MASTER

UCID- 17526 Rev. 2

Lawrence Livermore Laboratory

TECHNOLOGY TRAINING PROGRAM

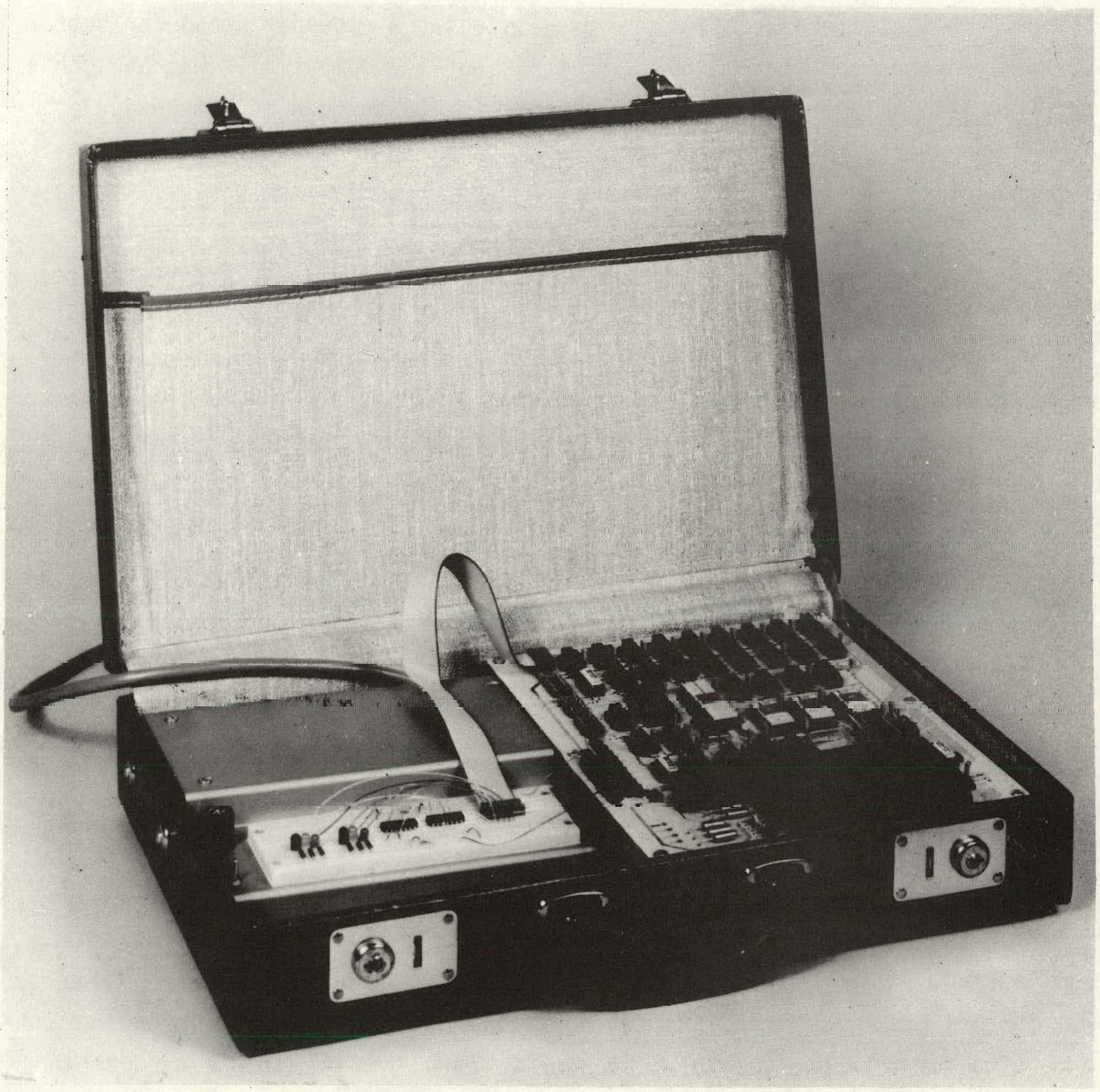
AIMED AT INCREASING THE PRODUCTIVITY OF INDUSTRIAL AMERICA

THE MST-80B MICROCOMPUTER TRAINER

G. D. Jones/E. R. Fisher/and J. M. Spann

April 1, 1980

Prepared for U. S. Department of Energy under contract No. W-7405-Eng-48.



MST-80B Microcomputer Trainer

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. Althrough many technical people have heard of microcomputers, relatively few know exactly what microcomputers are, what they do, and how best to apply them.

The Lawrence Livermore Laboratory (LLL) is a high-technology, energy-related research laboratory that has been a forerunner of microcomputer development and application. Their Technology Training Program (TTP) is patterned after hands-on training originally devised to rapidly educate their own employees about microcomputers. TTP was initiated to expand this hands-on instruction by reaching other energy-related industrial and governmental organizations as well as educational institutions. To expand this sphere of training even further, TTP loans videotaped lectures, provides lecture notes, and, in some cases, lends equipment or aids the instructor. The MST-80B trainer (the "trainer in a briefcase") was designed and fabricated as a part of this technology transfer effort and is now being built by more than 30 colleges for use in their own classes.

The MST-80B is a significant contribution to the effort of the electronics industry to effectively educate potential users of microcomputers. This "trainer in a briefcase" provides the user with hands-on experience in state-of-the-art microcomputer architecture, programming, interfacing and application design; learning these techniques is almost as easy as opening the MST-80B briefcase itself! This trainer, while simultaneously reinforcing and expanding comprehension, allows the user to immediately assemble hardware and apply the concepts developed in the classroom.

TABLE OF CONTENTS

CONTENTS PAGE

THE MST-80B MICROCOMPUTER TRAINER

G. D. Jones/E. R. Fisher/and J. M. Spann

Electronics Engineering Department
Lawrence Livermore Laboratory

ABSTRACT

The microcomputer revolution in electronics is spreading so rapidly it is difficult to educate enough people quickly and thoroughly in the new technology. Lawrence Livermore Laboratory'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 briefcase. The trainer uses the Intel 8080A* 8-Bit Microprocessor (CPU), and it has its own solid-state memory, a built-in keyboard, input and output ports, and a display for visual 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 microcomputer system self-contained in a briefcase for portability and easy usage. The microcomputer was designed as a training device for LLL's Technology Training Program (TTP), allowing students to explore the hardware and software capability of a typical microcomputer.

---

*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 United States Department of Energy to the exclusion of others that may be suitable.

The trainer uses the Intel 8080A Microprocessor and supporting integrated circuits. It 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 MST-80B has a 24-key keyboard and a 3-digit 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. Either number system can be selected by simply depressing a control key.

