An Application of Speech Synthesis

1984

Edward J. Purvis

University of Central Florida

Find similar works at: https://stars.library.ucf.edu/rtd

University of Central Florida Libraries http://library.ucf.edu

Part of the Engineering Commons

STARS Citation

https://stars.library.ucf.edu/rtd/4650

This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of STARS. For more information, please contact lee.dotson@ucf.edu.
AN APPLICATION OF SPEECH SYNTHESIS

BY

EDWARD JOHN PURVIS
B.S.E., University of Central Florida, 1981

THESIS

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering in the Graduate Studies Program of the College of Engineering University of Central Florida Orlando, Florida

Spring Term
1984
ABSTRACT

This paper discusses the implementation of a synthesized computer voice facility for a missile trainer. The actual facility hardware is described. Software used to control the hardware operation is presented and discussed.
ACKNOWLEDGEMENTS

First, I would like to thank the people who have helped me, in one way or another, in writing this thesis: Mr. Albert Marshall for allowing me to use the work done on one of his training devices as the basis for my thesis; Mr. Bon Shaw and Mr. Gary Bond for helping me debug the hardware; Dr. Forrest S. Mozer and Mr. Joseph B. Costello for their help in obtaining the information needed on National Semiconductor's "Digitalker". And I would like to thank the members of my committee: Dr. Robert Walker, Dr. Brian Petraske, and especially Dr. Herbert Towle for their time and patience.

I also need to acknowledge that certain registered trademarks are used in this paper. They are as follows: Multibus, iSBC 86/12A, SBC 86/12A and UPI 41A are trademarks of the Intel Corporation; Digitalker is a trademark of the National Semiconductor Corporation.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>CIRCUITS (HARDWARE)</td>
<td>5</td>
</tr>
<tr>
<td>Interface</td>
<td>11</td>
</tr>
<tr>
<td>Synthesizer</td>
<td>12</td>
</tr>
<tr>
<td>Speech ROMs</td>
<td>13</td>
</tr>
<tr>
<td>Address Decode</td>
<td>13</td>
</tr>
<tr>
<td>Filter</td>
<td>13</td>
</tr>
<tr>
<td>Controlling Processor</td>
<td>15</td>
</tr>
<tr>
<td>SOFTWARE</td>
<td>18</td>
</tr>
<tr>
<td>Interface</td>
<td>18</td>
</tr>
<tr>
<td>Controlling Processor</td>
<td>21</td>
</tr>
<tr>
<td>SYSTEM INTEGRATION</td>
<td>27</td>
</tr>
<tr>
<td>CONCLUSIONS</td>
<td>29</td>
</tr>
<tr>
<td>APPENDIX A. STANDARD VOCABULARY ONE</td>
<td>30</td>
</tr>
<tr>
<td>APPENDIX B. STANDARD VOCABULARY TWO</td>
<td>33</td>
</tr>
<tr>
<td>APPENDIX C. CUSTOM VOCABULARY</td>
<td>36</td>
</tr>
<tr>
<td>APPENDIX D. VOCABULARY ADDRESSES</td>
<td>38</td>
</tr>
<tr>
<td>APPENDIX E. INTERFACE PROGRAM</td>
<td>46</td>
</tr>
<tr>
<td>APPENDIX F. SCREEN EDITOR PROGRAM</td>
<td>51</td>
</tr>
<tr>
<td>LIST OF REFERENCES</td>
<td>72</td>
</tr>
</tbody>
</table>
A TOW missile simulator was built as a training device by the Naval Training Equipment Center (NTEC) (Marshall et al., 1983). As part of a contract between University of Central Florida and NTEC a Computer Voice Board was designed to provide a computer synthesized voice for the simulator. A screen editor program was also written, which allows a word or phrase to be entered from a keyboard. The computer voice board will speak the word or phrase upon command. The word or phrase may then be edited and spoken again until it is correct.

The computer voice board accomplishes two major purposes: First, the simulator uses it to issue commands which would normally be given by the squad leader. These include the command to fire the missile, which synchronizes the gunner trainee with the training scenario. Second, the computer voice provides interactive coaching.
The TOW is a wire guided missile. It must be flown to the target by the gunner. To do this, the gunner looks through a telescopic sight and keeps the crosshairs on the target. The simulator provides interactive coaching, when enabled, by saying "high", "low", "left", or "right" as appropriate.

The simulator also provides immediate debriefing at the end of the flight by saying "hit" or "miss". If a miss occurs, the simulator says by how much and in which direction.

National Semiconductor's "Digitalker" is a chip set consisting of a speech synthesizer chip and one or more ROMs. It was chosen for three reasons: one, it was available; two, "Digitalker" had the necessary vocabulary capability; and three, it had a high quality voice output. The algorithm used by "Digitalker" to synthesize speech is proprietary to National Semiconductor; but a brief description is given in the patent document (Mozer and Stauduhar, 1980). A detailed description of how the computer voice board is built using "Digitalker" is contained in the body of this paper.

The screen editor program was written to allow words and phrases to be entered and edited from a console. This program has a number of functions.
First, it provided a good stand-alone test of the computer speech board. This was important, because it allowed the board to be built and tested before it was integrated into the simulator. This greatly simplified debugging of the board during construction and testing.

Secondly, this program allows the board to be demonstrated independent of the simulator.

Third, if the board ever needs repair, this program will allow the repair and testing to be done independent of the simulator.

Fourth, programmers writing code to interface the computer voice board into the simulator can design and test the phrases that the simulator is to speak, before adding them to their code. This is useful because the code for this simulator is based in EPROMs. It would be a very time consuming process to recompile, relink, and relocate the code and burn a new set of EPROMs every time to fine-tune these phrases.

The computer speech board used in the simulator is the same board used in the stand-alone system, which consists of: (1) the computer voice board, (2) an Intel iSBC 86/12A board as the controlling processor running the screen editor program in EPROM, (3) an ADM-3A terminal, (4) a speaker, and (5) a cardcage with power
supply. It is the stand-alone system that is described in this paper.
The central piece of hardware is the MM54104 "Digitalker" speech synthesis chip from National Semiconductor. This chip along with one or more ROMs to store the vocabulary and a bandpass filter are the minimal configuration of a "Digitalker" system.

The simulator into which this computer speech board was eventually to be integrated was built using Intel's Multibus. The computer speech board was built to be a memory-mapped I/O device on the Multibus.

There are five principle blocks on this board (see figure 1). First is an Intel 8741 Universal Peripheral Interface (UPI). Second is the synthesizer chip. Third is the ROM memory. Fourth is the address decode logic. And fifth is the bandpass filter. Each of these blocks will be described below. Figures 2 through 5 contain the detailed circuit diagrams for the board described below.
Figure 1. Block Diagram
Figure 2. Component Layout
Figure 3. Schematic Diagram, page 1.
Figure 4. Schematic Diagram, page 2.
Figure 5. Schematic Diagram, page 3.
Interface. The 8741 Universal Peripheral Interface performs several functions. As the name implies, this chip is designed to interface complex digital devices to an Intel CPU and/or bus. The "Digitalker" is not directly compatible with Intel's bus system. Therefore, one of the jobs of the 8741 is to translate the bus control lines of Intel's Multibus to those of the National Semiconductor chip.

The second function of the 8741 is to provide some of the address decode for the vocabulary ROMs. This is necessary because the "Digitalker" does not have enough address lines to address the memory alone. The 8741, therefore, provides the upper address lines to select the proper page of memory for the "Digitalker."

The 8741 once had another function which, for reasons outlined below, is not currently implemented. As an experiment, a programmable gain operational amplifier was added to the circuit between the bandpass filter and the speaker. The idea was to add more inflection or emphasis to certain words by increasing their volume. The volume control worked quite well; but, it did not achieve its intended purpose of adding emphasis. It merely raised the volume. This did not change the inflection and the result sounded unnatural.
There was a second problem with the programmable gain amplifier. The chip functioned well; but, because it was implemented in CMOS there were frequent failures caused by DC voltage levels on the input and output lines. Since the programmable gain amplifier did not achieve its intended purpose, it was decided to remove the chip from the circuit rather than redesign the board to provide protection for it. The software, which will be described later, still sends the volume control information to the amplifier. However, the control lines on the 8741 which performed this function are not connected. This does not cause any problems because this is an open-loop operation. The software was not modified because of time constraints.

**Synthesizer.** The synthesizer chip, as previously noted is a National Semiconductor MM54104 Speech Synthesizer Chip. When given the address of the beginning of a word, phrase or sentence stored in ROM the "Digitalker" will read the stored data from the ROM and produce an analog output. The "Digitalker" requires a power supply connection of between seven and eleven volts. Since this is not readily available the board contains a uA723 voltage regulator which produces
approximately 7.2 volts from the 12 volt supply provided by the Multibus.

**Speech ROMs.** There are seven ROMs on the board containing three separate sets of vocabulary words. Each set provides a separate page of memory for the "Digitalker." The first set, page 0, labeled SSR1 and SSR2, is part of a DT1050 "Digitalker" Standard Vocabulary Kit which also includes the synthesizer chip. The second set, page 1, labeled SSR5 and SSR6, is a DT1057 "Digitalker" Standard Vocabulary Kit. The third set, page 2, labeled 2716, contains a custom vocabulary purchased from National Semiconductor.

**Address Decode.** The address decode logic for the speech ROMs is provided partly by the software in the 8741 UPI, as previously mentioned, and partly by a 74139 dual 2-to-4 line decoder and three inverters from a 7404. The address decode for the 8741 UPI is provided by a 7425, a 74125, one inverter from a 7404, and two NAND gates from a 7400. See figures 3 and 4.

**Filter.** The bandpass filter is constructed from a TL074 quad op-amp chip and a number of discrete resistors and capacitors. This filter was constructed
as prescribed in National Semiconductor's Application Note 252 (Smith and Weinrich, 1980) and DT1050 "Digitalker" Standard Vocabulary Kit manual (National Semiconductor, 1980) to have a low frequency cutoff of 200 Hz and a high frequency cutoff of 7 kHz. The component values were adjusted to give a response which more closely matched the curve given in Smith and Weinrich (see figure 6).

It can be shown that the circuit in figure 5 will give a response very similar to the curve in figure 6. It should be noted that this is a five pole open loop filter. An S-domain analysis was done where the poles were calculated as:

1. \( s = 2\pi/RC \).

This yields the equation:

\[ A = \frac{s^2}{a*b*c*d*e}, \]

where:

1. \( a = s + 7957.747 \),
2. \( b = s + 7957.747 \),
3. \( c = s + 194.09 \),
4. \( d = s + 88.223 \), and
5. \( e = s + 159.155 \).

Combining equations 2 through 7, substituting \( s = j\pi \), and taking the magnitude yields:

\[ A = \frac{\pi^2}{D} \]
(9) \[ D = (f_t^2 + 7957.747^2) \times (f_t^2 + 194.02^2)^{0.5} \times \\
(f_t^2 + 88.223^2)^{0.5} \times (f_t^2 + 159.155^2)^{0.5}. \]

This equation was plotted using an IBM PC to yield figure 7, which is a fair approximation of the curve in figure 6.

