 SAD: CLR A
01BC 90007F MOV DPTR, #$0FFH
01BF F0 MOVX @DPR, A
01C0 75B2FE MOV DPL, #$0FEH
01C3 04 INC A
01C4 F0 MOVX @DPR, A
01C5 75B2FD MOV DPL, #$0FDH
01C8 04 INC A
01C9 F0 MOVX @DPR, A
01CA 75B2F8 MOV DPL, #$0FBH
01CD 04 INC A
01CE F0 MOVX @DPR, A
01CF 75B2F7 MOV DPL, #$0F7H
01D2 04 INC A
01D3 F0 MOVX @DPR, A
01D4 75B2EF MOV DPL, #$0EFH

; if there is an error, set the error flag
; test for stuck-at-1 type faults in the low
; order 11 bits of the data memory
; addressing by writing a unique value
; to each memory address 2^i - 1 - 2^i
; (for i=0,11) and then reading back
; from these addresses
AVOCET SYSTEMS 8051 CROSS-ASSEMBLER - VERSION 1.09

SOURCE FILE NAME: SOURCE.ASM

01D7 04 INC A
01D8 F0 MOVX @DPTR, A
01D9 75B2DF MOV DPL, #0DFH
01DA 04 INC A
01DB F0 MOVX @DPTR, A
01DC 75B2BF MOV DPL, #08FH
01DE 04 INC A
01E0 F0 MOVX @DPTR, A
01E1 75B27F MOV DPL, #7FH
01E2 04 INC A
01E3 F0 MOVX @DPTR, A
01E4 9006FF MOV DPTR, #06FFH
01E5 04 INC A
01E6 F0 MOVX @DPTR, A
01E7 75B305 MOV DPH, #05H
01E8 04 INC A
01E9 F0 MOVX @DPTR, A
01EA 75B303 MOV DPH, #03H
01EB 04 INC A
01EC F0 MOVX @DPTR, A
01ED 9007FF MOV DPTR, #07FFH
01EE 75F000 MOV B, #00H
01EF E0 MOVX A, @DPTR
01F0 85F07B CJNE A, B, MEM2
0201 05F0 INC B
0202 75B2FE MOV DPL, #0FEH
0203 E0 MOVX A, @DPTR
0204 85F072 CJNE A, B, MEM2
0205 05F0 INC B
0206 75B2FD MOV DPL, #0FDH
0207 E0 MOVX A, @DPTR
0208 85F069 CJNE A, B, MEM2
0209 05F0 INC B
020A 75B2FB MOV DPL, #0FBH
020B E0 MOVX A, @DPTR
020C 85F060 CJNE A, B, MEM2
020D 05F0 INC B
020E 75B2FF MOV DPL, #0F7H
020F E0 MOVX A, @DPTR
0210 85F077 CJNE A, B, MEM2
0211 05F0 INC B
0212 75B2EF MOV DPL, #0EFH
0213 E0 MOVX A, @DPTR
0214 85F04E CJNE A, B, MEM2
0215 05F0 INC B
0216 75B2DF MOV DPL, #0DFH
0217 E0 MOVX A, @DPTR
0218 85F045 CJNE A, B, MEM2
0219 05F0 INC B
021A 75B2BF MOV DPL, #08FH
021B E0 MOVX A, @DPTR
AVOCET SYSTEMS 8051 CROSS-ASSEMBLER - VERSION 1.09