The trainer includes a breadboard socket so that experiments can be interfaced to the microcomputer through an 8-bit input port and an 8-bit output port. This allows the student to learn hardware interfacing techniques as well as software programming. The MST-80B also has ten uncommitted light-emitting diodes (LEDs) that can easily be connected to display the state of any desired signals (address lines, data lines, and status). These can be used when operating the trainer in the single-step mode or the normal operating mode.

HARDWARE FEATURES OF THE TRAINER

Figures 1a and 1b show 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.

- 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 Ø and PROM 1.

FIGURE 1a. The MST-80B Microcomputer Trainer

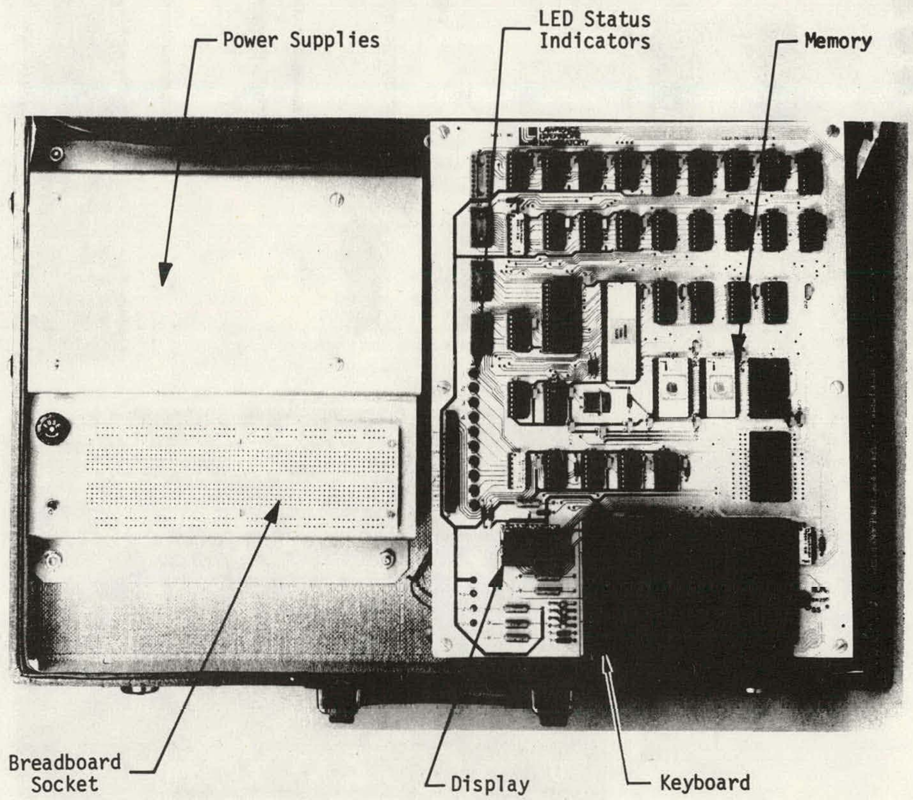


FIGURE 1b. Top view of the MST-80B Microcomputer Trainer showing location of parts.

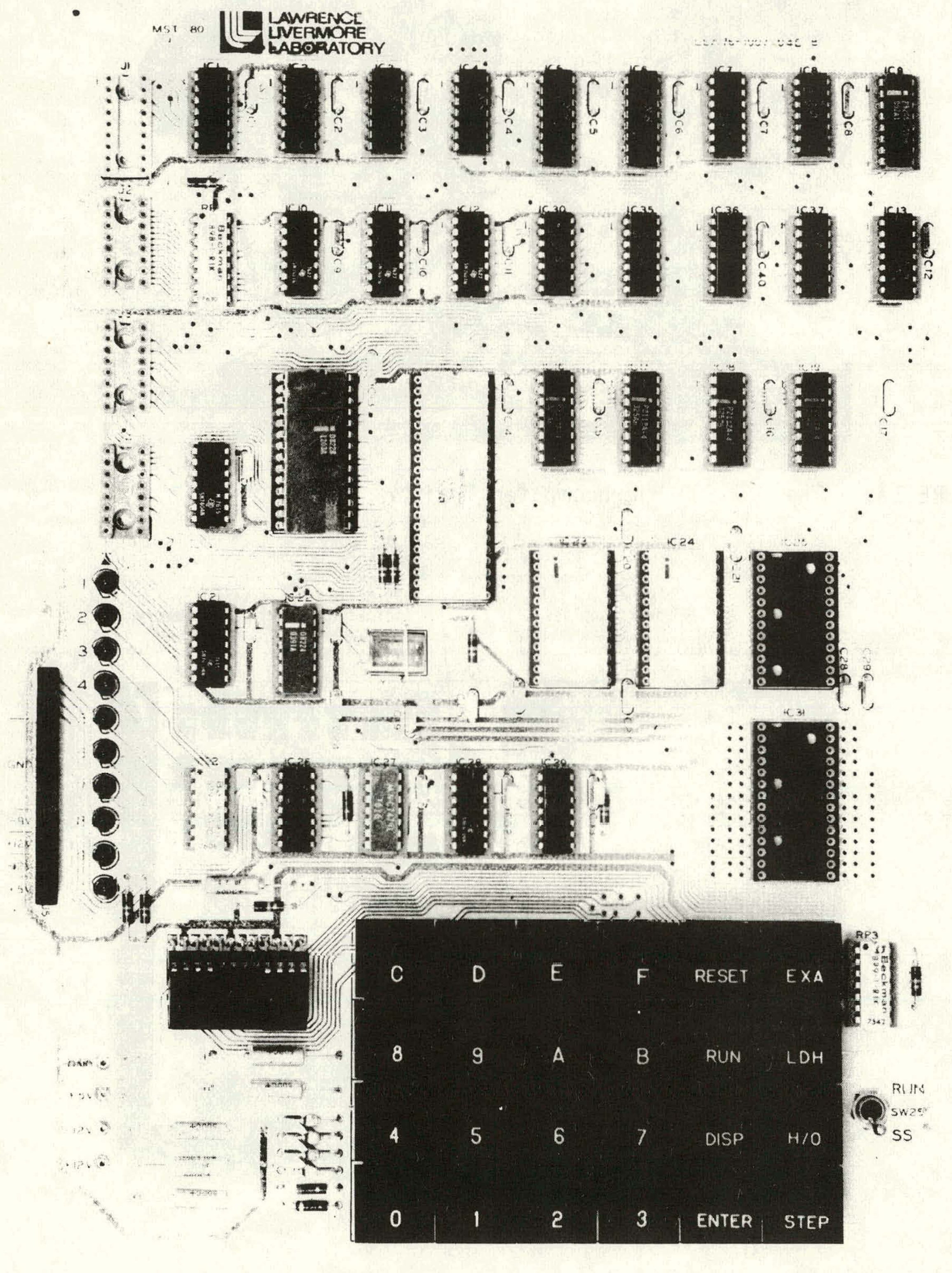


FIGURE 2. A closeup of the circuit board for the MST-80B Microcomputer Trainer. The display is just left of the "C" key.