**Controlling Processor.** The controlling processor, which will run the editor software is an Intel iSBC 86/12 board. The same type of board will be the controller in the simulator. The simulator control board however will contain a much different program.
Figure 6. Bandpass Filter Function (Smith and Weinrich, 1980) Copied with permission.
Figure 7. Bandpass Filter Function
SOFTWARE.

Two pieces of software are provided to program the hardware. First the 8741 Universal Peripheral Interface must be programmed; and, second, the controlling processor must be programmed.

**Interface.** The 8741 UPI program will be discussed first. Since this is an assembly language program, a flowchart is included in figure 8. A listing of this program is provided in Appendix E.

The first major problem encountered with the "Digitalkor" is that the control lines are incompatible with Intel's bus system. That is the purpose of adding the 8741 interface. The 8741 receives data from the Intel bus and generates the proper National Semiconductor control signals to transfer the data to the "Digitalkor."

The next problem encountered is that the "Digitalkor" does not have a hardware reset line; nor does it have a simple software reset command. Therefore
Figure 8. Flowchart of 8741 program.
the first thing that the 8741 must do on power-up is to bring the "Digitalker" to a known state. The gain must also be set to a known state.

First, the 8741 initializes the gain to unity. As mentioned previously, the programmable gain hardware is no longer in place.

Second, the "Digitalker" is interrupted. This is the closest thing to a reset available. This interrupt halts all speech output.

Third, the "Digitalker" is commanded to output a period of silence. This is necessary because the "Digitalker" maintains its busy/ready line in the busy state after an interrupt. By outputting a period of silence the "Digitalker" is caused to output a ready signal.

Fourth, the 8741 sets all of it's "Digitalker" control lines to the proper initial state and waits for the controlling processor to transmit data to it for the "Digitalker."

The 8741 is programmed to receive data in three-byte packets. The first byte is the "page" in ROM where the word is located. The second byte is the address of the word, or phrase, to be spoken. The third byte is the gain. The gain must be sent to the 8741 even though
the gain control chip is not currently present on the board.

Upon receipt of a data packet, the 8741 outputs the "page" onto the proper address lines; outputs the gain, and then sends the word address to the "Digitalker". Then the 8741 waits for the next packet.

Controlling Processor. The programming of the controlling processor will now be discussed. This program must send data packets consisting of three bytes; the page, word, and gain. That is the only absolute requirement. Please, note however, that it is very important to send a complete packet every time. If this is not done, the 8741 will lose its place, and all output from that point onward will be garbled. That is, the "Digitalker" will more than likely produce some sound. This sound, however, may or may not resemble speech and will almost certainly not be the desired words. A screen editor program has been written and will be described in the following paragraphs.

The screen editor program allows the operator to type page, word, and, optionally, volume into a line buffer at a console. The buffer may be edited. The program will send the buffer to the 8741, and therefore to the "Digitalker," as many times as desired. First, a
description of the editor is given and then the program is described in detail.

When the system is turned on or reset, the screen editor program outputs a dollar sign prompt and waits for an input.

At this point, the operator can enter the word packets for the word or words that he would like the "Digitalker" to speak. The words are listed in Appendixes A, B and C. Appendix D lists the page and word addresses for each word. The format for the packet is page (0, 1, or 2 in this system), followed by the word address (no spaces or other delimiters) and optionally followed by an "L" and the gain. The gain should be 0, 1, 2, or 3, where zero is unity gain. Commas or spaces should be used as delimiters between packets. The line terminates with a carriage return.

To repeat a previously spoken word or phrase, simply type "R" when not in the edit mode. To enter the edit mode type ctrl-P by holding down the ctrl (control) button and typing "P". The ctrl-P command will print the contents of the buffer and enter the edit mode. The edit control commands are as follows:

- **ctrl-H**: moves the cursor back one space
- **ctrl-L**: moves the cursor forward one space
ctrl-D: deletes the character under the cursor
ctrl-C: inserts a character to the left of the cursor

ctrl-A: prompts with an "_" and allows the insertion of text. Escape from insertion with a second ctrl-A.

ctrl-Z: prompts with an "@" and deletes all the text between the "@"s. The prompt will move with the cursor. Escape from deletion with a second ctrl-Z.

ctrl-X: terminates the line at the cursor
CR: a carriage return exits the edit mode.

Also note:
Shift-RUB is the rubout or delete function when not in the edit mode.

Typing an "R" when not in the edit mode will cause the previous buffer to be repeated.

When in the ctrl-A mode of the edit mode, typing more characters than the 70 character buffer can hold will automatically exit the ctrl-A mode.

The listing for the screen editor program is in Appendix F. A brief description of each part of the program is provided below.

On power-up or reset, the controlling computer sets up its serial communication port for talking to the terminal and waits for the 8741 to initialize the "Digitalker" speech board. Then the program enters an infinite loop which outputs a prompt, accepts inputs,
outputs the contents of the buffer to the screen, and finally sends the properly translated buffer to the "Digitalker."

It should be noted that this program was written to run on an ADM-3A terminal and is designed to take its characteristics into account.

The subroutine CI inputs a character from the terminal. If this procedure is called, a character must be typed in order to continue because CI waits for a character with no timeouts.

The subroutine CO outputs a character to the screen.

The subroutine SCRUB removes unwanted data from the buffer. This is necessary to prevent unwanted speech or sounds from being appended to the end of a word or phrase that is shorter than the previous word or phrase.

The subroutine PRNT prints the contents of the buffer to the screen.

The subroutine CBCK moves the cursor back one space while in the edit mode.

The subroutine CFWD moves the cursor forward one space while in the edit mode.

The subroutine DEL_EX deletes the character under the cursor while in the edit mode.
The subroutine **IN_DAT** filters the data coming from the terminal keyboard and rejects illegal characters.

The subroutine **INS_EX** inserts a character into the buffer to the left of the cursor when the program is in the edit mode.

The subroutine **CNTRL_A_EX** inputs characters from the terminal and inserts them into the buffer at the position of the cursor until a control A is encountered or until the buffer is full, whichever comes first.

The subroutine **CNTRL_Z_EX** will set the start point for a string delete and allow the cursor to be positioned forward to the end point of the string delete. Then the characters between these two points are deleted.

The subroutine **INPT** accepts characters from the terminal keyboard. This procedure also controls editing. If an edit control is encountered and the edit mode flag is set then the proper subroutine is called to edit the buffer. If not in the edit mode, and if the character typed is a legal character then it is placed in the buffer. If an "R" is encountered in the first position, when not in the edit mode, then the previous buffer is sent to the "Digitalker" board again.

The subroutine **OUTPT** sends the contents of the buffer to the terminal screen.
The subroutine TRANSLATE takes the contents of the input buffer and translates it into the packet form required by the "Digitalker" board. If no gain is specified, then this procedure will insert a gain of one.

The subroutine SPEECH sends a character to the 8741, which will send it to the "Digitalker" as appropriate.
The computer voice board will interface to any controller that communicates over the Multibus. The 8741 Universal Peripheral Interface on this board is memory mapped at OD000H and OD002H. The former is the Data address and the latter is the Status address of the 8741 UPI.

Before the controlling computer can send a byte of data to the computer voice board, it must first check the status of the 8741 interface. This is done by reading the status byte from the Status address and checking bit number one, with the byte numbered from zero to seven from right to left. It should be noted that the Multibus inverts all data transmitted on it. Therefore, the status byte that is received by a status read is inverted. The flag is set to one when the Input Buffer of the 8741 is full and reset to zero when it is empty. Recall, however, that the status byte is inverted by the Multibus. Therefore, the controlling processor should look for bit one to be a logical one
before sending a byte of data to the Data address. The status must be checked before each byte is sent. The other seven bits of the Status byte are not utilized by this application.

As mentioned in the introduction to this paper, this board has been integrated into a training device prototype as described by Marshall et al. (1983, p. 29).
CONCLUSIONS

The Computer Voice Board built using the "Digitalker" was integrated into the missile simulator and performed well. The voice is clear and easy to understand, and it sounds fairly natural. The computer voice facility has enhanced the function of the trainer by providing interactive coaching during the missile flight. It has also freed the instructor from the tedium of repeating the same phrases over and over. The facility also provides some immediate debriefing after the missile flight. The instructor is still important, however, to provide guidance and additional debriefing not yet programmed into the training device.
This appendix contains a list of the words found in National Semiconductor's DT1050 "Digitalkit" Standard Vocabulary Kit (National Semiconductor, 1980).
THIS IS DIGITALKER
ONE
TWO
THREE
FOUR
FIVE
SIX
SEVEN
EIGHT
NINE
TEN
ELEVEN
TWELVE
THIRTEEN
FOURTEEN
FIFTEEN
SIXTEEN
SEVENTEEN
EIGHTEEN
NINETEEN
TWENTY
THIRTY
FORTY
FIFTY
SIXTY
SEVENTY
EIGHTY
NINETY
HUNDRED
THOUSAND
MILLION
ZERO
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
AGAIN
AMPERE
AND
AT
CANCEL
CASE
CENT
400HZ TONE
80HZ TONE
20MS SILENCE
40MS SILENCE
80MS SILENCE
160MS SILENCE
320MS SILENCE
CENTI
CHECK
COMMA
CONTROL
DANGER
DEGREE
DOLLAR
DOWN
EQUAL
ERROR
FEET
FLOW
FUEL
GALLON
GO
GRAM
GREAT
GREATER
HAVE
HIGH
HIGHER
HOUR
IN
INCHES
IS
IT
KILO
LEFT
LESS POINT
LESSER POUND
LIMIT PULSES
LOW RATE
LOWER RE
MARK READY
METER RIGHT
MILE SS (Note 1)
MILLI SECOND
MINUS SET
MINUTE SPACE
NEAR SPEED
NUMBER STAR
OF START
OFF STOP
ON THAN
OUT THE
OVER TIME
PARENTHESIS TRY
OF UP
OVER VOLT
NUMBER WEIGHT
OF PLUS
OVER
NUMBER
OF
NOTE 1: "SS" makes any singular word plural.
This appendix contains a list of the words found in National Semiconductor's DT1057 "Digitalker" Standard Vocabulary Kit (National Semiconductor, 1981).
Note 1: "ED" is a suffix that can be used to make any present tense word become a past tense word. The way that "ED" is said, however, varies from word to word. The first two should be used with words ending with "T" or "D"; the third should be used with words ending in soft sounds, such as "ask"; and, the fourth should be used with all other words.

Note 2: "TH" is a suffix that can be used to form words like seventh, eighth and ninth from words such as "seven", "eight", and "nine".

