8/3/77

27

250 NTLS

UCID- 17526

Lawrence Livermore Laboratory

THE MST-80B MICROCOMPUTER TRAINER

G. D. Jones/J. W. Spencer

June 14, 1977

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.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

PREFACE

Microcomputers are bringing about a revolution in the design of electronic systems. All electronic design is being affected by the potential for better, cheaper, faster, and smarter new systems using microcomputers.

With all this activity, education has become a great concern: many technical people have heard of microcomputers, relatively few know exactly what microcomputers are, what they do, and how best to apply them.

As a high-technology, energy-related, research laboratory, LLL has been in the forefront of microcomputer development and application. We have had to devise effective means for rapidly educating our own employees about microcomputers. Now--through our Technology Training Program (TTP)--we are bringing some of this technology outside the Laboratory to other energy-related industrial and government organizations, and to educational institutions.

The LLL-developed MST-80B microcomputer trainer began as a part of this technology transfer effort. Now, in addition to the trainers used here at LLL more than 200 are being built by more than 30 colleges for use in their own classes.

The MST-80B is a significant contribution to the electronics industry's efforts to devise effective means of educating potential users of microcomputers. The trainer allows hands-on experience with state-of-the-art microcomputer architecture, programming, interfacing, and application design. When used in conjunction with a well-rounded classroom presentation, the trainer, by encouraging immediate "hardware" application of concepts developed in class, helps to both reinforce and expand comprehension.

NOTICE
This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research and Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

-i-

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

leg

<u>CONTENTS</u>	<u>PAGE</u>
Preface	i
Abstract	1
An Introduction To The MST-80B	1
Hardware Features Of The Trainer	2
MONITOR Program	3
Operation Of Keyboard Using The HEX/OCT MONITOR	4
MONITOR Subroutines As A Call From A User Program	7
Sample Program For The MST-80B	8
Using Breakpoints in Program Debugging	10
Acknowledgements	12
Figures	13
APPENDIX	
8080 CPU Instruction Set	
Alphabetical List, by mnemonic	22
Numerical List, HEX and OCTAL	23
HEX/OCT MONITOR Program Listing	24

THE MST-80B MICROCOMPUTER TRAINER

G. D. Jones/J. W. Spencer

Electronics Engineering Department
Lawrence Livermore Laboratory

ABSTRACT

The microcomputer revolution in electronics is spreading so rapidly that it is difficult to educate enough people in the technology both quickly and thoroughly. LLL's MST-80B was developed as a way to speed learning in our in-house training courses, and it is now being widely used outside LLL. The MST-80B trainer is a complete, self-contained, microcomputer system housed in a standard briefcase. The trainer uses the Intel 8080A* 8-bit microprocessor (CPU), and has its own solid-state memory and a built-in keyboard and display for Input/Output. The trainer is furnished with a permanent "monitor" program (in Read-Only Memory) that allows users to easily enter, debug, modify, and run programs of their own.

AN INTRODUCTION TO THE MST-80B

The LLL MST-80B is a complete, self-contained, microcomputer system housed in a briefcase for portability and convenience of use. It uses Intel's 8080A microprocessor and associated supporting integrated circuits.

* Reference to a company or product name here or elsewhere 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.

The microcomputer is specifically designed as a training device, a convenient means for allowing students to explore and learn the hardware and software capability of a typical microcomputer.

The trainer has its own set of solid-state memory elements so no external memory is required. Both random-access read/write memory (RAM) and programmable read-only memory (PROM) are provided. The trainer includes a keyboard and a numerical display for the student to communicate with the microcomputer. This Input/Output (I/O) combination eliminates the need for expensive and bulky I/O such as a teletypewriter. The keyboard and numerical display can be used with either the octal (base 8) number system or the hexadecimal (base 16) number system. A student can select either number system by simply depressing a control key.

The trainer includes a breadboard socket so that experiments can be interfaced to the microcomputer. This allows the student to learn hardware interfacing techniques as well as software programming.

HARDWARE FEATURES OF THE TRAINER

Figure 1 shows the complete trainer in its case. Figure 2 is a closeup of the computer circuit board showing the keyboard, display, and electronic circuitry.

The MST-80B uses Intel's 8080A central processor unit (microprocessor or CPU) and supporting integrated circuits. The 8080A is a second-generation microprocessor, with an 8-bit word and 78 instructions. (Appendix I lists the available instruction set.) The MST-80B:

- Has 512 bytes of RAM memory.
- Has sockets for three 1702A PROM's (768 bytes). It also includes one uncommitted socket that can be jumper-wired to a 24 PIN ROM of user's choice. Normally the monitor program resides in PROM 0 and PROM 1.

- Has a 24-key keyboard. This input device is accessed through memory mapped I/O. (See Figure 7 for a memory map.)
- Has a three digit display with full hex number capability. This output device is used by calling the DISPLAY subroutine in the monitor. (See the Monitor Subroutines section for a description of how this subroutine is used.)
- Has one 8-bit input port. Address = 1.
- Has one 8-bit output port (latched). Address = 1.
- Has single-step capability.
- Has ten uncommitted light-emitting diodes (LEDs) that can easily be connected to display the state of any desired signals (address lines, data lines, status, etc.). These can be used when operating in single-step mode.

Figure 3 is a block diagram of the MST-80B and Figure 4 a schematic diagram. Figure 5 shows the panel connectors used to interface the trainer and includes detailed information on each signal and its connector pin number.

MONITOR PROGRAM

The trainer, as supplied, includes a monitor program (HEX/OCT MONITOR) loaded in PROMS 0 and 1. This monitor program allows a user to enter programs into RAM memory, to select and examine memory locations, change contents of locations, and run user programs from specified starting addresses.

The monitor program also includes a debug routine to assist users in debugging their programs. This routine allows the user to insert breakpoints $(377_8)^*$ in programs. When such a breakpoint is encountered during program execution, the break routine in the monitor program is entered; it saves all the CPU registers and the breakpoint address, and will display 273 or BB** to signal the user that a breakpoint has been encountered.

The contents of the CPU registers and the breakpoint address are saved in a group of dedicated memory locations on memory page 7.*** These locations can be examined by using the display (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 explanation of these features is included in the sample program discussion later in this report.)

The RUN feature of the monitor program starts the user's program with the CPU registers initialized to the current values found in the dedicated memory locations. (This allows an operator to change these values before pushing RUN.) Figure 6 is a flowchart of the monitor program; a complete program listing is included as Appendix IV. Figure 7 is a memory map for the computer system. A sample program using the monitor is discussed later.

OPERATION OF KEYBOARD USING THE HEX/OCT MONITOR

The MST-80B keyboard layout is shown below. The keys function as follows:

C	D	E	F	RESET	EXA
8	9	A	B	RUN	LDH
4	5	6	7	DISP	H/O
0	1	2	3	ENTER	SS

*The subscript 8 on a number indicates an octal (base eight) representation.

