intel®

APPLICATION NOTE

AP-40

June 1978

Keyboard/Display Scanning With Intel's MCS-48™ Microcomputers

John Wharton
Microcomputer Applications

9800755

INTRODUCTION

This application notes presents a software package for interfacing members of Intel's MCS-48™ family of single-chip microcomputers with keyboards and displays using a minimum of external components. Because of the similarity of the architectures of the various members of the family (the 8035, 8048, 8748, 8039, 8049, 8021, and 8022 microcomputers; also the 8041 and 8741 universal peripheral interfaces in the UPI-41® family), the code included here could run with minor modifications on any member of the family.

Since keyboard and display logic can be just one of several functions handled by a microprocessor, the added cost of including these functions in a system is minimal. In fact, considering the extremely low cost of standard X-Y matrix keyboards and integrated displays, their use is often more cost effective than even a handful of discrete switches and indicators. Thus, the additional flexibility of keyboard input and display output can be added to inexpensive consumer products (such as games, clocks, thermostats, tape recorders, etc.), while producing a net savings in system cost.

Since each potential application will have its own unique combination of keys and display characters, the program is written so that very little modification is needed to interface it with a wide variety of hadware configurations. In general, the only changes required are within the set of initial EQUates at the beginning of the program.

Along with the basic software for driving a multiplexed display and/or scanning and debouncing an X-Y matrix of key switches, a collection of utility subroutines is also included for implementing the most commonly used keyboard and display utility functions, such as copying simple messages onto the display or determining the encoded value of each key in the key matrix. As a result of the versatile architecture and applications-oriented instruction set of the MCS-48 family, the entire package fits into about 250 bytes of internal program ROM or EPROM, leaving the rest of the ROM space for the program to cook the perfect piece of toast, or whatever. By tailoring the software to match a known hardware configuration, or by selecting only those functions needed for a given application, the program size could be even further reduced.

Since what is being presented in this application note is a software package, rather than the usual hardware/software system design, the format of this note is somewhat different from most — it consists primarily of a long program listing reproduced in the following pages. For the most part, the listing is self-explanatory, with comments introducing each subroutine and major code segment. Some parts of this introduction are reproduced in the program listing itself, explaining the configuration of the prototype system. However, an additional bit of explanation would make the listing easier to understand, especially for those readers unfamiliar with the concept of multiplexed displays and keyboards.

In traditional digital system design, various hardware registers or counters were used to hold binary or BCD values which had to be conveyed to the user. The standard way of presenting this information was by connecting each register to a seven-segment encoder (such as the 7447) driving a single display character, as represented by Figure 1. Thus, two ICs, seven current limiting resistors, and about 45 solder joints were required for each digit of output. Consider how traditional techniques might be (mis-)applied in designing a microprocessor system: the designer could add a latch, encoder, and resistors for each digit of the display. Still another latch and decoder could be used to turn on one of the decimal points (if used). The characters displayed could only be a sequence of decimal digits. In the same vein, a large matrix of key switches could be read by installing an MSI TTL priority encoder read by an additional input port. Not only would all this use a lot of extra I/O ports and increase the system price and part count drastically, but the flexibility and reliability of the system would be greatly reduced.

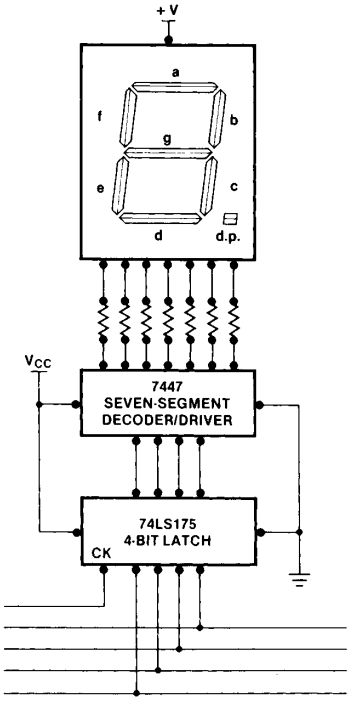

Figure 1. Wrong Way to Design Multiple Digit Displays for Microcomputer Systems

Instead, a scheme of time-multiplexing the display can be used to decrease costs, part count, and interconnections, while allowing a wider range of character types to be used on the display. The techniques used here are fairly typical of today's integrated subsystems designed especially for controlling keyboards and displays (such as in calculators or the Intel® 4269, 8278, and 8279 Keyboard/Display Controller Devices).

In a multiplexed display, all the segments of all the characters are interconnected in a regular two-dimensional array. One terminal of each segment is in common with the other segments of the same character; the other terminal is connected with the same segments of the other characters. This is represented schematically in Figure 2. A digit driver or segment driver is needed for each of these common lines.

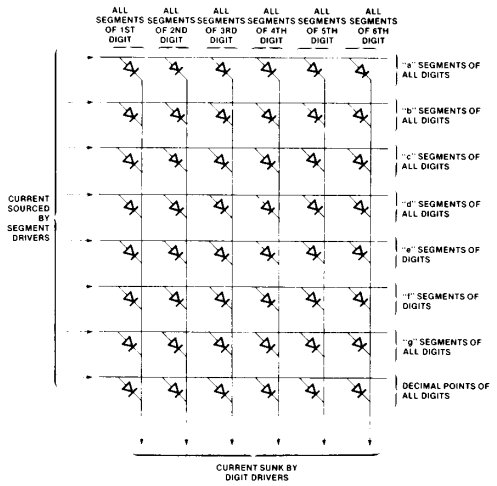

Figure 2. Schematic Representation of 6-Digit, 7-Segment Common-Cathod LED Multiplexed Display

The various characters of the display are not all on at once; rather, only one character at a time is energized. As each character is enabled, some combination of segment drivers is turned on, with the result that a digit appears on the enabled character. (For example, in Figure 3, if segment drivers 'a', 'b', and 'c' were on when character position #6 was enabled, the digit '7' would appear in the left-most place.) Each character is enabled in this way, in sequence, at a rate fast enough to ensure that the display characters seem to be on constantly, with no appearance of flashing or flickering.

In the system presented here, these rapid modifications to the display are all made under the control of the MCS-48™ microcomputer. At periodic intervals the computer quickly turns off all display segments, disables the character now being displayed and enables the next, looks up the pattern of segments for the next character to be displayed, and turns on the appropriate segments. With the next character now turned on, the processor may now resume whatever it had been doing before. The whole display updating task consumes only a small fraction of the processor's time.

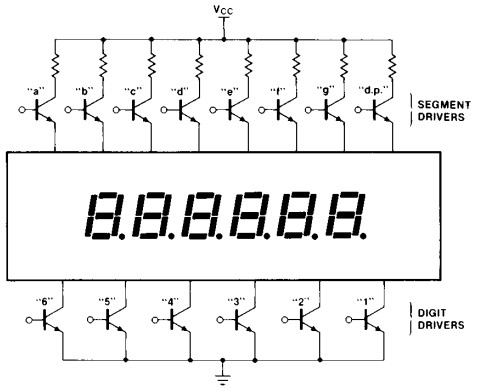

Figure 3. Segment and Digit Drivers used with 6-Position, 7-Segment LED Display

Moreover, since the computer rather than a standard decoder circuit is used to turn the segments off and on, patterns for characters other than decimal digits may be included in the display. Hexadecimal characters, special symbols, and many letters of the alphabet are possible. With sufficient imagination this feature can be exploited for some applications, as suggested by the examples in Figure 4.

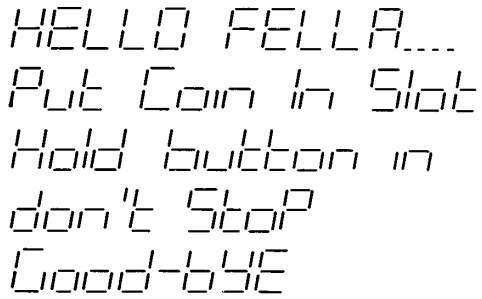

Figure 4. Examples of Typical Messages Possible with Simple 7-Segment Displays

As each character of the display is turned on, the same signal may be used to enable one row of the key matrix. Any keys in that row which are being pressed at the time will then pass the signal on to one of several "return lines", one corresponding to each column of the matrix. (See Figure 5.) By reading the state of these control lines, and knowing which row is enabled, it is possible to compute which (if any) of the keys are down. Note that the keys need not be physically arranged in a rectangular array; Figure 5 is merely a schematic.

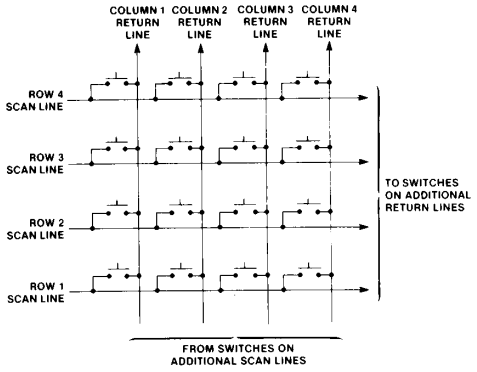

Figure 5. Schematic of X-Y Matrix Multiplexed Keyboard

Since each character is on for only a small fraction of the total display cycle, its segments must be driven with a proportionately higher current so that their brightness averages out over time. This requires character and segment drivers which can handle higher than normal levels of current. Various types of drivers can be used, ranging from specially designed circuits to integrated or discrete transistor arrays. The selection depends on several factors, including the type of display being used (LED, vacuum flourescent, neon, etc.), its size, the number of characters, and the polarity of the individual segments. Some drivers have active high inputs, some active low. Some invert their input logic levels, some do not. Some require insignificant input currents, some present a considerable load. Some systems use external logic to enable one of N characters or to produce the appropriate segment pattern for a given digit, some systems implement these functions through software.

