UCID-17145

Lawrence Livermore Laboratory

MST-80 MICROPROCESSOR TRAINER

Gordon D. Jones

May 21, 1976

This is an informal report intended primarily for internal or limited external distribution. The opinions and conclusions stated are those of the author and may or may not be those of the laboratory.

Prepared for U.S. Energy Research & Development Administration under contract No. W-7405-Eng-48.

DISCLAIMER

MST-80 MICROPROCESSOR TRAINER

CONTENTS PAGE

INTRODUCTION

This trainer is a complete, self-contained microcomputer system housed in a brief case for portability and convenience of use. It utilizes INTEL's* 8080A microprocessor and associated support chips.

The trainer is designed to allow the student to explore and learn the hardware and software capability of the 8080 microprocessor. It includes a breadboard socket so that experiments can be interfaced to the trainer. This option allows the student to learn both interfacing techniques and programming.

A keyboard and numerical display are provided for the student to communicate with the trainer. This combination eliminates the need for expensive and bulky I/O such as a teletype. The keyboard and numerical display can be used with either the octal number system or the hexadecimal number system. A header socket is provided to select which number system to be is utilized by the display. Two keyboard monitor programs are provided, one to allow octal number input from the keyboard and the other to allow hexadecimal input. The user can select which number system he prefers by plugging in the appropriate header socket along with its companion preprogrammed PROM.

A block diagram of the trainer is shown in Figure 1. Figure 6 shows the complete trainer in its case, Figure 7 a close-up of the circuit card with keyboard.

Hardware Features of the trainer are:

1. Uses INTEL's 8080A CPU and support chips.

*Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Energy Research and Development Administration to the exclusion of others that may be suitable.

2. Has 1024 bytes of read/write memory (RAM).

3. Has sockets for three 1702A PROM's (768 bytes). Also one uncommitted socket that can be jumper-wired to a 24 PIN ROM of user's choice. Normally a MONITOR program resides in PROM Ø.

4. Has a 24-key keyboard. This is an input device, accessed through memory mapped I/O. See Figure 3 for the memory map.

5. Has a three digit display with full hex number capability. This is an output device with output port address Ø.

6. Has one 8-bit input port. Address = 1.

7. Has one 8-bit output port (latched). Address = 1.

8. Has single step capability.

9. Has ten uncommitted LED's that can easily be connected to any desired signals (address lines, data lines, status, etc.). These are used in single step mode.

Figure 2 shows the connectors used to interface the trainer and also gives detailed information on each signal and its connector pin number.

MONITOR PROGRAM (OCTAL)

The trainer contains a monitor program that allows the user to enter a program in RAM, examine locations, change contents of locations and run the user program from a specified starting address.

The monitor program also contains a debug routine to assist the user in program debug. This routine allows the user to insert a break point (377_8) in his program. When the break point is encountered the break routine (in the monitor program) will be entered which will save all the CPU registers and the break point address, and will put 222 in the display to signal the user that a break point has been encountered.

The contents of the CPU registers and break point address are saved in the following dedicated page 7 memory locations:

BREAK POINT MEMORY STORAGE LOCATIONS (MEMORY PAGE 7)

LOC	CONTENTS		LOC	CONTENTS
222	PCL	Break Point Address	227	B REG
223	PCH		230	E REG
224	PSW		231	D REG
225	A REG		232	L REG
226	C REG		233	H REG

These locations can now be examined using the DISP feature of the monitor program and, if desired, can be changed to new values using the ENTER feature of the monitor program. A detailed description of how to do this is included in the SAMPLE PROGRAM write up.

The RUN feature of the monitor program starts the users program with the CPU registers intialized to the current values found in these dedicated memory locations. This allows you to change these values before pushing RUN.

A complete listing of this program is included at the end of this report along with a flow chart, Figures 4 and 5. Figure 3 is a memory map for the system. A sample program is included in the report.

OPERATION OF KEYBOARD USING OCTAL MONITOR

KEYBOARD LAYOUT

C	D	E	F	RESET	EXA
8	9	A	B	RUN	LDH
4	5	6	7	DISP	S3
0	1	2	3	ENTER	S4

RESET: Resets the system and starts the monitor program running.

NUMBER KEYS: Pushing these keys cause a number to be entered into the display in a left shift mode. Care must be exercised when entering numbers to ensure that the intended number is entered, since the display is not cleared but simply shifted left. For instance if you want to enter a 1 into the display, you should push ØØ1 to insure the old number is completely replaced.

The current value in the display is also stored in a memory location called KYTEM.

Keys 8 thru F are ignored by the octal monitor program.

LDH: In order to address any location in memory the user needs to specify the complete address. The high order address is specified by keying in the desired value into the display and then pushing LDH (LOAD H). This stores the high value in a memory location called HVALU for later use by the monitor program.

The low order address is specified by the current contents of the display whenever it is needed, i.e., in RUN or DISP operations. Its current value is kept in a memory location called LVALU.

DISP: When it is desired to examine the contents of a memory location the DISP key is used. The high order address is selected by entering the desired value and using the LDH key, as explained above. The low order address is then keyed into the display, then the DISP key is pushed. This will cause the contents of the desired address to be displayed.

ENTER: The ENTER key is used to enter new values into specified locations. ENTER also automatically increments the address value, allowing the user to quickly examine or enter new values into consecutive locations in memory.

The address is set by using the DISP key since the present value should be displayed before you enter a new value. After pushing DISP a new value may be keyed into the display and when ENTER is pushed this value is entered into the currently addressed location.

In addition, the address is incremented and the contents of the next consecutive location is displayed. That value can be re-entered by pressing ENTER again or a new value can be keyed in before pressing ENTER.

RUN: This allows you to start a user program at any specified address. The address is specified by using the LDH key and keying in to the display the low order address before pushing RUN. Remember RUN initializes all CPU registers from dedicated memory locations before starting the user program.

EXA: Pushing this key displays the current value of the low order address. This is useful when examining a program (stepping through using ENTER) and you forget where you are.

USE OF THE KEYBOARD READ ROUTINE AS A CALL FROM A USER PROGRAM (OCTAL MONITOR)

The KEY routine in the monitor program is written as a subroutine and may be called by a user program. This is useful when the user's program requires operator interaction since the keyboard is convenient for this purpose. The subroutine is called by a CALL KEY instruction (315 Ø26 ØØØ) and returns to the user with the value of the number key pushed in the C register. The A register also contains this number in the low order octal digit, and in addition contains the last two key entries in the high order digits.

Two precautions must be observed when using this subroutine. First, the KEY routine uses the A, B, C, H and L registers. If the user program also uses the registers, they must be saved before calling KEY. Second, only number keys can be used when KEY is called. The control keys are not decoded in the KEY subroutine and should not be used. Also number keys larger than 7 will be ignored when using the octal monitor. Therefore a number key between Ø and 7 must be pushed before a return is completed to the user program.