Note 3: "UTH" is a suffix that can be used to form words like thirtieth and fortieth form words such as "thirty and forty".
APPENDIX C

CUSTOM VOCABULARY

This appendix contains the custom vocabulary used in the training system that this device is built for. This vocabulary was obtained from National Semiconductor.
TANK
DEPTH
DAY
COMMAND
USE
AT
FIRE
ALERT
NIGHT
NORTH
SOUTH
EAST
WEST
THERMAL
CEASE
FRONTAL
TRACKING
SIGHT
MISS
HIT
CROSS
MY
TARGET
TRANSFER
SQUAD
OPTICAL
APPENDIX D

VOCABULARY ADDRESSES

This appendix contains the entire "Digitalker" vocabulary listed in alphabetical order along with page and word addresses. Note that some words are duplicated for reasons best described as "historical."
<table>
<thead>
<tr>
<th>WORD</th>
<th>PAGE--ADDRESS--WORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>160MS SILENCE</td>
<td>0 46H</td>
</tr>
<tr>
<td>20MS SILENCE</td>
<td>0 43H</td>
</tr>
<tr>
<td>320MS SILENCE</td>
<td>0 47H</td>
</tr>
<tr>
<td>400HZ TONE</td>
<td>0 41H</td>
</tr>
<tr>
<td>40MS SILENCE</td>
<td>0 44H</td>
</tr>
<tr>
<td>80HZ TONE</td>
<td>0 42H</td>
</tr>
<tr>
<td>80MS SILENCE</td>
<td>0 45H</td>
</tr>
<tr>
<td>A</td>
<td>0 20H</td>
</tr>
<tr>
<td>ABORT</td>
<td>1 00H</td>
</tr>
<tr>
<td>ADD</td>
<td>1 01H</td>
</tr>
<tr>
<td>ADJUST</td>
<td>1 02H</td>
</tr>
<tr>
<td>AGAIN</td>
<td>0 3AH</td>
</tr>
<tr>
<td>ALARM</td>
<td>1 03H</td>
</tr>
<tr>
<td>ALERT</td>
<td>1 04H</td>
</tr>
<tr>
<td>ALERT</td>
<td>2 07H</td>
</tr>
<tr>
<td>ALL</td>
<td>1 05H</td>
</tr>
<tr>
<td>AMPERE</td>
<td>0 3BH</td>
</tr>
<tr>
<td>AND</td>
<td>0 3CH</td>
</tr>
<tr>
<td>ASK</td>
<td>1 06H</td>
</tr>
<tr>
<td>ASSISTANCE</td>
<td>1 07H</td>
</tr>
<tr>
<td>AT</td>
<td>0 3DH</td>
</tr>
<tr>
<td>AT</td>
<td>2 05H</td>
</tr>
<tr>
<td>ATTENTION</td>
<td>1 08H</td>
</tr>
<tr>
<td>B</td>
<td>0 21H</td>
</tr>
<tr>
<td>BRAKE</td>
<td>1 09H</td>
</tr>
<tr>
<td>BUTTON</td>
<td>1 0AH</td>
</tr>
<tr>
<td>BUY</td>
<td>1 0BH</td>
</tr>
<tr>
<td>C</td>
<td>0 22H</td>
</tr>
<tr>
<td>CALL</td>
<td>1 0CH</td>
</tr>
<tr>
<td>CANCEL</td>
<td>0 3EH</td>
</tr>
<tr>
<td>CASE</td>
<td>0 3FH</td>
</tr>
<tr>
<td>CAUTION</td>
<td>1 0DH</td>
</tr>
<tr>
<td>CEASE</td>
<td>2 0EH</td>
</tr>
<tr>
<td>CENT</td>
<td>0 40H</td>
</tr>
<tr>
<td>CENTI</td>
<td>0 46H</td>
</tr>
<tr>
<td>CHANGE</td>
<td>1 0EH</td>
</tr>
<tr>
<td>CHECK</td>
<td>0 49H</td>
</tr>
<tr>
<td>CIRCUIT</td>
<td>1 0FH</td>
</tr>
<tr>
<td>CLEAR</td>
<td>1 10H</td>
</tr>
<tr>
<td>CLOSE</td>
<td>1 11H</td>
</tr>
<tr>
<td>COMMA</td>
<td>0 4AH</td>
</tr>
<tr>
<td>COMMAND</td>
<td>2 03H</td>
</tr>
<tr>
<td>COMPLETE</td>
<td>1 12H</td>
</tr>
<tr>
<td>CONNECT</td>
<td>1 13H</td>
</tr>
<tr>
<td>CONTINUE</td>
<td>1 14H</td>
</tr>
<tr>
<td>CONTROL</td>
<td>0 4BH</td>
</tr>
<tr>
<td>COPY</td>
<td>1 15H</td>
</tr>
<tr>
<td>CORRECT</td>
<td>1 16H</td>
</tr>
<tr>
<td>CROSS</td>
<td>2 14H</td>
</tr>
<tr>
<td>WORD</td>
<td>PAGE--ADDRESS--WORD</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
</tr>
<tr>
<td>DANGER</td>
<td>0</td>
</tr>
<tr>
<td>DATE</td>
<td>1</td>
</tr>
<tr>
<td>DAY</td>
<td>2</td>
</tr>
<tr>
<td>DECREASE</td>
<td>1</td>
</tr>
<tr>
<td>DEGREE</td>
<td>0</td>
</tr>
<tr>
<td>DEPOSIT</td>
<td>1</td>
</tr>
<tr>
<td>DEPTH</td>
<td>2</td>
</tr>
<tr>
<td>DIAL</td>
<td>1</td>
</tr>
<tr>
<td>DIVIDE</td>
<td>1</td>
</tr>
<tr>
<td>DOLLAR</td>
<td>0</td>
</tr>
<tr>
<td>DOOR</td>
<td>1</td>
</tr>
<tr>
<td>DOWN</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
</tr>
<tr>
<td>EAST</td>
<td>1</td>
</tr>
<tr>
<td>EAST</td>
<td>2</td>
</tr>
<tr>
<td>ED</td>
<td>1</td>
</tr>
<tr>
<td>ED</td>
<td>1</td>
</tr>
<tr>
<td>ED</td>
<td>1</td>
</tr>
<tr>
<td>ED</td>
<td>1</td>
</tr>
<tr>
<td>EIGHT</td>
<td>0</td>
</tr>
<tr>
<td>EIGHTEEN</td>
<td>0</td>
</tr>
<tr>
<td>EIGHTY</td>
<td>0</td>
</tr>
<tr>
<td>ELEVEN</td>
<td>0</td>
</tr>
<tr>
<td>EMERGENCY</td>
<td>1</td>
</tr>
<tr>
<td>END</td>
<td>1</td>
</tr>
<tr>
<td>ENTER</td>
<td>1</td>
</tr>
<tr>
<td>ENTRY</td>
<td>1</td>
</tr>
<tr>
<td>EQUAL</td>
<td>0</td>
</tr>
<tr>
<td>ER</td>
<td>1</td>
</tr>
<tr>
<td>ERROR</td>
<td>0</td>
</tr>
<tr>
<td>EVACUATE</td>
<td>1</td>
</tr>
<tr>
<td>EXIT</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
</tr>
<tr>
<td>FAIL</td>
<td>1</td>
</tr>
<tr>
<td>FAILURE</td>
<td>1</td>
</tr>
<tr>
<td>FARAD</td>
<td>1</td>
</tr>
<tr>
<td>FAST</td>
<td>1</td>
</tr>
<tr>
<td>FASTER</td>
<td>1</td>
</tr>
<tr>
<td>FEET</td>
<td>0</td>
</tr>
<tr>
<td>FIFTEEN</td>
<td>0</td>
</tr>
<tr>
<td>FIFTH</td>
<td>1</td>
</tr>
<tr>
<td>FIFTY</td>
<td>0</td>
</tr>
<tr>
<td>FIRE</td>
<td>1</td>
</tr>
<tr>
<td>FIRE</td>
<td>2</td>
</tr>
<tr>
<td>FIRST</td>
<td>1</td>
</tr>
<tr>
<td>FIVE</td>
<td>0</td>
</tr>
<tr>
<td>FLOOR</td>
<td>1</td>
</tr>
</tbody>
</table>