Because of these and the other variables which make each application unique, provisions are made in the first page of symbol EQUates to allow the user to specify such things as the number of characters in the display or the polarity of the drivers used, and the program will be assembled accordingly. The display is refreshed on each timer interrupt, which occurs every 32× (TICK) machine cycles. (One machine cycle occurs every 30 crystal oscillations for the 8021 and 8022, or every 15 oscillations for all other members of the family.) A more detailed explanation of these variables is included in the listing.

Port assignment is also at the discretion of the user — all port references in the listing are "logical" rather than physical port names. The port used to specify which character is enabled is referred to as "PDIGIT". The output segment pattern is written to "PSGMNT" and the keyboard return lines are read by "PINPUT". These logical port names may be assigned to whichever ports the user pleases.

By way of example, the breadboard used to develop and debug this software used a matrix of 16 single-pole pushbuttons and an 8-character common-cathode LED display with right-hand decimal point. No decoders external to the 8748 microcomputer were used; all logic was handled through software. PDIGIT was the 8-bit bus, PSGMNT was port 1, and PINPUT was port 2. The drivers used were 75491 and 75492 logically non-inverting buffers: high level inputs were used to turn a segment or character on. Pull-up resistors were used on the 8748 output lines to source the current levels needed by the buffers. The 8748 was socketed on the breadboard, and was driven with an inexpensive 3.59 MHz television crystal. The short test program included in this listing was used to echo key depressions as they were detected, and to invoke four demonstration subroutines. A summary of the subroutines included in this listing with a short explanation of the function of each is included in Figure 6; Figure 7 shows how the various utilities interact.

KBDIN	**Keyboard Input.** Waits until one keystroke input has been received from the keyboard; determines the meaning or legend of that key; and returns with the encoded value in the accumulator.
CLEAR	Blank out the display.
ENCACC	Encode accumulator with bit pattern corresponding to the segment pattern needed by the display to represent that symbol or character. Uses the value of the accumulator when called to access a table containing the patterns for all legal input values.
WDISP	**Write into Display.** Writes the bit pattern in the accumulator into the next character position of the display. Maintains a character position counter so that repeated calls will automatically write characters into sequential positions.
RENTRY	**Right-hand Entry.** Stores the accumulator segment pattern in the display in the right-most character position. Shifts all other characters to the left one place.
PRINT	Print a string of arbitrary characters onto the display. Useful for prompting messages, warnings, etc. Uses a table of segment patterns in ROM, so that messages will not be restricted to numbers, letters, etc.
FILL	Fill the display with the character pattern in the accumulator. Useful for writing dashes, segment test patterns, etc., into all character positions.
ECHO	Wait for a key to be pressed by the operator and write that key onto the display. Used for providing feedback to the operator when entering numeric data, etc.
RDPADD	Adds or deletes a decimal point to the character at the right-hand side of the display, for entering floating point numbers.
HOLD	Called when a key is known to be down. Does not return until all keys have been released. Used for organ-type keyboards, or when some action should not be initiated until the key invoking that action has been released.
DELAY	Provides a crude real-time delay corresponding to the value of the accumulator when called. Can be used to cause display characters to blink, to momentarily flash information, to enable a buzzer, etc. Could also be used by the program when delays are needed, such as to slow down the computer reaction rate while playing a game against the human operator.

Figure 6. Utility Subroutine Definitions

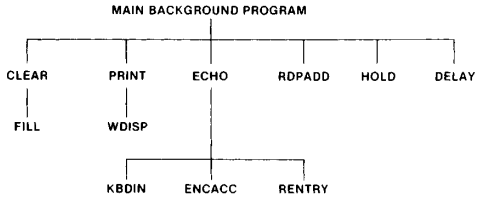

Figure 7. Subroutine Interrelationships

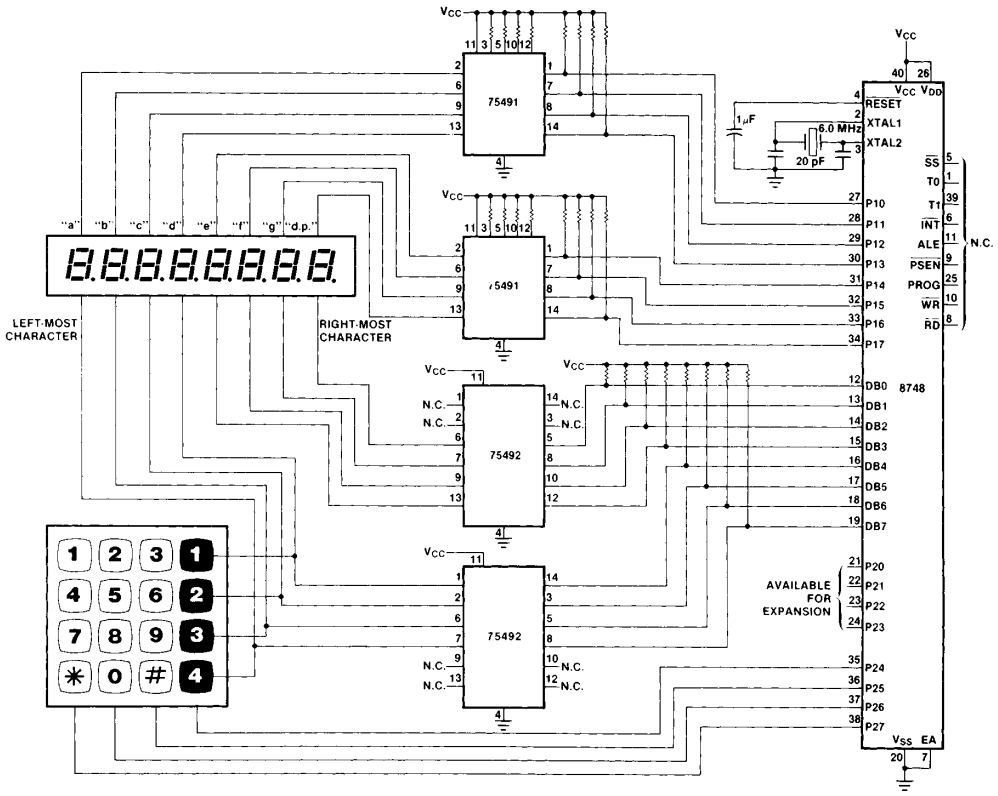

Figure 8 Prototype System Schematic

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0
AP40  INTEL MCS-48 KEYBOARD/DISPLAY APPLICATION NOTE APPENDIX

 LOC  OBJ        SEQ        SOURCE STATEMENT

                   1 $MACROFILE XREF
                   2 $TITLE('AP40  INTEL MCS-48 KEYBOARD/DISPLAY APPLICATION NOTE APPENDIX')
                   3 ;
                   4 ;THE FOLLOWING SOFTWARE PACKAGE PROVIDES A SEVEN SEGMENT DISPLAY
                   5 ;INTERFACE FOR MICROCOMPUTERS IN THE INTEL MCS-48 FAMILY
                   6 ;THE CODE IS WRITTEN SO THAT VARIOUS HARDWARE
                   7 ;CONFIGURATIONS CAN BE ACCOMODATED BY REDEFINING THE INITIAL VARIABLES
                   8 ;IN MOST SITUATIONS, THE KEYBOARD/DISPLAY INTERFACE WILL BE REQUIRED TO
                   9 ;IMPLEMENT MORE SOPHISTICATED SINGLE-CHIP SYSTEMS (CALCULATORS,SCALES,CLOCKS,
                  10 ;ETC.), WITH SECTIONS OF THE FOLLOWING CODE SELECTED AND MODIFIED AS NECESSARY
                  11 ;FOR EACH APPLICATION
                  12 ;
                  13 ;A SINGLE SUBROUTINE (CALLED REFRSH) IS USED TO IMPLEMENT BOTH THE DISPLAY
                  14 ;MULTIPLEXING AND KEYBOARD SCANNING, USING THE SAME SIGNAL BOTH TO ENABLE
                  15 ;ONE CHARACTER OF THE DISPLAY AND TO STROBE ONE ROW OF THE X-Y KEY MATRIX
                  16 ;THE SUBROUTINE MUST BE CALLED SUFFICIENTLY OFTEN TO ENSURE THE DISPLAY
                  17 ;CHARACTERS DO NOT FLICKER- AT LEAST 50 COMPLETE DISPLAY SCANS PER SECOND
                  18 ;TO ACCOMODATE SWITCHES OF ARBITRARY CHEAPNESS, THE DEBOUNCE TIME CAN BE
                  19 ;SET TO BE ANY DESIRED NUMBER OF COMPLETE SCANS
                  20 ;THUS THE DEBOUNCE TIME IS A FUNCTION OF BOTH THE SCAN RATE AND THE VALUE
                  21 ;OF CONSTANT 'DEBNCE'
                  22 ;
                  23 ;IN THIS LISTING, THE INTERNAL TIMER IS USED TO GENERATE INTERRUPTS THAT
                  24 ;SERVE AS A TIME BASE FOR THE REFRESH SUBROUTINE.
                  25 ;ALTERNATE TIME BASES MIGHT BE AN EXTERNAL OSCILLATOR (DRIVING THE INTERRUPT
                  26 ;PIN OR POLLED BY A TEST OR INPUT PIN), A SOFTWARE DELAY LOOP IN THE BACKGROUND
                  27 ;PROGRAM, OR PERIODIC CALLS TO THE SUBROUTINE FROM THROUGHOUT THE USER'S PROGRAM
                  28 ;AT APPROPRIATE PLACES
                  29 ;IN THESE CASES, THE CODE STARTING AT LABEL TIINT (TIMER INTERRUPT) AND TIRET
                  30 ;(TIINT RETURN) COULD STILL BE USED TO SAVE AND RESTORE ACCUMULATOR CONTENTS
                  31 ;THE INTERRUPT SERVICING ROUTINE SELECTS REGISTER BANK 1
                  32 ;FOR THE NEEDED REGISTERS
                  33 ;
                  34 ;
                  35 ;WRITTEN BY JOHN WHARTON, INTEL SINGLE-CHIP COMPUTER APPLICATIONS
                  36 ;
                  37 $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0
AP40  INTEL MCS-48 KEYBOARD/DISPLAY APPLICATION NOTE APPENDIX