SAMPLE PROGRAM

```
 MEMORY      MACHINE
LOCATION  CODE IN OCTAL
  ØØØ          Ø76                    MVI A, Ø        ; CLEAR AC
  ØØ1          ØØØ
  ØØ2          323       AGAIN:       OUT Ø           ; SEND AC TO DISPLAY
  ØØ3          ØØØ
  ØØ4          ØØ6                    MVI B, Ø        ; CLR B REGISTER
  ØØ5          ØØØ
  ØØ6          Ø16                    MVI C, 1ØØ      ; PUT 100 IN C REGISTER
  ØØ7          1ØØ
  Ø1Ø          ØØ4       LOOP:        INR B           ; INCREMENT B
  Ø11          312                    JZ LOOP         ; DO IT AGAIN
  Ø12          Ø1Ø
  Ø13          ØØ4
  Ø14          Ø15                    DCR C           ; DECREMENT C
  Ø15          3Ø2                    JNZ LOOP        ; LOOP UNTIL ZERO
  Ø16          Ø1Ø
  Ø17          ØØ4
  Ø2Ø          3Ø6                    ADI ØØ1         ; ADD ONE TO AC
  Ø21          ØØ1
  Ø22          3Ø3                    JMP AGAIN       ; GO DISPLAY AC & DO AGAIN
  Ø23          ØØ2
  Ø24          ØØ4
```

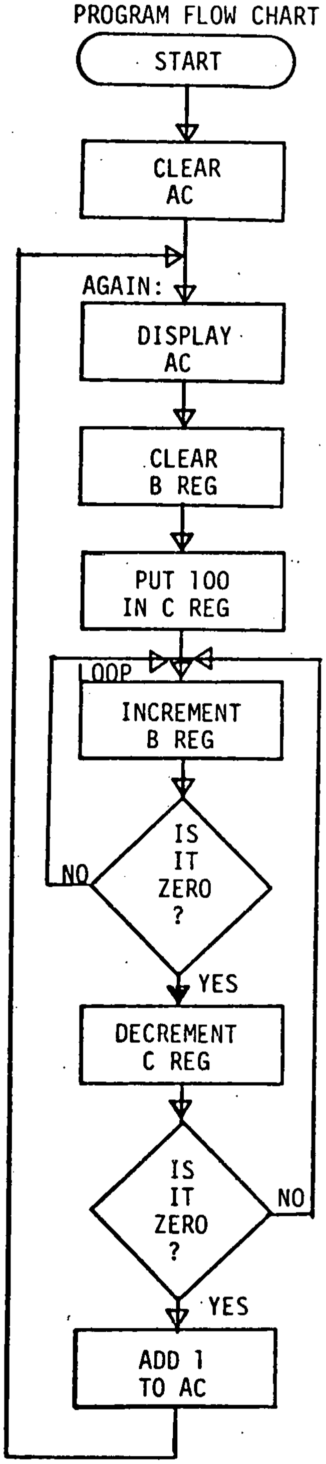
This program can be used to demonstrate the use of the OCTAL monitor program. Load the sample program into memory as follows:

Before you start, you need to decide where to load it. Let's put it in page 4 starting at location Ø (absolute address = Ø4ØØ hex). First, key ØØ4 into the display and then push the LDH (load H) key. This sets the high order address (High byte) to page 4. Next key ØØØ into the display, and push the DISP key. This will display the current contents of location Ø on page 4. Now you can key in the machine language code for the first instruction, Ø76 (MVI A), and push the ENTER key. This will enter the Ø76 into location Ø and will also display the contents of the next location (loc 1). Now you can key in the next code, ØØØ, and push ENTER again. The ØØØ will be entered into location 1 and then location 2 will be displayed. Continue this process until the entire program is entered.

If at any time a mistake is made while keying in a number, just continue to key in until the correct value appears in the display. (This number is not used until a control key is pressed.) If at any time while loading a program you forget where you are, just press EXA (examine address) and the current low order address will appear in the display. You can continue on from that point by pushing the DISP key and then the ENTER key. Or you can key in a new address into the display; then pushing the DISP key will allow you to continue from that address.

After the entire program has been keyed in, you may want to check it for correctness. This is done by keying the starting address into the display (ØØØ for our sample program), pushing the DISP key and then repeatedly pushing the ENTER key. This will step though the program sequentially and display each location so it can be checked. If a mistake is found, just key in the correct value before the ENTER key is pushed.

After the program is loaded satisfactorily you can run it if so desired. To run the program, key the starting address (ØØØ for our sample program) into the display and push RUN. If you are not sure what the current high order address (HVALU) is, you should set it to the correct value using the LDH key as explained previously.



USING A BREAK POINT IN PROGRAM DEBUGGING

The use of a break point in program debugging can be demonstrated using this sample program.

The program is a simple count routine that will cause the display to count up at a fixed rate determined by the constants in the counting loops. If you execute the program as it is written, you will notice the display is counting very rapidly. This is not intentional and is caused by a program bug. Let's use the break point to find it. Looking at the flow chart at the left, you will see there are two counting loops. The first one counts up to 377_8 and then goes back to Ø. Then the second count loop is entered. It counts the number of times the first loop must go thru a full count (400_8 counts). Since the C register is initialized to 100_8, the second loop counts 100_8 counts, hence the total counts for both loops is 400_8 x 100_8 ($=16384_{10}$) counts. After the full count is reached, 1 is added to the A register and its contents are displayed. Then the count loop starts over. This program runs endlessly until stopped by the user. The first thing to check is to see if the registers are initialized correctly. This is done by inserting a break point (break point code = 377_8) in place of the INR B instruction at memory location Ø1Ø. (Remember to set the high order address to page 4.) Run the program. It will break when the 377 is encountered and a 222 will appear in the display to signal the user that a break has occured. The break routine automatically sets HVALU to page 7 and 222 is being displayed so if you now push the DISP key, the contents of memory location 222 page 7 will be displayed. This location contains the low byte of the address where the break occured. The high byte of the break address is stored in location 223, so pushing the ENTER key will cause it to be displayed. Repeated use of the ENTER key allows you to examine the contents of all the CPU registers. The BREAK routine stores these away in the following memory locations:

BREAK ROUTINE MEMORY STORAGE LOCATIONS (MEMORY PAGE 7)
(Octal Monitor)

LOC	CONTENTS		LOC	CONTENTS
222	PCL	Break Point Address	227	B REG
223	PCH		230	E REG
224	PSW		231	D REG
225	A REG		232	L REG
226	C REG		233	H REG

Register C is stored in location 226 and upon examination should contain 100_8. Location 225 (A REG) and 227 (B REG) should contain zero. If these are O.K. replace the INR B instruction (code ØØ4) in location Ø1Ø and put a break point (377) in location Ø14 in place of the DCR C instruction. Run the program. When it breaks, examine loc 227 again to see what the B REG is now. It should be a zero when the count loop is exited. But it is not zero! The bug must be in this loop. Upon inspection of the program it is apparent that the JZ Loop instruction which tests for completion of the count, is testing the wrong condition. It exits the loop on nonzero count rather than zero count, so you need to replace the JZ instruction with a JNZ (code 3Ø2) instruction. Replace the break point in Ø14 with DCR C (Ø15) and run the program. It should run O.K. with the display counting much slower.

COMMENT: This may appear to be a trivial bug and should be apparent by just inspecting the program listing. But this is one of the most common programming errors (that is, using the wrong sense of a test instruction), and is usually quite difficult to find in a more complex program.

ACKNOWLEDGMENTS

I wish to acknowledge the contributions of Eugene R. Fisher and James M. Spann to the design of the trainer and its operating features.