PAGE--ADDRESS--WORD

DANGER 0 4CH
DATE 1 17H
DAY 1 18H
DAY 2 02H
DECREASE 1 19H
DEGREE 0 4DH
DEPOSIT 1 1AH
DEPTH 2 01H
DIAL 1 1BH
DIVIDE 1 1CH
DOLLAR 0 4EH
DOOR 1 1DH
DOWN 0 4FH
E 0 24H
EAST 1 1EH
EAST 2 0BH
ED 1 1FH
ED 1 20H
ED 1 21H
ED 1 22H
EIGHT 0 0BH
EIGHTEEN 0 12H
EIGHTY 0 1AH
ELEVEN 0 0BH
EMERGENCY 1 23H
END 1 24H
ENTER 1 25H
ENTRY 1 26H
EQUAL 0 50H
ER 1 27H
ERROR 0 51H
EVACUATE 1 28H
EXIT 1 29H
F 0 25H
FAIL 1 2AH
FAILURE 1 2BH
FARAD 1 2CH
FAST 1 2DH
FASTER 1 2EH
FEET 0 52H
FIFTEEN 0 0FH
FIFTH 1 2FH
FIFTY 0 17H
FIRE 1 30H
FIRE 2 06H
FIRST 1 31H
FIVE 0 05H
FLOOR 1 32H
<table>
<thead>
<tr>
<th>WORD</th>
<th>PAGE -- ADDRESS -- WORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOW</td>
<td>0</td>
</tr>
<tr>
<td>FORTY</td>
<td>0</td>
</tr>
<tr>
<td>FORWARD</td>
<td>1</td>
</tr>
<tr>
<td>FOUR</td>
<td>0</td>
</tr>
<tr>
<td>FOURTEEN</td>
<td>0</td>
</tr>
<tr>
<td>FROM</td>
<td>1</td>
</tr>
<tr>
<td>FRONTAL</td>
<td>2</td>
</tr>
<tr>
<td>FUEL</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
</tr>
<tr>
<td>GALLON</td>
<td>0</td>
</tr>
<tr>
<td>GAS</td>
<td>1</td>
</tr>
<tr>
<td>GET</td>
<td>1</td>
</tr>
<tr>
<td>GO</td>
<td>0</td>
</tr>
<tr>
<td>GOING</td>
<td>1</td>
</tr>
<tr>
<td>GRAM</td>
<td>0</td>
</tr>
<tr>
<td>GREAT</td>
<td>0</td>
</tr>
<tr>
<td>GREATER</td>
<td>0</td>
</tr>
<tr>
<td>H</td>
<td>0</td>
</tr>
<tr>
<td>HALF</td>
<td>1</td>
</tr>
<tr>
<td>HAVE</td>
<td>0</td>
</tr>
<tr>
<td>HELLO</td>
<td>1</td>
</tr>
<tr>
<td>HELP</td>
<td>1</td>
</tr>
<tr>
<td>HERTZ</td>
<td>1</td>
</tr>
<tr>
<td>HIGH</td>
<td>0</td>
</tr>
<tr>
<td>HIGHER</td>
<td>0</td>
</tr>
<tr>
<td>HIT</td>
<td>2</td>
</tr>
<tr>
<td>HOLD</td>
<td>1</td>
</tr>
<tr>
<td>HOUR</td>
<td>0</td>
</tr>
<tr>
<td>HUNDRED</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0</td>
</tr>
<tr>
<td>IN</td>
<td>0</td>
</tr>
<tr>
<td>INCHES</td>
<td>0</td>
</tr>
<tr>
<td>INCORRECT</td>
<td>1</td>
</tr>
<tr>
<td>INCREASE</td>
<td>1</td>
</tr>
<tr>
<td>INTRUDER</td>
<td>1</td>
</tr>
<tr>
<td>IS</td>
<td>0</td>
</tr>
<tr>
<td>IT</td>
<td>0</td>
</tr>
<tr>
<td>J</td>
<td>0</td>
</tr>
<tr>
<td>JUST</td>
<td>1</td>
</tr>
<tr>
<td>K</td>
<td>0</td>
</tr>
<tr>
<td>KEY</td>
<td>1</td>
</tr>
<tr>
<td>KILO</td>
<td>0</td>
</tr>
<tr>
<td>L</td>
<td>0</td>
</tr>
<tr>
<td>LEFT</td>
<td>0</td>
</tr>
<tr>
<td>LESS</td>
<td>0</td>
</tr>
<tr>
<td>LESSER</td>
<td>0</td>
</tr>
<tr>
<td>LEVEL</td>
<td>1</td>
</tr>
<tr>
<td>LIMIT</td>
<td>0</td>
</tr>
<tr>
<td>LOAD</td>
<td>1</td>
</tr>
<tr>
<td>WORD</td>
<td>PAGE ADDRESS</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------</td>
</tr>
<tr>
<td>LOCK</td>
<td>1</td>
</tr>
<tr>
<td>LOW</td>
<td>0</td>
</tr>
<tr>
<td>LOWER</td>
<td>0</td>
</tr>
<tr>
<td>M</td>
<td>0</td>
</tr>
<tr>
<td>MARK</td>
<td>0</td>
</tr>
<tr>
<td>MEG</td>
<td>1</td>
</tr>
<tr>
<td>MEGA</td>
<td>1</td>
</tr>
<tr>
<td>METER</td>
<td>0</td>
</tr>
<tr>
<td>MICRO</td>
<td>1</td>
</tr>
<tr>
<td>MILE</td>
<td>0</td>
</tr>
<tr>
<td>MILLI</td>
<td>0</td>
</tr>
<tr>
<td>MILLION</td>
<td>0</td>
</tr>
<tr>
<td>MINUS</td>
<td>0</td>
</tr>
<tr>
<td>MINUTE</td>
<td>2</td>
</tr>
<tr>
<td>MISS</td>
<td>1</td>
</tr>
<tr>
<td>MORE</td>
<td>1</td>
</tr>
<tr>
<td>MOVE</td>
<td>2</td>
</tr>
<tr>
<td>MY</td>
<td>0</td>
</tr>
<tr>
<td>N</td>
<td>1</td>
</tr>
<tr>
<td>NANO</td>
<td>0</td>
</tr>
<tr>
<td>NEAR</td>
<td>1</td>
</tr>
<tr>
<td>NEED</td>
<td>1</td>
</tr>
<tr>
<td>NEXT</td>
<td>0</td>
</tr>
<tr>
<td>NIGHT</td>
<td>2</td>
</tr>
<tr>
<td>NINE</td>
<td>0</td>
</tr>
<tr>
<td>NINETEEN</td>
<td>0</td>
</tr>
<tr>
<td>NINETY</td>
<td>1</td>
</tr>
<tr>
<td>NO</td>
<td>1</td>
</tr>
<tr>
<td>NORMAL</td>
<td>1</td>
</tr>
<tr>
<td>NORTH</td>
<td>2</td>
</tr>
<tr>
<td>NOT</td>
<td>1</td>
</tr>
<tr>
<td>NOTICE</td>
<td>1</td>
</tr>
<tr>
<td>NUMBER</td>
<td>0</td>
</tr>
<tr>
<td>O</td>
<td>0</td>
</tr>
<tr>
<td>OFF</td>
<td>0</td>
</tr>
<tr>
<td>OFF</td>
<td>0</td>
</tr>
<tr>
<td>OHMS</td>
<td>0</td>
</tr>
<tr>
<td>ON</td>
<td>0</td>
</tr>
<tr>
<td>ONE</td>
<td>0</td>
</tr>
<tr>
<td>ONWARD</td>
<td>1</td>
</tr>
<tr>
<td>OPEN</td>
<td>1</td>
</tr>
<tr>
<td>OPERATOR</td>
<td>1</td>
</tr>
<tr>
<td>OPTICAL</td>
<td>2</td>
</tr>
<tr>
<td>OR</td>
<td>0</td>
</tr>
<tr>
<td>OUT</td>
<td>0</td>
</tr>
<tr>
<td>OVER</td>
<td>0</td>
</tr>
<tr>
<td>P</td>
<td>0</td>
</tr>
<tr>
<td>PARENTHESIS</td>
<td>0</td>
</tr>
<tr>
<td>WORD</td>
<td>PAGE -- ADDRESS -- WORD</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>PASS</td>
<td>1</td>
</tr>
<tr>
<td>PER</td>
<td>1</td>
</tr>
<tr>
<td>PERCENT</td>
<td>0</td>
</tr>
<tr>
<td>PICO</td>
<td>1</td>
</tr>
<tr>
<td>PLACE</td>
<td>1</td>
</tr>
<tr>
<td>PLEASE</td>
<td>0</td>
</tr>
<tr>
<td>PLUS</td>
<td>0</td>
</tr>
<tr>
<td>POINT</td>
<td>0</td>
</tr>
<tr>
<td>POUND</td>
<td>0</td>
</tr>
<tr>
<td>PRESS</td>
<td>1</td>
</tr>
<tr>
<td>PRESSURE</td>
<td>1</td>
</tr>
<tr>
<td>PULSES</td>
<td>0</td>
</tr>
<tr>
<td>Q</td>
<td>0</td>
</tr>
<tr>
<td>QUARTER</td>
<td>1</td>
</tr>
<tr>
<td>R</td>
<td>0</td>
</tr>
<tr>
<td>RANGE</td>
<td>1</td>
</tr>
<tr>
<td>RATE</td>
<td>0</td>
</tr>
<tr>
<td>RE</td>
<td>0</td>
</tr>
<tr>
<td>REACH</td>
<td>1</td>
</tr>
<tr>
<td>READY</td>
<td>0</td>
</tr>
<tr>
<td>RECEIVE</td>
<td>1</td>
</tr>
<tr>
<td>RECORD</td>
<td>1</td>
</tr>
<tr>
<td>REPLACE</td>
<td>1</td>
</tr>
<tr>
<td>REVERSE</td>
<td>1</td>
</tr>
<tr>
<td>RIGHT</td>
<td>0</td>
</tr>
<tr>
<td>ROOM</td>
<td>1</td>
</tr>
<tr>
<td>S</td>
<td>0</td>
</tr>
<tr>
<td>SAFE</td>
<td>1</td>
</tr>
<tr>
<td>SECOND</td>
<td>0</td>
</tr>
<tr>
<td>SECURE</td>
<td>1</td>
</tr>
<tr>
<td>SELECT</td>
<td>1</td>
</tr>
<tr>
<td>SEND</td>
<td>1</td>
</tr>
<tr>
<td>SERVICE</td>
<td>1</td>
</tr>
<tr>
<td>SET</td>
<td>0</td>
</tr>
<tr>
<td>SEVEN</td>
<td>0</td>
</tr>
<tr>
<td>SEVENTEEN</td>
<td>0</td>
</tr>
<tr>
<td>SEVENTY</td>
<td>0</td>
</tr>
<tr>
<td>SIDE</td>
<td>1</td>
</tr>
<tr>
<td>SIGHT</td>
<td>2</td>
</tr>
<tr>
<td>SIX</td>
<td>0</td>
</tr>
<tr>
<td>SIXTEEN</td>
<td>0</td>
</tr>
<tr>
<td>SIXTY</td>
<td>0</td>
</tr>
<tr>
<td>SLOW</td>
<td>1</td>
</tr>
<tr>
<td>SLOWER</td>
<td>1</td>
</tr>
<tr>
<td>SMOKE</td>
<td>1</td>
</tr>
<tr>
<td>SOUTH</td>
<td>1</td>
</tr>
<tr>
<td>SOUTH</td>
<td>2</td>
</tr>
<tr>
<td>SPACE</td>
<td>0</td>
</tr>
<tr>
<td>SPEED</td>
<td>0</td>
</tr>
<tr>
<td>WORD</td>
<td>PAGE</td>
</tr>
<tr>
<td>---------------</td>
<td>------</td>
</tr>
<tr>
<td>SQUAD</td>
<td>2</td>
</tr>
<tr>
<td>SS</td>
<td>0</td>
</tr>
<tr>
<td>STAR</td>
<td>0</td>
</tr>
<tr>
<td>START</td>
<td>0</td>
</tr>
<tr>
<td>STATION</td>
<td>1</td>
</tr>
<tr>
<td>STOP</td>
<td>0</td>
</tr>
<tr>
<td>SWITCH</td>
<td>1</td>
</tr>
<tr>
<td>SWITCH</td>
<td>1</td>
</tr>
<tr>
<td>SYSTEM</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
</tr>
<tr>
<td>TANK</td>
<td>2</td>
</tr>
<tr>
<td>TARGET</td>
<td>2</td>
</tr>
<tr>
<td>TEN</td>
<td>0</td>
</tr>
<tr>
<td>TEST</td>
<td>1</td>
</tr>
<tr>
<td>TH</td>
<td>1</td>
</tr>
<tr>
<td>THAN</td>
<td>0</td>
</tr>
<tr>
<td>THANK</td>
<td>1</td>
</tr>
<tr>
<td>THE</td>
<td>0</td>
</tr>
<tr>
<td>THERMAL</td>
<td>2</td>
</tr>
<tr>
<td>THIRD</td>
<td>1</td>
</tr>
<tr>
<td>THIRTEEN</td>
<td>0</td>
</tr>
<tr>
<td>THIRTY</td>
<td>0</td>
</tr>
<tr>
<td>THIS</td>
<td>1</td>
</tr>
<tr>
<td>THIS IS DIGITALKER</td>
<td>0</td>
</tr>
<tr>
<td>THOUSAND</td>
<td>0</td>
</tr>
<tr>
<td>THREE</td>
<td>0</td>
</tr>
<tr>
<td>TIME</td>
<td>0</td>
</tr>
<tr>
<td>TOTAL</td>
<td>1</td>
</tr>
<tr>
<td>TRACK</td>
<td>2</td>
</tr>
<tr>
<td>TRANSFER</td>
<td>2</td>
</tr>
<tr>
<td>TRY</td>
<td>0</td>
</tr>
<tr>
<td>TURN</td>
<td>1</td>
</tr>
<tr>
<td>TWELVE</td>
<td>0</td>
</tr>
<tr>
<td>TWENTY</td>
<td>0</td>
</tr>
<tr>
<td>TWO</td>
<td>0</td>
</tr>
<tr>
<td>U</td>
<td>0</td>
</tr>
<tr>
<td>UP</td>
<td>0</td>
</tr>
<tr>
<td>USE</td>
<td>1</td>
</tr>
<tr>
<td>USE2</td>
<td>2</td>
</tr>
<tr>
<td>UTH</td>
<td>1</td>
</tr>
<tr>
<td>V</td>
<td>0</td>
</tr>
<tr>
<td>VOLT</td>
<td>0</td>
</tr>
<tr>
<td>W</td>
<td>0</td>
</tr>
<tr>
<td>WAITING</td>
<td>1</td>
</tr>
<tr>
<td>WARNING</td>
<td>1</td>
</tr>
<tr>
<td>WORD</td>
<td>PAGE ADDRESS WORD</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------</td>
</tr>
<tr>
<td>WATER</td>
<td>1 7DH</td>
</tr>
<tr>
<td>WEIGHT</td>
<td>0 8FH</td>
</tr>
<tr>
<td>WEST</td>
<td>1 7EH</td>
</tr>
<tr>
<td>WEST</td>
<td>2 0CH</td>
</tr>
<tr>
<td>WINDOW</td>
<td>1 60H</td>
</tr>
<tr>
<td>X</td>
<td>0 37H</td>
</tr>
<tr>
<td>Y</td>
<td>0 38H</td>
</tr>
<tr>
<td>YES</td>
<td>1 81H</td>
</tr>
<tr>
<td>Z</td>
<td>0 39H</td>
</tr>
<tr>
<td>ZERO</td>
<td>0 1FH</td>
</tr>
<tr>
<td>ZONE</td>
<td>1 82H</td>
</tr>
</tbody>
</table>
APPENDIX E