 LOC  OBJ        SEQ        SOURCE STATEMENT

                   38 ;IN THIS IMPLEMENTATION OF THE DISPLAY SCAN, IT IS ASSUMED THAT THERE WILL
                   39 ;BE RELATIVELY LITTLE I/O OTHER THAN FOR THE KEYBOARD/DISPLAY
                   40 ;IF THIS IS THE CASE, THEN THERE IS NO NEED FOR FOR ANY ADDITIONAL EXTERNAL
                   41 ;LOGIC (SUCH AS ONE-OF-EIGHT DECODERS OR SEVEN-SEGMENT ENCODERS), THOUGH
                   42 ;THERE WILL STILL BE A NEED FOR CURRENT OR VOLTAGE DRIVERS, ACCORDING TO
                   43 ;THE TYPE OF DISPLAY BEING USED
                   44 ;
                   45 ;IN THIS LISTING, THE PROCESSOR I/O PORTS ARE LOGICALLY DIVIDED AS FOLLOWS
                   46 ;
                   47 ;PDIGIT-EIGHT BIT PORT USED TO ENABLE, ONE AT A TIME, THE INDIVIDUAL
                   48 ;       CHARACTERS OF AN EIGHT DIGIT SEVEN-SEGMENT DISPLAY, WHILE ALSO
                   49 ;       STROBING THE ROWS OF AN X-Y MATRIX KEYBOARD
                   50 ;       BIT7 ENABLES THE LEFTMOST CHARACTER AND THE BOTTOM ROW OF THE KBD.
                   51 ;       BIT4 ENABLES THE TOP ROW OF THE 4X4 KBD AND THE FOURTH CHARACTER.
                   52 ;       BIT0 ENABLES THE RIGHTMOST CHARACTER
                   53 ;       (A 4X8 KEYBOARD COULD BE STROBED BY ALSO USING BIT3-BIT0
                   54 ;       AND EXTENDING OR ELIMINATING THE TABLE, "LEGNDS".)
                   55 ;       THE ENABLING OF ONE BIT (ACTIVE HIGH OR LOW) IS ACCOMODATED BY
                   56 ;       ACCESSING A LOOK-UP TABLE CALLED CHRSTB
                   57 ;       THIS TECHNIQUE TAKES ABOUT FOUR BYTES MORE ROM THAN A TECHNIQUE
                   58 ;       OF ROTATING A 'ONE' THROUGH A FIELD OF 'ZEROES' IN THE ACC
                   59 ;       AN APPROPPIATE NUMBER OF TIMES, BUT IT ALLOWS SOME ADDITIONAL
                   60 ;       FLEXABILITY.  IF THE DRIVERS BEING USED HAVE A COMBINATORIAL INPUT
                   61 ;       (AS IN THE 7545X FAMILY OF HIGH-CURRENT, HIGH-VOLTAGE DRIVERS),
                   62 ;       THE CHRSTB TABLE COULD PPOVIDE ENCODED OUTPUTS.  NINE DIGITS, FOR
                   63 ;       EXAMPLE, COULD BE ENABLED WITH SIX BITS OF (BUFFERED) OUTPUT
                   64 ;       (001001,001010,001100,010001,010010,010100,100001,100010,100100)
                   65 ;       IF I/O LINES NEED TO BE CONSERVED, OR IF MANY DIGITS
                   66 ;       MUST BE DISPLAYED, AN EXTERNAL DECODER COULD BE ADDED TO THE SYSTEM
                   67 ;       DURING CHARACTER TRANSITIONS A 'BLANK' CHARACTER IS
                   68 ;       EXPLICITLY WRITTEN TO THE DISPLAY.  THUS,
                   69 ;       THERE WILL BE NO CHARACTER 'SHADOWING' CAUSED BY THE
                   70 ;       FACT THAT THE HARDWARE OR SOFTWARE DECODER KEEPS ONE
                   71 ;       OUTPUT, AND THUS ONE CHARACTER, ACTIVE AT ALL TIMES
                   72 ;
                   73 ;PSGMNT-EIGHT BIT PORT TO ENABLE THE SEVEN SEGMENTS & D.P. OF A STANDARD
                   74 ;       DISPLAY
                   75 ;       BIT7-BIT0 CORRESPOND TO THE DP AND SEGMENTS G THROUGH A, RESPECTIVELY
                   76 ;       IT IS POSSIBLE TO ACCOMODATE
                   77 ;       DRIVERS WHICH ARE EITHER LOGICALLY INVERTING OR NON-INVERTING BY
                   78 ;       SETTING VARIABLE 'SEGPOL' (SEGMENT POLARITY).
                   79 ;       NOTE THAT BY HAVING ARBITRARY CONTROL OVER EACH SEGMENT, NON-NUMERIC
                   80 ;       CHARACTERS CAN BE REPRESENTED ON A SEVEN SEGMENT DISPLAY.
                   81 ;       AS SHOWN IN EXAMPLE SUBROUTINE 'TEST2'.
                   82 ;
                   83 $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0
AP40: INTEL MCS-48 KEYBOARD/DISPLAY APPLICATION NOTE APPENDIX

 LOC  OBJ        SEQ        SOURCE STATEMENT

                   84 ;PINPUT-FOUR HIGH-ORDER BITS USED AS INPUTS FROM THE KEYBOARD RETURN LINES
                   85 ;       ASSUMES THAT A KEY DOWN IN  HE CURRENTLY ENABLED ROW WOULD RETURN
                   86 ;       A LOW LEVEL.
                   87 ;       IN THIS CASE, BIT7 RETURNS THE LEFTMOST COLUMN, BIT4 THE RIGHTMOST
                   88 ;       THE HIGH-ORDER BITS ARE USED SO THAT IF AN OFF-CHIP DECODER IS USED
                   89 ;       TO ENABLE UP TO 16 CHARACTERS. FOR EXAMPLE, IT COULD BE DRIVEN BY
                   90 ;       THE LOW ORDER BITS OF THE SAME PORT.
                   91 ;       NOTE ALSO THAT IF A SIXTEEN KEY MATRIX WERE ELECTRICALLY ORGANIZED
                   92 ;       IN A 2X8 ARRAY, ONLY TWO RETURN LINES WOULD BE NEEDED.
                   93 ;       (IN THIS CASE, PERHAPS T0 AND T1 COULD BE USED FOR INPUT BITS.)
                   94 ;
                   95 ;PULL-UP RESISTORS ON THE RETURN LINES MIGHT BE IN ORDER IF THERE IS ANY
                   96 ;POSSIBILITY OF A HIGH-IMPEDENCE CONDUCTIVE PATH THROUGH THE SWITCH WHEN
                   97 ;IT IS SUPPOSED TO BE 'OPEN'.
                   98 ;(THIS PHENOMENON HAS ACTUALLY BEEN OBSERVED.)
                   99 ;
                  100 ;THE DRIVERS USED IN THE PROTOTYPE WERE ALL NON-INVERTING IN THAT
                  101 ;A HIGH LEVEL ON AN OUTPUT LINE IS USED TO TURN A CHARACTER OR SEGMENT ON
                  102 ;THERE ARE A TOTAL OF SEVEN I/O LINES LEFT OVER
                  103 ;
                  104 ;THE ALGORITHM FOR DRIVING THE DISPLAY USES A BLOCK OF INTERNAL RAM
                  105 ;AS DISPLAY REGISTERS, WITH ONE BYTE CORRESPONDING TO EACH CHARACTER OF THE
                  106 ;DISPLAY.  THE EIGHT BITS OF EACH BYTE CORRESPOND TO THE SEVEN SEGMENTS & DP
                  107 ;OF EACH CHARACTER.  IF AN EXTERNAL ENCODER IS USED (SUCH AS A FOUR-BIT TO
                  108 ;SEVEN-SEGMENT ENCODER OR A ROM FOR TRANSLATING ASCII TO
                  109 ;SIXTEEN-SEGMENT "STARBURST" DISPLAY PATTERNS), THE TABLE ENTRIES WOULD HOLD
                  110 ;THE CHARACTER CODES.  (IN THE FORMER CASE, AN UNUSED BIT COULD BE USED TO
                  111 ;ENABLE THE D.P.)
                  112 ;THUS, WRITING CHARACTERS TO THE DISPLAY FROM THE BACKGROUND PROGRAM
                  113 ;REALLY ENTAILS WRITING THE APPROPRIATE SEGMENT
                  114 ;PATTERNS TO A DISPLAY REGISTER.  THE ACTUAL OUTPUTTING IS AUTOMATIC.
                  115 ;THE LEFTMOST CHARACTER CORRESPONDS TO THE LAST BYTE OF THE DISPLAY
                  116 ;REGISTERS, AND IS ACCESSED BY NEXTPL=8 (SEE SOURCE); THE RIGHTMOST
                  117 ;CHARACTER IS THE FIRST DISPLAY BYTE, WHEN NEXTPL=1.
                  118 ;UTILITY SUBROUTINES ARE INCLUDED HERE TO TRANSLATE FOUR BIT NUMBERS TO HEX
                  119 ;DIGIT PATTERNS, AND WRITE THEM INTO THE DISPLAY REGISTERS SEQUENTIALLY
                  120 ;(EITHER FILLING FROM THE LEFT- H.P. CALCULATOR STYLE OR FROM THE
                  121 ;RIGHT- T.I. STYLE, SUBROUTINES WDISP AND RENTRY, RESPECTIVELY).
                  122 ;
                  123 ;THE KEYBOARD SCANNING ALGORITHM SHOWN HERE REQUIRES A KEY BE DOWN FOR
                  124 ;SOME NUMBER OF COMPLETE DISPLAY SCANS TO BE ACKNOWLEGED.  SINCE IT IS
                  125 ;INTENDED FOR 'ONE-FINGER' OPERATION, TWO-KEY ROLLOVER/N-KEY LOCKOUT HAS
                  126 ;BEEN IMPLEMENTED.  HOWEVER, MODIFICATIONS WOULD BE POSSIBLE TO ALLOW, FOR
                  127 ;EXAMPLE, ONE KEY IN THE MATRIX TO BE USED AS A SHIFT KEY OR CONTROL KEY
                  128 ;TO BE HELD DOWN WHILE ANOTHER KEY IN THE MATRIX IS PRESSED.  (SEE NOTE WITHIN
                  129 ;THE BODY OF THE LISTING.)
                  130 ;
                  131 $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0            PAGE    4
AP40  INTEL MCS-48 KEYBOARD/DISPLAY APPLICATION NOTE APPENDIX