**Display depends on user-selected mode: 273 is the octal display, BB the hexadecimal.

***The locations used are tabulated on page 10 of this report.

RESET: This key resets the system and starts the monitor program running.

DIGIT

KEYS: These keys cause the selected digits 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 001 to insure that any existing number is completely replaced.) Keys 8 through F are functional only in HEX mode; they are ignored when in OCTAL mode.

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

LDH: Load High Order Address. In order to address any location in memory the user needs to specify the complete address. The MST-80B addresses are two 8-bit bytes: the high order address and the low order address. The high order address is specified by keying the desired value into the display and then pushing LDH (LOAD H). This stores the 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: Display. 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 to 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; when ENTER is pushed this value will be entered into the currently-addressed location.

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

EXA: Examine address. This key displays the current value of the low order address. It is particularly useful if, when you are examining a program (stepping through, using ENTER), you forget where you are.

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

SS: Single Step. This key is used in single step mode to advance the program to the next step. The toggle switch labeled SS-RUN must be in the SS position before the SS key is functional.

H/O: Hex/Octal. This key selects the desired keyboard mode. When RESET is pushed after first turning on power, the keyboard will be in HEX mode. Depressing the H/O key will then cause a switch to OCTAL mode. Depressing the H/O key again will cause the mode to switch back to HEX. In short, depressing the H/O key changes the keyboard mode from its present mode to the other mode.

MONITOR SUBROUTINES AS A CALL FROM A USER PROGRAM

Two of the routines in HEX/OCT MONITOR are written as subroutines and may be called by a user program:

The KEY routine in the monitor program is useful when a user's program requires operator interaction since the keyboard is convenient for this purpose.

KEY as a subroutine is called by a CALL KEY instruction $(315\ 131\ 000)_8$. When called, an appropriate number key for the mode in use must be depressed by the user before a return to the user program will be completed. KEY returns to the user with the C register containing the value of the number key depressed. (The C register contains this number in the low order octal digit, and in addition contains the previous two key entries in the high order digits.)

Two precautions must be observed when using the KEY subroutine. First, the routine uses the A, B, C, H and L registers. If the user program also requires these 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 in OCTAL mode.

The DISPLAY routine in the monitor program is another useful subroutine available to the user. Whenever the user wants to send a number to the digital display, this routine should be used. The subroutine is called by a CALL DISPLAY instruction $(315\ 122\ 001)_8$ and will display the number currently in the A register in whichever mode (HEX or OCTAL) is presently in use. This subroutine uses the A, B and C registers.

SAMPLE PROGRAM FOR THE MST-80B

A sample program for the MST-80B is given below; Figure 8 is a flowchart for the program. This sample program can be used to demonstrate the operation of the MST-80B and the use of the monitor program in OCTAL mode. (Since the MST-80B is programmed in machine language, program "steps" are often written as mnemonics--abbreviated indications of what the instruction does. For example, MVI means "MoVe ImmEDIATEly"; MVI A, 0 means "MoVe ImmEDIATEly (into the Accumulator) zero(s)". Many of the mnemonics found in MST-80B programs such as this one can be easily understood from the context. If you have questions, Appendix I includes a complete list of 8080A instruction mnemonics and their meanings.)

<u>MEMORY LOCATION</u>	<u>MACHINE CODE IN OCTAL</u>	<u>OPERATIONS</u>
000	076	MVI A, 0 ; CLEAR AC
001	000	
002	127	AGAIN: MOV D, A ; SAVE A
003	315	CALL DISPLAY ; SEND AC TO DISPLAY
004	122	
005	001	
006	172	MOV A, D ; RESTORE A
007	006	MVI B, 0 ; CLR B REGISTER
010	000	
011	016	MVI C, 100 ; PUT 100 IN C REGISTER
012	100	
013	004	LOOP: INR B ; INCREMENT B
014	312	JZ LOOP ; DO IT AGAIN
015	013	
016	006	
017	015	DCR C ; DECREMENT C
020	302	JNZ LOOP ; LOOP UNTIL ZERO
021	013	
022	006	
023	306	ADI 001 ; ADD ONE TO AC
024	001	
025	303	JMP AGAIN ; GO DISPLAY AC & DO AGAIN
026	002	
027	006	

First you must load the sample program into memory: before you start, you need to decide where to load it. Let's put it in memory page 6, starting at location 0 (absolute address = 0600 hex). First, key 006 into the display and then push the LDH (load H) key. This sets the high-order address (High byte) to page 6. Next, key 000 into the display and push the DISP key. This will display the current contents of location 0 on page 6. Now you can key in the machine language code for the first instruction, 076 (MVI A), and push the ENTER key. This will enter the 076 into location 0, and will also display the contents of the next location (location 1). Now you can key in the next code, 000, and push ENTER again. The 000 will be entered into location 1, and then location 2 will be displayed. Continue this process until the entire program is entered.

If you make a mistake while keying in a number, just continue to key in until the correct value appears in the display. (The entered, displayed, number is not used until a control key is pressed.) If you forget where you are at any time while loading the program, just press EXA (examine address), and the current low-order address will appear in the display. You can continue on from that point by first pushing the DISP key and then the ENTER key. Or you can key 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 (000 for our sample program), pushing the DISP key, and then repeatedly pushing the ENTER key. This will step through the program sequentially and display each location so it can be checked. If you find a mistake, just key in the correct value before the ENTER key is pushed.

After the program is loaded satisfactorily, you can run it if you desire. To run the program, key the starting address (000 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 BREAKPOINTS IN PROGRAM DEBUGGING

The monitor program for the MST-80B allows users to set breakpoints at desired locations in their programs. This can be a very useful capability, particularly when debugging a program. The use of breakpoints in program debugging can be demonstrated using the BREAK routine with the sample program introduced in the preceding section.

As can be seen from the flowchart of the sample program, Figure 8, 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 breakpoints to find it.

Looking at the flow chart, you can see that there are two counting loops. The first loop counts up to 377_8 and then goes back to 0. Then the second count loop is entered. This second loop counts the number of times the first loop must go through 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 $400_8 \times 100_8 (=16,384_{10})$ 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 can be done by inserting a breakpoint (breakpoint code = 377_8) in place of the INR B instruction at memory location 013. Now run the program. (Remember to set the high order address to page 6.) When the breakpoint is encountered during the running of the program, the BREAK routine will stop execution of the program at that point and store the contents of all CPU registers in the dedicated memory locations shown below. A 273 will appear in the display to signal you that a break has occurred.

BREAK ROUTINE MEMORY STORAGE LOCATIONS (MEMORY PAGE 7)
(OCTAL Mode)

<u>LOC</u>	<u>CONTENTS</u>	<u>LOC</u>	<u>CONTENTS</u>
273	PCL	300	B REG
274	PCH	301	E REG
275	PSW	302	D REG
276	A REG	303	L REG
277	C REG	304	H REG

The BREAK routine also automatically sets HVALU to page 7. So, since 273 is already being displayed, if you now push the DISP key, the contents of memory location 273 on page 7 will be displayed. This location contains the low byte of the address where the break occurred. The high byte of the break address is stored in location 274, so pushing the ENTER key will cause the high byte to be displayed. Repeated use of the ENTER key will allow you to examine the contents of all the CPU registers.