INTERFACE PROGRAM

This appendix contains a listing of the program that is burned into the EPROM of the 8741 Universal Peripheral Interface chip. Refer to the UPI-41A User's Manual (Intel, 1980b).
LOC  OBJ  SEQ  SOURCE STATEMENT

1
2
3 ;UPI41 PORT 1 (P1) WILL OUTPUT DATA TO SPEECH CHIP (SPC)
4 ;UPI41 PORT 2 (P2) WILL FURNISH CONTROL SIGNALS
5

0000 6  ORG 0               ;RESERVE INTERRUPT LOCATIONS
0000 0409 7  JMP 9
6
8

0003 9  ORG 3
0003 042A 10 JMP ISERV      ;TO PROCESS PHRASE
11
12
13 ; PORT 2 BITS:
14 ; BIT 0 == SPC WR/
15 ; BIT 1 == SPC CS/
16 ; BIT 2 == PAGE1. HIGH == ROM BANK "1"
17 ; BIT 3 == SET. THE ROM SET TO BE USED.
18 ; BIT 4 == CMS
19 ; BIT 5 == GAIN (LSB)
20 ; BIT 6 == GAIN (MSB)
21 ; BIT 7 == GAIN CONTROL CLOCK
22
23 ; TO (PIN 1 ON THE B741) = SPC "INTR." HIGH == DONE
24
25
26 ;**************************************************************************
27 ; MAIN TEST PROGRAM LOOP
28 ;**************************************************************************
29
30
31  ORG 9
32
33  MOV  A,#OFFH
0009 23FF 34  OUTL P1,A
000B 39 35  OUTL P2,A
000C 3A 36
37
38 ;***** SET GAIN TO 1 FOR TEST
000D 9A7F 39  ANL  P2,#1111111B               ;LOWER CLOCK LINE
0000 F9A9F 40 ANL P2,#10011111B ;SET GAIN TO 1
0001 8AB0 41 ORL P2,#10000000B ;RAISE CLOCK LINE
0003 9A7F 42 ANL P2,#01111111B ;LOWER CLOCK LINE
43
44
45 ;** WILL FORCE SPC TO STOP PROCESSING ANY STARTING GARBAGE AND RESET
46 ;** SPC INTR ( = 0). THIS MAKES SPC APPEAR BUSY ****
47
0015 2311 48 MOV A,#11H ;SPC (WR/=1;CS/=0;CMS=1)
0017 3A 49 OUTL P2,A
0019 9AFE 50 ANL P2,#11111110B ;SPC WR/=0.
001A BA01 51 ORL P2,#00000001B ;SPC WR/=1.
001C BA02 52 ORL P2,#00000010B ;SPC CS/=1
53
54 ;********* WILL NOW FORCE A VALID COMMAND TO THE SPC TO SET SPC INTR *********
55
001E 2347 56 MOV A,#47H ;320MS SILENCE
0020 39 57 OUTL P1,A ;P1 DATA OUT TO SPC
58
59
60 ; P2A P23 P22 P21 P20
;SPC CMS SET P1 CS/ WR/
61
62 ; INITIAL P2
; 1 0 0 1 1
63
64
65
66
67
68
69 ;*******************************************************************************
70 ; TEST LOOP
71 ;*******************************************************************************
72
0027 05 73 EN I
0028 0428 74 BACK: JMP BACK ;WAIT FOR DATA READY INTERRUPT
75
002A 00 76 ISERV: NOP
002B D62B 77 LOOP1: JNIBF LOOP1 ;INPUT PAGE TO A
002D 22 78 IN A,DbB ;MOVE A TO R6
002E AE 79 MOV R6,A
002F D62F 80 LOOPP: JNIBF LOOPP ;INPUT WORD TO A
0031 22 81 IN A,DbB ;MOVE A TO R7
0032 AF 82 MOV R7,A
0033 D633 83 LOOPD: JNIBF LOOPD ;INPUT GAIN TO A
0035 22 84 IN A,DbB ;MASK OFF UPPER PORTION
0036 5303 85 ANL A,#00000011B ;MOVE A TO R5
0038 AD 86 MOV R5,A
0039 144A 87 CALL SETT ;CALL SETT
003B 85 88 CLR F0 ;CLEAR F0 FLAG
<table>
<thead>
<tr>
<th>Address</th>
<th>Code</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>003C FE</td>
<td>89</td>
<td>MOV A,R6</td>
<td>MOVE R6 TO A</td>
</tr>
<tr>
<td>003D C640</td>
<td>90</td>
<td>JZ CONT</td>
<td>IF PAGE=0 JUMP TO CONT</td>
</tr>
<tr>
<td>003F 95</td>
<td>91</td>
<td>CPL FO</td>
<td>IF PAGE=1 COMPLEMENT FO FO ==&gt; 1</td>
</tr>
<tr>
<td>0040 FF</td>
<td>92</td>
<td>MOV A,R7</td>
<td>MOVE R7 TO A</td>
</tr>
<tr>
<td>0041 B647</td>
<td>93</td>
<td>JFO CONT1</td>
<td>IF FO=1 JUMP TO CONT1</td>
</tr>
<tr>
<td>0043 1459</td>
<td>94</td>
<td>CALL WRITE0</td>
<td>ELSE CALL WRITE0</td>
</tr>
<tr>
<td>0045 0449</td>
<td>95</td>
<td>JMP LEAVE</td>
<td></td>
</tr>
<tr>
<td>0047 145E</td>
<td>96</td>
<td>CONT1: CALL WRITE1</td>
<td></td>
</tr>
<tr>
<td>0049 93</td>
<td>97</td>
<td>LEAVE: RETR</td>
<td></td>
</tr>
<tr>
<td>004A FE</td>
<td>100</td>
<td>SETT: MOV A,R6</td>
<td>MOVE PAGE TO A</td>
</tr>
<tr>
<td>004B 324F</td>
<td>101</td>
<td>JRP SETT1</td>
<td>IF SETT = 1, GOTO SETT1</td>
</tr>
<tr>
<td>004D 0454</td>
<td>102</td>
<td>JMP NOSETT</td>
<td>ELSE JUMP TO NOSETT</td>
</tr>
<tr>
<td>004F BA08</td>
<td>103</td>
<td>SETT1: ORL P2,#00001000B</td>
<td>SETT = 1</td>
</tr>
<tr>
<td>0051 27</td>
<td>104</td>
<td>CLR A</td>
<td>CLEAR A</td>
</tr>
<tr>
<td>0052 045B</td>
<td>105</td>
<td>JMP ENDTT</td>
<td>JUMP TO ENDTT</td>
</tr>
<tr>
<td>0054 9A67</td>
<td>106</td>
<td>NOSETT: ANL P2,#11110111B</td>
<td>SETT = 0</td>
</tr>
<tr>
<td>0056 53F7</td>
<td>107</td>
<td>ANL A,#11110111B</td>
<td>MASK OFF SETT</td>
</tr>
<tr>
<td>0058 83</td>
<td>108</td>
<td>ENDTT: RET</td>
<td></td>
</tr>
<tr>
<td>0059 9AF8</td>
<td>111</td>
<td>WRITE0: ANL P2,#0FBH</td>
<td>P22 = 0 ==&gt; PAGE = 0</td>
</tr>
<tr>
<td>005B 1463</td>
<td>112</td>
<td>CALL OUTPUT</td>
<td></td>
</tr>
<tr>
<td>005D 83</td>
<td>113</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>005E BA04</td>
<td>116</td>
<td>WRITE1: ORL P2,#04H</td>
<td>P22 = 1 ==&gt; PAGE = 1</td>
</tr>
<tr>
<td>0060 1463</td>
<td>117</td>
<td>CALL OUTPUT</td>
<td></td>
</tr>
<tr>
<td>0062 83</td>
<td>118</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>0063 2663</td>
<td>121</td>
<td>OUTPUT: JNTO OUTPUT</td>
<td>MUST INSURE SPC IS READY</td>
</tr>
<tr>
<td>0065 39</td>
<td>122</td>
<td>OUTL P1,A</td>
<td>P1 DATA OUT TO SPC</td>
</tr>
<tr>
<td>0066 FD</td>
<td>123</td>
<td>MOV A,R5</td>
<td></td>
</tr>
<tr>
<td>0067 9A7F</td>
<td>124</td>
<td>ANL P2,#01111111B</td>
<td>MOVE GAIN TO A</td>
</tr>
<tr>
<td>0069 9A9F</td>
<td>125</td>
<td>ANL P2,#10011111B</td>
<td>LOWER CLOCK LINE</td>
</tr>
<tr>
<td>006B 126F</td>
<td>126</td>
<td>JBO SETS</td>
<td>MASK OFF OLD GAIN</td>
</tr>
<tr>
<td>006D 0471</td>
<td>127</td>
<td>JMP TST1</td>
<td></td>
</tr>
<tr>
<td>006F BA20</td>
<td>128</td>
<td>SET5: ORL P2,#00100000B</td>
<td>IF BIT 0 IS 1 THEN JUMP TO SET5</td>
</tr>
<tr>
<td>0071 3275</td>
<td>129</td>
<td>TST1: JRP SET6</td>
<td>ELSE JUMP TO TST1</td>
</tr>
<tr>
<td>0073 0477</td>
<td>130</td>
<td>JMP NSET6</td>
<td>SET BIT 5 OF PORT 2</td>
</tr>
<tr>
<td>0075 BA40</td>
<td>131</td>
<td>SET6: ORL P2,#01000000B</td>
<td>IF BIT 1 IS 1 THEN JUMP TO SET6</td>
</tr>
<tr>
<td>0077 00</td>
<td>132</td>
<td>JMP NSET6: NOP</td>
<td>ELSE JUMP TO NSET6</td>
</tr>
<tr>
<td>0078 BA80</td>
<td>133</td>
<td>ORL P2,#10000000B</td>
<td>CONTINUE</td>
</tr>
<tr>
<td>007A 9A7F</td>
<td>134</td>
<td>ANL P2,#01111111B</td>
<td>RAISE CLOCK LINE</td>
</tr>
<tr>
<td>007C 135</td>
<td>136</td>
<td>JMP NSET5: NOP</td>
<td>LOWER CLOCK LINE</td>
</tr>
<tr>
<td>137</td>
<td>138</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
139
140 ; INITIAL P2
141
007C 9AED
007E 9AFE
0080 8A03
0082 26B2
0084 9A7F
0086 9A9F
0088 B800
008A 9A7F
00BC 83
0151
0152
0153
0154