  LOC  OBJ        SEQ        SOURCE STATEMENT

                   132 ;(BE AWARE THAT NO MORE THAN TWO KEYS CAN EVER BE DOWN UNLESS DIODES
                   133 ;ARE PLACED IN SERIES WITH ALL OF THE SWITCHES- CERTAINLY NOT THE CASE FOR EL
                   134 ;CHEAPO KEYBOARDS- BECAUSE SOME COMBINATIONS OF THREE KEYS DOWN WILL RESULT
                   135 ;IN A 'PHANTOM' FOURTH KEY BEING PERCEIVED
                   136 ;THE PHANTOM KEY WOULD BE THE FOURTH 'CORNER' WHEN THREE KEYS FORMING
                   137 ;A RECTANGULAR PATTERN (IN THE X-Y KEY MATRIX) ARE DOWN.)
                   138 ;IF DIODES ARE PLACED IN THE SCANNING ARRAY, CONSIDERATIONS MUST BE MADE
                   139 ;ABOUT HOW THE DIODE VOLTAGE DROP WILL AFFECT INPUT LOGIC LEVELS
                   140 ;
                   141 ;WHEN A DEBOUNCED KEY IS DETECTED, THE NUMBER OF ITS POSITION IN THE KEY
                   142 ;MATRIX (LEFT-TO-RIGHT, BOTTOM-TO-TOP, STARTING FROM 00) IS PLACED INTO
                   143 ;RAM LOCATION 'KBDBUF'.  AN INPUT SUBROUTINE THEN NEED ONLY READ THIS LOCATION
                   144 ;REPEATEDLY TO DETERMINE WHEN A KEY HAS BEEN PRESSED.  WHEN A KEY IS DETECTED,
                   145 ;A SPECIAL CODE BYTE SHOULD BE WRITTEN BACK TO INTO 'KBDBUF' TO PREVENT
                   146 ;REPEATED DETECTIONS OF THE SAME KEY.
                   147 ;THE ROUTINE 'KBDIN' DEMONSTRATES A TYPICAL INPUT PROTOCOL, ALONG WITH A METHOD
                   148 ;FOR TRANSLATING A KEY POSITION TO ITS ASSOCIATED SIGNIFICANCE BY ACCESSING
                   149 ;TABLE 'LEGNDS' IN ROM.
                   150 ;
                   151 $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0            PAGE    5
AP40: INTEL MCS-48 KEYBOARD/DISPLAY APPLICATION NOTE APPENDIX

 LOC  OBJ        SEQ        SOURCE STATEMENT

                  152 ;*******************************************************
                  153 ;
                  154 ;       INITIAL EQUATES TO DEFINE SYSTEM CONFIGURATION
                  155 ;
                  156 ;*******************************************************
                  157 ;
0010              158 PDIGIT  EQU     BUS     ;USED TO ENABLE CHARACTERS AND STROBE ROWS OF KEYBOARD
0008              159 PSGMNT  EQU     P1      ;USED TO TURN ON SEGMENTS OF CURRENTLY ENABLED DIGIT
0009              160 PINPUT  EQU     P2      ;PORT USED TO SCAN FOR KEY CLOSURES
                  161                         ;(NOTE THAT THIS PORT ALLOCATION USES THE HIGHER
                  162                         ;CURRENT SOURCING ABILITY OF THE BUS TO SWITCH ON THE
                  163                         ;DIGIT DRIVERS, AND LEAVES P23-P20 FREE FOR USING
                  164                         ;AN 8243 PORT EXPANDER IN THE SYSTEM.)
                  165 ;
0000              166 POSLOG  EQU     00H
00FF              167 NEGLOG  EQU     0FFH
                  168 ;
0000              169 CHRPOL  EQU     POSLOG  ;DEFINES WHETHER OUTPUT LINES ARE ACTIVE HI OR LOW
0000              170 SEGPOL  EQU     POSLOG  ;\FOR DRIVING CHARACTERS AND SEGMENT PATTERNS
00F0              171 INPMSK  EQU     0F0H    ;DEFINES BITS USED AS INPUT
                  172 ;
0008              173 CHARNO  EQU     8       ;NUMBER OF DIGITS IN DISPLAY
0004              174 NROWS   EQU     4       ;ROWS OF KEYS (LESS THAN OR EQUAL TO CHARNO)
0004              175 NCOLS   EQU     4       ;LESSER DIMENSION OF KEYBOARD MATRIX
                  176 ;
FFF0              177 TICK    EQU     -10H    ;DETERMINES INTERRUPT INTERVAL
0004              178 DEBNCE  EQU     4       ;NUMBER OF SUCESSIVE SCANS BEFORE KEY CLOSURE ACCEPTED
0000              179 BLANK   EQU     00H     ;CODE TO BLANK DISPLAY CHARACTERS.
                  180                         ;(WOULD BE 20H IF ASCII DECODING ROM USED OR 0FH IF
                  181                         ;7447-TYPE SEVEN-SEGMENT DECODER EXTERNAL TO 8748)
                  182 ;
000F              183 ENCMSK  EQU     0FH     ;SELECTS WHICH BITS ARE RELEVANT TO ENCACC SUBROUTINE
                  184 ;
                  185 $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0            PAGE    6
AP40: INTEL MCS-48 KEYBOARD/DISPLAY APPLICATION NOTE APPENDIX

 LOC  OBJ        SEQ        SOURCE STATEMENT

                  186 ;*********************************************************
                  187 ;
                  188 ;       BANK 0 REGISTERS USED
                  189 ;
                  190 ;POINTERS USED FOR INDIRECT RAM ACCESSING:
0000              191 PNTR0   EQU     R0
0001              192 PNTR1   EQU     R1
0007              193 NEXTPL  EQU     R7      ;USED TO KEEP TRACK OF CHARACTER POSITION BEING
                  194                         ;WRITTEN INTO
                  195 ;
                  196 ;*********************************************************
                  197 ;
                  198 ;       BANK 1 REGISTER ALLOCATION
                  199 ;.
                  200 ;PNTR0  EQU     R0      (ALREADY DEFINED)
                  201 ;PNTR1  EQU     R1
0002              202 ASAVE   EQU     R2      ;HOLDS ACCUMULATOR VALUE DURING SERVICE ROUTINE
0004              203 ROTPAT  EQU     R4      ;USED TO HOLD INPUT PATTERN BEING ROTATED THROUGH CY
0005              204 ROTCNT  EQU     R5      ;COUNTS NUMBER OF BITS ROTATED THROUGH CY
0006              205 LASTKY  EQU     R6      ;HOLDS KEY POSITION OF LAST KEY DEPRESSION DETECTED
0007              206 CURDIG  EQU     R7      ;HOLDS POSITION OF NEXT CHARACTER TO BE DISPLAYED
                  207 ;
                  208 ;*********************************************************
                  209 ;
                  210 ;       DATA RAM ALLOCATION
                  211 ;
0020              212 NREPTS  EQU     32      ;KEEPS TRACK OF SUCCESSIVE READS OF SAME KEYSTROKE
0021              213 KEYLOC  EQU     33      ;INCREMENTED AS SUCCESSIVE KEY LOCATIONS SCANNED
0022              214 KBDBUF  EQU     34      ;CARRIES POSITION OF DEBOUNCED KEY FROM REFRSH ROUTINE
                  215                         ;\ BACK TO BACKGROUND PROGRAM
0023              216 RDELAY  EQU     35      ;NON-ZERO WHEN DISPLAY IN PROGRESS
                  217 ;
                  218 ;       THE LAST <CHARNO> REGISTERS HOLD THE DISPLAY SEGMENT PATTERNS
                  219 ;
0037              220 SEGMAP  EQU     (63-CHARNO)     ;BASE OF REGISTER ARRAY FOR DISPLAY PATTERNS
                  221                                 ;\ (COULD BE ANYWHERE IN INTERNAL RAM)
                  222 ;
                  223 ;*********************************************************
                  224 ;
                  225 ;       NOTE THAT LASTKY, CURDIG, AND F1 RETAIN STATUS INFORMATION FROM
                  226 ;       ONE INTERRUPT TO THE NEXT.  ALL OTHER REGISTERS MAY BE USED IN
                  227 ;       THE USER'S OWN INTERRUPT SERVICING ROUTINE
                  228 ;
                  229 ;*********************************************************
                  230 ;
                  231 $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0            PAGE    7
AP40: INTEL MCS-48 KEYBOARD/DISPLAY APPLICATION NOTE APPENDIX