SOURCE FILE NAME: SOURCE.ASM

0230 85F03C  CJNE A, B, MEME2
0240 05F0  INC B
0242 75827F  MOV DPL, $7FH
0245 E0  MOVX A, @DPR
0246 85F033  CJNE A, B, MEME2
0249 05F0  INC B
024B 9006FF  MOV DPTR, #06FFH
024E E0  MOVX A, @DPR
024F 85F02A  CJNE A, B, MEME2
0252 05F0  INC B
0254 79305  MOV DPH, #05H
0257 E0  MOVX A, @DPR
0258 85F021  CJNE A, B, MEME2
025B 05F0  INC B
025D 758303  MOV DPH, #03H
0260 E0  MOVX A, @DPR
0261 85F01B  CJNE A, B, MEME2
0264 74FF  MOV A, #0FFH ; test for stuck-at-0 faults in the data
0266 45F0  MOV B, A ; path of RAM
0268 F0  MOVX @DPTR, A
0269 7400  MOV A, #000H
026B E0  MOVX A, @DPR
026C 85F00D  CJNE A, B, MEME2
026F 7400  MOV A, #000H ; test for stuck-at-1 faults in the data
0271 55F0  MOV B, A ; path of RAM
0273 F0  MOVX @DPTR, A
0274 74FF  MOV A, #0FFH
0276 E0  MOVX A, @DPR
0277 85F002  CJNE A, B, MEME2
027A 417E  AJMP IRAM
027C 0285  MEME2: SETB P3.5 ; if there is an error, set the error flag
027E 44  IRAM: CLR A ; test internal registers for stuck-at-1 faults
027F 55F0  MOV B, A
0281 85F048  CJNE A, B, IRAME
0284 55B1  MOV SP, A
0286 858143  CJNE A, SP, IRAME
0289 F521  MOV DSM, A
028B 85213E  CJNE A, DSM, IRAME
028E F522  MOV TSLICE, A
0290 852239  CJNE A, TSLICEL, IRAME
0293 F523  MOV TSLICEH, A
0295 852234  CJNE A, TSLICEH, IRAME
029B F555  MOV TEMPL, A
0299 85552F  CJNE A, TEMPL, IRAME
029D F556  MOV TEMPH, A
029F 85562A  CJNE A, TEMPH, IRAME
02A2 F531  MOV XYBUF, A
02A4 853125  CJNE A, XYBUF, IRAME
02A7 F532  MOV ZRBUF, A
02A9 853220  CJNE A, ZRBUF, IRAME
02AC F533  MOV QBUF, A
AVOCET SYSTEMS 8051 CROSS-ASSEMBLER - VERSION 1.09

SOURCE FILE NAME: SOURCE.ASM

02AE B5331B CJNE A, OBUF, IRAME
02B1 F534 MOV LENGTH, A
02B3 B53416 CJNE A, LENGTH, IRAME
02B6 F500 MOV OOH, A
02BB 850011 CJNE A, OOH, IRAME
02BB F501 MOV 01H, A
02BD B5010C CJNE A, 01H, IRAME
02CO F502 MOV 02H, A
02C2 B50207 CJNE A, 02H, IRAME
02C5 F507 MOV 07H, A
02C7 B50702 CJNE A, 07H, IRAME
02CA 41CE AJMP SA02
02CC D2B5 IRAME: SETB P3.5 ;if there is an error, set the error flag
02CE 74FF SA02: MOV A, 0FFH ;test internal registers for stuck-at-0
02D0 F500 MOV B, A
02D2 B5F048 CJNE A, B, IRAME2
02D5 F581 MOV SP, A
02D7 B5B143 CJNE A, SP, IRAME2
02DA F521 MOV DSM, A
02DC B5213E CJNE A, DSM, IRAME2
02DF F522 MOV TSLICE, A
02E1 B52239 CJNE A, TSLICE, IRAME2
02E4 F523 MOV TSLICEH, A
02E6 B52334 CJNE A, TSLICEH, IRAME2
02E9 F535 MOV TEMPL, A
02EB B53508 CJNE A, TEMPL, IRAME
02EE F536 MOV TEMPH, A
02F0 B53609 CJNE A, TEMPH, IRAME
02F3 F531 MOV YBUF, A
02F5 B53125 CJNE A, YBUF, IRAME2
02F8 F532 MOV ZBUF, A
02FA B53200 CJNE A, ZBUF, IRAME2
02FF F533 MOV OBUF, A
02FF B5333A CJNE A, OBUF, IRAME
0302 F534 MOV LENGTH, A
0304 B53416 CJNE A, LENGTH, IRAME2
0307 F500 MOV OOH, A
0309 B500C0 CJNE A, OOH, IRAME
030C F501 MOV 01H, A
030E B50108 CJNE A, 01H, IRAME
0311 F502 MOV 02H, A
0313 B50206 CJNE A, 02H, IRAME
0316 F507 MOV 07H, A
0318 B50701 CJNE A, 07H, IRAME
031B 61F AJMP INIT
031D D2B5 IRAME2: SETB P3.5 ;if there is an error, set the error flag
031F E4 INIT: CLR A ;initialize status words
0320 F521 MOV CSM, A
0322 C206 CLR ISW.6
0324 7523FF MOV TSLICEH, #0FFH ;initialize timeslice for default setting of
0327 752280 MOV TSLICE, #B0H ; maximum speed
032A 77D000  MOV  PSW,  #00H ;select register bank 00
032D 75921  MOV  TMOD,  #21H ;TIMER0:MODE1 16-bit / TIMER1:mode2 autoload
0330 75B50  MOV  TCON,  #50H
0333 75B80  MOV  PCON,  #80H
0336 700C08  JNB  ISW.3,  BYPASS ;bypass next 3 steps if not a hardware reset
0339 75B90  MOV  SCON,  #40H ;conf. serial port for 10-bit frame (MODE1)
033C 75B148  MOV  SP,  #48H ;initialize stack pointer
033F 75B0F3  CLR  ISW.4
0341 75BD0F3  MOV  TH1,  #243 ;BAUD=crystal freq./12*16*(256-value in TH1)
0344 75BB02  MOV  IP,  #02H ;interrupt priority: TIMER0 > UART
0347 75B90  MOV  IE,  #90H ;enable serial interrupt
034A 029C  SETB  REN ;enable serial port reception
034C 200307  JB  ISW.3,  HARDOK ;check for hardware reset
034F 3B050  JNB  P3.5,  HARDOK ;goto ERROR1 if hardware error was detected
0352 B170  AJMP  ERROR1
0354 B15E  HARDOK:  AJMP  RPLY ;send reply in response to software RESET
0356 C203  HARDOK:  CLR  ISW.3 ;clear "hardware reset" flag
0358 B1F5  AJMP  WAIT ;wait for an interrupt