INITIAL P2

; 0 ? 0/1 1 1
ANL P2,#11101101B 0
ANL P2,#11111110B 0
DRL P2,#00000011B 0
146 LOOP1: JNTO LOOP1 ; WAIT UNTIL SPC IS NOT BUSY
147 ANL P2,#01111111B 0
148 ANL P2,#10011111B 0
149 ORL P2,#10000000B 0
150 ANL P2,#01111111B 0
151 RET
152
153
154

USER SYMBOLS
BACK 0028 CONT 0040 CONT1 0047 ENDTT 005B ISERV 002A LEAVE 0049
LOOP1 0082 LOOP1 002B LOOPP 002F LOOPB 0033 MOSETT 0054 NSET6 0077
OUTPUT 0063 SET5 006F SET6 0075 SETT 0075 SETT1 004A SETT1 004F TST1 0071
WRITE0 0059 WRITE1 005E

ASSEMBLY COMPLETE, NO ERRORS
APPENDIX F

SCREEN EDITOR PROGRAM

This appendix contains a listing of the program that is burned into the EPROMs on the 86/12A controlling processor. Refer to the ISBC 86/12A Single Board Computer Hardware Reference Manual (Intel, 1979) and the PLM86 Programming Manual (Intel, 1980a).
SERIES-III PL/M-86 V2.0 COMPILATION OF MODULE DIGITALKER_TEST
OBJECT MODULE PLACED IN :F1:WRDTST.OBJ
COMPILER INVOKED BY: PLMB6.B6 :F1:WRDTST.023 DEBUG ROM XREF NOPAGING PAGewidth(100)

DIGITALKER_TEST:DO;

/**********************************************************
* THIS IS A SIMPLE TEXT EDITOR PROGRAM TO ALLOW A SINGLE LINE OF TEXT (A SE-
* QUENCE OF WORDS FOR THE DIGITALKER) TO BE ENTERED FROM AN ADMSA TERMINAL
* AND TRANSLATED TO HEX CODE AND TRANSMITTED TO THE DIGITALKER WHERE IT WILL
* BE OUTPUT AS SPEECH.
**********************************************************/

/**********************************************************
* THIS PROGRAM WAS LAST REVISED: DECEMBER 7, 1982
**********************************************************/

DECLARE BUFFER (80) BYTE, INDEX BYTE, DATA_IN BYTE;
DECLARE CR LITERALLY '0DH', SP LITERALLY '20H', COMMA LITERALLY '2CH',
RUB OUT LITERALLY '7FH', BELL LITERALLY '07H';
DECLARE ARRAY (61) STRUCTURE(PAGE BYTE, WRD BYTE, VOL BYTE);
DECLARE IODATA LITERALLY '008H', IOSTATUS LITERALLY '0DAH', MASK LITERALLY '7FH';
DECLARE LIT LITERALLY 'LITERALLY';
DECLARE COUNTER_2 LIT '004H', CONTROL LIT '006H', CNTR2MODE LIT '10010110B',
SETCOUNT LIT '20H'; /* FOR TIMER SETUP */
DECLARE USART_CONTROL LIT '0DAH', USART_MODE LIT '4EH', USART_COMMAND LIT '37H';
DECLARE LF LIT '0AH', FINISHED BYTE, BS LIT '08H', FF LIT '0CH', DEL LIT '04H',
INS LIT '03H', (CNT, DONE, CNTRL, FLAG, EMPTY) BYTE, PRT LIT '010H';
DECLARE KILL LIT '18H', VT LIT '08H', CNTRL_A LIT '01H', CNTRL_Z LIT '1AH';
DECLARE (STOP, CNT1, CNT2, A_FLAG, Z_FLAG, FIRST, FLAG1) BYTE;

/**********************************************************
* THE FOLLOWING SUBROUTINE INPUTS A BYTE OF ASCII DATA FROM THE TERMINAL
**********************************************************/
CIN: PROCEDURE BYTE;
DO WHILE NOT SHR(INPUT(IOSTATUS)),1);
END;
RETURN MASK AND INPUT(IODATA);
END CIN;

COUT: PROCEDURE (ITEM);
DECLARE ITEM BYTE;
DO WHILE NOT(INPUT(IOSTATUS));
END;
OUTPUT(IODATA)=ITEM;
END COUT;

SCRUB: PROCEDURE;
CNT = 0;
DO WHILE BUFFER(CNT) <> CR;
CNT = CNT + 1;
IF CNT > 70D THEN GOTO QUIT;
END;
DO WHILE CNT < 75D;
CNT = CNT + 1;
BUFFER(CNT) = SP;
END;
QUIT: END SCRUB;

PRNT: PROCEDURE;
CNT = 0;
CALL SCRUB;
CALL COUT(CR);
CALL COUT(FF);
DO WHILE CNT < 75D;
CALL COUT(SP);
CNT = CNT + 1;
END;
44 2  CNT = 0;
45 2  CALL COUT(CR);
46 2  CALL COUT(FF);
47 2  DO WHILE BUFFER(CNT) <> CR;
48 3  CALL COUT(BUFFER(CNT));
49 3  CNT = CNT + 1;
50 3  IF BUFFER(CNT) = CR THEN CALL COUT('\^');
52 3  END;
53 2  CNT = CNT + 1;
54 2  DO WHILE CNT < 70D;
55 3  CALL COUT(SP);
56 3  CNT = CNT + 1;
57 3  END;
58 2  CALL COUT(CR);
59 2  CALL COUT(FF);
60 2  END PRNT;

/*******************************************************************************
# THE FOLLOWING SUBROUTINE MOVES THE CURSOR BACK ONE SPACE
*******************************************************************************/

61 1 CBCK: PROCEDURE;
62 2  IF INDEX > 0 THEN DO;
63 3    INDEX = INDEX - 1;
64 3    CALL COUT(BS);
66 3  END;
67 2 ELSE CALL COUT(BELL);
68 2  END CBCK;

/*******************************************************************************
# THE FOLLOWING SUBROUTINE MOVES THE CURSOR FORWARD ONE SPACE
*******************************************************************************/

69 1 CFWD: PROCEDURE;
70 2  IF INDEX < 70D THEN DO;
72 3    INDEX = INDEX + 1;
73 3    CALL COUT(FF);
74 3  END;
75 2 ELSE CALL COUT(BELL);
76 2  END CFWD;

/*******************************************************************************
# THE FOLLOWING SUBROUTINE DELETES THE CHARACTER UNDER THE CURSOR
*******************************************************************************
DEL_EX: PROCEDURE;
78 2 CNT = INDEX;
79 2 DO WHILE CNT < 70D:
80 3 BUFFER(CNT) = BUFFER(CNT + 1);
81 3 CNT = CNT + 1;
82 3 END;
83 2 CALL SCRUB;
84 2 CALL PRINT;
85 2 CNT = INDEX;
86 2 DO WHILE CNT > 0;
87 3 CALL COUT(FF);
88 3 CNT = CNT - 1;
89 3 END;
90 2 END DEL_EX;

DWORD SUBROUTINE FILTERS THE INCOMING DATA AND REJECTS ILLEGAL DATA
******************************************************************************

IN_DAT: PROCEDURE;
91 1 DONE = 0;
92 2 DO WHILE NOT DONE;
93 2 DATA_IN = CIN AND MASK;
94 3 IF DATA_IN = CTRL_A THEN
95 3 IF A_FLAG THEN DONE = 1;
96 3 IF DATA_IN = RUB_OUT THEN
97 3 IF A_FLAG THEN DONE = 1;
98 3 IF DATA_IN = SP THEN DONE = 1;
99 3 IF DATA_IN = COMMA THEN DONE = 1;
100 3 IF DATA_IN > 2FH THEN
101 3 IF DATA_IN < 3AH THEN DONE = 1;
102 3 IF DATA_IN > 40H THEN
103 3 IF DATA_IN < 47H THEN DONE = 1;
104 3 IF DATA_IN = 'H' THEN DONE = 1;
105 3 IF DATA_IN = 'L' THEN DONE = 1;
106 3 IF NOT DONE THEN CALL COUT(BELL);
107 3 END;
108 2 END IN_DAT;

******************************************************************************
THE FOLLOWING SUBROUTINE INSERTS A CHARACTER TO THE LEFT OF THE CURSOR
******************************************************************************

INS_EX: PROCEDURE;
119 1 BUFFER(70) = CR;
120 2 CNT = 69D;
121 2 DO WHILE CNT <> INDEX;
122 2
BUFFER(CNT) = BUFFER(CNT - 1);
CNT = CNT - 1;
END;
CALL IN_DAT;
BUFFER(INDEX) = DATA_IN;
CALL PRNT;
CNT = INDEX;
DO WHILE CNT > 0;
CALL COUT(FF);
CNT = CNT - 1;
END;
END INS_EX;

///////////////////////////////
THE FOLLOWING SUBROUTINE ALLOWS THE INSERTION OF A STRING OF CHARACTERS
///////////////////////////////