Register C is stored in location 277 and, for our sample program, should contain 100₈. Location 276 (A register) and 300 (B register) should contain zero. If these locations contain the correct values, replace the INR B instruction (code 004) in location 013 and put a breakpoint (377) in location 017 in place of the DCR C instruction. Run the program. When it breaks, examine location 300 again to see what the B register 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.

When you examine 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 302) instruction. Replace the breakpoint in 017 with DCR C (015) and run the program. It should now run correctly, with the display counting much more slowly.

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.

ACKNOWLEDGEMENTS

The contributions of Eugene R. Fisher and James M. Spann to the design of the trainer and its operating features are gratefully acknowledged.

Figure 1. The MST-80B microcomputer trainer in its case.

Figure 2.

A closeup of the circuit board for the MST-80B microcomputer trainer. The display is just left of the "C" key.

Figure 3. Operational block diagram of MST-80B Microcomputer trainer.

Figure 4. Schematic of MST-80B Microcomputer.

Figure 5. Panel connectors used to interface MST-80B microcomputer trainer.

Figure 6. Flow chart for
HEX/OCT monitor program for MST-80B
Microcomputer trainer.

HEX		SPLIT OCTAL
0000	PAGE 0 (PROM) MONITOR PROGRAM	000 000
00FF		000 377
0100	PAGE 1 (PROM) MONITOR PROGRAM	001 000
01FF		001 377
0200	PAGE 2 (PROM)	002 000
02FF		002 377
0300	PAGE 3 UNCOMMITTED SOCKET	003 000
03FF		003 377
0400	PAGE 4 UNCOMMITTED SOCKET	004 000
04FF		004 377
0500	PAGE 5 KEYBOARD	005 000
05FF		005 377
0600	PAGE 6 (RAM)	006 000
06FF		006 377
0700	PAGE 7 (RAM) REGISTER STORAGE & STACK	007 000
07FF		007 377
0800	NOT USED IN MST-80B	010 000
FFFF		377 377

Page 7 locations used by monitor program:

OCTAL/HEX
LOCATION

CONTENTS

267/B7	KYTEM	} current value of display
271/B9	LVALU	
272/BA	HVALU	
273/BB	PCL	} PCSTO
274/BC	PCH	
275/BD	PSW	} PSWST
276/BE	A REG	
277/BF	C REG	} BSTOR
300/C0	B REG	
301/C1	E REG	} DSTOR
302/C2	D REG	
303/C3	L REG	} HSTOR
304/C4	H REG	
305/C5	OFLAG	

360/F0 STACK PTR

FLAGWORD

D7 D6 D5 D4 D3 D2 D1 D0

S Z 0 AC 0 P 1 CY

SIGN ———

ZERO ———

AUX CARRY ———

PARITY ———

CARRY ———

Figure 7. Memory map for MST-80B Microcomputer trainer.

Figure 8. Flow chart for MST-80B sample program.

APPENDIX

I. Summary of 8080 instruction set.

Summary of Processor Instructions By Alphabetical Order

Mnemonic	Description	Instruction Code ⁽¹⁾								Clock ⁽²⁾ Cycles
		D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	7
ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0	7
ADC r	Add register to A with carry	1	0	0	0	1	S	S	S	4
ADI M	Add memory to A	1	0	0	0	0	1	0	1	7
ADD r	Add register to A	1	0	0	0	0	S	S	S	4
ADI	Add immediate to A	1	1	0	0	0	1	1	0	7
ANA M	And memory with A	1	0	1	0	0	1	1	0	7
ANA r	And register with A	1	0	1	0	0	S	S	S	4
ANI	And immediate with A	1	1	1	0	0	1	1	0	7
CALL	Call unconditional	1	1	0	0	1	1	0	1	17
CC	Call on carry	1	1	0	1	1	1	0	0	11/17
CM	Call on minus	1	1	1	1	1	1	0	0	11/17
CMA	Complement A	0	0	1	0	1	1	1	1	4
CNC	Complement carry	0	0	1	1	1	1	1	1	4
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	7
CMP r	Compare register with A	1	0	1	1	1	S	S	S	4
CNC	Call on no carry	1	1	0	1	0	1	0	0	11/17
CNZ	Call on no zero	1	1	0	0	0	1	0	0	11/17
CP	Call on positive	1	1	1	1	0	1	0	0	11/17
CPE	Call on parity even	1	1	1	0	1	1	0	0	11/17
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	7
CPO	Call on parity odd	1	1	1	0	0	1	0	0	11/17
CZ	Call on zero	1	1	0	0	1	1	0	0	11/17
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	4
DAD B	Add B & C to H & L	0	0	0	0	1	0	0	1	10
DAD D	Add D & E to H & L	0	0	0	1	1	0	0	1	10
DAD H	Add H & L to H & L	0	0	1	0	1	0	0	1	10
DAD SP	Add stack pointer to H & L	0	0	1	1	1	0	0	1	10
DCR M	Decrement memory	0	0	1	1	0	1	0	1	10
DCR r	Decrement register	0	0	0	0	0	1	0	1	5
DCX B	Decrement B & C	0	0	0	0	1	0	1	1	5
DCX D	Decrement D & E	0	0	0	1	1	0	1	1	5
DCX H	Decrement H & L	0	0	1	0	1	0	1	1	5
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	5
DI	Disable Interrupts	1	1	1	1	0	0	1	1	4
EI	Enable Interrupts	1	1	1	1	1	0	1	1	4
HLT	Halt	0	1	1	1	0	1	1	0	7
IN	Input	1	1	0	1	1	0	1	1	10
INR M	Increment memory	0	0	1	1	0	1	0	0	10
INR r	Increment register	0	0	0	0	0	1	0	0	5
INX B	Increment B & C registers	0	0	0	0	0	0	1	1	5
INX D	Increment D & E registers	0	0	0	1	0	0	1	1	5
INX H	Increment H & L registers	0	0	1	0	0	0	1	1	5
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	5
JC	Jump on carry	1	1	0	1	1	0	1	0	10
JM	Jump on minus	1	1	1	1	1	0	1	0	10
JMP	Jump unconditional	1	1	0	0	0	0	1	1	10
JNC	Jump on no carry	1	1	0	1	0	0	1	0	10
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	10
JP	Jump on positive	1	1	1	1	0	0	1	0	10
JPE	Jump on parity even	1	1	1	0	1	0	1	0	10
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	10
JZ	Jump on zero	1	1	0	0	1	0	1	0	10
LDA	Load A direct	0	0	1	1	1	0	1	0	13
LDAX B	Load A indirect	0	0	0	0	1	0	1	0	7
LDAX D	Load A indirect	0	0	0	1	1	0	1	0	7
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	16
LXI B	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	10
LXI D	Load immediate register Pair D & E	0	0	0	1	0	0	0	1	10
LXI H	Load immediate register Pair H & L	0	0	1	0	0	0	0	1	10
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	10
MVI M	Move immediate memory	0	0	1	1	0	1	1	0	10
MVI r	Move immediate register	0	0	0	0	0	1	1	0	7
MOV M, r	Move register to memory	0	1	1	1	0	S	S	S	7
MOV r, M	Move memory to register	0	1	0	0	0	1	1	0	7
MOV r ₁ , r ₂	Move register to register	0	1	1	1	0	S	S	S	5
NOP	No-operation	0	0	0	0	0	0	0	0	4
ORA M	Or memory with A	1	0	1	1	0	1	1	0	7
ORA r	Or register with A	1	0	1	1	0	S	S	S	4
ORI	Or immediate with A	1	1	1	1	0	1	1	0	7
OUT	Output	1	1	0	1	0	0	1	1	10
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	5
POP B	Pop register pair B & C off stack	1	1	0	0	0	0	0	1	10
POP D	Pop register pair D & E off stack	1	1	0	1	0	0	0	1	10
POP H	Pop register pair H & L off stack	1	1	1	0	0	0	0	1	10
POP PSW	Pop A and Flags off stack	1	1	1	1	0	0	0	1	10
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	11
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	11
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	11
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1	11
RAL	Rotate A left through carry	0	0	0	1	0	1	1	1	4
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	4
RC	Return on carry	1	1	0	1	1	0	0	0	5/11
RET	Return	1	1	0	0	1	0	0	1	10
RLC	Rotate A left	0	0	0	0	0	1	1	1	4
RM	Return on minus	1	1	1	1	0	0	0	0	5/11
RNC	Return on no carry	1	1	0	1	0	0	0	0	5/11
RNZ	Return on no zero	1	1	0	0	0	0	0	0	5/11
RP	Return on positive	1	1	1	1	0	0	0	0	5/11
RPE	Return on parity even	1	1	1	0	1	0	0	0	5/11
RPO	Return on parity odd	1	1	1	0	0	0	0	0	5/11
RRC	Rotate A right	0	0	0	0	1	1	1	1	4
RST	Restart	1	1	A	A	A	1	1	1	11
RZ	Return on zero	1	1	0	0	1	0	0	0	5/11
SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	7
SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S	4
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	7
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	16
SPHL	Store H & L to stack pointer	1	1	1	1	1	0	0	1	5
STA	Store A direct	0	0	1	1	0	0	1	0	13
STAX B	Store A indirect	0	0	0	0	0	0	1	0	7
STAX D	Store A indirect	0	0	0	1	0	0	1	0	7
STC	Set carry	0	0	1	1	0	1	1	1	4
SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	7
SUB r	Subtract register from A	1	0	0	1	0	S	S	S	4
SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	7
XCHG	Exchange D & E, H & L Registers	1	1	1	0	1	0	1	1	4
XRA M	Exclusive Or memory with A	1	0	1	0	1	1	1	0	7
XRA r	Exclusive Or register with A	1	0	1	0	1	S	S	S	4
XRI	Exclusive Or immediate with A	1	1	1	0	1	1	1	0	7
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1	18