- A 24-key keyboard. This input device is accessed through memory mapped I/O. (See Figure 7 for a memory map.)

- 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...., page 13, for a description of how this subroutine is used.)

- One 8-bit input port. Address = 1.

- One 8-bit output port (latched). Address = 1.

- Single-step capability.

- 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.

- A self-contained power supply.

- A prototyping area for user experiments.

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

MONITOR PROGRAM

The trainer, as supplied, includes a Monitor Program (Hex/Oct Monitor) loaded in PROMS Ø 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 FF_{16}* 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 DISP (display) 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 a sample program using the Monitor Program, Figure 7 is a memory map for the computer system. The flowchart for the monitor program itself is displayed in Figure 8 and a complete program listing is included as Appendix IV.

OPERATION OF KEYBOARD USING THE HEX/OCT MONITOR

The MST-80B keyboard layout is:

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

*The subscript 16 on a number indicates hexadecimal (base 16) representation.
**Display depends on a user-selected mode: 273 is the octal display, BB the hexadecimal.
***The locations used are tabulated on page 15 of this report.

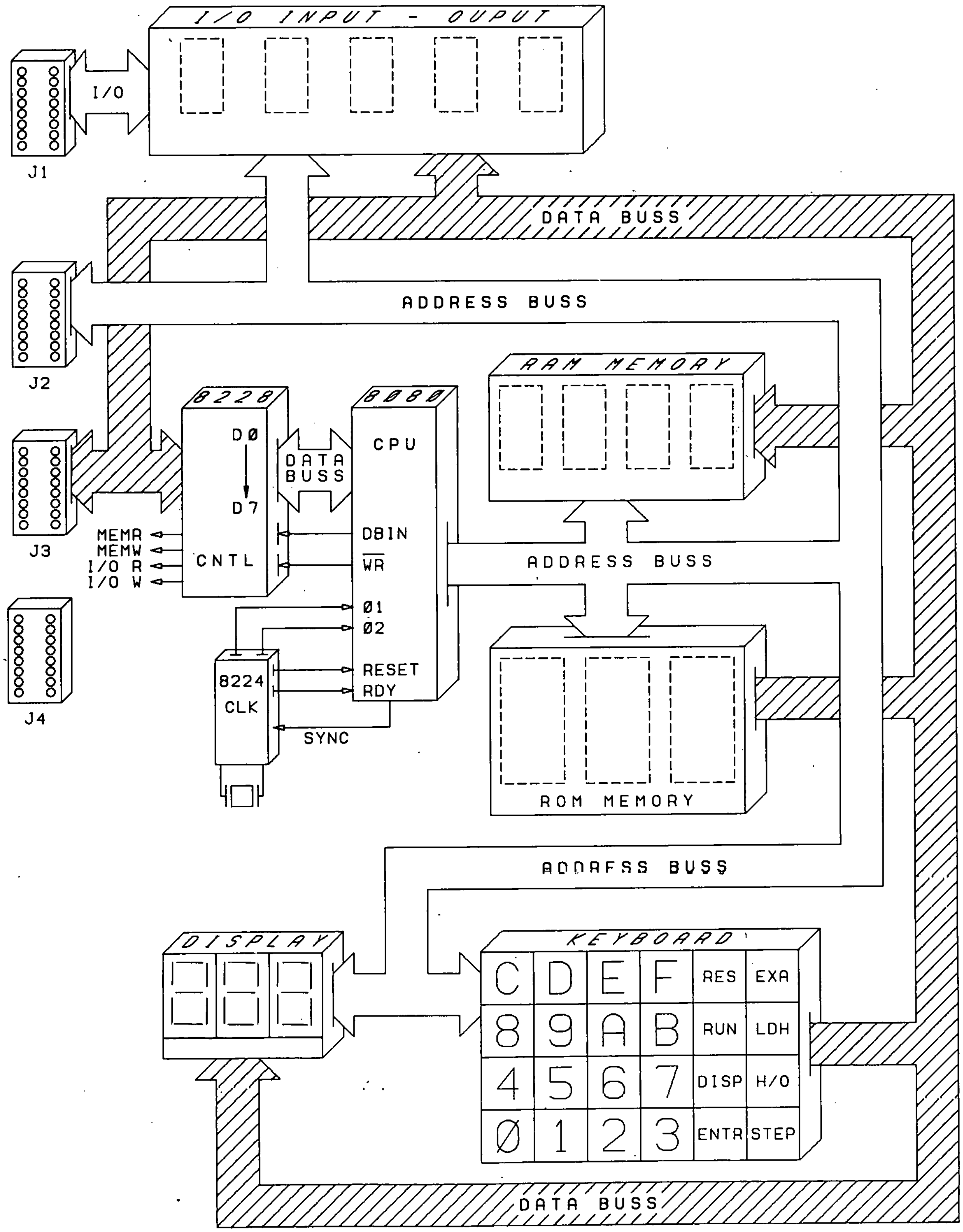


FIGURE 3. Operational block diagram of MST-80B Microcomputer Trainer.