 LOC  OBJ        SEQ        SOURCE STATEMENT

                  232 ;
                  233 ;*********************************************************
                  234 ;
 0000             235 ORG     000H
 0000 0460        236         JMP     INIT
                  237 ;
                  238 ;
                  239 ;*********************************************************
                  240 ;
 0007             241 ORG     007H
                  242 ;
                  243 ;TIINT  TIMER INTERRUPT SUBROUTINE
                  244 ;       CALL MADE TO LOC 007H WHEN TIMER TIMES OUT
                  245 ;       TIMER CAN BE RE-INITIALLIZED AT THIS POINT IF DESIRED
                  246 ;       USED HERE TO CAUSE THE DISPLAY REFRESH AND KEY SCAN ROUTINES TO
                  247 ;       BE CALLED PERIODICALLY.
 0007 D5          248 TIINT:  SEL     RB1
 0008 AA          249         MOV     ASAVE,A
 0009 23F0        250         MOV     A,#TICK
 000B 62          251         MOV     T,A             ;RELOAD TIMER INTERVAL
                  252 ;
                  253 ;*********************************************************
                  254 ;
                  255 ;       THE USER'S OWN TIMER INTERRUPT ROUTINE (IF IT EXISTS) COULD
                  256 ;       BE PLACED AT THIS POINT
                  257 ;
                  258 ;*********************************************************
                  259 ;
 000C 1410        260         CALL    REFRSH          ;CAUSE DISPLAY TO BE UPDATED
                  261 ;
                  262 ;       THE COMPLETE INTERRUPT ROUTINE SHOULD BE COPIED HERE
                  263 ;       TO SAVE A FULL LEVEL OF SUBROUTINE NESTING.
                  264 ;       IT WAS WRITTEN AS A SUBROUTINE HERE FOR THE SAKE OF CLARITY.
                  265 ;
                  266 ;*********************************************************
                  267 ;
                  268 ;TIRET  TIMER INTERRUPT RETURN CODE- RESTORES MCC VALUE
 000E FA          269 TIRET:  MOV     A,ASAVE
 000F 93          270         RETR
                  271 ;
                  272 $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0            PAGE    8
AP40: INTEL MCS-48 KEYBOARD/DISPLAY APPLICATION NOTE APPENDIX

 LOC  OBJ        SEQ        SOURCE STATEMENT

                  273 ;*********************************************************
                  274 ;REFRSH SUBROUTINE TO MULTIPLEX SEVEN-SEGMENT DISPLAYS
                  275 ;       EACH CALL CAUSES THE NEXT CHARACTER TO BE DISPLAYED,
                  276 ;       ACCORDING TO THE CONTENTS OF THE SEGMAP REGISTER ARRAY
                  277 ;       REFRSH SHOULD BE CALLED AT LEAST EVERY MSEC OR SO
                  278 ;*********************************************************
                  279 ;
0010 2300         280 REFRSH: MOV     A,#BLANK XOR SEGPOL
0012 39           281         OUTL    PSGMNT,A        ;WRITE BLANK PATTERN TO SEG DRIVERS
0013 2357         282 REFR1:  MOV     A,#CHRSTB       ;LOOK UP DIGIT ENABLE PATTERN
0015 6F           283         ADD     A,CURDIG        ;ADD CURDIG DISPLACEMENT
0016 A3           284         MOVP    A,@A            ;ENABLE ONE BIT OF ACCUMULATOR
0017 02           285         OUTL    PDIGIT,A        ;ENERGIZE CHARACTER
                  286 ;
                  287                                 ;WRITE NEXT SEGMENT PATTERN
0018 2337         288         MOV     A,#SEGMAP       ;LOAD BASE OF REGISTER ARRAY
001A 6F           289         ADD     A,CURDIG        ;ADD CURDIG DISPLACMENT
001B A9           290         MOV     PNTR1,A
001C F1           291         MOV     A,@PNTR1        ;LOAD ACC W/ NEXT SEGMENT PATTERN
001D 39           292         OUTL    PSGMNT,A        ;ENABLE APPROPRIATE SEGMENTS
                  293 ;
                  294 ;*********************************************************
                  295 ;       THE NEXT CHARACTER IS NOW BEING DISPLAYED.
                  296 ;       THE KEYBOARD SCAN ROUTINE IS INTEGRATED INTO THE DISPLAY SCAN.
                  297 ;       WITH THE CURRENT ROW ENERGIZED, CHECK IF THERE ARE ANY INPUTS
                  298 ;*********************************************************
                  299 ;
001E B821         300 SCAN:   MOV     PNTR0,#KEYLOC   ;SET POINTER FOR SEVERAL KEYLOC REFERENCES
0020 0A           301         IN      A,PINPUT        ;LOAD ANY SWITCH CLOSURES
                  302 ;
                  303 ;#########################################################################
                  304 ;##     THIS BLOCK OF CODE IS NOT NEEDED BY THE KEYBOARD SCAN LOGIC     ###
                  305 ;##     HOWEVER, ITS INCLUSION WOULD SPEED THINGS UP A BIT BY          ###
                  306 ;##     SKIPPING OVER ROWS IN WHICH NO KEYS ARE DOWN.                  ###
                  307 ;##     IT WAS OMITTED HERE TO CONSERVE ROM SPACE, BUT MIGHT BE        ###
                  308 ;##     RESTORED IF VERY LARGE KEYBOARDS (ESPECIALLY THOSE WITH EIGHT  ###
                  309 ;##     KEYS PER ROW) ARE TO BE USED WITH THIS ALGORITHM               ###
                  310 ;#########################################################################
                  311 ;##     CPL     A               ;ANY CLOSURES DETECTED ARE NOW ONE BITS ###
                  312 ;##     ANL     A,#INPMSK                                              ###
                  313 ;##     JNZ     SCAN1   ;-IF A KEY IN THE CURRENTLY ENABLED ROW IS DOWN ###
                  314 ;##;    NO KEY IS NOW DOWN SO THE KEYLOC COUNT MAY BE UPDATED DIRECTLY  ###
                  315 ;##     MOV     A,@PNTR0                                               ###
                  316 ;##     ADD     A,#NCOLS                                               ###
                  317 ;##     MOV     @PNTR0,A                                               ###
                  318 ;##     JMP     SCAN6                                                  ###
                  319 ;#########################################################################
                  320 ;##     IF THIS CODE IS USED, SUBSTITUTE THE 'JC  SCAN5' FOUR LINES     ###
                  321 ;##     HENCE WITH 'JNC  SCAN5' TO ACCOMODATE THE INVERTED POLARITY     ###
                  322 ;#########################################################################
                  323 $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0            PAGE    9
AP40: INTEL MCS-48 KEYBOARD/DISPLAY APPLICATION NOTE APPENDIX

 LOC  OBJ        SEQ        SOURCE STATEMENT

                  324 ;*********************************************************
                  325 ;       ROTATE BITS THROUGH THE CY WHILE INCREMENTING KEYLOC
                  326 ;*********************************************************
                  327 ,
 0021 BD04        328 SCAN1:  MOV     ROTCNT,#NCOLS   ;SET UP FOR <NCOLS> LOOPS THROUGH 'NXTLOC'
 0023 F7          329 NXTLOC: RLC     A
 0024 AC          330         MOV     ROTPAT,A        ;SAVE SHIFTED BIT PATTERN
 0025 F63F        331         JC      SCAN5           ;ONE BIT IN CY INDICATES KEY NOT DOWN
                  332 ,
                  333 ;*********************************************************
                  334 ;
                  335 ;       AT THIS POINT IT HAS JUST BEEN DETERMINED THAT THE VALUE
                  336 ;       OF KEYLOC IS THE POSITION OF A KEY WHICH IS NOW DOWN
                  337 ;       THE FOLLOWING CODE DEBOUNCES THE KEY, ETC.
                  338 ;       IF MODIFICATIONS TO THE KEYBOARD LOGIC, I.E. THE INCLUSION
                  339 ;       OF A SHIFT, CONTROL, OR MODE KEY IN THE KEY MATRIX ITSELF)
                  340 ;       ARE DESIRED, THEY SHOULD BE MADE AT THIS POINT, BEFORE
                  341 ;       THE DEBOUNCE LOGIC BEGINS.  FOR EXAMPLE, AT THIS POINT
                  342 ;       KEYLOC COULD BE COMPARED AGAINST THE POSITION OF THE MODE
                  343 ;       KEY, AND IF THEY MATCH SET SOME FLAG BIT AND JUMP TO
                  344 ;       LABEL 'SCAN5'.  OR, BY COMPARING KEYLOC AGAINST THE LAST
                  345 ;       KEY DEBOUNCED, IMMEDIATE TWO-KEY ROLLOVER COULD BE
                  346 ;       IMPLEMENTED.
                  347 ;
                  348 ;*********************************************************
                  349 ;
 0027 A5          350         CLR     F1              ;MARK THAT AT LEAST ONE KEY WAS DETECTED
 0028 B5          351         CPL     F1              ;\ IN THE CURRENT SCAN
                  352 ;
                  353 ;*********************************************************
                  354 ;       A KEYSTROKE WAS DETECTED FOR THE CURRENT COLUMN.  ITS
                  355 ;       POSITION IS IN REGISTER KEYLOC.  SEE IF SAME KEY SENSED LAST CYCLE
                  356 ;*********************************************************
                  357 ,
 0029 F0          358         MOV     A,@PNTR0        ;PNTR0 STILL HOLDS #KEYLOC
 002A 2E          359         XCH     A,LASTKY
 002B DE          360         XRL     A,LASTKY
 002C B820        361         MOV     PNTR0,#NREPTS   ;PREPARE TO CHECK AND/OR MODIFY REPEAT COUNT
 002E C634        362         JZ      SCAN3
                  363 ;
                  364 $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0            PAGE   10
AP40  INTEL MCS-48 KEYBOARD/DISPLAY APPLICATION NOTE APPENDIX