NOTES: 1. DDD or SSS — 000 B — 001 C — 010 D — 011 E — 100H — 101L — 110 Memory — 111 A.
2. Two possible cycle times, (5/11) indicate instruction cycles dependent on condition flags.

From the Intel 8080 Microprocessor Systems User's Manual,
Courtesy of Intel Corporation.

APPENDIX

II. 8080 ASSEMBLY LANGUAGE REFERENCE CARD

ALPHABETICAL LISTING

OCT	HEX	MNEMONIC	OCT	HEX	MNEMONIC	OCT	HEX	MNEMONIC	OCT	HEX	MNEMONIC	OCT	HEX	MNEMONIC
316	CE	ACI D8	71	39	DAD SP	174	7C	MOV A,H	167	77	MOV M,A	347	E7	RST 4
217	8F	ADC A	75	3D	DCR A	175	7D	MOV A,L	160	70	MOV M,B	357	EF	RST 5
210	88	ADC B	05	05	DCR B	176	7E	MOV A,M	161	71	MOV M,C	367	F7	RST 6
211	89	ADC C	15	0D	DCR C	107	47	MOV B,A	162	72	MOV M,D	377	FF	RST 7
212	8A	ADC D	25	15	DCR D	100	40	MOV B,B	163	73	MOV M,E	310	C8	RZ
213	8B	ADC E	35	1D	DCR E	101	41	MOV B,C	164	74	MOV M,H	327	9F	SBB A
214	8C	ADC H	45	25	DCR H	102	42	MOV B,D	165	75	MOV M,L	230	98	SBB B
215	8D	ADC L	55	2D	DCR L	103	43	MOV B,E	76	3E	MVI A,D8	231	99	SBB C
216	8E	ADC M	65	35	DCR M	104	44	MOV B,H	06	06	MVI B,D8	232	9A	SBB D
207	87	ADD A	13	0B	DCX B	105	45	MOV B,L	16	0E	MVI C,D8	233	9B	SBB E
200	80	ADD B	33	1B	DCX D	106	46	MOV B,M	26	16	MVI D,D8	234	9C	SBB H
201	81	ADD C	53	2B	DCX H	117	4F	MOV C,A	36	1E	MVI E,D8	235	9D	SBB L
202	82	ADD D	73	3B	DCX SP	110	48	MOV C,B	46	26	MVI H,D8	236	9E	SBB M
203	83	ADD E	363	F3	DI	111	49	MOV C,C	56	2E	MVI L,D8	336	DE	SBI D8
204	84	ADD H	373	FB	EI	112	4A	MOV C,D	66	36	MVI M,D8	42	22	SHLD Adr
205	85	ADD L	166	76	HLT	113	4B	MOV C,E	00	00	NOP	371	F9	SPHL
206	86	ADD M	333	DB	IN D8	114	4C	MOV C,H	267	B7	ORA A	62	32	STA Adr
306	C6	ADI D8	74	3C	INR A	115	4D	MOV C,L	260	B0	ORA B	02	02	STAX B
247	A7	ANA A	04	04	INR B	116	4E	MOV C,M	261	B1	ORA C	22	12	STAX D
240	A0	ANA B	14	0C	INR C	127	57	MOV D,A	262	B2	ORA D	67	37	STC
241	A1	ANA C	24	14	INR D	120	50	MOV D,B	263	B3	ORA E	227	97	SUB A
242	A2	ANA D	34	1C	INR E	121	51	MOV D,C	264	B4	ORA H	220	90	SUB B
243	A3	ANA E	44	24	INR H	122	52	MOV D,D	265	B5	ORA L	221	91	SUB C
244	A4	ANA H	54	2C	INR L	123	53	MOV D,E	266	B6	ORA M	222	92	SUB D
245	A5	ANA L	64	34	INR M	124	54	MOV D,H	366	F6	ORI D8	223	93	SUB E
246	A6	ANA M	03	03	INX B	125	55	MOV D,L	323	D3	OUT D8	224	94	SUB H
346	E6	ANI D8	23	13	INX D	126	56	MOV D,M	351	E9	PCHL	225	95	SUB L
315	CD	CALL Adr	43	23	INX H	137	5F	MOV E,A	301	C1	POP B	226	96	SUB M
334	DC	CC Adr	63	33	INX SP	130	58	MOV E,B	321	D1	POP D	326	D6	SUI D8
374	FC	CM Adr	332	DA	JC Adr	131	59	MOV E,C	341	E1	POP H	353	EB	XCHG
57	2F	CMA	372	FA	JM Adr	132	5A	MOV E,D	361	F1	POP PSW	257	AF	XRA A
77	3F	CMC	303	C3	JMP Adr	133	5B	MOV E,E	305	C5	PUSH B	250	A8	XRA B
277	BF	CMP A	322	D2	JNC Adr	134	5C	MOV E,H	325	D5	PUSH D	251	A9	XRA C
270	B8	CMP B	302	C2	JNZ Adr	135	5D	MOV E,L	345	E5	PUSH H	252	AA	XRA D
271	B9	CMP C	362	F2	JP Adr	136	5E	MOV E,M	365	F5	PUSH PSW	253	AB	XRA E
272	BA	CMP D	352	EA	JPE Adr	147	67	MOV H,A	27	17	RAL	254	AC	XRA H
273	BB	CMP E	342	E2	JPO Adr	140	60	MOV H,B	37	1F	RAR	255	AD	XRA L
274	BC	CMP H	312	CA	JZ	141	61	MOV H,C	330	D8	RC	256	AE	XRA M
275	BD	CMP L	72	3A	LDA Adr	142	62	MOV H,D	311	C9	RET	356	EE	XRI D8
276	BE	CMP M	12	0A	LDAX B	143	63	MOV H,E	07	07	RLC	343	E3	XTHL
324	D4	CNC Adr	32	1A	LDAX D	144	64	MOV H,H	370	F8	RM	10	08	---
304	C4	CNZ Adr	52	2A	LHLD Adr	145	65	MOV H,L	320	D0	RNC	20	10	---
364	F4	CP Adr	01	01	LXI B,D16	146	66	MOV H,M	300	C0	RNZ	30	18	---
354	EC	CPE Adr	21	11	LXI D,D16	157	6F	MOV L,A	360	F0	RP	40	20	---
376	FE	CPI D8	41	21	LXI H,D16	150	68	MOV L,B	350	E8	RPE	50	28	---
344	E4	CPO Adr	61	31	LXI SP,D16	151	69	MOV L,C	340	E0	RPO	60	30	---
314	CC	CZ Adr	177	7F	MOV A,A	152	6A	MOV L,D	17	0F	RRC	70	38	---
47	27	DAA	170	78	MOV A,B	153	6B	MOV L,E	307	C7	RST 0	313	CB	---
11	09	DAD B	171	79	MOV A,C	154	6C	MOV L,H	317	CF	RST 1	331	D9	---
31	19	DAD D	172	7A	MOV A,D	155	6D	MOV L,L	327	D7	RST 2	335	DD	---
51	29	DAD H	173	7B	MOV A,E	156	6E	MOV L,M	337	DF	RST 3	355	ED	---
												375	FD	---

D8 = constant, or expression that evaluates to an 8 bit data quantity.

D16 = constant, or expression that evaluates to a 16 bit data quantity.

Adr = 16 bit address.