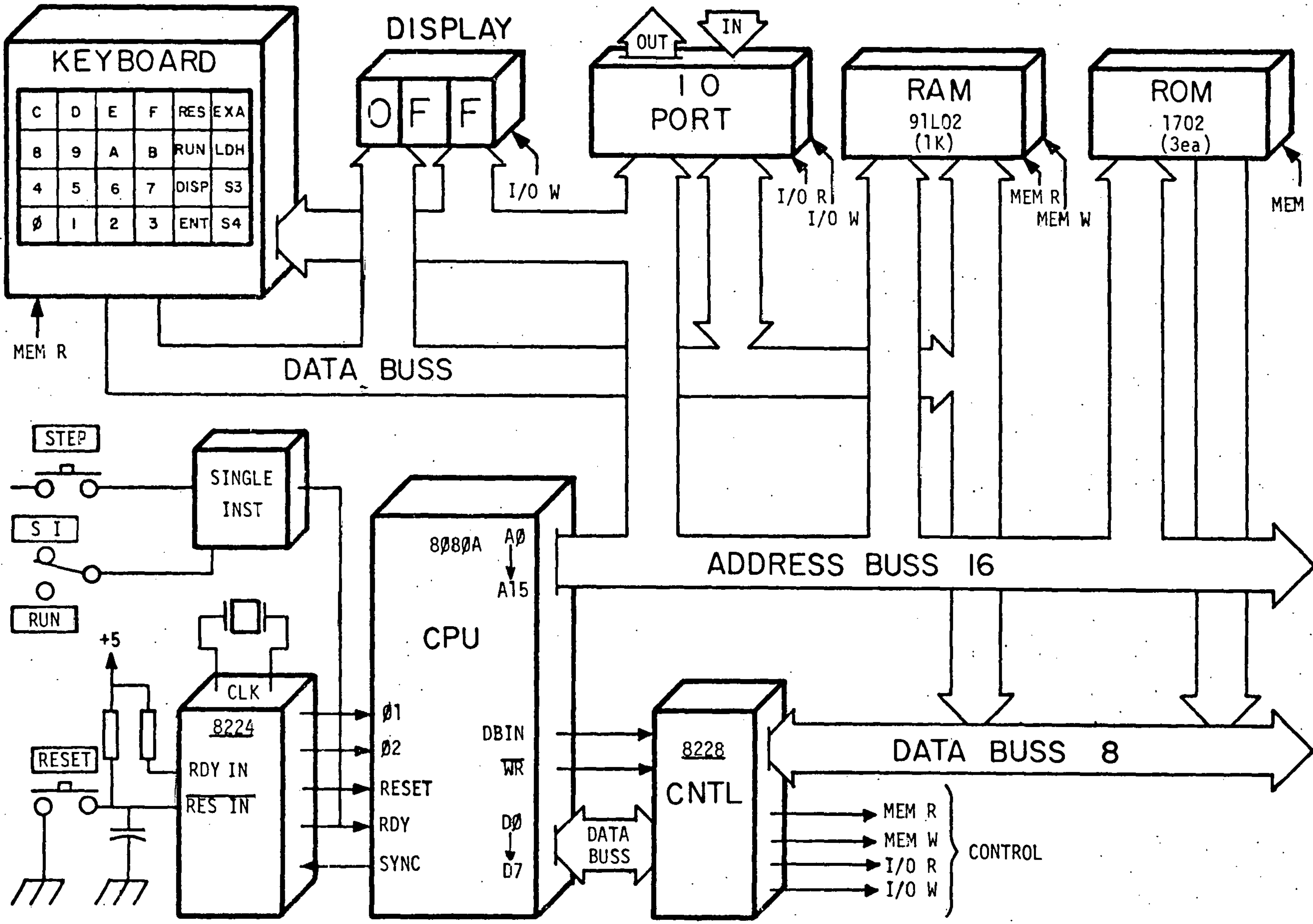


Figure 1. Operational Block Diagram of MST-80 Microprocessor Trainer

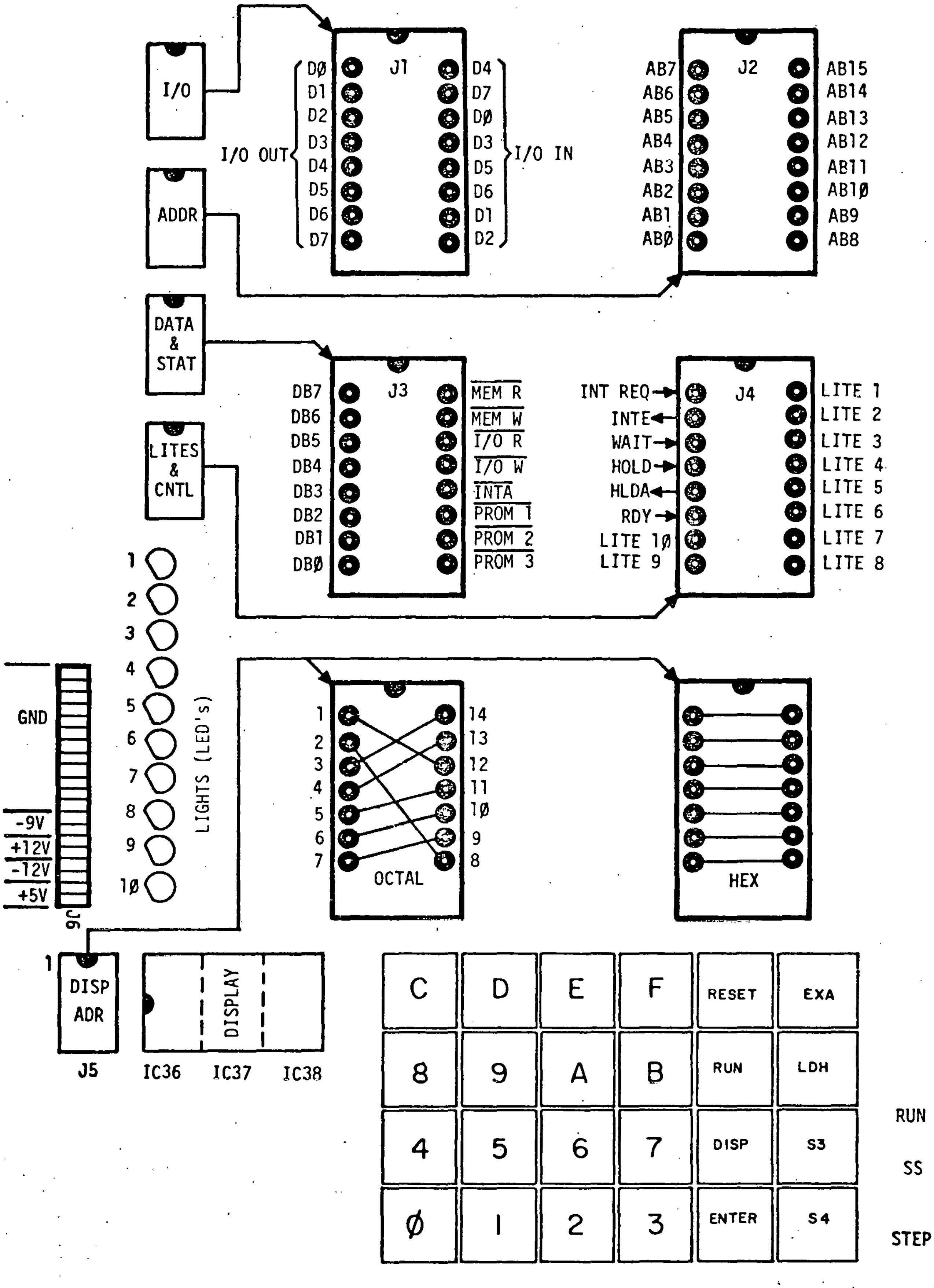


Figure 2. Panel Connectors Used to Interface MST-80 Microprocessor Trainer

HEX		OCTAL
ØØØØ – ØØFF	PROM Ø MONITOR PROGRAM	ØØØØØØ – ØØØ377
Ø1ØØ – Ø1FF	PROM 1	ØØØ4ØØ – ØØØ777
Ø2ØØ – Ø2FF	PROM 2	ØØ1ØØØ – ØØ1377
Ø3ØØ – Ø3FF	PROM 3 UNCOMMITTED SOCKET	ØØ14ØØ – ØØ1777
Ø4ØØ – Ø4FF	RAM 4	ØØ2ØØØ – ØØ2377
Ø5ØØ – Ø5FF	RAM 5	ØØ24ØØ – ØØ2777
Ø6ØØ – Ø6FF	RAM 6	ØØ3ØØØ – ØØ3377
Ø7ØØ – Ø7FF	RAM 7 REGISTER STORAGE & STACK	ØØ34ØØ – ØØ3777
Ø8ØØ – ØFFF	KEYBOARD (PAGES 10-17 OCTAL)	ØØ4ØØØ – ØØ7777