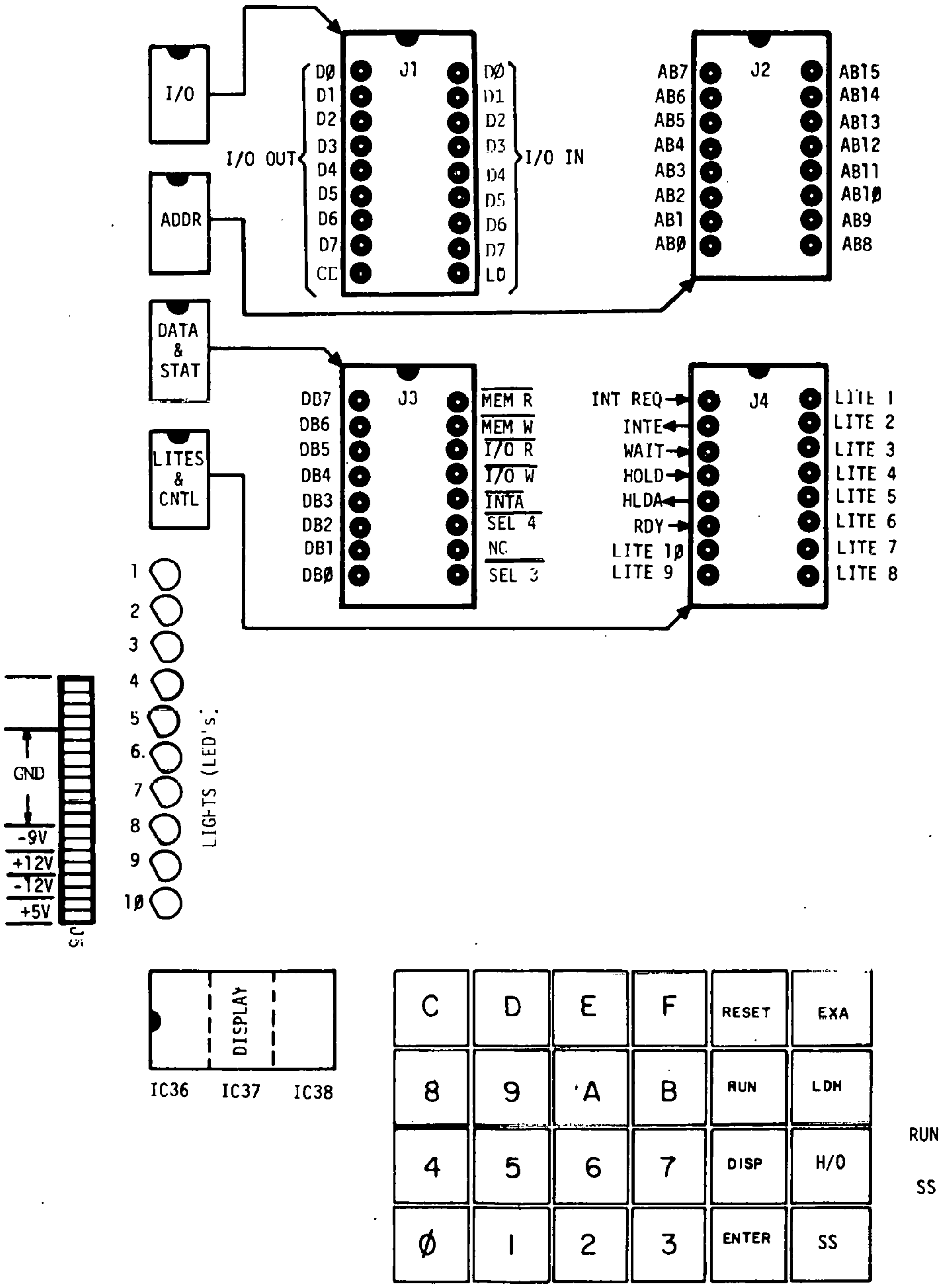


FIGURE 4. Panel connectors used to interface MST-80B Microcomputer Trainer.

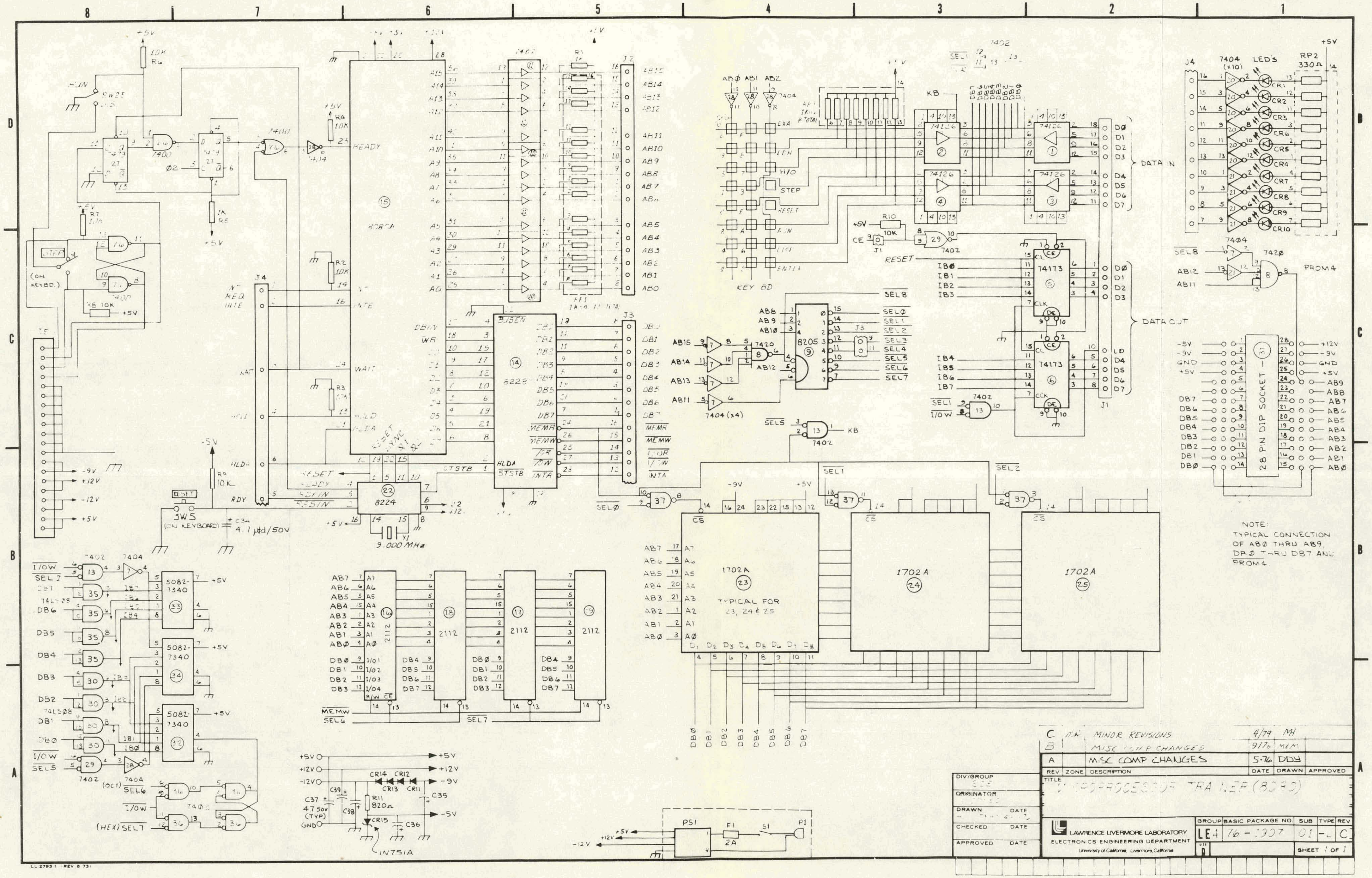


FIGURE 5. Schematic of MST-80B Microcomputer Trainer.