APPENDIX

III. 8080 ASSEMBLY LANGUAGE REFERENCE CARD

NUMERICAL LISTING

OCT	HEX	MNEMONIC	OCT	HEX	MNEMONIC	OCT	HEX	MNEMONIC	OCT	HEX	MNEMONIC	OCT	HEX	MNEMONIC
00	00	NOP	63	33	INX SP	146	66	MOV H,M	231	99	SBB C	314	CC	CZ Adr
01	01	LXI B,D16	64	34	INR M	147	67	MOV H,A	232	9A	SBB D	315	CD	CALL Adr
02	02	STAX B	65	35	DCR M	150	68	MOV L,B	233	9B	SBB E	316	CE	ACI D8
03	03	INX B	66	36	MVI M,D8	151	69	MOV L,C	234	9C	SBB H	317	CF	RST 1
04	04	INR B	67	37	STC	152	6A	MOV L,D	235	9D	SBB L	320	D0	RNC
05	05	DCR B	70	38	---	153	6B	MOV L,E	236	9E	SBB M	321	D1	POP D
06	06	MVI B,D8	71	39	DAD SP	154	6C	MOV L,H	237	9F	SBB A	322	D2	JNC Adr
07	07	RLC	72	3A	LDA Adr	155	6D	MOV L,L	240	A0	ANA B	323	D3	OUT D8
10	08	---	73	3B	DCX SP	156	6E	MOV L,M	241	A1	ANA C	324	D4	CNC Adr
11	09	DAD B	74	3C	INR A	157	6F	MOV L,A	242	A2	ANA D	325	D5	PUSH D
12	0A	LDAX B	75	3D	DCR A	160	70	MOV M,B	243	A3	ANA E	326	D6	SUI D8
13	0B	DCX B	76	3E	MVI A,D8	161	71	MOV M,C	244	A4	ANA H	327	D7	RST 2
14	0C	INR C	77	3F	CMC	162	72	MOV M,D	245	A5	ANA L	330	D8	RC
15	0D	DCR C	100	40	MOV B,B	163	73	MOV M,E	246	A6	ANA M	331	D9	---
16	0E	MVI C,D8	101	41	MOV B,C	164	74	MOV M,H	247	A7	ANA A	332	DA	JC Adr
17	0F	RRC	102	42	MOV B,D	165	75	MOV M,L	250	A8	XRA B	333	DB	IN D8
20	10	---	103	43	MOV B,E	166	76	HLT	251	A9	XRA C	334	DC	CC Adr
21	11	LXI D,D16	104	44	MOV B,H	167	77	MOV M,A	252	AA	XRA D	35	DD	---
22	12	STAX D	105	45	MOV B,L	170	78	MOV A,B	253	AB	XRA E	336	DE	SBI D8
23	13	INX D	106	46	MOV B,M	171	79	MOV A,C	254	AC	XRA H	337	DF	RST 3
24	14	INR D	107	47	MOV B,A	172	7A	MOV A,D	255	AD	XRA L	340	E0	RPO
25	15	DCR D	110	48	MOV C,B	173	7B	MOV A,E	256	AE	XRA M	341	E1	POP H
26	16	MVI D,D8	111	49	MOV C,C	174	7C	MOV A,H	257	AF	XRA A	342	E2	JPO Adr
27	17	RAL	112	4A	MOV C,D	175	7D	MOV A,L	260	B0	ORA B	343	E3	XTHL
30	18	---	113	4B	MOV C,E	176	7E	MOV A,M	261	B1	ORA C	344	E4	CPO Adr
31	19	DAD D	114	4C	MOV C,H	177	7F	MOV A,A	262	B2	ORA D	345	E5	PUSH H
32	1A	LDAX D	115	4D	MOV C,L	200	80	ADD B	263	B3	ORA E	346	E6	ANI D8
33	1B	DCX D	116	4E	MOV C,M	201	81	ADD C	264	B4	ORA H	347	E7	RST 4
34	1C	INR E	117	4F	MOV C,A	202	82	ADD D	265	B5	ORA L	350	E8	RPE
35	1D	DCR E	120	50	MOV D,B	203	83	ADD E	266	B6	ORA M	351	E9	PCHL
36	1E	MVI E,D8	121	51	MOV D,C	204	84	ADD H	267	B7	ORA A	352	EA	JPE Adr
37	1F	RAR	122	52	MOV D,D	205	85	ADD L	270	B8	CMP B	353	EB	XCHG
40	20	---	123	53	MOV D,E	206	86	ADD M	271	B9	CMP C	354	EC	CPE Adr
41	21	LXI H,D16	124	54	MOV D,H	207	87	ADD A	272	BA	CMP D	355	ED	---
42	22	SHLD Adr	125	55	MOV D,L	210	88	ADC B	273	BB	CMP E	356	EE	XRI D8
43	23	INX H	126	56	MOV D,M	211	89	ADC C	274	BC	CMP H	357	EF	RST 5
44	24	INR H	127	57	MOV D,A	212	8A	ADC D	275	BD	CMP L	360	F0	RP
45	25	DCR H	130	58	MOV E,B	213	8B	ADC E	276	BE	CMP M	361	F1	POP PSW
46	26	MVI H,D8	131	59	MOV E,C	214	8C	ADC H	277	BF	CMP A	362	F2	JP Adr
47	27	DAA	132	5A	MOV E,D	215	8D	ADC L	300	C0	RNZ	363	F3	DI
50	28	---	133	5B	MOV E,E	216	8E	ADC M	301	C1	POP B	364	F4	CP Adr
51	29	DAD H	134	5C	MOV E,H	217	8F	ADC A	302	C2	JNZ Adr	365	F5	PUSH PSW
52	2A	LHLD Adr	135	5D	MOV E,L	220	90	SUB B	303	C3	JMP Adr	366	F6	ORI D8
53	2B	DCX H	136	5E	MOV E,M	221	91	SUB C	304	C4	CNZ Adr	367	F7	RST 6
54	2C	INR L	137	5F	MOV E,A	222	92	SUB D	305	C5	PUSH B	370	F8	RM
55	2D	DCR L	140	60	MOV H,B	223	93	SUB E	306	C6	ADI D8	371	F9	SPHL