1ØØØ – 17FF	NOT USED	Ø1ØØØØ – Ø13777
18ØØ – 1FFF	KEYBOARD (PAGES 30-37 OCTAL)	Ø14ØØØ – Ø17777
2ØØØ – FFFF	NOT USED	Ø2ØØØØ – 177777

Page 7 locations used by monitor program

LOC	CONTENTS	
216	KYTEM (current value of display)	
22Ø	LVALU	
221	HVALU	
222	PCL	PCSTO
223	PCH	PCSTO
224	PSW	PSWST
225	A REG	PSWST
226	C REG	BSTOR
227	B REG	BSTOR
23Ø	E REG	DSTOR
231	D REG	DSTOR
232	L REG	HSTOR
233	H REG	HSTOR

36Ø STACK PTR

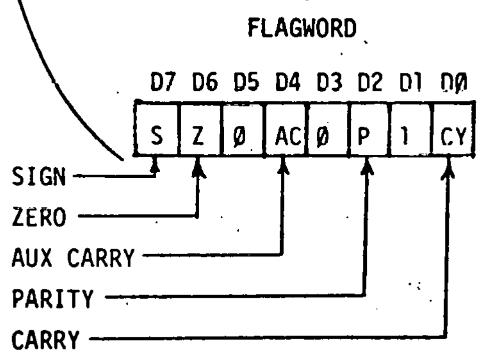


Figure 3. Memory Map for MST-80 Microprocessor Trainer

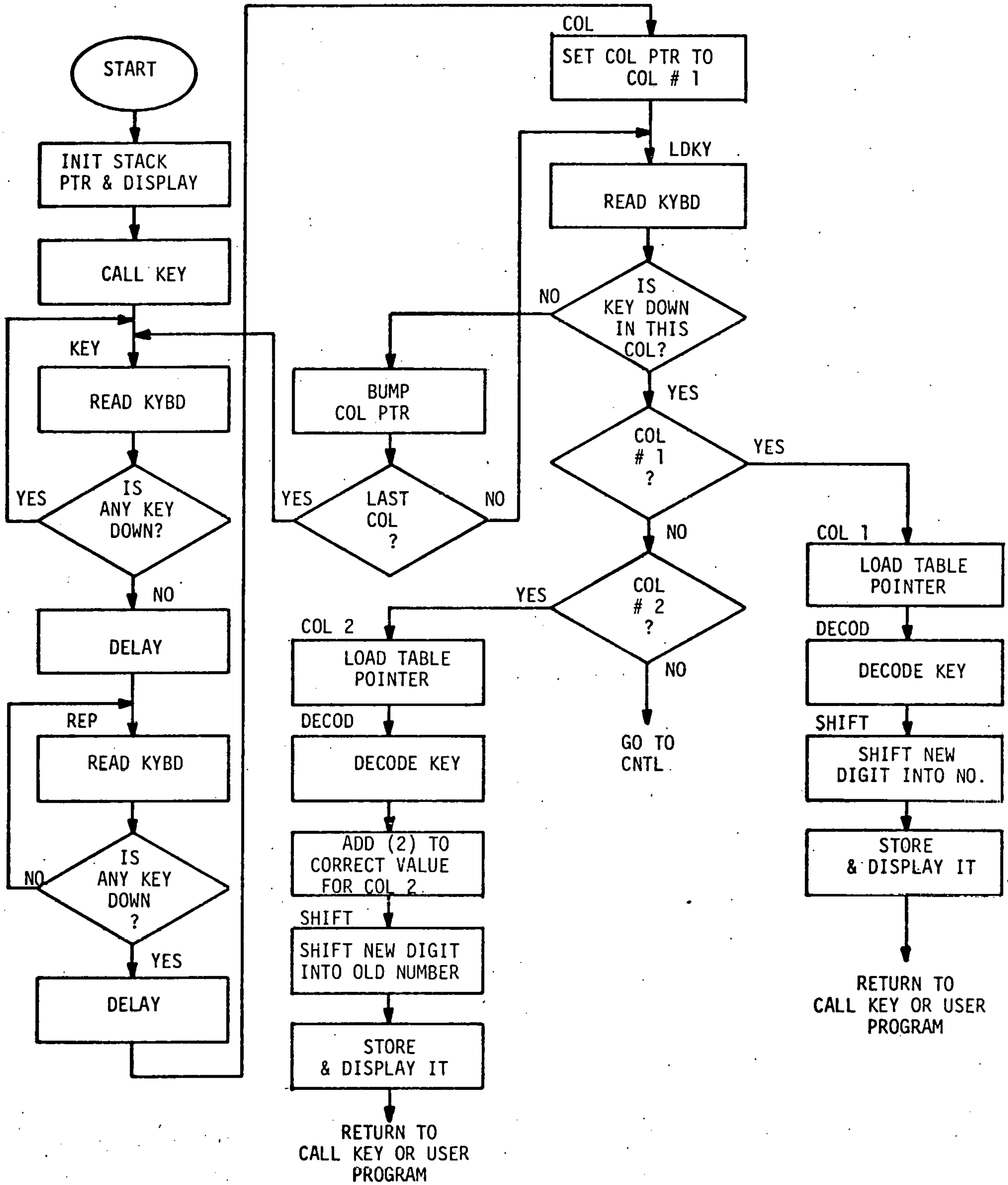


Figure 4. Flowchart for Octal MONITOR Program for MST-80 Microprocessor Trainer (Continued in Figure 5)