; SERIAL is an interrupt service routine for processing commands
; and data received from the central controller through the serial
; port.

035A 209B05  SERIAL:  JB  RI,  RECEIVE ;goto RECEIVE if there was a reception
035D C299  CLR  TI ;otherwise clear transmit flags and return
035F C204  CLR  ISW.4
0361 32  RETI
0362 D0E0  RECEIVE:  PUSH  ACC ;save the Acc.
0364 E599  MOV  A,  SBUF
0366 B4D02  CJNE  A,  #002H,  NOTRES ;check for Restart command
0369 2107  AJMP  TEST
036B 2BB536  NOTRES:  JB  P3.5,  JSRET ;ignore command if error flag is set
036E 3D000  JNB  P,  POK ;goto ERROR4 if the word has odd parity
0371 B17C  AJMP  ERROR4
0373 E521  POK:  MOV  A,  DSW ;goto DATA if expecting data
0375 7002  JNZ  DATA
0377 61F4  AJMP  COMMAND
0379 E599  DATA:  MOV  A,  SBUF ;decide what type of data was received
037B 20E62E  JB  ACC.4,  JUMP3
037D 30E528  JNB  ACC.5,  JUMP3
0380 10E429  JNB  ACC.4,  JUMP3
0384 540F  ANL  A,  #0FH ;isolate 4-bit data in low nibble of Acc.
0386 5F86  JNB  ACC.4,  JUMP3
0388 20F14  JB  DSW.7,  LHN ;Length High Nibble?
038A 20F14  JB  DSW.6,  LHN ;Length Low Nibble?
038C 20F25  JB  DSW.5,  XD ;X data?
038F 20C2A  JB  DSW.4,  YD ;Y data?
0392 20B30  JB  DSW.3,  ID ;I data?
SOURCE FILE NAME: SOURCE.ASM