 LOC  OBJ        SEQ        SOURCE STATEMENT

                  365 ;*********************************************************
                  366 ;       A DIFFERENT KEY WAS READ ON THIS CYCLE THAN ON THE PREVIOUS CYCLE.
                  367 ;       SET NREPTS TO THE DEBOUNCE PARAMETER FOR A NEW COUNTDOWN
                  368 ;*********************************************************
                  369 ;
 0030 B004        370         MOV     @PNTR0,#DEBNCE
 0032 043F        371         JMP     SCAN5
                  372 ;
                  373 ;*********************************************************
                  374 ;       SAME KEY WAS DETECTED AS ON PREVIOUS CYCLE
                  375 ;       LOOK AT NREPTS.  IF ALREADY ZERO, DO NOTHING
                  376 ;       ELSE DECREMENT NREPTS
                  377 ;       IF THIS RESULTS IN ZERO, MOVE LASTKY INTO KBDBUF
                  378 ;*********************************************************
                  379 ;
 0034 F0          380 SCAN3:  MOV     A,@PNTR0
 0035 C63F        381         JZ      SCAN5           ;IF ALREADY ZERO
 0037 07          382         DEC     A               ;INDICATE ONE MORE SUCCESIVE KEY DETECTION
 0038 A0          383         MOV     @PNTR0,A
 0039 963F        384         JNZ     SCAN5           ;IF DECREMENT DOES NOT RESULT IN ZERO
 003B FE          385         MOV     A,LASTKY
 003C B822        386         MOV     PNTR0,#KBDBUF
 003E A0          387         MOV     @PNTR0,A        ;TO MARK NEW KEY CLOSURE
                  388 ;
 003F B821        389 SCAN5:  MOV     PNTR0,#KEYLOC
 0041 10          390         INC     @PNTR0
 0042 FC          391         MOV     A,ROTPAT
 0043 ED23        392         DJNZ    ROTCNT,NXTLOC
                  393 ;
                  394 ;
 0045 EF57        395 SCAN6:  DJNZ    CURDIG,SCAN9
                  396 ;
                  397 $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0            PAGE   11
AP40: INTEL MCS-48 KEYBOARD/DISPLAY APPLICATION NOTE APPENDIX

 LOC  OBJ        SEQ        SOURCE STATEMENT

                  398 ;
                  399 ;*********************************************************
                  400 ;       THE FOLLOWING CODE SEGMENT IS USED BY THE KEYBOARD SCANNING ROUTINE
                  401 ;       IT IS EXECUTED ONLY AFTER A REFRESH SEQUENCE OF ALL
                  402 ;       THE CHARACTERS IN THE DISPLAY IS COMPLETED
                  403 ;*********************************************************
                  404 ;
0047 BF08         405         MOV     CURDIG,#CHARNO
0049 B000         406         MOV     @PNTR0,#0       ;PNTR0 STILL CONTAINS #KEYLOC
004B 764F         407         JF1     SCAN8           ;JUMP IF ANY KEYS WERE DETECTED
004D BEFF         408         MOV     LASTKY,#0FFH    ;CHANGE <LASTKY> WHEN NO KEYS ARE DOWN
004F A5           409 SCAN8:  CLR     F1
                  410 ;
                  411 ;*********************************************************
                  412 ;       THE NEXT CODE SEGMENT IS THE INTERRUPT-DRIVEN PORTION OF THE 'DELAY'
                  413 ;       UTILITY.  IT DECREMENTS RAM LOCATION 'RDELAY' ONCE PER DISPLAY SCAN
                  414 ;       IF 'RDELAY' IS NOT ALREADY ZERO.
                  415 ;*********************************************************
                  416 ;
0050 B923         417         MOV     PNTR1,#RDELAY
0052 F1           418         MOV     A,@PNTR1
0053 C657         419         JZ      SCAN9
0055 07           420         DEC     A
0056 A1           421         MOV     @PNTR1,A
                  422 ;
0057 83           423 SCAN9:  RET
                  424 ;
                  425 ;*********************************************************
                  426 ;
                  427 ;CHRSTB IS THE BASE FOR THE PATTERNS TO ENABLE ONE-OF-CHARNO CHARACTERS.
0057              428 CHRSTB  EQU     (#-1) AND 0FFH
0058 01           429         DB      (00000001B XOR CHRPOL)
0059 02           430         DB      (00000010B XOR CHRPOL)
005A 04           431         DB      (00000100B XOR CHRPOL)
005B 08           432         DB      (00001000B XOR CHRPOL)
005C 10           433         DB      (00010000B XOR CHRPOL)
005D 20           434         DB      (00100000B XOR CHRPOL)
005E 40           435         DB      (01000000B XOR CHRPOL)
005F 80           436         DB      (10000000B XOR CHRPOL)
                  437
                  438 $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0            PAGE   12
AP40: INTEL MCS-48 KEYBOARD/DISPLAY APPLICATION NOTE APPENDIX

 LOC  OBJ        SEQ        SOURCE STATEMENT

                 439 ;INIT   INITIALIZES PROCESSOR REGISTERS
 0060 D5         440 INIT    SEL     RB1
 0061 BF08       441         MOV     CURDIG,#CHARNO
 0063 B822       442         MOV     PNTR0,#KBDBUF
 0065 B0FF       443         MOV     @PNTR0,#0FFH
 0067 B821       444         MOV     PNTR0,#KEYLOC
 0069 B000       445         MOV     @PNTR0,#0
 006B 23F0       446         MOV     A,#INPMSK
 006D 3A         447         OUTL    P1INPUT,A       ;SET BIDIRECTIONAL INPUT LINES
 006E C5         448         SEL     RB0
 006F 149E       449         CALL    CLEAR           ;UTILITY FOR SETTING INITIAL DISPLAY REGISTERS.
 0071 A5         450         CLR     F1
 0072 23F0       451         MOV     A,#TICK         ;LOAD INTERRUPT RATE VALUE
 0074 62         452         MOV     T,A
 0075 55         453         STRT    T
 0076 25         454         EN      TCNTI           ;ENABLE TIMER INTERRUPTS
                 455 ;
                 456 ;
                 457 ;*********************************************************
                 458 ;
                 459 ;ECHO   CHECK FOR ANY NEW KEYSTROKES DETECTED
                 460 ;       TRANSLATE EACH KEYSTROKE INTO A SEGMENT PATTERN
                 461 ;       AND WRITE IT INTO THE APPROPRIATE DISPLAY REGISTER.
                 462 ;
                 463 ;*********************************************************
                 464 ;
 0077 1483       465 ECHO:   CALL    KBDIN           ;GET NEXT KEYSTROKE
 0079 B281       466         JB5     FKEY            ;JUMP IF KEY IN RIGHTHAND COLUMN
                 467 ;       SINCE THE ACC IS USED BY ENCACC AND RENTRY, ITS CONTENTS MUST
                 468 ;       BE PROCESSED OR SAVED BEFORE ENCACC IS CALLED
 007B 14BA       469         CALL    ENCACC          ;FORM APPROPRIATE SEGMENT PATTERN
 007D 14DB       470         CALL    RENTRY          ;WRITE PATTERN INTO DISPLAY REGISTERS
 007F 0477       471         JMP     ECHO            ;LOOP INDEFINITELY
                 472 ;
 0081 2400       473 FKEY    JMP     FUNCTN          ;JUMP TO OFF-PAGE CODE TO CALL DEMO ROUTINE
                 474 ;
                 475 $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0            PAGE   13
AP40  INTEL MCS-48 KEYBOARD/DISPLAY APPLICATION NOTE APPENDIX

  LOC  OBJ        SEQ        SOURCE STATEMENT

                  476 ;*******************************************************
                  477 ;
                  478 ;       THE FOLLOWING SUBROUTINES IMPLEMENT THE UTILITIES COMMONLY USED FOR
                  479 ;       MOST KEYBOARD/DISPLAY APPLICATIONS.
                  480 ;       THEY COULD BE USED EXACTLY AS SHOWN HERE OR ADAPTED FOR SPECIAL CASES.
                  481 ;
                  482 ;*******************************************************
                  483 ;
                  484 ;KBDIN  KEYBOARD INPUT SUBROUTINE.
                  485 ;       COULD BE USED TO INTERFACE THE USER'S BACKGROUND PROGRAM WITH
                  486 ;       THE INTERRUPT DRIVEN KEYBOARD SCANNER.
                  487 ;       RETURNS ONLY AFTER A NEW KEYSTROKE HAS BEEN DETECTED AND DEBOUNCED.
                  488 ;       ENCODED VALUE OF KEY (RATHER THAN ITS POSITION IN SWITCH MATRIX) IS
                  489 ;       RETURNED IN THE ACCUMULATOR.
0083 B922         490 KBDIN:  MOV     R1,#KBDBUF
0085 2380         491         MOV     A,#80H          ;KBDBUF WILL BE MARKED AS CLEAR
0087 21           492         XCH     A,@R1           ;LOAD BUFFER VALUE
0088 F283         493         JB7     KBDIN
008A 038E         494         ADD     A,#LEGNDS       ;ADD BASE OF KEY ENCODING TABLE
008C A3           495         MOVP    A,@A            ;OBTAIN BYTE REPRESENTING KEY SIGNIFICANCE
008D 83           496         RET
                  497 ;
                  498 ;
                  499 ;LEGNDS IS THE BASE FOR TABLE SHOWING KEY MATRIX SIGNIFICANCE
                  500 ;       FOR THE KEYBOARD USED IN THE PROTOTYPE.
                  501 ;       KEY LAYOUT IS AS SHOWN TO THE RIGHT.
                  502 ;
                  503         NOTE THAT BIT6-BIT4 MAY BE USED TO ENCODE KEY TYPE.  IN THIS CASE:
                  504 ;               BIT4 INDICATES REGULAR DECIMAL DIGITS,
                  505 ;               BIT5 INDICATES RIGHT-COLUMN FUNCTION KEYS,
                  506 ;               BIT6 INDICATES PUNCTUATION MARKS ( * AND # ).
                  507 ;
008E              508 LEGNDS: EQU     ($ AND 0FFH)    ;USE LOW ORDER BITS AS TABLE INDEX
008E 4F           509         DB      4FH
008F 10           510         DB      10H
0090 4E           511         DB      4EH
0091 28           512         DB      28H     ; PDIGIT4==>  1       2       3       <1>
0092 17           513         DB      17H
0093 18           514         DB      18H     ; PDIGIT5==>  4       5       6       <2>
0094 19           515         DB      19H
0095 24           516         DB      24H     ; PDIGIT6==>  7       8       9       <3>
0096 14           517         DB      14H
0097 15           518         DB      15H     ; PDIGIT7==>  *       0       #       <4>
0098 16           519         DB      16H
0099 22           520         DB      22H     ;             !       !       !       !
009A 11           521         DB      11H     ;             !       !       !       !
009B 12           522         DB      12H     ;             V       V       V       V
009C 13           523         DB      13H     ;         PINPUT7 PINPUT6 PINPUT5  PINPUT4
009D 21           524         DB      21H
                  525 $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0              PAGE   14
AP40  INTEL MCS-48 KEYBOARD/DISPLAY APPLICATION NOTE APPENDIX