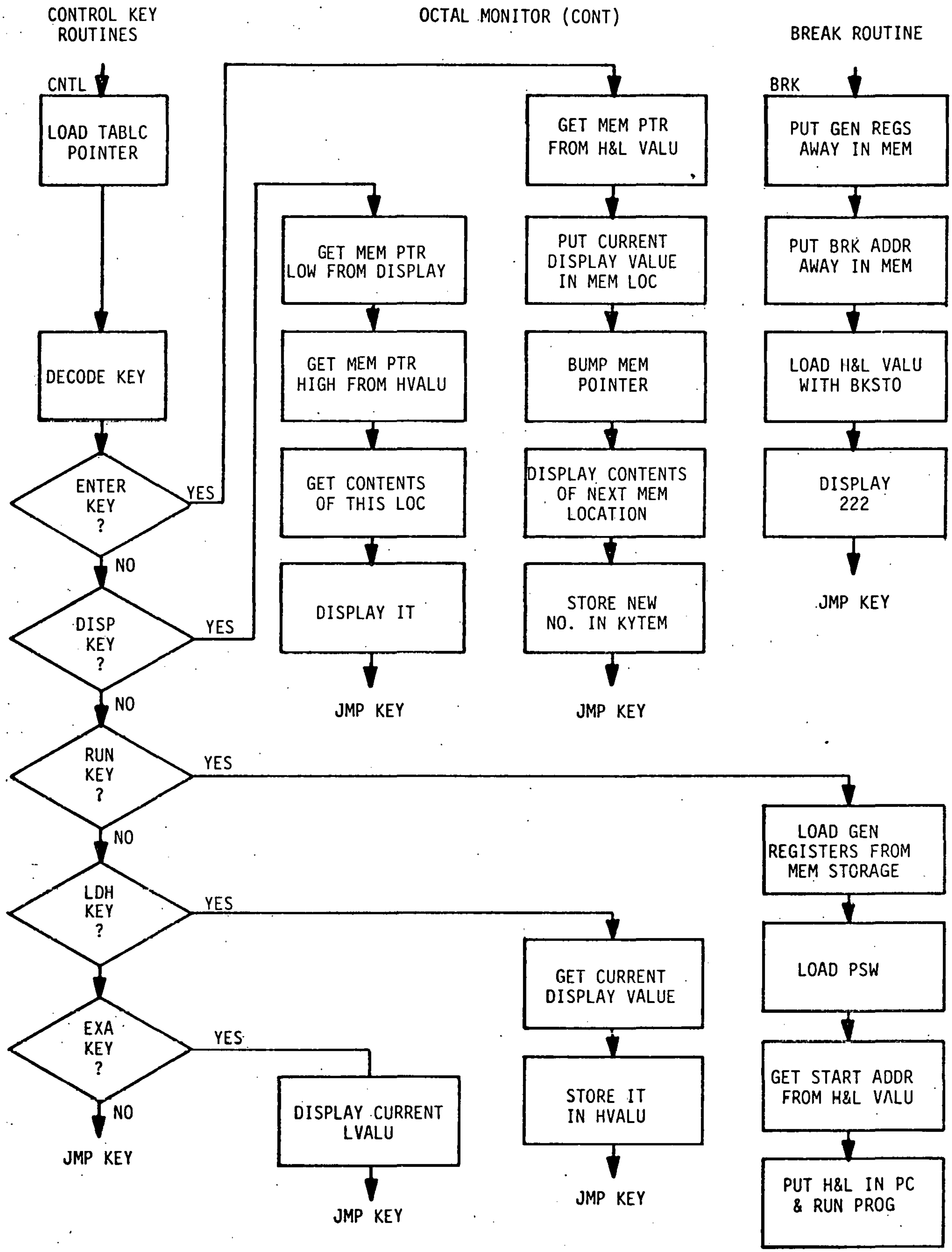


Figure 5. Flowchart for Octal MONITOR Program for MST-80 Microprocessor Trainer

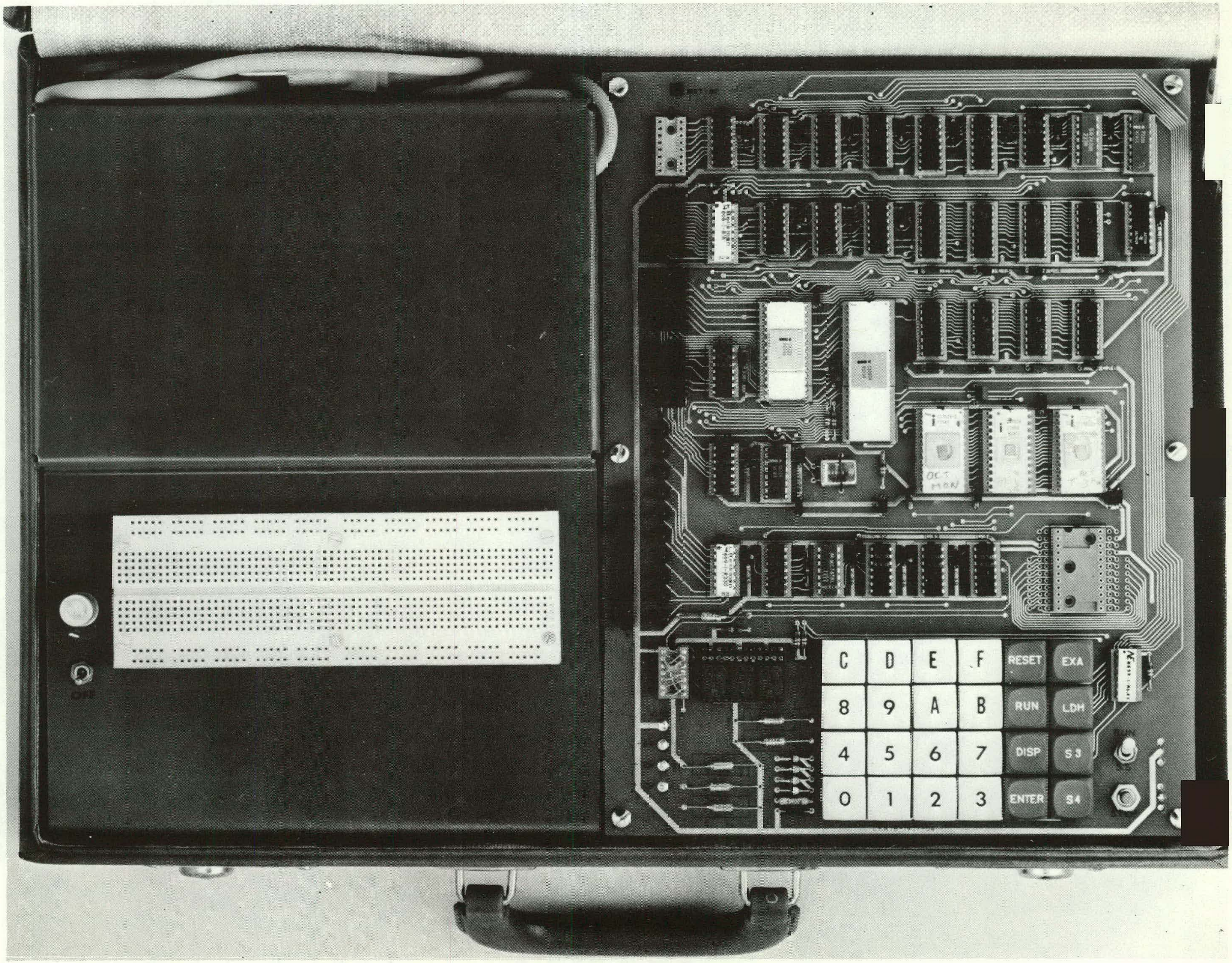


Figure 6. Complete MST-80 in Attache Case

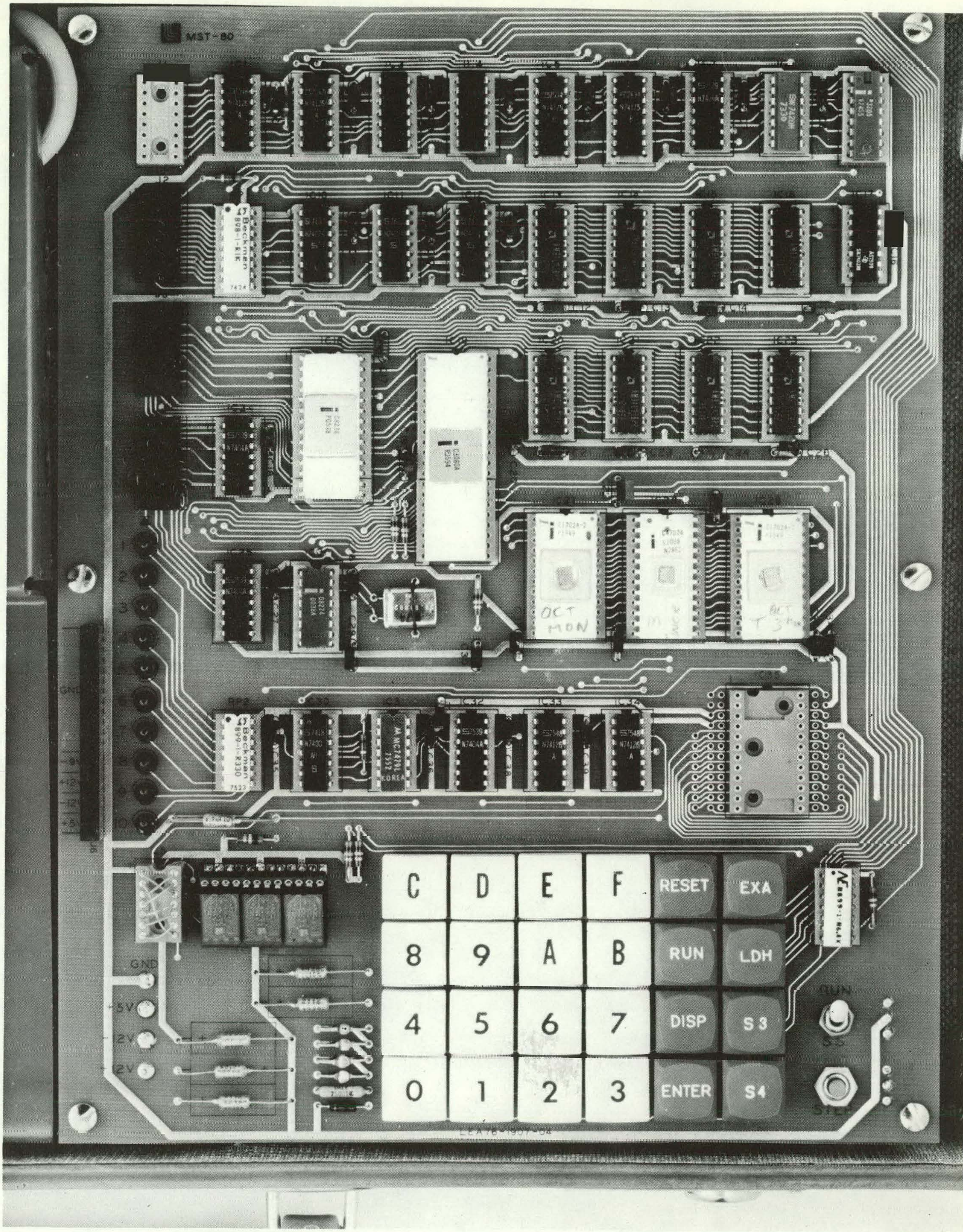


Figure 7. Close Up of MST-80 Circuit Card and Keyboard

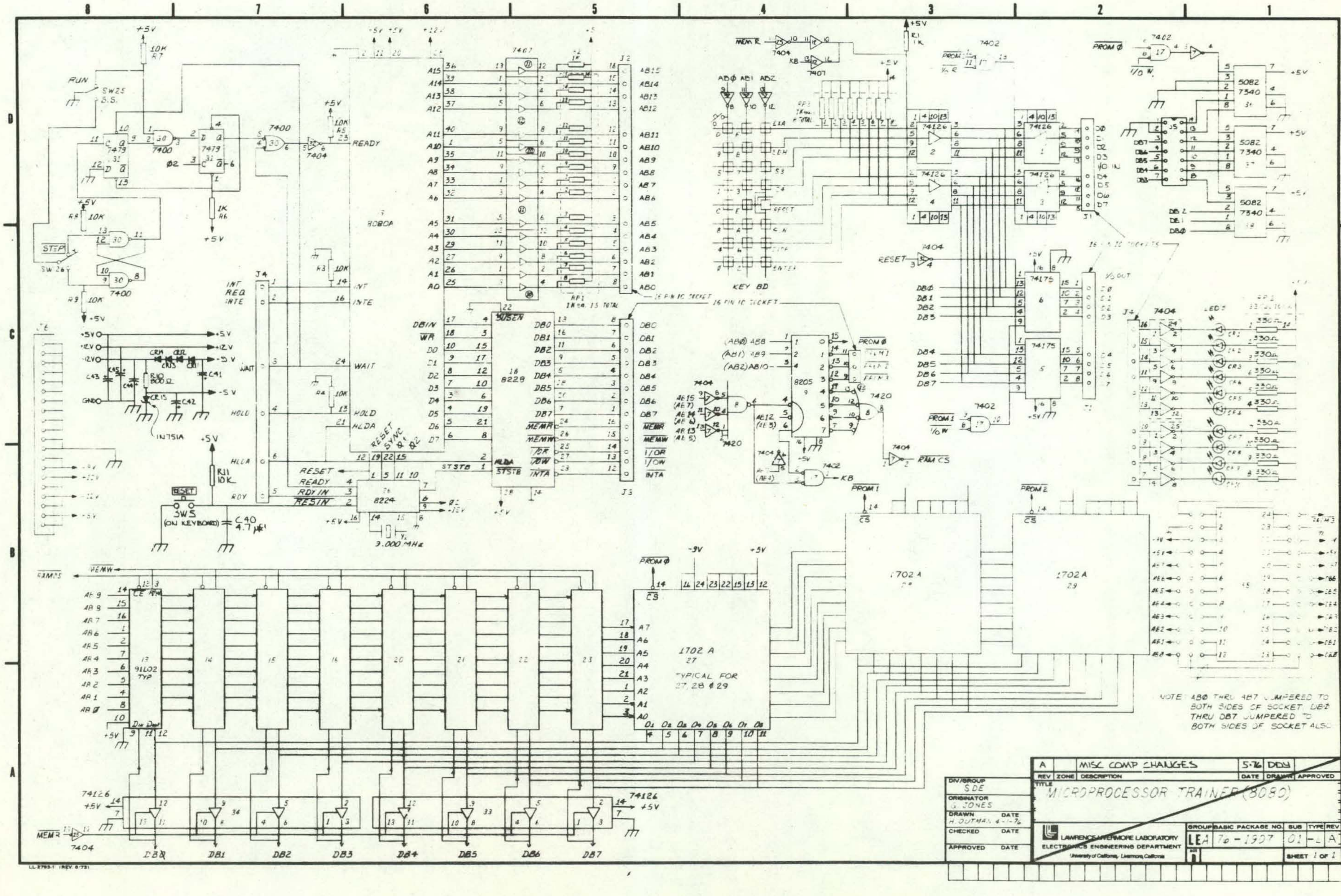


Figure 8. Circuit Schematic of MST-80 Microprocessor Trainer

```
8080 MACRO ASSEMBLER, VER 2.2  ERRORS = 0 PAGE 1

                                              ;++++++++++OCTAL MONITOR++++++++++
                                            ;++++++FOR MST-08 MICROPROCESSOR TRAINER+++++++

                                              ;WRITTEN BY GORDON JONES
                                              ;DATE: 4-15-76
                         ;
                         ;
                         ;
003616                   KYTEM     EQU    078EH
003620                   LVALU     EQU    0790H
003621                   HVALU     EQU    0791H
003622                   PCSTO     EQU    0792H
003624                   PSWST     EQU    0794H
003626                   BSTOR     EQU    0796H
003630                   DSTOR     EQU    0798H
003632                   HSTOR     EQU    079AH
                         ;
                         ;
                         ;
014007                   KEYBD     EQU    1807H
014001                   KYBD1     EQU    1801H
                         ;
000360                   TOP       EQU    0F0H
000017                   BOT       EQU    0FH
000000                   DISPL     EQU    0
000002                   RREAD     EQU    2Q
003622                   BKSTO     EQU    0792H

                                        ;++++++KEYBOARD READ ROUTINE+++++

000000                   ORG 0
000000   061 360 007 INIT:         LXI    SP,07F0H           ;INITIALIZE STACK POINTER
000003   076 000                   MVI    A,0
000005   323 000                   OUT    DISPL              ;PUT 000 IN DISPLAY
000007   062 216 007               STA    KYTEM              ;INITIALIZE DISPLAY STORAGE
000012   315 026 000 ST:           CALL   KEY                ;GO TO KEY ROUTINE
000015   303 012 000               JMP    ST                 ;GET BACK TO KEY ROUTINE IF NOT A CALL

000020   072 007 030 READ:         LDA    KEYBD              ;READ KEYBOARD
000023   057                       CMA                       ;COMPLEMENT
000024   267                       ORA    A                  ;SET FLAGS
000025   311                       RET

000026   327         KEY:          RST    RREAD              ;GO READ KEYBOARD
000027   302 026 000               JNZ    KEY                ;LOOP IF KEY DOWN
000032   315 243 000               CALL   DELAY              ;DEBOUNCE

000035   327         REP:          RST    RREAD              ;GO READ KEYBOARD
000036   312 035 000               JZ     REP                ;LOOP IF NO KEY DOWN
```

```
000041   315 243 000           CALL    DELAY            ;DEBOUNCE
000044   041 001 030 COL:      LXI     H,KYBD1          ;SET UP COLUMN POINTER
000047   176         LDKY:     MOV     A,M              ;READ KEYBOARD COLUMN
000050   057                   CMA                      ;COMPLEMENT
000051   267                   ORA     A                ;SET FLAGS
000052   302 131 000           JNZ     LUT              ;GOTO LOOK UP TABLE IF KEY FOUND
000055   175                   MOV     A,L              ;NO KEY FOUND - BUMP COLUMN POINTER
000056   027                   RAL                      ;ROTATE TO NEXT COLUMN
000057   157                   MOV     L,A              ;PUT BACK
000060   346 010               ANI     10Q              ;CHECK FOR LAST COLUMN
000062   312 047 000           JZ      LDKY             ;NOT LAST COLUMN - GO READ A KEY
000065   303 026 000           JMP     KEY              ;LAST COLUMN START OVER

000070               ORG       70Q

                     ;+++++++THIS IS THE BREAK ROUTINE+++++