56	2E	MVI L,D8	141	61	MOV H,C	224	94	SUB H	307	C7	RST 0	372	FA	JM Adr
57	2F	CMA	142	62	MOV H,D	225	95	SUB L	310	C8	RZ	373	FB	EI
60	30	---	143	63	MOV H,E	226	96	SUB M	311	C9	RET	374	FC	CM Adr
61	31	LXI SP,D16	144	64	MOV H,H	227	97	SUB A	312	CA	JZ	375	FD	---
62	32	STA Adr	145	65	MOV H,L	230	98	SBB B	313	CB	---	376	FE	CPI D8
												377	FF	RST 7

D8 = constant, or expression that evaluates to an 8 bit data quantity.

D16 = constant, or expression that evaluates to a 16 bit data quantity.

Adr = 16 bit address.

APPENDIX

IV. Program Listing, MST-80B Microcomputer Monitor Program

1
8080 MACRO ASSEMBLER, VER 2.4

ERRORS = 0 PAGE 1

;*****HEX/OCT MONITOR*****
;*****FOR MST-80 MICROPROCESSOR TRAINER*****

;WRITTEN BY GORDON JONES
;DATE: 8-23-76

```

;
;
;
003667      KYTEM      EQU      07B7H
003671      LVALU      EQU      07B9H
003672      HVALU      EQU      07BAH
003673      PCSTO      EQU      07BBH
003675      PSWST      EQU      07BDH
003677      BSTOR      EQU      07BFH
003701      DSTOR      EQU      07C1H
003703      HSTOR      EQU      07C3H
003705      OFLAG      EQU      HSTOR+2
;
;
;
002407      KEYBD      EQU      0507H
002401      KYBD1      EQU      0501H
;
;
000360      TOP        EQU      0F0H
000017      BOT        EQU      0FH
000002      RREAD      EQU      2H
003673      BKSTO      EQU      07BBH
000006      DISO       EQU      6
000007      DISH       EQU      7

```

;*****INITIALIZE ROUTINE*****

```

000000      ORG 0
000000      061 360 007      INIT:      LXI      SP,07F0H      ;INITIALIZE STACK POINTER
000003      257              XRA      A              ;CLEAR AC
000004      062 267 007      STA      KYTEM          ;INITIALIZE DISPLAY STORAGE
000007      062 305 007      STA      OFLAG          ;CLEAR OCTAL FLAG - SET TO HEX DISPLAY
000012      323 007          OUT      DISH           ;SET HDWR FLAG TO HEX
000014      315 117 001      CALL     DIS           ;PUT 000 IN DISPLAY
000017      315 131 000      ST:      CALL     KEY          ;GO TO KEY ROUTINE
000022      303 017 000      JMP      ST             ;GET BACK TO KEY ROUTINE IF NOT A CALL

000025      316              TABLC:      DB      ENTER      ;CONTROL ROUTINES ADDRESS TABLE
000026      270              DB      DISP
000027      234              DB      RUN
000030      131              DB      KEY
000031      131              DB      KEY
000032      331              DB      HO
000033      223              DB      LDH

```

```

000034 215          DB      EXA
000070          ORG      700

```

*****THIS IS THE BREAK ROUTINE*****

```

000070 042 303 007  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 042 273 007  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 042 275 007  SHLD  PSWST     ;PUT AC & PSW IN MEMORY
000105 305          PUSH   B          ;GET B&C
000106 341          POP    H          ;PUT B&C IN MEMORY
000107 042 277 007  SHLD  BSTOR     ;PUT B&C IN MEMORY
000112 353          XCHG          ;PUT D&E IN H&L
000113 042 301 007  SHLD  DSTOR     ;PUT D&E IN MEMORY
000116 041 273 007  LXI    H,BKSTO   ;LOAD BREAK MEMORY LOCATION
000121 042 271 007  SHLD  LVALU     ;PUT IT IN PROPER LOCATION
000124 076 273      MVI    A,0BBH    ;PUT BB IN AC
000126 303 305 000  JMP     BACK     ;DISPLAY BB AND RETURN TO KEY