The MST-80B keys function as follows:

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 Ø1 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. 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 depressed. 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. The key 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 depressing 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. For the single step mode, this key advances 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. After first turning on power, when RESET is pushed 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. The keyboard is convenient for this purpose. When KEY is 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 hex digit, and in addition contains the previous key entry in the high order digit. The KEY routine is called by a CALL KEY instruction $(CD\ 59\ ØØ)_{16}$. 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 numerical keys can be used when KEY is called. The control keys are not decoded in the KEY subroutine and should not be used. Also, numerical 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 $(CD\ 52\ Ø1)_{16}$ 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 6 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 HEX 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, Ø 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.

MEMORY LOCATION	MACHINE CODE		OPERATIONS	
ØØ	3E		MVI A, Ø	; CLEAR AC
Ø1	ØØ			
Ø2	57	AGAIN:	MOV D, A	; SAVE A
Ø3	CD		CALL DISPLAY	; SEND AC TO DISPLAY
Ø4	52			
Ø5	Ø1			
Ø6	7A		MOV A, D	; RESTORE A
Ø7	Ø6		MVI B, Ø	; CLR B REGISTER
Ø8	ØØ			
Ø9	ØE		MVI C, 4Ø	; PUT 4Ø IN C REGISTER
ØA	4Ø			
ØB	Ø4	LOOP:	INR B	; INCREMENT B
ØC	CA		JZ LOOP	; DO IT AGAIN
ØD	ØB			
ØE	Ø6			
ØF	ØD		DCR C	; DECREMENT C
1Ø	C2		JNZ LOOP	; LOOP UNTIL ZERO
11	ØB			
12	Ø6			
13	C6		ADI Ø1	; ADD ONE TO AC
14	Ø1			
15	C3		JMP AGAIN	; GO DISPLAY AC & DO AGAIN
16	Ø2			
17	Ø6			

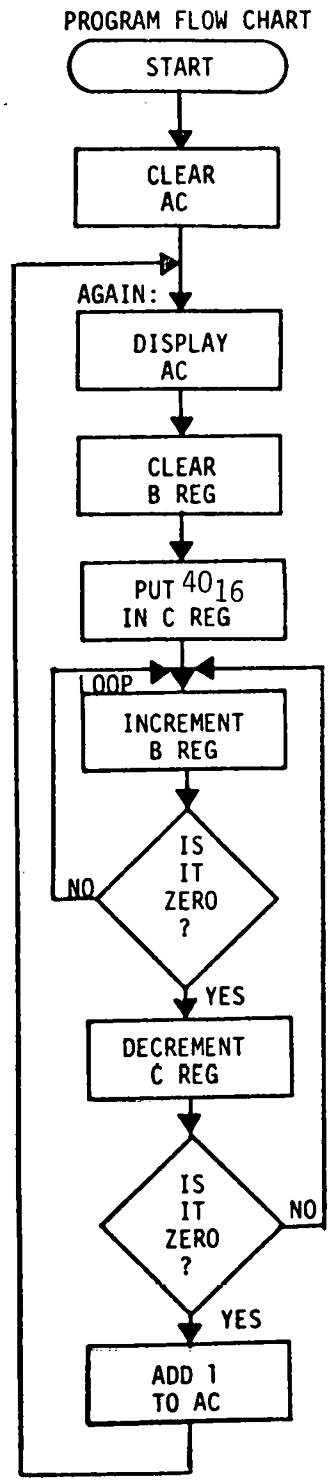


FIGURE 6. Flowchart for MST - 80B sample program.

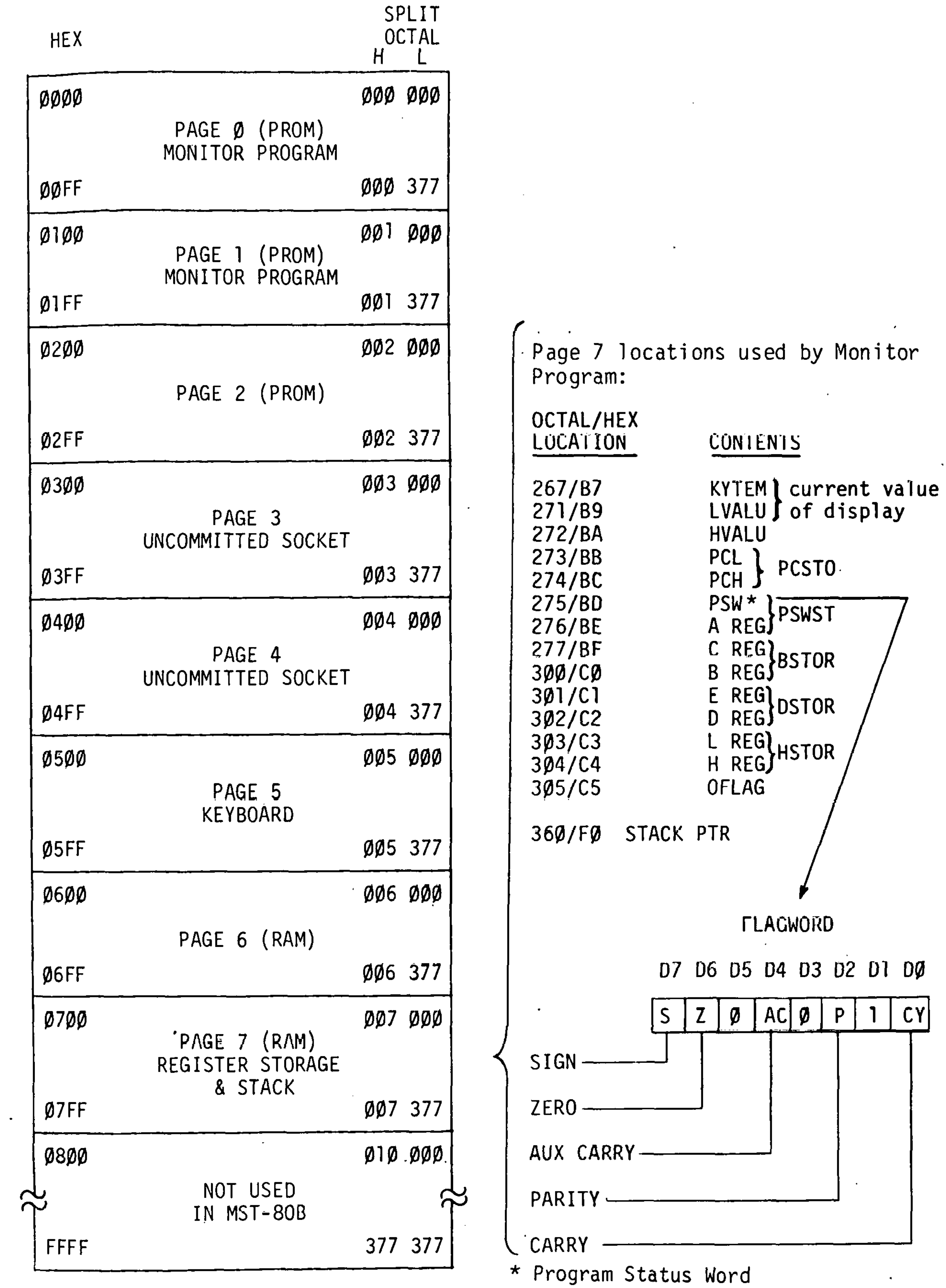


FIGURE 7. Memory map for MST-80B Microcomputer Trainer.