0395 200A35  JB DSW.2, RFD ;R data?
0399 20094A  JB DSW.1, TSB ;TൽsSlice?
039B B170  AJMP ERROR1 ;never executed under proper operation
039D C4  LHN:  SWAP A
039E F534  MOV LENGTH, A
03A0 C20F  CLR DSW.7
03A2 D20E  SETB DSW.6
03A4 B16B  JSRET: AJMP SRET
03A6 4234  LLN:  ORL LENGTH, A
03AA E534  MOV A, LENGTH ;goto ERROR3 if (LENGTH) = 0
03AC B178  JUMP:  AJMP ERROR3
03AE C20E  LDK:  CLR DSW.6
03B0 D20D  SETB DSW.5
03B2 B16B  AJMP SRET
03B4 C4  ID:  SWAP A ;process X coordinate of data point
03B5 FF  MOV R7, A
03B6 C20D  CLR DSW.5
03B8 D20C  SETB DSW.4
03BA B16B  AJMP SRET
03BC 4F  YD:  ORL A, R7 ;process Y coordinate of data point
03BD F0  MOVX @DPTR, A ;save XY in external memory
03BE A3  INC DPTR
03BF C20C  CLR DSW.4
03C1 D20B  SETB DSW.3
03C3 B16B  AJMP SRET
03C5 C4  ID:  SWAP A ;process Z coordinate of data point
03C6 FF  MOV R7, A
03C7 C20B  CLR DSW.3
03C9 D20A  SETB DSW.2
03CA B16B  AJMP SRET
03CD 24F7  RFD:  ADD A, #247 ;process R intensity factor of data point
03CF 40BB  JC JUMP3 ;goto ERROR4 if R intensity factor > 8
03D1 94F7  SUBB A, #247
03D3 4F  ORL A, R7
03D4 F0  MOVX @DPTR, A ;save ZR in external memory
03D5 A3  INC DPTR
03D6 C20A  CLR DSW.2
03DB EA  MOV A, R2 ;goto INCDF if (LENGTH) not = (COUNT)
03D9 B53404  CJNE A, LENGTH, INCDF
03DC D206  SETB ISW.6 ;set flag and send reply if data is complete
03DE B15E  AJMP RPLY
03E0 0A  INCDF: INC R2 ;prepare for next data point
03E1 D20D  SETB DSW.5
03E3 B16B  AJMP SRET
03E5 44F0  TSD:  ORL A, #0F0H ;process TIME SLICE data
03E7 D3  SET C
03E8 13  RRC A ; 11111ABCD00000000 (ABCD=TSLICE data)
03E9 F523  MOV TSLICEN, A
03EA 752200  MOV TSLICE, #00H ; TSLICE= TSLICEL
03EE 9217  MOV TSLICEL.7, C
AVOCET SYSTEMS 8051 CROSS-ASSEMBLER - VERSION 1.09

SOURCE FILE NAME: SOURCE.ASM

03F0 C209 CLR DSW.1
03F2 B15E AJMP RPLY
03F4 E599 COMMAND: MOV A, SBUF ;decide what command was received
03F6 B44E02 CINE A, #4Eh, BEG ;No Operation?
03F9 B15E AJMP RPLY
03FB B44202 BEG: CINE A, #42h, ST ;Begin?
03FE B121 AJMP BEGIN
0400 B45302 ST: CINE A, #53h, DAT ;Stop?
0403 B158 AJMP STOP
0405 B45002 DAT: CINE A, #50h, TS ;Path?
0408 B15E AJMP PATH
040A B44042 TS: CINE A, #04h, IN ;Time Slice data?
040D B151 AJMP TSLICE
040F 5430 IN: ANL A, #30h ;an intensity factor?
0411 B43069 CINE A, #30h, ERROR2 ;not recognizable as a command
0414 E599 MOV A, SBUF
0416 540F ANL A, #0Fh ;mask off all but last 4 bits
0418 750F7 MOV B, #27 ;goto ERROR2 if factor is out of bound
041B 25F0 ADD A, B
041D 4055 JC ERROR2
041F B156 AJMP INTEN
0421 30065C BEGIN: JNB ISW.6, ERROR5 ;cannot begin without data for scanpath
0424 B523BE MOV TH0, TSLICEH ;initialize cycle of timer interrupts
0427 B5228A MOV TL0, TSLICEL
042A C2B0 CLR TF0
042C B249 SETB IE.1
042E B207 SETB ISW.7 ;prepare to begin scan with first data point
0430 7800 MOV R0, #00h
0432 7901 MOV R1, #01h
0434 AA34 MOV R2, LENGTH
0436 B15E AJMP RPLY
043B C249 STOP: CLR IE.1 ;disable timer interrupt cycle
043A C2B4 CLR PJ.4 ;turn off the antenna array
043C B15E AJMP RPLY
043E C2B4 PATH: CLR PJ.4 ;turn off the array
0440 C249 CLR IE.1 ;disable timer interrupt cycle
0442 752180 MOV DSW, #80h ;initialize for reception of a new data path
0445 C268 MOV ISW.6
0447 758200 MOV DPL, #00h
044A 75300 MOV DPH, #00h
044D 7A01 MOV R2, #01h
044F B168 AJMP SRET
0451 752102 TSLICE: MOV DSW, #02h ;set the Data Status Word
0454 B168 AJMP SRET
0456 859933 INTEN: MOV OBUF, SBUF ;store new D factor
0459 53330F ANL OBUF, #0Fh
045C B15E AJMP RPLY
045E 300405 RPLY: JNB ISW.4, SDK1 ;wait for completion of current transmission
0461 3099FA JNB TI, RPLY
0464 C299 CLR TI
0466 D204 SDK1: SETB ISW.4
AVOCET SYSTEMS 8051 CROSS-ASSEMBLER - VERSION 1.09