CNT = INDEX;
DO WHILE CNT < 75D;
CALL COUT(SP);
CNT = CNT + 1;
END;
CALL COUT(CR);
CALL COUT(FF);
CNT = 0;
DO WHILE CNT <> INDEX;
CALL COUT(FF);
CNT = CNT + 1;
END;
CALL COUT('_');
CALL COUT(BS);
STOP = 0;
CNT = INDEX;
CNT1 = 1;
DO WHILE BUFFER(CNT) <> CR;
CNT = CNT + 1;
CNT1 = CNT1 + 1;
END;
CNT = INDEX;
DO WHILE NOT STOP;
IF CNT < (71 - CNT1) THEN DO;
CALL IN_DAT;
IF DATA_IN = CNTRL_A THEN STOP = 1;
ELSE DO;
IF DATA_IN = RUB_OUT THEN DO;
IF INDEX = 0 THEN
CALL COUT(BELL);
169  ELSE DO;
170    CALL COUT(SP);
171    CALL COUT(BS);
172    CALL COUT(BS);
173    CNT = CNT - 1;
174    END;
175  END;
176  ELSE DO;
177    CNT2 = 700;
178    DO WHILE CNT2 <> CNT;
179      BUFFER(CNT2) = BUFFER(CNT2 - 1);
180    END;
181    CNT2 = CNT2 - 1;
182    BUFFER(CNT) = DATA_IN;
183    END;
184  END;
185  ELSE DO;
186    IF NOT STOP THEN DO;
187      IF DATA_IN <> RUB.OUT THEN DO;
188        CNT = CNT + 1;
189      END;
190    END;
191    CALL COUT(BS);
192    CALL COUT('_.');
193    CALL COUT(BS);
194    END;
195  END;
196  ELSE DO;
197    BUFFER(70) = CR;
198    CALL COUT(BELL);
199    STOP = 1;
200  END;
201  END;
202  INDEX = CNT;
203  CALL SCRUB;
204  CNT = INDEX;
205  DO WHILE CNT1 > 0;
206    CALL COUT(BUFFER(CNT));
207    CNT = CNT + 1;
208    IF BUFFER(CNT) = CR THEN CALL COUT('^^');
209    CNT1 = CNT1 - 1;
210    END;
211    CALL COUT(CR);
212    CALL COUT(FF);
213    CNT = INDEX;
214  DO WHILE CNT > 0;
215    CALL COUT(FF);
216    CNT = CNT - 1;
217    END;
218  END;
219  END CTRL_A_EX;
THE FOLLOWING SUBROUTINE ALLOWS THE DELETION OF A STRING OF CHARACTERS

220   1   CNTRL_Z.EX: PROCEDURE;
221   2   CNT = INDEX;
222   2   CNT1 = 0;
223   2   CALL COUT('¥');
224   2   CALL COUT(BS);
225   2   FIRST = 1;
226   2   STOP = 0;
227   2   DO WHILE NOT STOP;
228   3   FLAG1 = 0;
229   3   DATA_IN = CIN AND MASK;
230   3   IF DATA_IN = CNTRL_Z THEN STOP = 1;
231   3   IF NOT STOP THEN DO;
232   3   IF DATA_IN = FF THEN DO;
233   4   IF CNT < 700 THEN DO;
234   5   IF BUFFER(CNT) <> CR THEN DO;
235   6   IF NOT FIRST THEN CALL COUT(BUFFER(CNT));
236   6   ELSE CALL COUT(FF);
237   7   CNT1 = CNT1 + 1;
238   7   CNT = CNT + 1;
239   7   FIRST = 0;
240   7   FLAG1 = 1;
241   7   CALL COUT('¥');
242   7   CALL COUT(BS);
243   7   END;
244   8   ELSE CALL COUT(BELL);
245   9   END;
246   9   IF DATA_IN = BS THEN DO;
247  10   IF CNT > INDEX THEN DO;
248  11   CALL COUT(BUFFER(CNT));
249  11   CALL COUT(BS);
250  11   CALL COUT('¥');
251  11   CALL COUT(BS);
252  11   CNT = CNT - 1;
253  11   CNT1 = CNT1 - 1;
254  11   FLAG1 = 1;
255  11   END;
256  12   ELSE CALL COUT(BELL);
257  13   END;
258  14   IF NOT FLAG1 THEN CALL COUT(BELL);
259  15   END;
260  16   END;
261  17   CALL COUT(BELL);
262  18   BUFFER(70) = CR;
263  19   DO WHILE CNT1 > 0;
264  20   CNT = INDEX;
DO WHILE CNT < 70D:
BUFFER(CNT) = BUFFER(CNT + 1);
CNT = CNT + 1;
END;
CNT1 = CNT1 - 1;
END;
CALL SCRUB;
CALL PRT;
CNT = INDEX;
DO WHILE CNT > 0:
CALL COUT(FF);
CNT = CNT - 1;
END;
END CNTRL_Z_EX;

/***************************************************************************/
/* THE FOLLOWING SUBROUTINE LOADS THE INPUT BUFFER FROM THE CONSOLE */
/***************************************************************************/

DO WHILE NOT FINISHED;
CALL SCRUB;
IF BUFFER(0) = CR THEN EMPTY = 1;
ELSE EMPTY = 0;
A_FLAG = 0;
Z_FLAG = 0;
CNTRL = 0;
DATA_IN = CIN AND MASK;

/***************************************************************************/
/* IF THE INPUT IS AN UP-CURSOR OR DOWN-CURSOR COMMAND THEN DO NOTHING */
/***************************************************************************/

IF DATA_IN = VT THEN CNTRL = 1;
IF DATA_IN = LF THEN CNTRL = 1;
IF THE INPUT IS A CNTRL-P THEN CALL THE PRNT ROUTINE

IF DATA_IN = PRN THEN DO:
    CALL PRNT;
    CNTRL = 1;
    IF NOT EMPTY THEN FLAG = 1;
    END;

IF THE INPUT IS A CURSOR-BACK COMMAND THEN CALL THE CBCK ROUTINE

IF DATA_IN = BS THEN DO:
    CNTRL = 1;
    IF FLAG THEN CALL CBCK;
    ELSE CALL COUT(BELL);
    END;

IF THE INPUT IS A CURSOR-FORWARD COMMAND THEN CALL THE CFWD ROUTINE

IF DATA_IN = FF THEN DO:
    CNTRL = 1;
    IF FLAG THEN CALL CFWD;
    ELSE CALL COUT(BELL);
    END;

IF THE INPUT IS A CNTRL-D THEN CALL THE DEL_EX ROUTINE

IF DATA_IN = DEL THEN DO:
    CNTRL = 1;
    IF FLAG THEN CALL DEL_EX;
    ELSE CALL COUT(BELL);
    END;

IF THE INPUT IS A CNTRL-C THEN CALL THE INS_EX ROUTINE  

IF DATA_IN = INS THEN DO:
    CNTRL = 1;
    IF FLAG THEN CALL INS_EX;
339 4 ELSE CALL COUT(BELL);
340 4 END;

/********************************************************
* IF THE INPUT IS A CTRL-A THEN CALL THE CNTRL_A_EX ROUTINE *
***********************************************************/

341 3 IF DATA_IN = CNTRL_A THEN DO;
342 4 A_FLAG = 1;
343 4 CNTRL = 1;
344 4 IF FLAG THEN CALL CNTRL_A_EX;
345 4 ELSE CALL COUT(BELL);
346 4 END;

/********************************************************
* IF THE INPUT IS A CTRL-Z THEN CALL THE CNTRL_Z_EX ROUTINE *
***********************************************************/

349 3 IF DATA_IN = CNTRL_Z THEN DO;
350 4 Z_FLAG = 1;
351 4 CNTRL = 1;
352 4 IF FLAG THEN CALL CNTRL_Z_EX;
353 4 ELSE CALL COUT(BELL);
354 4 END;

/********************************************************
* IF THE INPUT IS A KILL THEN INSERT A CR INTO THE BUFFER AT THE CURRENT *
* POSITION, DELETE THE REST OF THE BUFFER AND EXIT THE INPUT ROUTINE *
***********************************************************/

357 3 IF DATA_IN = KILL THEN DO;
358 4 CNTRL = 1;
359 4 IF FLAG THEN DO;
360 5 BUFFER(INDEX) = CR;
361 5 CALL SCRUB;
362 5 FINISHED = 1;
363 5 CALL COUT(CR);
364 5 CALL COUT(LF);
365 5 END;
366 5 ELSE CALL COUT(BELL);
367 5 END;

/********************************************************
* THIS IS A FILTER WHICH REJECTS UNWANTED INPUTS *
***********************************************************/
370  3  IF DATA_IN < 30H THEN CNTRL = 1;
372  3  IF DATA_IN > 39H THEN
373  3  IF DATA_IN < 41H THEN CNTRL = 1;
375  3  IF DATA_IN > 46H THEN CNTRL = 1;
377  3  IF DATA_IN = SP THEN CNTRL = 0;
379  3  IF DATA_IN = 'H' THEN CNTRL = 0;
381  3  IF DATA_IN = 'L' THEN CNTRL = 0;
383  3  IF DATA_IN = COMMA THEN CNTRL = 0;
385  3  IF DATA_IN = CR THEN CNTRL = 0;
387  3  IF DATA_IN = 'R' THEN CNTRL = 0;
389  3  IF DATA_IN = RUB_OUT THEN CNTRL = 0;

#ifndef

#define DATA_IN

391  3  IF DATA_IN = 'R' THEN DO;
393  4  IF NOT FLAG THEN DO;
395  5  FINISHED = 1;
396  5  CALL COUT(CR);
397  5  CALL COUT(LF);
398  5  CALL COUT('!');
399  5  END;
400  4  ELSE DO;
401  5  CALL COUT(BELL);
402  5  CNTRL = 1;
403  5  END;
404  4  END;

#ifndef

#define DATA_IN

405  3  ELSE IF NOT CNTRL THEN
406  3  IF NOT FINISHED THEN DO;
408 4 IF DATA_IN = RUB_OUT THEN
409 4 DO;
410 5 IF INDEX = 0 THEN
411 5 CALL COUT(BELL);
412 5 ELSE DO;
413 6 CALL COUT(SP);
414 6 CALL COUT(BS);
415 6 CALL COUT(BS);
416 6 INDEX = INDEX - 1;
417 6 END;
418 5 END;
419 4 ELSE DO;
420 5 IF DATA_IN = CR THEN
421 5 IF NOT FLAG THEN CALL COUT('"');
422 5 CALL COUT(DATA_IN);
423 5 IF DATA_IN <> CR THEN DO;
424 6 IF BUFFER(INDEX) = CR THEN
425 6 BUFFER(INDEX + 1) = CR;
426 6 BUFFER(INDEX) = DATA_IN;
427 6 END;
428 6 END;
429 6 IF DATA_IN = CR THEN DO;
430 5 IF NOT FLAG THEN BUFFER(INDEX) = DATA_IN;
431 5 FINISHED = 1;
432 5 CALL COUT(LF);
433 6 END;
434 6 IF INDEX < 70D THEN
435 5 INDEX = INDEX + 1;
436 6 ELSE DO;
437 6 CALL COUT(BELL);
438 6 CALL COUT(BS);
439 6 CALL COUT(SP);
440 6 CALL COUT(BS);
441 6 CALL COUT(BS);
442 6 CALL COUT(SL);
443 6 CALL COUT(BL);
444 6 END;
445 5 END;
446 4 END;
447 3 END INPT;
THE FOLLOWING ROUTINE OUTPUTS THE CONTENTS OF THE INPUT BUFFER. THIS
ROUTINE IS CALLED ONLY FROM THE MAIN PROGRAM.