000070   042 232 007 BRK:      SHLD    HSTOR            ;STORE H&L IN MEMORY
000073   341                   POP     H                ;PUT BREAK ADDRESS IN H&L REG
000074   053                   DCX     H                ;CORRECT BRK ADDR
000075   042 222 007           SHLD    PCSTOR           ;STORE BREAK ADDR IN MEMORY
000100   365                   PUSH    PSW              ;GET AC AND PSW IN STACK
000101   341                   POP     H                ;PUT AC &PSW IN H&L
000102   042 224 007           SHLD    PSWST            ;PUT AC &PSW IN MEMORY
000105   305                   PUSH    B                ;GET B&C
000106   341                   POP     H                ;PUT D&C IN MEMORY
000107   042 226 007           SHLD    BSTOR            ;PUT B&C IN MEMORY
000112   353                   XCHG                     ;PUT D&E IN H&L
000113   042 230 007           SHLD    DSTOR            ;PUT D&E IN MEMORY
000116   041 222 007           LXI     H,BKSTO          ;LOAD BREAK MEMORY LOCATION
000121   042 220 007           SHLD    LVALU            ;PUT IT IN PROPER LOCATION
000124   076 222               MVI     A,222Q           ;PUT 222 IN AC
000126   303 354 000           JMP     BACK             ;DISPLAY 222 AND RETURN TO KEY

                               ;+++++++THIS ROUTINE DETERMINES THE COLUMN
                               ;+++++++++THE KEY WAS FOUND IN AND LOOKS UP
                               ;+++++++++VALUE IN THE APPROPRIATE TABLE.

000131   107         LUT:      MOV     D,A              ;SAVE AC
000132   175                   MOV     A,L              ;GET COLUMN POINTER
000133   017                   RRC                      ;ROTATE COL POINTER RIGHT
000134   332 146 000           JC      COL1             ;IS IT COL1?
000137   017                   RRC                      ;ROTATE AGAIN
000140   332 157 000           JC      COL2             ;IS IT COL2?
000143   303 254 000           JMP     CNTL             ;MUST BE CONTROL COLUMN

000146   041 222 000 COL1:     LXI     H,TABLE-1        ;GET TABLE POINTER
```

```
000151  315 211 000          CALL   DECOD        ;GO GET VALUE FROM TABLE
000154  303 170 000          JMP    SHIFT        ;STORE AND SEND TO DISPLAY
000157  041 222 000 COL2:    LXI    H,TABLE-1    ;GET TABLE POINTER
000162  315 211 000          CALL   DECOD        ;GET VALUE FROM TABLE
000165  306 002              ADI    2Q           ;CORRECT VALUE FOR COLUMN 2
000167  117                  MOV    C,A
000170  372 026 000 SHIFT:   JM     KEY          ;ILLEGAL CHARACTER CHECK
000173  041 216 007          LXI    H,KYTEM      ;GET OLD DISPLAY VALUE
000176  176                  MOV    A,M
000177  007                  RLC                 ;ROTATE ONE OCTAL DIGIT LEFT
000200  007                  RLC
000201  007                  RLC
000202  346 370              ANI    370Q         ;MASK OFF BOTTOM DIGIT
000204  261                  ORA    C            ;OR NEW DIGIT TO OLD NUMBER
000205  167                  MOV    M,A          ;PUT BACK IN DISPLAY STORAGE
000206  323 000              OUT    DISPL        ;SEND TO DISPLAY
000210  311                  RET                 ;END OF NUMBER KEY ROUTINE

000211  170         DECOD:   MOV    A,B          ;GET KEY VALUE
000212  027         AGAIN:   RAL                 ;ROTATE INTO CARRY
000213  043                  INX    H            ;BUMP TABLE POINTER
000214  322 212 000          JNC    AGAIN        ;
000217  116                  MOV    C,M          ;SAVE KEY CODE
000220  171                  MOV    A,C          ;PUT KEY CODE IN AC
000221  267                  ORA    A            ;SET FLAGS
000222  311                  RET

000223  000         TABLE:   DB     0Q           ;NUMBER KEYS CODE TABLE
000224  004                  DB     4Q
000225  200                  DB     200Q
000226  200                  DB     200Q
000227  001                  DB     1Q