SOURCE FILE NAME: SOURCE.ASM

0468 7599C3  MOV  SBUF, #0C3H ;send "command execution completed" code
046B 00E0  SRET: POP  ACC ;restore Acc.
046D C29B  CLR  RI ;clear serial interrupt flag
046F 32  RETI

; These routines each transmit a unique error message to the central
; controller through the serial port. In addition, a routine
; disables the timer interrupt, turns off the antenna array, and sets
; an external error flag.

0470 745A  ERROR1: MOV  A, #5AH ;load Acc. with the type 1 error code
0472 81B4  AJMP  ERROR
0474 7459  ERROR2: MOV  A, #59H ;load Acc. with the type 2 error code
0476 81B4  AJMP  ERROR
0478 745B  ERROR3: MOV  A, #03B8H ;load Acc. with the type 3 error code
047A 81B4  AJMP  ERROR
047C 7407  ERROR4: MOV  A, #0D7H ;load Acc. with the type 4 error code
047E 81B4  AJMP  ERROR
0480 7456  ERROR5: MOV  A, #55H ;load Acc. with the type 5 error code
0482 81B4  AJMP  ERROR
0484 300405  ERROR: JNB  ISW.4, SDK2 ;wait for completion of current transmission
0487 3009FA  JNB  TI, ERROR
048A C299  CLR  TI
048C 9204  SDK2: SETB  ISW.4
048E F399  MOV  SBUF, A
0490 C2A9  CLR  IE.1 ;disable timer interrupts and counters
0492 C2B4  CLR  P3.4 ;turn off the array
0494 82B5  SETB  P3.5 ;set external error flag
0496 816B  AJMP  SRET

; TINT is an interrupt routine for repeatedly "scanning" a series of
; data points. The routine is initiated by TIMERO overflow at
; regular intervals (real time) when timer interrupt is enabled.
; Note: ON = (R0), OFF = (R1), and COUNT = (R2)

0498 85228A  TINT: MOV  TL0, TSLICEH ;reload the timer
049B 85230C  MOV  TH0, TSLICEH
049E C0E0  PUSH  ACC ;save the Acc.
04A0 30070B  JNB  ISW.7, ONOFF ;goto ONOFF unless new coordinates required
04A3 C284  CLR  P3.4
04A5 60  MOVX  A, @DPTR
04A6 853582  MOV  DPL, TEMPL
04A9 853583  MOV  DPH, TEMPH
04AC C207  CLR  ISW.7
04AE B80905  ONOFF: CJNE  RO, #00H, DECON ;goto DECON if (ON) not = 0
04B1 C284  CLR  P3.4
04B3 19  DEC  R1 ;decrement (OFF)
AVOCET SYSTEMS 8051 CROSS-ASSEMBLER - VERSION 1.09

SOURCE FILE NAME: SOURCE.ASM

0484 81B9 AJMP OPOFF
0486 0284 DECON: SETB P3.4
0488 18 DEC RO ; decrement (ON)
0489 8B ONOFF: MOV A, R0 ; goto ENDR if [(ON) + (OFF)] not = 0
048A 29 ADD A, R1
048B 7035 JNZ END
048D 0207 SETB ISW.7 ; set "new X,Y,Z" flag
048F EA MOV A, R2 ; goto LOADBUF if (COUNT) not = (LENGTH)
04C0 853406 CINE A, LENGTH, LOADBUF
04C3 E4 CLR A ; reset memory pointer to first data point
04C4 F583 MOV DPH, A
04C6 F5B2 MOV DPL, A
04CB FA MOV R2, A
04C9 E0 LOADBUF: MOVX A, @DPTR ; load XBUFF and IRBUF with next data point
04CA F531 MOV XBUFF, A
04CC A3 INC DPTR
04CD E0 MOVX A, @DPTR
04CE F532 MOV IRBUF, A
04D0 A3 INC DPTR
04D1 E0 INC R2 ; increment data point counter
04D3 E532 MOV A, IRBUF ; (ON) = 0 * R for next data point
04D4 S40F ANL A, $0FH
04D6 8533F0 MOV B, DBUF
04DA F3 MUL AB
04DB F6 MOV RO, A
04DC C3 CLR C
04DE 744A MOV A, $64 ; (OFF) = 64 - (ON)
04DF 9B SUBB A, R0
04E0 F9 MOV R1, A
04E0 858235 MOV TEMPL, DPL ; place 8XY Hex, in the DPTR
04E3 858236 MOV TEMPH, DPH
04E6 853192 MOV DPL, XBUFF
04E9 E532 MOV A, IRBUF
04EB C4 SWAP A
04EC S40F ANL A, $0FH
04EE 02E7 SETB ACC.7
04F0 F583 MOV DPH, A
04F2 D0E0 ENDR: POP ACC ; restore Acc.
04F4 32 RETI