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 ∅ (absolute address = ∅6∅∅ hex). First, key ∅6 into the display and then push the LDH (load H) key. This sets the high-order address (high byte) to page 6. Next, key ∅∅ into the display and push the DISP key. This will display the current contents of location ∅ on page 6. Now you can key in the machine language code for the first instruction, 3E (MVI A), and push the ENTER key. This will enter the 3E into location ∅, and will also display the contents of the next location (location 1). Now you can key in the next code, ∅∅, and push ENTER again. The ∅∅ 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 (∅∅ 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 (∅∅ 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 flow chart of the sample program, Figure 6, 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 FF_{16} and then goes back to Ø. Then the second count loop is entered. This second loop counts the number of times the first loop must go through a full count ($1ØØ_{16}$ counts). Since the C register is initialized to $4Ø_{16}$, the second loop counts $4Ø_{16}$ counts; hence the total counts for both loops is $1ØØ_{16} \times 40_{16}$ ($=16{,}384_{10}$) 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 = FF_{16}) in place of the INR B instruction at memory location ØB. 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 BB will appear in the display to signal you that a break has occurred.

BREAK ROUTINE MEMORY STORAGE LOCATIONS (MEMORY PAGE 7)
(HEX MODE)

ADDRESS	CONTENTS		ADDRESS	CONTENTS
BB	PCL	Breakpoint Address	CØ	B REG
BC	PCH		C1	E REG
BD	PSW		C2	D REG
BE	A REG		C3	L REG
BF	C REG		C4	H REG

The BREAK routine also automatically sets HVALU to page 7. So, since BB is already being displayed, if you now push the DISP key, the contents of memory location BB 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 BC, 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 BF and, for our sample program, should contain $4Ø_{16}$. Location BE (A register) and CØ (B register) should contain zero. If these locations contain the correct values, replace the INR B instruction (code Ø4) in location ØB and put a breakpoint (FF) in location ØF in place of the DCR C instruction. Run the program. When it breaks, examine location CØ again to see what the B register is now. It should be a zero when the count loop is existed. 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 zero count rather than non-zero count, so you need to replace the JZ instruction with a JNZ (code C2) instruction. Replace the breakpoint in ØF with DCR C (ØD) 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 the following people are gratefully acknowledged.

- Stanley A. Nielsen - - TTP Program Engineer
- Stephan A. Mick - - Trouble shooting and checkout
- Alan E. Ragsdale - - Conversion of software from octal to hexadecimal
- J. W. Spencer - - Technical writing/editing
- C. W. Jensen - - Technical writing/editing

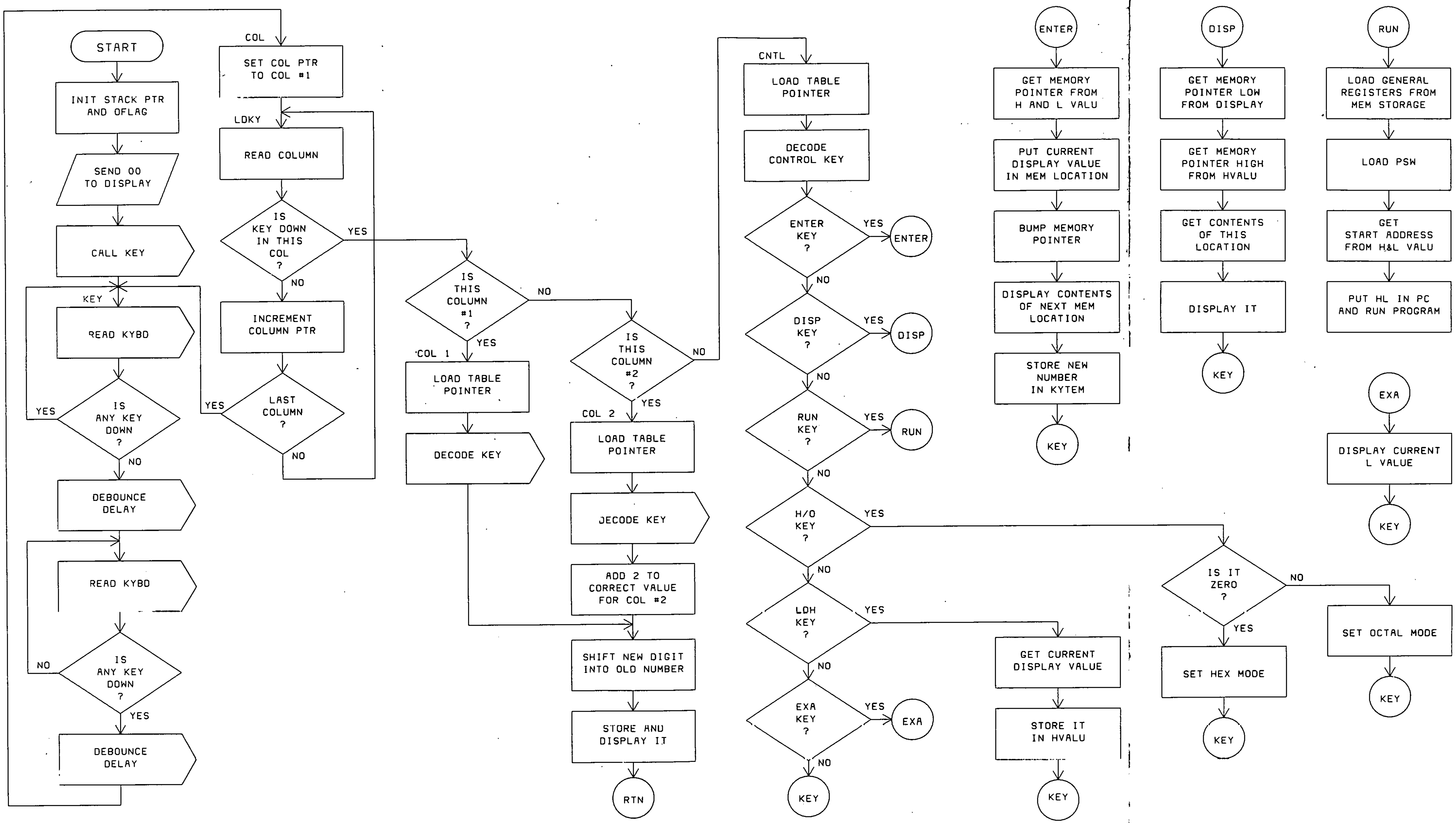


FIGURE 8. Flowchart for HEX/OCT monitor program for MST-80B Microcomputer Trainer.