000230  005                  DB     5Q
000231  200                  DB     200Q
000232  200                  DB     200Q
000233  364         TABLC:   DB     ENTER        ;CONTROL ROUTINES ADDRESS TABLE
000234  337                  DB     DISP
000235  303                  DB     RUN
000236  026                  DB     KEY
000237  026                  DB     KEY
000240  026                  DB     KEY
000241  272                  DB     LDH
000242  264                  DB     EXA


                             ;THIS IS A DELAY SUBROUTINE TO DEBOUNCE THE KEY SWITCHES

000243  006 000     DELAY:   MVI    B,0          ;INITIALIZE COUNTER
000245  004         LOOP:    INR    B            ;BUMP COUNTER
000246  343                  XTHL                ;EXTRA DELAY IN LOOP
```

```
000247   343                    XTHL
000250   302 245 000            JNZ     LOOP            ; LOOP UNTIL ZERO
000253   311                    RET

                             ;THESE ARE THE CONTROL KEY ROUTINES
000254   041 232 000 CNTL:      LXI     H,TABLC-1       ;GET TABLE POINTER
000257   315 211 000            CALL    DECOD           ;GET ADRESS FROM TABLE
000262   151                    MOV     L,C             ;MOVE ADDRESS INTO L REG
000263   351                    PCHL                    ;JUMP TO PROPER CONTROL ROUTINE
000264   072 220 007 EXA:       LDA     LVALU           ;GET L REGISTER VALUE
000267   303 354 000            JMP     BACK            ;DISPLAY IT & JUMP TO KEY
000272   072 216 007 LDH:       LDA     KYTEM           ;GET KEY VALUE FROM TEMP
000275   062 221 007            STA     HVALU           ;PUT IN H REGISTER STORAGE
000300   303 026 000            JMP     KEY             ;DONE- GO TO START


000303   072 216 007 RUN:       LDA     KYTEM           ;GET CURRENT DISPLAY VALUE
000306   062 220 007            STA     LVALU           ;STORE IN L REG LOCATION
000311   052 226 007            LHLD    BSTOR           ;GET CONTENTS OF B&C REGS
000314   345                    PUSH    H               ;PUT ON STACK
000315   301                    POP     B               ;PUT IN B&C REGS
000316   052 230 007            LHLD    DSTOR           ;GET CONTENTS OF D&E REGS
000321   353                    XCHG                    ;EXCHANGE H&L WITH D&E
000322   052 224 007            LHLD    PSWST           ;GET OLD AC AND PSW
000325   345                    PUSH    H               ;PUT AC & PSW ON STACK
000326   361                    POP     PSW             ;RESTORE AC & STATUS
000327   052 220 007            LHLD    LVALU           ;GET STARTING ADDRESS
000332   345                    PUSH    H               ;PUT STARTING ADDR ON STACK
000333   052 232 007            LHLD    HSTOR           ;RESTORE H&L
000336   311                    RET                     ;GET STARTING ADDR FROM STACK AND RUN

000337   072 216 007 DISP:      LDA     KYTEM           ;GET CURRENT DISPLAY VALUE
000342   062 220 007            STA     LVALU           ;STORE IN LREG STORAGE
000345   052 220 007            LHLD    LVALU           ;GET VALUE JUST KEYED IN
000350   042 220 007 NEXT:      SHLD    LVALU           ;STORE IN MEMORY POINTER
000353   176                    MOV     A,M             ;GET VALUE POINTED TO BY MEM POINTER
000354   062 216 007 BACK:      STA     KYTEM           ;PUT THIS VALUE IN KEY STORAGE
000357   323 000                OUT     DISPL           ;DISPLAY IT
000361   303 026 000            JMP     KEY             ;GO BACK AND START OVER
000364   052 220 007 ENTER:     LHLD    LVALU           ;GET MEMORY POINTER
000367   072 216 007            LDA     KYTEM           ;GET DISPLAY VALUE
000372   167                    MOV     M,A             ;PUT VALUE IN LOC POINTED TO BY H&L
000373   043                    INX     H               ;BUMP TO NEXT LOCATION
000374   303 350 000            JMP     NEXT            ;PUT INC PTR AWAY AND DISPLAY NEXT LOC

                                END
```