***************

449 1 OUTPT: PROCEDURE;
450 2 INDEX = 0;
451 2 FINISHED = 0;
452 2 DO WHILE NOT FINISHED;
453 3 IF BUFFER(INDEX) = CR THEN CALL COUT('"');
454 3 CALL COUT(BUFFER(INDEX));
455 3 IF BUFFER(INDEX) = CR THEN FINISHED = 1;
456 3 IF INDEX < 79D THEN
457 3 INDEX = INDEX + 1;
458 3 END;
459 3 CALL COUT(LF);
460 3 END OUTPT;

463 1 TRANS: PROCEDURE;
464 2 DECLARE I BYTE, TEMP BYTE;
465 2 I = 0;
466 2 INDEX = 0;
467 2 FINISHED = 0;
468 2 DO WHILE NOT FINISHED;
469 3 IF BUFFER(INDEX) <> CR THEN
470 3 ARRAY(I).PAGE = BUFFER(INDEX) AND 0FH;
471 3 ELSE DO;
472 4 ARRAY(I).PAGE = CR;
473 4 FINISHED = 1;
474 4 END;
475 3 IF NOT FINISHED THEN DO;
476 4 INDEX = INDEX + 1;
477 4 IF BUFFER(INDEX) = COMMA THEN
478 4 INDEX = INDEX + 1;
479 4 IF BUFFER(INDEX) = SP THEN
480 4 INDEX = INDEX + 1;
IF BUFFER(INDEX) <> CR THEN DO;

IF BUFFER(INDEX) < 40H THEN
    TEMP = BUFFER(INDEX) AND 0FH;
ELSE TEMP = (BUFFER(INDEX) AND 0FH) + 9H;

INDEX = INDEX + 1;
END;

ELSE DO;

FINISHED = 1;
ARRAY(I).PAGE = CR;
END;

IF BUFFER(INDEX) <> CR THEN DO;

IF BUFFER(INDEX) < 40H THEN
    ARRAY(I).WRD = (BUFFER(INDEX) AND 0FH) OR SHL(TEMP,4);
ELSE ARRAY(I).WRD = ((BUFFER(INDEX) AND 0FH) + 9H) OR SHL(TEMP,4);

INDEX = INDEX + 1;
END;

ELSE DO;

FINISHED = 1;
ARRAY(I).PAGE = CR;
END;

IF BUFFER(INDEX) = 'H' THEN
    INDEX = INDEX + 1;

IF BUFFER(INDEX) = SP THEN
    INDEX = INDEX + 1;

IF BUFFER(INDEX) = COMMA THEN
    INDEX = INDEX + 1;

IF BUFFER(INDEX) <> CR THEN DO;

INDEX = INDEX + 1;

ARRAY(I).VOL = (BUFFER(INDEX) AND 07H);

INDEX = INDEX + 1;
517 6      END;
518 5      ELSE ARRAY(I).VOL = 0;
519 5      END;

520 4      ELSE DO;
521 5      FINISHED = 1;
522 5      ARRAY(I).PAGE = CR;
523 5      END;

524 4      IF BUFFER(INDEX) = COMMA THEN
525 4      INDEX = INDEX + 1;
526 4      IF BUFFER(INDEX) = SP THEN
527 4      INDEX = INDEX + 1;
528 4      I = I + 1;

529 4      END;
530 3      END;
531 2      END TRANSLATE;

/*************************************************************************/
/* THE FOLLOWING ROUTINE OUTPUTS A CHARACTER TO THE 8741 WHEN CALLED FROM */
/* THE MAIN PROGRAM                                                      */
/*************************************************************************/

532 1      SPEECH: PROCEDURE (PHRASE);
533 2  DECLARE PHRASE BYTE, STAT_COM BYTE AT (0D002H), P_DATA BYTE AT (0D000H);
534 2  DO WHILE NOT SHR(STAT_COM,1); /* WAIT UNTIL UPI41 IBF = 0 */
535 3      END;
536 2  P_DATA = NOT PHRASE; /* NOT BECAUSE MULTIBUS INVERTS */
537 2  END SPEECH;

/*************************************************************************/
PROGRAM STARTS
************************************************************************/

538 1  DECLARE K BYTE;
539 1  DISABLE; /* DISABLE ALL INTERRUPTS */

/* THE FOLLOWING FOUR LINES SET UP THE 8253 "PIT" (PROGRAMMABLE INTERVAL */
/* TIMER) TO SUPPLY A BAUD RATE OF 2400 BAUD, AND TO SET UP THE 8251 USART. */
540 1 OUTPUT(CONTROL)=ENT2MODE;
541 1 OUTPUT(COUNTER_2)=SETCOUNT;
542 1 OUTPUT(USART_CONTROL)=USART_MODE;
543 1 OUTPUT(USART_CONTROL)=USART_COMMAND;

/***************************************************************************/
544 1 ARRAY(61).PAGE = CR;  /* THIS WILL STOP BUFFER OVERFLOW */

545 1 CNT = 0;
546 1 COUNT: DO WHILE CNT < 60;
547 2 CALL TIME(250);         /* 1.5 SEC. DELAY TO ALLOW FOR SETUP */
548 2 CNT = CNT + 1;
549 2 END COUNT;

550 1 LOOP: DO WHILE K < ARRAY(61).PAGE:
551 2 CALL COUT("*");
552 2 CALL INPT;
553 2 CALL OUTPT;
554 2 CALL TRANSLATE;
555 2 K = K + 1;
556 2 END LOOP;
557 2 END LOOP;
561 3 END DIGITALKER_TEST;
<table>
<thead>
<tr>
<th>DEFN</th>
<th>ADDR</th>
<th>SIZE</th>
<th>NAME, ATTRIBUTES, AND REFERENCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0052H</td>
<td>183</td>
<td>STRUCTURE ARRAY(61)</td>
</tr>
<tr>
<td>0000H</td>
<td>1</td>
<td>PAGE</td>
<td>BYTE 470# 472# 491# 502# 522# 544# 556</td>
</tr>
<tr>
<td>11</td>
<td>0112H</td>
<td>1</td>
<td>A_FLAG</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>BELL</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td>BS</td>
</tr>
<tr>
<td>2</td>
<td>0000H</td>
<td>80</td>
<td>BUFFER</td>
</tr>
<tr>
<td>61</td>
<td>0189H</td>
<td>34</td>
<td>CBCK</td>
</tr>
<tr>
<td>69</td>
<td>018EH</td>
<td>34</td>
<td>CFRD</td>
</tr>
<tr>
<td>12</td>
<td>009EH</td>
<td>24</td>
<td>CIN.</td>
</tr>
<tr>
<td>9</td>
<td>010AH</td>
<td>1</td>
<td>CNT.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0110H</td>
<td>1</td>
<td>CNT1</td>
</tr>
<tr>
<td></td>
<td>0111H</td>
<td>1</td>
<td>CNT2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>010CH</td>
<td>1</td>
<td>CNTRL</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td>CNTRL_A.</td>
</tr>
<tr>
<td>135</td>
<td>0388H</td>
<td>491</td>
<td>CNTRL_A_EX</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td>CNTRL_Z.</td>
</tr>
<tr>
<td>220</td>
<td>0576H</td>
<td>389</td>
<td>CNTRL_Z_EX</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>COMMA.</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>CONTROL.</td>
</tr>
<tr>
<td>546</td>
<td>0025H</td>
<td></td>
<td>COUNT.</td>
</tr>
<tr>
<td>Address</td>
<td>Operation</td>
<td>Description</td>
<td>Data</td>
</tr>
<tr>
<td>---------</td>
<td>-----------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>0051H</td>
<td>DATA_IN</td>
<td>BYTE</td>
<td>94 95 98 101 103 105 106</td>
</tr>
<tr>
<td>0098H</td>
<td>DONE</td>
<td>BYTE</td>
<td>92 93 97 100 102 104 107</td>
</tr>
<tr>
<td>010EH</td>
<td>EMPTY</td>
<td>BYTE</td>
<td>29 29 31 30</td>
</tr>
<tr>
<td>FF</td>
<td></td>
<td>LITERALLY '0CH'</td>
<td>39 46 59 73 87</td>
</tr>
<tr>
<td>0109H</td>
<td>FINISHED</td>
<td>BYTE</td>
<td>29 29 36 34 40 43 45</td>
</tr>
<tr>
<td>1114H</td>
<td>FIRST</td>
<td>BYTE</td>
<td>22 24 25 24 25</td>
</tr>
<tr>
<td>00H</td>
<td>FLAG</td>
<td>BYTE</td>
<td>29 31 31 36 33 30 34 34</td>
</tr>
<tr>
<td>1115H</td>
<td></td>
<td>BYTE</td>
<td>22 24 26 25 26</td>
</tr>
<tr>
<td>0116H</td>
<td>I</td>
<td>BYTE IN PROC (TRANSLATE)</td>
<td>46 52 28 52 52</td>
</tr>
<tr>
<td>0050H</td>
<td>INDEX</td>
<td>BYTE</td>
<td>62 64 64 70 72 72 78</td>
</tr>
<tr>
<td>06FBH</td>
<td>INPT</td>
<td>PROCEDURE STACK=0010H</td>
<td>552</td>
</tr>
</tbody>
</table>

```
17 0086H 24 COUT ....... LITERALLY '004H' 541
   46 48 51 55 58 59 65 67 73
   75 87 116 131 138 141 142 145 148
  149 168 170 171 172 190 192 193 198
  206 209 212 213 216 223 224 241 242
  247 248 251 257 258 259 260 261 266
  269 285 318 325 332 339 347 355 365
  366 368 396 397 398 401 411 413 414
  415 422 423 435 440 441 442 443 454
  455 461 551

3  CR ......... LITERALLY '00H' 25 38 45 47 50
   58 120 141 153 197 208 212 212 272
  295 362 385 385 396 420 424 426 427
  430 453 456 469 472 482 491 493 502
  510 522 544 556

2 0051H 1 DATA_IN ....... BYTE 94 95 98 101 103 105 106

77 01FDH B1 DEL_EI ...... PROCEDURE STACK=000CH 331

0008H 150 DIGITALKER_TEST ...... PROCEDURE STACK=0012H

9 0108H 1 DONE ....... BYTE 92 93 97 100 102 104 107

9 010EH 1 EMPTY ....... BYTE 29 29 31 30

9 FF ....... LITERALLY '0CH' 39 46 59 73 87

9 0109H 1 FINISHED ....... BYTE 29 29 36 34 40 43 45

11 0114H 1 FIRST ....... BYTE 22 24 24 25

9 0100H 1 FLAG ....... BYTE 29 31 31 36 33 30 34 34

11 0115H 1 FLAG ....... BYTE 22 24 26 25 26

464 0116H 1 I ....... BYTE IN PROC (TRANSLATE) 46 52 28 52 52

2 0050H 1 INDEX ....... BYTE 62 64 64 70 72 72 78
```

```
9 INS ....... LITERALLY '03H' 334
```
MODULE INFORMATION:
CODE AREA SIZE = 0055H 3413D
CONSTANT AREA SIZE = 0000H 0D
VARIABLE AREA SIZE = 0119H 2810
MAXIMUM STACK SIZE = 0012H 180
783 LINES READ
0 PROGRAM WARNINGS
0 PROGRAM ERRORS

END OF PL/M-86 COMPILATION
LIST OF REFERENCES


Marshall, Albert; Towle, Herbert; Shaw, Bon; Bond, Gary; Lohman, Jeff; and Purvis, Ed. Simulated Tank Anti-Armor Gunnery System (STAGS-T). Orlando: Naval Training Equipment Center/Simulation Technology Branch, [1983].