APPENDIX

I. Summary of 8080 instruction set.

Summary of Processor Instructions
By Alphabetical Order

Mnemonic	Description	Instruction Code[1] D7	D6	D5	D4	D3	D2	D1	D0	Clock[2] Cycles
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
ADD M	Add memory to A	1	0	0	0	0	1	1	0	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	Compliment A	0	0	1	0	1	1	1	1	4
CMC	Compliment 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	D	D	D	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 Interrupt	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	D	D	D	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	D	D	D	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	D	D	D	1	1	0	7
MOV r1,r2	Move register to register	0	1	D	D	D	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	1	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	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 – 100 H – 101 L – 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	335	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 - 8/23/76
                                           ;ADDED JUMP VECTORS FOR INTERRUPTS - 3/14/79
                          ;
                          ;
07B7                      KYTEM     EQU    07B7H
07B9                      LVALU     EQU    07B9H
07BA                      HVALU     EQU    07BAH
07BB                      PCSTO     EQU    07BBH
07BD                      PSWST     EQU    07BDH
07BF                      BSTOR     EQU    07BFH
07C1                      DSTOR     EQU    07C1H
07C3                      HSTOR     EQU    07C3H
07C5                      OFLAG     EQU    HSTOR+2
                          ;
                          ;
                          ;
0507                      KEYBD     EQU    0507H
0501                      KYBD1     EQU    0501H
                          ;
00F0                      TOP       EQU    0F0H
000F                      BOT       EQU    0FH
0002                      RREAD     EQU    2H
07BB                      DIGSTO    EQU    07BBH
0006                      DISO      EQU    6
0007                      DISH      EQU    7
                                ;+++++INITIALIZE ROUTINE+++++

0000                      ORG 0
0000      31 F0 07        INIT:     LXI    SP,07F0H        ;INITIALIZE STACK POINTER
0003      AF                        XRA    A               ;CLEAR AC
0004      32 B7 07                  STA    KYTEM           ;INITIALIZE DISPLAY STORAGE
0007      32 C5 07                  STA    OFLAG           ;CLEAR OCTAL FLAG - SET TO HEX DISPLAY
000A      D3 07                     OUT    DISH            ;SET HDWR FLAG TO HEX
000C      CD 4F 01                  CALL   DIS             ;PUT 000 IN DISPLAY
000F      CD 59 00        ST:       CALL   KEY             ;GO TO KEY ROUTINE
0012      C3 0F 00                  JMP    ST              ;GET BACK TO KEY ROUTINE IF NOT A CALL

0015      CE              TABLC:    DB     ENTER           ;CONTROL ROUTINES ADDRESS TABLE
0016      D3                        DB     DISP
0017      9C                        DB     RUN
0018      59                        DB     KEY
0019      59                        DB     KEY
001A      D9                        DB     HO
001B      93                        DB     LBH
001C      8D                        DB     EXA
```

```
                              ;          +++++THESE ARE THE JUMP VECTORS THAT MAY BE+++++
                              ;               USED WITH INTERRUPTS.

0020                          ORG        20H
0020      C3 80 02                       JMP     0280H

0028                          ORG        28H
0028      C3 80 06                       JMP     0680H

0030                          ORG        30H
0030      C3 80 07                       JMP     0780H

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

0038                          ORG        38H

0038      22 C3 07            BRK:       SHLD    HSTOR            ;STORE H&L IN MEMORY
003B      E1                             POP     H                ;PUT BREAK ADDRESS IN H&L REG
003C      2B                             DCX     H                ;CORRECT BRK ADDR
003D      22 BB 07                       SHLD    PCSTOR           ;STORE BREAK ADDR IN MEMORY
0040      F5                             PUSH    PSW              ;GET AC AND PSW IN STACK
0041      E1                             POP     H                ;PUT AC &PSW IN H&L
0042      22 BD 07                       SHLD    PSWST            ;PUT AC &PSW IN MEMORY
0045      C5                             PUSH    B                ;GET B&C
0046      E1                             POP     H                ;PUT B&C IN MEMORY
0047      22 BF 07                       SHLD    BSTOR            ;PUT B&C IN MEMORY
004A      EB                             XCHG                     ;PUT D&E IN H&L
004B      22 C1 07                       SHLD    DSTOR            ;PUT D&E IN MEMORY
004E      21 BB 07                       LXI     H,BKSTO          ;LOAD BREAK MEMORY LOCATION
0051      22 B9 07                       SHLD    LVALU            ;PUT IT IN PROPER LOCATION
0054      3E BB                          MVI     A,0BBH           ;PUT BB IN AC
0056      C3 C5 00                       JMP     BACK             ;DISPLAY BB AND RETURN TO KEY

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

0059      CD 49 01            KEY:       CALL    READ             ;GO READ KEYBOARD
005C      C2 59 00                       JNZ     KEY              ;LOOP IF KEY DOWN
005F      CD 71 01                       CALL    DELAY            ;DEBOUNCE

0062      CD 4F 01            REP:       CALL    DIS              ;CHECK FOR CHANGE IN DISP MODE
0065      CD 49 01                       CALL    READ             ;GO READ KEYBOARD
0068      CA 62 00                       JZ      REP              ;LOOP IF NO KEY DOWN
006B      CD 71 01                       CALL    DELAY            ;DEBOUNCE
006E      21 01 05            COL:       LXI     H,KYBDI          ;SET UP COLUMN POINTER
0071      7E                  LDKY:      MOV     A,M              ;READ KEYBOARD COLUMN
0072      2F                             CMA                      ;COMPLEMENT
0073      B7                             ORA     A                ;SET FLAGS
0074      C2 EE 00                       JNZ     LUT              ;GOTO LOOK UP TABLE IF KEY FOUND
```

```
0077    7D                         MOV   A,L          ;NO KEY FOUND - BUMP COLUMN POINTER
0078    17                         RAL                ;ROTATE TO NEXT COLUMN
0079    6F                         MOV   L,A          ;PUT BACK
007A    E6 08                      ANI   08H          ;CHECK FOR LAST COLUMN
007C    CA 71 00                   JZ    LDKY         ;NOT LAST COLUMN - GO READ A KEY
007F    C3 59 00                   JMP   KEY          ;NO KEY DOWN GO BACK