 LOC  OBJ        SEQ        SOURCE STATEMENT

                  526 ;*************************************************************
                  527
                  528 ;CLEAR  WRITES 'BLANK' CHARACTERS INTO ALL DISPLAY REGISTERS.
                  529 ;       RETURNS WITH NEXTPL SET TO LEFTMOST CHARACTER POSITION
                  530 ;FILL   WRITES SEGMENT PATTERN NOW IN ACC INTO ALL DISPLAY REGISTERS
009E 2300         531 CLEAR  MOV     A,#BLANK XOR SEGPOL
00A0 B938         532 FILL   MOV     PNTR1,#SEGMAP+1
00A2 BF08         533        MOV     NEXTPL,#CHARNO
00A4 A1           534 CLR1:  MOV     @PNTR1,A        ;STORE THE BLANK CODE
00A5 19           535        INC     PNTR1           ;POINT TO NEXT CHARACTER TO THE LEFT
00A6 EFA4         536        DJNZ    NEXTPL,CLR1
00A8 BF08         537        MOV     NEXTPL,#CHARNO
00AA 83           538        RET
                  539 ;
                  540 ;*************************************************************
                  541 ;
                  542 ;PRINT  SUBROUTINE TO COPY A STRING OF BIT PATTERNS FROM ROM TO THE
                  543 ;       DISPLAY REGISTERS.  STRING STARTS AT LOCATION POINTED TO BY PNTR0.
                  544 ;       CONTINUES UNTIL AN ESCAPE CODE (0FFH) IS REACHED.
                  545 ;       NOTE THAT THE CHARACTER STRING PUT OUT MUST BE LOCATED ON THE SAME
                  546 ;       PAGE AS THIS SUBROUTINE, SINCE SAME-PAGE MOVES ARE USED.
                  547 ;       PRINT IN TURN CALLS EITHER SUBROUTINE 'WDISP' OR 'RENTRY'
                  548 ;       TO ACTUALLY EFFECT WRITING INTO THE DISPLAY REGISTERS.
00AB F8           549 PRINT: MOV     A,PNTR0         ;LOAD NEXT CHARACTER LOCATION
00AC A3           550        MOVP    A,@A            ;LOAD BIT PATTERN INDIRECT
00AD C6B4         551        JZ      PRNT1           ;ESCAPE PATTERN
00AF 14D0         552        CALL    WDISP           ;OUTPUT TO NEXT CHARACTER POSITION
                  553 ;##    (CALL   RENTRY          INSTEAD IF MESSAGE IS TO BE RIGHT JUSTIFIED)
00B1 18           554        INC     PNTR0           ;INDEX POINTER
00B2 04AB         555        JMP     PRINT
00B4 83           556 PRNT1: RET                     ;DONE
                  557 ;
                  558 ;*************************************************************
                  559 ;
                  560 ;JOHN   ARRAY HOLDS THE BIT PATTERNS FOR THE LETTERS 'JOHN' (SEE 'TEST2')
                  561 ;       (NOTE THAT 'OHN' IS WRITTEN IN LOWER CASE LETTERS)
00B5              562 JOHN   EQU     $ AND 0FFH
00B5 1E           563        DB      00011110B XOR SEGPOL
00B6 5C           564        DB      01011100B XOR SEGPOL
00B7 74           565        DB      01110100B XOR SEGPOL
00B8 54           566        DB      01010100B XOR SEGPOL
00B9 00           567        DB      00
                  568 ;
                  569 $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0            PAGE   15
AP40  INTEL MCS-48 KEYBOARD/DISPLAY APPLICATION NOTE APPENDIX

 LOC  OBJ        SEQ        SOURCE STATEMENT

                  570 ;*********************************************************
                  571 ;
                  572 ;ENCACC ENCODES LSNIBBLE OF ACC INTO HEX BIT PATTERN INTO ACC
 00BA 530F        573 ENCACC  ANL     A,#ENCMSK
 00BC 03C0        574         ADD     A,#DGPATS
 00BE A3          575         MOVP    A,@A
 00BF 83          576         RET
                  577 ;DGPATS IS THE BASE FOR THE TABLE OF SEGMENT PATTERNS FOR THE BASIC
                  578 ;DIGITS.  HERE THE FULL HEX SET (0-F) IS INCLUDED.
                  579 ;FOR MANY USER APPLICATIONS, THE CHARACTER SET MAY BE AMENDED OR AUGMENTED
                  580 ;TO INCLUDE ADDITIONAL SPECIAL PURPOSE PATTERNS.
                  581 ;FORMAT IS      PGFEDCBA        IN STANDARD SEVEN-SEGMENT ENCODING CONVENTION
                  582 ;                               WHERE P REPRESENTS THE DECIMAL POINT
 00C0             583 DGPATS  EQU     $ AND 0FFH
 00C0 3F          584         DB      00111111B XOR SEGPOL
 00C1 06          585         DB      00000110B XOR SEGPOL
 00C2 5B          586         DB      01011011B XOR SEGPOL
 00C3 4F          587         DB      01001111B XOR SEGPOL
 00C4 66          588         DB      01100110B XOR SEGPOL
 00C5 6D          589         DB      01101101B XOR SEGPOL
 00C6 7D          590         DB      01111101B XOR SEGPOL
 00C7 07          591         DB      00000111B XOR SEGPOL
 00C8 7F          592         DB      01111111B XOR SEGPOL
 00C9 67          593         DB      01100111B XOR SEGPOL
 00CA 77          594         DB      01110111B XOR SEGPOL
 00CB 7C          595         DB      01111100B XOR SEGPOL
 00CC 39          596         DB      00111001B XOR SEGPOL
 00CD 5E          597         DB      01011110B XOR SEGPOL
 00CE 79          598         DB      01111001B XOR SEGPOL
 00CF 71          599         DB      01110001B XOR SEGPOL
                  600 ;
                  601 ;*********************************************************
                  602 ;
                  603 ;WDISP  WRITES BIT PATTERN NOW IN ACC INTO NEXT CHARACTER POSITION
                  604 ;       OF THE DISPLAY (NEXTPL).  ADJUSTS NEXTPL POINTER VALUE.
                  605 ;       RESULTS IN DISPLAY BEING FILLED LEFT TO RIGHT, THEN RESTARTING
 00D0 A9          606 WDISP:  MOV     PNTR1,A
 00D1 FF          607         MOV     A,NEXTPL
 00D2 0337        608         ADD     A,#SEGMAP
 00D4 29          609         XCH     A,PNTR1
 00D5 A1          610         MOV     @PNTR1,A
 00D6 EFDA        611         DJNZ    NEXTPL,WDISP1
 00D8 BF08        612         MOV     NEXTPL,#CHARNO
 00DA 83          613 WDISP1  RET
                  614 ;
                  615 $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0            PAGE   16
AP40  INTEL MCS-48 KEYBOARD/DISPLAY APPLICATION NOTE APPENDIX

 LOC  OBJ        SEQ        SOURCE STATEMENT

                616 ;*********************************************************
                617 ;
                618 ;RENTRY SUBROUTINE TO ENTER ACC CONTENTS INTO THE RIGHTMOST DIGIT
                619 ;       AND SHIFT EVERYTHING ELSE ONE PLACE TO THE LEFT
 00DB B938      620 RENTRY: MOV     PNTR1,#SEGMAP+1
 00DD BF08      621         MOV     NEXTPL,#CHARNO
 00DF 21        622 RENTR1  XCH     A,@PNTR1
 00E0 19        623         INC     PNTR1
 00E1 EFDF      624         DJNZ    NEXTPL,RENTR1
 00E3 BF08      625         MOV     NEXTPL,#CHARNO  ;POINT TO LEFTMOST CHARACTER
 00E5 83        626         RET
                627 ;
                628 ;*********************************************************
                629 ;
                630 ;RDPADD TOGGLE DECIMAL POINT IN LAST CHARACTER DISPLAY CHARACTER
                631 ;DPADD  TOGGLES DECIMAL POINT IN THE CHARACTER POINTED TO BY THE ACC
                632 ;
 00E6 2301      633 RDPADD: MOV     A,#01H          ;SET INDEX TO RIGHTMOST POSITION
 00E8 0337      634 DPADD:  ADD     A,#SEGMAP       ;ACCESS DISPLAY REGISTER FOR DESIRED PLACE
 00EA A9        635         MOV     PNTR1,A
 00EB F1        636         MOV     A,@PNTR1
 00EC D380      637         XRL     A,#80H
 00EE A1        638         MOV     @PNTR1,A
 00EF 83        639         RET
                640 ;
                641 ;*********************************************************
                642 ;
                643 ;HOLD   SUBROUTINE CALLED WHEN KEY IS KNOWN TO BE DOWN.
                644 ;       WILL NOT RETURN UNTIL KEY IS RELEASED.
 00F0 D5        645 HOLD:   SEL     RB1
 00F1 FE        646         MOV     A,LASTKY        ;<LASTKY>=0FFH IFF NO KEYS DOWN
 00F2 C5        647         SEL     RB0
 00F3 37        648         CPL     A
 00F4 96F0      649         JNZ     HOLD
 00F6 83        650         RET
                651 ;
                652 ;*********************************************************
                653 ;
                654 ;DELAY  SUBROUTINE HANGS UP FOR THE NUMBER OF COMPLETE DISPLAY SCANS EQUAL
                655 ;       TO THE CONTENTS OF THE ACCUMULATOR WHEN CALLED.
 00F7 B923      656 DELAY:  MOV     PNTR1,#RDELAY
 00F9 A1        657         MOV     @PNTR1,A
 00FA F1        658 DELAY1: MOV     A,@PNTR1
 00FB 96FA      659         JNZ     DELAY1
 00FD 83        660         RET
                661 #EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER  V2.0            PAGE   17
AP40:  INTEL MCS-48 KEYBOARD/DISPLAY APPLICATION NOTE APPENDIX