```

*****KEYBOARD READ ROUTINE*****

```

000131 315 111 001  KEY:    CALL  READ      ;GO READ KEYBOARD
000134 302 131 000  JNZ    KEY          ;LOOP IF KEY DOWN
000137 315 161 001  CALL    DELAY        ;DEBOUNCE

000142 315 117 001  REP:    CALL  DIS      ;CHECK FOR CHANGE IN DISP MODE
000145 315 111 001  CALL    READ      ;GO READ KEYBOARD
000150 312 142 000  JZ      REP          ;LOOP IF NO KEY DOWN
000153 315 161 001  CALL    DELAY        ;DEBOUNCE
000156 041 001 005  COL:    LXI    H,KYBDI  ;SET UP COLUMN POINTER
000161 176          LDY:    MOV    A,M      ;READ KEYBOARD COLUMN
000162 057          CMA          ;COMPLEMENT
000163 267          ORA    A          ;SET FLAGS
000164 302 356 000  JNZ    LUT        ;GOTO LOOK UP TABLE IF KEY FOUND
000167 175          MOV    A,L        ;NO KEY FOUND - BUMP COLUMN POINTER
000170 027          RAL          ;ROTATE TO NEXT COLUMN
000171 157          MOV    L,A        ;PUT BACK
000172 346 010      ANI    0BH        ;CHECK FOR LAST COLUMN
000174 312 161 000  JZ      LDY        ;NOT LAST COLUMN - GO READ A KEY
000177 303 131 000  JMP     KEY        ;NO KEY DOWN GO BACK

```

*****THESE ARE THE CONTROL KEY ROUTINES

```

000202 041 024 000  CNTL:  LXI  H,TABLC-1  ;GET TABLE POINTER
000205 170          MOV  A,B              ;GET KEY VALUE
000206 027          LPI:  RAL              ;ROTATE INTO CARRY
000207 043          INX  H                ;BUMP TABLE POINTER
000210 322 206 000  JNC  LPI              ;
000213 156          MOV  L,M              ;MOVE ADDRESS INTO L REG
000214 351          PCHL              ;JUMP TO PROPER CONTROL ROUTINE
000215 072 271 007  EXA:  LDA  LVALU      ;GET L REGISTER VALUE
000220 303 305 000  JMP  BACK            ;DISPLAY IT & JUMP TO KEY
000223 072 267 007  LDH:  LDA  KYTEM      ;GET KEY VALUE FROM TEMP
000226 062 272 007  STA  HVALU      ;PUT IN H REGISTER STORAGE
000231 303 131 000  JMP  KEY            ;DONE- GO TO START

000234 072 267 007  RUN:  LDA  KYTEM      ;GET CURRENT DISPLAY VALUE
000237 062 271 007  STA  LVALU      ;STORE IN L REG LOCATION
000242 052 277 007  LHLD  B&C        ;GET CONTENTS OF B&C REGS
000245 345          PUSH H              ;PUT ON STACK
000246 301          POP  B              ;PUT IN B&C REGS
000247 052 301 007  LHLD  D&E        ;GET CONTENTS OF D&E REGS
000252 353          XCHG              ;EXCHANGE H&L WITH D&E
000253 052 275 007  LHLD  PSW        ;GET OLD AC AND PSW
000256 345          PUSH H              ;PUT AC & PSW ON STACK
000257 361          POP  PSW           ;RESTORE AC & STATUS
000260 052 271 007  LHLD  LVALU      ;GET STARTING ADDRESS
000263 345          PUSH H              ;PUT STARTING ADDR ON STACK
000264 052 303 007  LHLD  H&L        ;RESTORE H&L
000267 311          RET                ;GET STARTING ADDR FROM STACK AND RUN

000270 072 267 007  DISP:  LDA  KYTEM      ;GET CURRENT DISPLAY VALUE
000273 062 271 007  STA  LVALU      ;STORE IN LREG STORAGE
000276 052 271 007  LHLD  LVALU      ;GET VALUE JUST KEYED IN
000301 042 271 007  NEXT:  SHLD  LVALU     ;STORE IN MEMORY POINTER
000304 176          MOV  A,M            ;GET VALUE POINTED TO BY MEM POINTER
000305 062 267 007  BACK:  STA  KYTEM      ;PUT THIS VALUE IN KEY STORAGE
000310 315 117 001  CALL  DIS          ;DISPLAY IT
000313 303 131 000  JMP  KEY            ;GO BACK AND START OVER
000316 052 271 007  ENTER:  LHLD  LVALU     ;GET MEMORY POINTER
000321 072 267 007  LDA  KYTEM      ;GET DISPLAY VALUE
000324 167          MOV  M,A            ;PUT VALUE IN LOC POINTED TO BY H&L
000325 043          INX  H              ;BUMP TO NEXT LOCATION
000326 303 301 000  JMP  NEXT          ;PUT INC PTR AWAY AND DISPLAY NEXT LOC

000331 072 305 007  HO:  LDA  OFLAG      ;FETCH HEX/OCTAL FLAG
000334 057          CMA                ;CHANGE TO OTHER BASE
000335 062 305 007  STA  OFLAG      ;PUT IT BACK
000340 267          ORA  A              ;SET-UP FOR TESTING IT
000341 312 351 000  JZ  HOHO          ;JMP IF 0 FOR HEX
000344 323 006  OUT  DISO            ;MUST BE 1'S FOR OCTAL - SET DISPLAY

```

```

000346 303 131 000  JMP  KEY            ;
000351 323 007  HOHO:  OUT  DISH          ;SET DISPLAY FOR HEX 3 DIGITS
000353 303 131 000  JMP  KEY            ;