;;
;; WAIT is a loop which is executed when waiting for an interrupt
;;
04F5 81F5 WAIT: AJMP WAIT

;;
0000 END
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Location</th>
<th>Description</th>
<th>Location</th>
<th>Description</th>
<th>Location</th>
<th>Description</th>
<th>Location</th>
<th>Description</th>
<th>Location</th>
<th>Description</th>
<th>Location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC</td>
<td>00E0</td>
<td>ERROR4</td>
<td>047C</td>
<td>LLN</td>
<td>03A6</td>
<td>RI</td>
<td>059B</td>
<td>THO</td>
<td>008C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>00F0</td>
<td>ERRORS</td>
<td>0480</td>
<td>LOADBUF</td>
<td>04C9</td>
<td>PPLY</td>
<td>045E</td>
<td>TH1</td>
<td>0020</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BES</td>
<td>03FB</td>
<td>HARD0</td>
<td>0354</td>
<td>LOK</td>
<td>03AE</td>
<td>SAD</td>
<td>019B</td>
<td>TI</td>
<td>0099</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BEGIN</td>
<td>0421</td>
<td>HARDR</td>
<td>0356</td>
<td>MEME</td>
<td>0189</td>
<td>SA02</td>
<td>02CE</td>
<td>TINT</td>
<td>049B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BYPASS</td>
<td>0341</td>
<td>IE</td>
<td>00A8</td>
<td>MEME2</td>
<td>027C</td>
<td>SBUF</td>
<td>0099</td>
<td>TLO</td>
<td>008A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMMAND</td>
<td>03F4</td>
<td>IN</td>
<td>040F</td>
<td>NOTRES</td>
<td>03B8</td>
<td>SC0N</td>
<td>009B</td>
<td>TMOD</td>
<td>0089</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONF16</td>
<td>0195</td>
<td>INCDP</td>
<td>03E0</td>
<td>QBUF</td>
<td>0333</td>
<td>SERIAL</td>
<td>035A</td>
<td>TS</td>
<td>040A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA</td>
<td>0405</td>
<td>INIT</td>
<td>031F</td>
<td>ONOFF</td>
<td>04AE</td>
<td>SDK1</td>
<td>0466</td>
<td>TSD</td>
<td>03E5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECON</td>
<td>0379</td>
<td>INT'N</td>
<td>0456</td>
<td>ONPOFF</td>
<td>04B9</td>
<td>SDK2</td>
<td>04EC</td>
<td>TSLICE</td>
<td>0451</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DPM</td>
<td>0486</td>
<td>IP</td>
<td>008B</td>
<td>P</td>
<td>00D0</td>
<td>SP</td>
<td>0081</td>
<td>TSLICEH</td>
<td>0023</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DPL</td>
<td>0082</td>
<td>IRAM</td>
<td>027E</td>
<td>P3</td>
<td>00B0</td>
<td>SRET</td>
<td>048B</td>
<td>TSLICEL</td>
<td>0022</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSW</td>
<td>0021</td>
<td>IRAME2</td>
<td>031D</td>
<td>PC0N</td>
<td>0087</td>
<td>STOP</td>
<td>043B</td>
<td>XD</td>
<td>0384</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENDR</td>
<td>04F2</td>
<td>ISW</td>
<td>0020</td>
<td>P6K0</td>
<td>0373</td>
<td>TCON</td>
<td>00BB</td>
<td>XYBUF</td>
<td>0031</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERR01</td>
<td>0484</td>
<td>JSRET</td>
<td>03A4</td>
<td>PSW</td>
<td>00D0</td>
<td>TEMPH</td>
<td>0036</td>
<td>YD</td>
<td>03B8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERROR2</td>
<td>047A</td>
<td>JUMP3</td>
<td>03AC</td>
<td>RECEIVE</td>
<td>0362</td>
<td>TEMPL</td>
<td>0035</td>
<td>ZD</td>
<td>03C5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERROR3</td>
<td>047F</td>
<td>LENGTH</td>
<td>0034</td>
<td>REN</td>
<td>009C</td>
<td>TEST</td>
<td>0107</td>
<td>ZIRBUF</td>
<td>0032</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SOURCE FILE NAME: SOURCE.ASM