 LOC  OBJ        SEQ        SOURCE STATEMENT

 0100            662 ORG    100H
                 663 ;
                 664 ;*************************************************************
                 665 ;
                 666 ;THE CODE ON THIS PAGE IS FOR DEMONSTRATION PURPOSES ONLY-
                 667 ;I TRUELY DOUBT WHETHER ANY END USERS WOULD LIKE TO SEE A NAME
                 668 ;POPPING UP ON THEIR CALCULATOR SCREENS
                 669 ;HOWEVER  THE CODE SHOWN HERE DOES INDICATE HOW THE UTILITY SUBROUTINES
                 670 ;INCLUDED HERE COULD BE ACCESSED
                 671 ;THE ROUTINES THEMSELVES ARE CALLED WHEN ONE OF THE FOUR BUTTONS
                 672 ;ON THE RIGHT-HAND SIDE OF THE PROTOTYPE KEYBOARD IS PRESSED
                 673 ;
                 674 ;*************************************************************
                 675 ;
                 676 ;FUNCTN ROUTINE TO IMPLEMENT ONE OF FOUR DEMO UTILITIES, ACCORDING
                 677 ;       TO WHICH OF THE FOUR FUNCTION KEYS WAS PRESSED
 0100 1212       678 FUNCTN: JB0    FUNCT1
 0102 320E       679         JB1    FUNCT2
 0104 5208       680         JB2    FUNCT3
                 681 ;
 0106 14E6       682 FUNCT4: CALL   RDPADD
 0108 0477       683         JMP    ECHO
                 684 ;
 010A 342E       685 FUNCT3: CALL   TEST3
 010C 0477       686         JMP    ECHO
                 687 ;
 010E 3424       688 FUNCT2: CALL   TEST2
 0110 0477       689         JMP    ECHO
                 690 ;
 0112 3416       691 FUNCT1: CALL   TEST1
 0114 0477       692         JMP    ECHO
                 693 ;
                 694 ;*************************************************************
                 695 ;
                 696 ;TEST1  CODE SEGMENT TO FILL DISPLAY REGISTERS WITH DIGITS DOWN TO '1'
 0116 BF08       697 TEST1:  MOV    NEXTPL,#CHARNO
 0118 B808       698         MOV    PNTR0,#CHARNO   ;SET FOR EIGHT LOOP REPETITIONS
 011A FF         699 TST11:  MOV    A,NEXTPL
 011B 14BA       700         CALL   ENCACC
 011D 14D0       701         CALL   WDISP
 011F E81A       702         DJNZ   PNTR0,TST11     ;COPY NEXT DIGIT INTO DISPLAY REGISTERS
 0121 BF08       703         MOV    NEXTPL,#CHARNO
 0123 83         704         RET
                 705 ;
                 706 $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0            PAGE   18
AP40: INTEL MCS-48 KEYBOARD/DISPLAY APPLICATION NOTE APPENDIX

 LOC  OBJ        SEQ        SOURCE STATEMENT

                 707 ;*********************************************************
                 708 ;
                 709 ;TEST2  WRITES THE SEGMENT PATTERN FOR 'JOHN' ONTO THE DISPLAY,
                 710 ;       WAITS FOR A WHILE, AND THEN CLEARS THE DISPLAY
 0124 B8B5       711 TEST2:  MOV     PNTR0,#JOHN
 0126 14AB       712         CALL    PRINT
 0128 2364       713         MOV     A,#100  ;SCAN DISPLAY FOR 100 CYCLES
 012A 14F7       714         CALL    DELAY
 012C 049E       715         JMP     CLEAR
                 716 ;
                 717 ;*********************************************************
                 718 ;
                 719 ;TEST3  SUBROUTINE TO FILL DISPLAY WITH DASHES
                 720 ;       JUMPS INTO SUBROUTINE 'CLEAR'
                 721 ;       AS SOON AS THE KEY IS RELEASED
 012E 2340       722 TEST3:  MOV     A,#01000000B XOR SEGPOL ;PATTERN FOR '-'
 0130 14A0       723         CALL    FILL
 0132 14F0       724         CALL    HOLD
 0134 049E       725         JMP     CLEAR
                 726 ;
                 727 ;*********************************************************
                 728 ;
                 729 END

USER SYMBOLS
ASAVE  0002   BLANK  0000   CHARNO 0008   CHRPOL 0000   CHRSTB 0057   CLEAR  009E   CLR1   00A4   CURDIG 0007
DEBNCE 0004   DELAY  00F7   DELAY1 00FA   DGPATS 00C0   DPADD  00E8   ECHO   0077   ENCACC 00BA   ENCMSK 000F
FILL   00A0   FKEY   0081   FUNCT1 0112   FUNCT2 010E   FUNCT3 010A   FUNCT4 0106   FUNC1N 0100   HOLD   00F0
INIT   0060   INPMSK 00F0   JOHN   00B5   KBDBUF 0022   KBDIN  0083   KEYLOC 0021   LASTKY 0006   LEGNDS 008E
NCOLS  0004   NEGLOG 00FF   NEXTPL 0007   NPEPTS 0020   NROWS  0004   NXTLOC 0023   PDIGIT 0010   PINPUT 0009
PNTR0  0000   PNTR1  0001   POSLOG 0000   PRINT  00AB   PRNT1  00B4   PSGMNT 0008   RDELAY 0023   RDPADD 00E6
REFR1  0013   REFRSH 0010   RENTR1 00DF   RENTRY 00DB   ROTCNT 0005   ROTPAT 0004   SCAN   001E   SCAN1  0021
SCAN3  0034   SCAN5  003F   SCAN6  0045   SCAN8  004F   SCAN9  0057   SEGMAP 0037   SEGPOL 0000   TEST1  0116
TEST2  0124   TEST3  012E   TICK   FFF0   TIINT  0007   TIRET  000E   TST11  011A   WDISP  00D0   WDISP1 00DA

ASSEMBLY COMPLETE,  NO ERRORS
```

```
ISIS-II ASSEMBLER SYMBOL CROSS REFERENCE, V2 0                  PAGE  1

ASAVE   202#  249   269
BLANK   179#  280   531
CHARNO  173#  220   405   441   533   537   612   621   625   697   698   703
CHRPOL  169#  429   430   431   432   433   434   435   436
CHRSTB  282   428#
CLEAR   449   531#  715   725
CLR1    534#  536
CURDIG  206#  283   289   395   405   441
DEBNCE  178#  370
DELAY   656#  714
DELAY1  658#  659
DSPATS  574   583#
DPADD   634#
ECHO    465#  471   683   686   689   692
ENCACC  469   573#  700
ENCMSK  183#  573
FILL    532#  723
FKEY    466   473#
FUNCT1  678   691#
FUNCT2  679   688#
FUNCT3  680   685#
FUNCT4  682#
FUNCTN  473   678#
HOLD    645#  649   724
INIT    236   440#
INPMSK  171#  446
JOHN    562#  711
KBDBUF  214#  386   442   490
KBDIN   465   490#  493
KEYLOC  213#  300   389   444
LASTKY  205#  359   360   385   408   646
LEGNDS  494   508#
NCOLS   175#  328
NEGLOG  167#
NEXTPL  193#  533   536   537   607   611   612   621   624   625   697   699   703
NREPTS  212#  361
NROWS   174#
NXTLOC  329#  392
PDIGIT  158#  285
PINPUT  160#  301   447
PNTR0   191#  300   358   361   370   380   383   386   387   389   390   406   442   443   444   445
        549   554   698   702   711
PNTR1   192#  290   291   417   418   421   490   492   532   534   535   606   609   610   620   622
        623   635   636   638   656   657   658
POSLOG  166#  169   170
PRINT   549#  555   712
PRNT1   551   556#
PSGMNT  159#  281   292
RDELAY  216#  417   656
RDPADD  633#  682
REFR1   282#
REFRSH  260   280#
RENTR1  622#  624
RENTRY  470   620#
```

```
ISIS-II ASSEMBLER SYMBOL CROSS REFERENCE, V2 0                  PAGE  2

ROTCNT  204#  328   392
ROTPAT  203#  330   391
SCAN    300#
SCAN1   328#
SCAN3   362   380#
SCAN5   331   371   381   384   389#
SCAN6   395#
SCAN8   407   409#
SCAN9   395   419   423#
SEGMAP  220#  288   532   608   620   634
SEGPOL  170#  280   531   563   564   565   566   584   585   586   587   588   589   590   591   592
        593   594   595   596   597   598   599   722
TEST1   691   697#
TEST2   688   711#
TEST3   685   722#
TICK    177#  250   451
TIINT   248#
TIRET   269#
TST11   699#  702
WDISP   552   606#  701
WDISP1  611   613#

CROSS REFERENCE COMPLETE
```