```

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

000356	107	LUT:	MOV	B,A	;SAVE AC
000357	175		MOV	A,L	;GET COLUMN POINTER
000360	017		RRC		;ROTATE COL POINTER RIGHT
000361	332 373 000		JC	COL1	;IS IT COL1?
000364	017		RRC		;ROTATE AGAIN
000365	332 004 001		JC	COL2	;IS IT COL2?
000370	303 202 000		JMP	CNTL	;MUST BE CONTROL COLUMN
000373	041 171 001	COL1:	LXI	H,TABLE-1	;GET TABLE POINTER
000376	315 063 001		CALL	DECOD	;GO GET VALUE FROM TABLE
000401	303 016 001		JMP	SHIFT	;STORE AND SEND TO DISPLAY
000404	041 171 001	COL2:	LXI	H,TABLE-1	;GET TABLE POINTER
000407	315 063 001		CALL	DECOD	;GET VALUE FROM TABLE
000412	171		MOV	A,C	;PUT TABLE VALUE IN AC
000413	306 002		ADI	2H	;CORRECT VALUE FOR COLUMN 2
000415	117		MOV	C,A	
000416	041 267 007	SHIFT:	LXI	H,KYTEM	;GET OLD DISPLAY VALUE
000421	000		NOP		
000422	000		NOP		
000423	072 305 007		LDA	OFLAG	;CHECK HEX/OCT FLAG
000426	267		ORA	A	;SET FLAGS
000427	302 047 001		JNZ	OCT1	;GOTO OCTAL IF FLAG IS A 1
000432	176		MOV	A,M	;GET KEY CODE
000433	007	HEX1:	RLC		;ROTATE ONE HEX DIGIT LEFT
000434	007		RLC		
000435	007		RLC		
000436	007		RLC		
000437	346 360		ANI	0F0H	;MASK OFF BOTTOM DIGIT
000441	261		ORA	C	;OR NEW DIGIT TO OLD NUMBER
000442	167		MOV	M,A	;PUT BACK IN DISPLAY STORAGE
000443	315 117 001		CALL	DIS	;SEND TO DISPLAY
000446	311		RET		;END OF NUMBER KEY ROUTINE
000447	176	OCT1:	MOV	A,M	;GET KEY CODE
000450	007		RLC		;ROTATE ONE OCTAL DIGIT LEFT
000451	007		RLC		
000452	007		RLC		
000453	346 370		ANI	3700	;MASK OFF BOTTOM DIGIT
000455	261		ORA	C	;OR NEW DIGIT TO OLD NUMBER


```

000456 167          MOV    M,A          ;PUT BACK IN DISPLAY STORAGE
000457 315 117 001  CALL   DIS          ;SEND TO DISPLAY
000462 311          RET

000463 170          DECOD: MOV    A,B          ;GET KEY VALUE
000464 027          AGAIN: RAL          ;ROTATE INTO CARRY
000465 043          INX    H          ;BUMP TABLE POINTER
000466 322 064 001  JNC    AGAIN        ;
000471 116          MOV    C,M          ;SAVE KEY CODE

000472 072 305 007  LDA    OFLAG        ;CHECK HEX/OCT FLAG
000475 267          ORA    A          ;SET FLAGS
000476 302 102 001  JNZ    OCT2        ;IF IN OCTAL MODE JUMP TO CHAR CHECK
000501 311          RET

000502 171          OCT2: MOV    A,C          ;GET KEY VALUE
000503 346 370      ANI    370Q        ;MASK OFF LOWER DIGIT
000505 310          RZ              ;RETURN IF LEGAL OCTAL NUMBER
000506 303 131 000  JMP    KEY          ;ILLEGAL CHAR GOTO KEY

```

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

```

000511 072 007 005  READ:  LDA    KEYBD        ;READ KEYBOARD
000514 057          CMA          ;COMPLEMENT
000515 267          ORA    A          ;SET FLAGS
000516 311          RET

```

;++++++ROUTINE TO DISPLAY HEX OR OCTAL++++++

```

000517 072 267 007  DIS:   LDA    KYTEM        ;GET CURRENT DISPLAY VALUE
000522 117          DISPLAY: MOV    C,A          ;SAVE A REG
000523 072 305 007  LDA    OFLAG        ;CHECK HEX/OCT FLAG
000526 267          ORA    A          ;SET FLAGS
000527 302 136 001  JNZ    OCT          ;SIGN BIT=1 FOR OCT DISPLAY
000532 171          HEX:   MOV    A,C          ;HEX - GET AC
000533 323 000      OUT    0          ;SEND TO DISPLAY
000535 311          RET

000536 171          OCT:   MOV    A,C          ;GET NUMBER TO DISPLAY
000537 007          RLC          ;GET HIGH ORDER DIGIT
000540 007          RLC          ;ROTATE INTO POSITION
000541 346 003      ANI    3Q          ;SAVE HIGH ORDER DIGIT
000543 323 005      OUT    5          ;DISPLAY HIGH ORDER DIGIT
000545 171          MOV    A,C          ;GET NUMBER AGAIN
000546 027          RAL          ;MOVE 2ND DIGIT INTO POSTION
000547 346 160      ANI    160Q        ;SAVE MIDDLE DIGIT
000551 107          MOV    B,A          ;SAVE MIDDLE DIGIT
000552 171          MOV    A,C          ;GET NUMBER AGAIN
000553 346 007      ANI    7Q          ;GET 1ST DIGIT
000555 260          ORA    B          ;COMBINE DIGITS 1 & 2

```

1

8080 MACRO ASSEMBLER, VER 2.4

ERRORS = 0 PAGE 6

000556 323 000
000560 311

OUT 0 ;DISPLAY THEM
RET

;+++++THIS IS A DELAY ROUTINE TO DEBOUNCE THE SWITCHES+++++

000561 006 000
000563 004
000564 343
000565 343
000566 302 163 001
000571 311

DELAY: MVI B,0 ;INITIALIZE COUNTER
LOOP: INR B ;BUMP COUNTER
XTHL ;EXTRA DELAY IN LOOP
XTHL
JNZ LOOP ; LOOP UNTIL ZERO
RET

000572 000
000573 004
000574 010
000575 014
000576 001
000577 005
000600 011
000601 015

TABLE: DB 00H ;NUMBER KEY CODE TABLE
DB 04H
DB 08H
DB 0CH
DB 01H
DB 05H
DB 09H
DB 0DH
END

NO PROGRAM ERRORS

SYMBOL TABLE

• 01

A	000007	AGAIN	000464	B	000000	BACK	000305
BKSTO	003673	BOT	000017 •	BRK	000070 •	BSTOR	003677
C	000001	CNTL	000202	COL	000156 •	COLI	000373
COL2	000404	D	000002	DECOD	000463	DELAY	000561
DIS	000517	DISH	000007	DISO	000006	DISP	000270
DISPL	000522 •	DSTOR	003701	E	000003	ENTER	000316
EXA	000215	H	000004	HEX	000532 •	HEXI	000433 •
HO	000331	HOMO	000351	HSTOR	003703	HVALU	003672
INIT	000000 •	KEY	000131	KEYBD	002407	KYBDI	002401
KYTEM	003667	L	000005	LDH	000223	LDKY	000161
LOOP	000563	LPI	000206	LUT	000356	LVALU	003671
M	000006	NEXT	000301	OCT	000536	OCTI	000447
OCT2	000502	OFLAG	003705	PCSTO	003673	PSW	000006
PSWST	003675	READ	000511	REP	000142	RREAD	000002 •
RUN	000234	SHIFT	000416	SP	000006	ST	000017
TABLC	000025	TABLE	000572	TOP	000360 •		

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research & Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately-owned rights.

NOTICE

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 & Development Administration to the exclusion of others that may be suitable.

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 \$3.00

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	5.50	426-450	11.75
126-150	6.00	451-475	12.00
151-175	6.75	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.