NO PROGRAM ERRORS

SYMBOL TABLE

* 01

A	000007		AGAIN	000212		B	000000		BACK	000354	
BKSTO	003622		BOT	000017	*	BRK	000070	*	BSTOR	003626	
C	000001		CNTL	000254		COL	000044	*	COL1	000146	
COL2	000157		D	000002		DECOD	000211		DELAY	000243	
DISP	000337		DISPL	000000		DSTOR	003630		E	000003	
ENTER	000364		EXA	000264		H	000004		HSTOR	003632	
HVALU	003621		INIT	000000	*	KEY	000026		KEYBD	014007	
KYBD1	014001		KYTEM	003616		L	000005		LDH	000272	
LDKY	000047		LOOP	000245		LUT	000131		LVALU	003620	
M	000006		NEXT	000350		PCSTO	003622		PSW	000006	
PSWST	003624		READ	000020	*	RTP	000035		RREAD	000002	
RUN	000303		SHIFT	000170		SP	000006		ST	000012	
TABLC	000233		TABLE	000223		TOP	000360	*			

DISTRIBUTION

Internal Distribution

L.L. Cleland, L-156	
G.D. Jones, L-156	(10)
E.A. Lafranchi, L-151	
H.C. McDonald, L-161	
A.W. Olson, L-152	
C.A. Larsen, L-73	
ERD File	(02)
TID	(15)

External Distribution

Argonne Code Center Att'n: M. Butler Argonne National Lab. 9700 South Cass Ave. Argonne, IL 60439	(01)
TIC, Oak Ridge	(27)
Distribution through UCLLL TIP Program	(100)

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161
Price: Printed Copy $; Microfiche $2.25

Page Range	Domestic Price	Page Range	Domestic Price
001–025	$ 3.50	326–350	10.00
026–050	4.00	351–375	10.50
051–075	4.50	376–400	10.75
076–100	5.00	401–425	11.00
101–125	5.25	426–450	11.75
126–150	5.50	451–475	12.00
151–175	6.00	476–500	12.50
176–200	7.50	501–525	12.75
201–225	7.75	526–550	13.00
226–250	8.00	551–575	13.50
251–275	9.00	576–600	13.75
276–300	9.25	601–up	*
301–325	9.75		

*Add $2.50 for each additional 100 page increment from 601 to 1,000 pages: add $4.50 for each additional 100 page increment over 1,000 pages.

Technical Information Department
LAWRENCE LIVERMORE LABORATORY
University of California | Livermore, California | 94550