---- SYMBOL TABLE ----
APPENDIX B

GENERATING EPROM LOOK-UP TABLES

This appendix contains a crude and simple program which will generate look-up tables for the EPROMs on the element driver board(s). It has been written in Apple-Basic, and can be executed on an Apple IIe computer which contains an 80-column display card and a disk drive. The file name is LUTG (short for: Look-Up-Table Generator).
10 REM THIS APPLE II PROGRAM WAS WRITTEN WITH THE AID OF JOHN MCCARTHY
12 REM AND DAVE PROGSITT.
13 REM
14 REM
15 REM
50 REM THE FOLLOWING PROGRAM CREATES THE LOOK-UP-TABLES FOR THE EPROMS
51 REM OF THE PHASED ARRAY CONTROLLER ON THE ELEMENT DRIVER BOARD(S).
52 REM THE PROGRAM CAN HANDLE UP TO 64 ELEMENTS. IT GENERATES A GROUP
53 REM OF BINARY FILES ON DISK WHICH CAN THEN BE USED TO BURN LOOK-UP-
54 REM TABLE EPROMS.
55 REM
56 REM THE FIRST PART OF THE PROGRAM PROMPTS THE USER FOR ALL
57 REM NECESSARY INFORMATION ABOUT THE ARRAY CONFIGURATION.
100 D$ = CHR$(4)
150 PRINT D$; "PR45"  
200 DIM F(16);I(16);X(16)
300 INPUT "NUMBER OF ELEMENTS ? "; N
350 T = INT(N)
355 IF T < > N THEN GOTO 30300
360 IF N < 1 THEN GOTO 30300
365 IF N > 64 THEN GOTO 30300
400 INPUT 'ELEMENT SPACING (MM) ? '; S
450 IF S < 0 THEN GOTO 30400
455 IF S > 50 THEN GOTO 30400
500 FOR H = 0 TO 15
510 PRINT
600 PRINT *Y*;H; " CORRESPONDS TO ? (KHI) *"
700 INPUT F(H)
710 PRINT
750 IF F(H) < 400 THEN GOTO 30000
755 IF F(H) > 1000 THEN GOTO 30000
800 NEXT H
900 PRINT "X AXIS DISTANCES RELATIVE TO THE CENTER OF THE ARRAY"
910 PRINT
1000 FOR H = 0 TO 15
1010 PRINT
1100 PRINT "I*;H* CORRESPONDS TO ? (MM)*"
1200 INPUT X(H)
1210 PRINT
1250 IF X(H) < ( - 100 THEN GOTO 30100
1255 IF X(H) > 100 THEN GOTO 30100
1300 NEXT H
1500 PRINT "I AXIS DISTANCES RELATIVE TO THE CENTER OF THE ARRAY"
1505 PRINT
1510 FOR H = 0 TO 15
1515 PRINT
1600 PRINT "I*;H* CORRESPONDS TO ? (MM)*"
1700 INPUT I(H)
1710 PRINT
1750 IF I(H) < 50 THEN GOTO 30200
1755 IF I(H) > 200 THEN GOTO 30200
1800 NEXT H
1900 HOME
1955 REM
1956 REM
1960 PRINT "NUMBER OF ELEMENTS IS "; N
1962 PRINT "ELEMENT SPACING IS "; S; " (MM)*"
2000 PRINT TAB(10);"X (IN MM)"; TAB(15);"Y (IN KHz)"; TAB(15);"Z (IN MM)"
2100 PRINT *
2200 FOR H = 0 TO 15
2300 PRINT TAB(10);X(H); TAB(19);F(H); TAB(20);Z(H)
2400 NEXT H
2500 INPUT "ARE THESE VALUES CORRECT? (Y/N)"; AS
2600 IF AS = "N" THEN GOTO 300
2700 IF AS = "Y" THEN GOTO 2900
2800 GOTO 2500
2840 REM
2850 REM THE FOLLOWING SERIES OF NESTED LOOPS WILL GENERATE THE BINARY
2851 REM LOOK-UP-TABLE FILES, AND STORE THEM ON THE DISK. THE FILES
2852 REM WILL ALL BE NAMED "ELEMENTS" WITH A POSTSCRIPT INDICATING
2853 REM WHICH PAIR OF ELEMENTS THE INDIVIDUAL FILES ARE FOR.
2854 REM
2855 REM THE DATA FOR ONE ELEMENT IS STORED IN THE UPPER NIBBLE OF
2856 REM THE X, Y, AND Z COORDINATE VALUES. THE DATA FOR THE NEXT
2857 REM ELEMENT FOR THE SAME COORDINATES IS STORED IN THE LOWER
2858 REM NIBBLE OF THE SAME ADDRESS.
2859 REM
2860 REM THE EQUATION USED TO CALCULATE THE BINARY VALUES IS:
2861 REM MOD 16((16*F(J)/1500 + SQRT((XI-I)*(XI-I)+I(K)*I(K))))
2862 REM WHERE (I,J,K) IS THE COORDINATE FOCUS, AND X(E) IS THE
2863 REM DISTANCE FROM THE ORIGIN TO THE CENTER OF ELEMENT # E.
2900 FOR H = 0 TO 15
2900 F(H) = F(H) * 16 / 1500
3100 Z(H) = Z(H) ^ 2
3200 NEXT H
3205 REM OFFSET IS THE DISTANCE FROM THE ORIGIN TO THE CENTER
3206 REM OF ELEMENT NUMBER 0
3210 OFFSET = (N - 1) * S / 2
3250 FOR E = 0 TO N - 1 STEP 2
3260 PRINT "GENERATING LOOK-UP-TABLE FOR ELEMENTS "; "E "; "E + 1"
3300 FOR I = 0 TO 15
3400 FOR J = 0 TO 15
3500 FOR K = 0 TO 15
4000 ARGUMENT = (((I) + OFFSET - (E * S)) ^ 2 + Z(K)
4100 HIGHNIBBLE = F(J) * SQRT(ARGUMENT)
4200 ARGUMENT = (((I) + OFFSET - ((E + 1) * S)) ^ 2 + Z(K)
4300 LOWNIBBLE = F(J) * SQRT(ARGUMENT)
4400 TEMP = LOWNIBBLE
4500 GSUB 20000
4600 BUFFER = TEMP
4700 TEMP = HIGHNIBBLE
4800 GSUB 20000
4900 BUFFER = (TEMP * 16) + BUFFER
4950 REM THE ADDRESS FOR DATA WITHIN THE LOOK-UP-TABLE IS
4952 REM OF THE FORM ZXY
5000 POKE 16384 + (256 * K) + (16 * I) + J + BUFFER
5002 PRINT I;J;K; "*,BUFFER
5100 NEXT K
5200 NEXT J
5300 NEXT I
5400 I$ = STR$(I)
5500 L$ = STR$(E + 1)
5500 PRINT 0;"BSAVE ELEMENTS ";L$;","L$","I$","E","A#4000,L#1000"
19499 PRINT CHR$ (12); CHR$ (21)
19500 END
19900 REM THIS SUBROUTINE PERFORMS A MOD 16 FUNCTION ON THE
19901 REM CONTENTS OF TEMP. THE RESULTING 4 BIT VALUE IS RETURNED
19902 REM IN THE LOW NIBBLE OF TEMP.
20000 D = INT (TEMP / 16)
20100 TEMP = TEMP - (D * 16)
20200 TEMP = INT (TEMP)
20300 RETURN
30000 PRINT "VALUE MUST BE BETWEEN 400 AND 1000, TRY AGAIN"
30010 PRINT
30020 GOTO 600
30100 PRINT "VALUE MUST BE BETWEEN -100 AND 100, TRY AGAIN"
30110 PRINT
30120 GOTO 1100
30200 PRINT "VALUE MUST BE BETWEEN 50 AND 200, TRY AGAIN"
30210 PRINT
30220 GOTO 1600
30300 PRINT "PLEASE ENTER INTEGER VALUE BETWEEN 0 AND 64"
30310 PRINT
30320 GOTO 300
30400 PRINT "ELEMENT SPACINGS MUST BE BETWEEN 0 AND 50 MM., TRY AGAIN"
30410 PRINT
30420 GOTO 400
REFERENCES