                           ;+++++THESE ARE THE CONTROL KEY ROUTINES+++++

0082    21 14 00        CNTL:      LXI   H,TABLC-1    ;GET TABLE POINTER
0085    78                         MOV   A,B          ;GET KEY VALUE
0086    17              LP1:       RAL                ;ROTATE INTO CARRY
0087    23                         INX   H            ;BUMP TABLE POINTER
0088    D2 86 00                   JNC   LP1          ;
008B    6E                         MOV   L,M          ;MOVE ADDRESS INTO L REG
008C    E9                         PCHL               ;JUMP TO PROPER CONTROL ROUTINE
008D    3A B9 07        EXA:       LDA   LVALU        ;GET L REGISTER VALUE
0090    C3 C5 00                   JMP   BACK         ;DISPLAY IT & JUMP TO KEY
0093    3A B7 07        LDH:       LDA   KYTEM        ;GET KEY VALUE FROM TEMP
0096    32 BA 07                   STA   HVALU        ;PUT IN H REGISTER STORAGE
0099    C3 59 00                   JMP   KEY          ;DONE- GO TO START


009C    3A B7 07        RUN:       LDA   KYTEM        ;GET CURRENT DISPLAY VALUE
009F    32 B9 07                   STA   LVALU        ;STORE IN L REG LOCATION
00A2    2A BF 07                   LHLD  BSTOR        ;GET CONTENTS OF B&C RECS
00A5    E5                         PUSH  H            ;PUT ON STACK
00A6    C1                         POP   B            ;PUT IN B&C RECS
00A7    2A C1 07                   LHLD  DSTOR        ;GET CONTENTS OF D&E RECS
00AA    EB                         XCHG               ;EXCHANGE H&L WITH D&E
00AB    2A BD 07                   LHLD  PSWST        ;GET OLD AC AND PSW
00AE    E5                         PUSH  H            ;PUT AC & PSW ON STACK
00AF    F1                         POP   PSW          ;RESTORE AC & STATUS
00B0    2A B9 07                   LHLD  LVALU        ;GET STARTING ADDRESS
00B3    E5                         PUSH  H            ;PUT STARTING ADDR ON STACK
00B4    2A C3 07                   LHLD  HSTOR        ;RESTORE H&L
00B7    C9                         RET                ;GET STARTING ADDR FROM STACK AND RUN

00B8    3A B7 07        DISP:      LDA   KYTEM        ;GET CURRENT DISPLAY VALUE
00BB    32 B9 07                   STA   LVALU        ;STORE IN LREG STORAGE
00BE    2A B9 07                   LHLD  LVALU        ;GET VALUE JUST KEYED IN
00C1    22 B9 07        NEXT:      SHLD  LVALU        ;STORE IN MEMORY POINTER
00C4    7E                         MOV   A,M          ;GET VALUE POINTED TO BY MEM POINTER
00C5    32 B7 07        BACK:      STA   KYTEM        ;PUT THIS VALUE IN KEY STORAGE
00C8    CD 4F 01                   CALL  DIS          ;DISPLAY IT
00CB    C3 59 00                   JMP   KEY          ;GO BACK AND START OVER
00CE    2A B9 07        ENTER:     LHLD  LVALU        ;GET MEMORY POINTER
00D1    3A B7 07                   LDA   KYTEM        ;GET DISPLAY VALUE
```

```
00D4    77                      MOV   M,A          ;PUT VALUE IN LOC POINTED TO BY H&L
00D5    23                      INX   H            ;BUMP TO NEXT LOCATION
00D6    C3 C1 00                JMP   NEXT         ;PUT INC PTR AWAY AND DISPLAY NEXT LOC

00D9    3A C5 07      HO:       LDA   OFLAG        ;FETCH HEX/OCTAL FLAG
00DC    2F                      CMA                ;CHANGE TO OTHER BASE
00DD    32 C5 07                STA   OFLAG        ;PUT IT BACK
00E0    B7                      ORA   A            ;SET-UP FOR TESTING IT
00E1    CA E9 00                JZ    HOHO         ;JMP IF 0 FOR HEX
00E4    D3 06                   OUT   DISO         ;MUST BE 1'S FOR OCTAL - SET DISPLAY
00E5    C3 59 00                JMP   KEY
00E9    D3 07         HOHO:     OUT   DISH         ;SET DISPLAY FOR HEX 3 DIGITS
00EB    C3 59 00                JMP   KEY


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

00EE    47            LUT:      MOV   B,A          ;SAVE AC
00EF    7D                      MOV   A,L          ;GET COLUMN POINTER
00F0    0F                      RRC                ;ROTATE COL POINTER RIGHT
00F1    DA FB 00                JC    COL1         ;IS IT COL1?
00F4    0F                      RRC                ;ROTATE AGAIN
00F5    DA 04 01                JC    COL2         ;IS IT COL2?
00F8    C3 82 00                JMP   CNTL         ;MUST BE CONTROL COLUMN

00FB    21 79 01      COL1:     LXI   H,TABLE-1    ;GET TABLE POINTER
00FE    CD 33 01                CALL  DECOD        ;GO GET VALUE FROM TABLE
0101    C3 0E 01                JMP   SHIFT        ;STORE AND SEND TO DISPLAY
0104    21 79 01      COL2:     LXI   H,TABLE-1    ;GET TABLE POINTER
0107    CD 33 01                CALL  DECOD        ;GET VALUE FROM TABLE
010A    79                      MOV   A,C          ;PUT TABLE VALUE IN AC
010B    C6 02                   ADI   2H           ;CORRECT VALUE FOR COLUMN 2
010D    4F                      MOV   C,A
010E    21 B7 07      SHIFT:    LXI   H,KYTEM      ;GET OLD DISPLAY VALUE
0111    00                      NOP
0112    00                      NOP
0113    3A C5 07                LDA   OFLAG        ;CHECK HEX/OCT FLAG
0116    B7                      ORA   A            ;SET FLAGS
0117    C2 27 01                JNZ   OCT1         ;GOTO OCTAL IF FLAG IS A 1

011A    7E                      MOV   A,M          ;GET KEY CODE
011B    07            HEX1:     RLC                ;ROTATE ONE HEX DIGIT LEFT
011C    07                      RLC
011D    07                      RLC
011E    07                      RLC
011F    E6 F0                   ANI   0F0H         ;MASK OFF BOTTOM DIGIT
```

```
0121    B1                      ORA     C               ;OR NEW DIGIT TO OLD NUMBER
0122    77                      MOV     M,A             ;PUT BACK IN DISPLAY STORAGE
0123    CD 4F 01                CALL    DIS             ;SEND TO DISPLAY
0126    C9                      RET                     ;END OF NUMBER KEY ROUTINE

0127    7E              OCT1:   MOV     A,M             ;GET KEY CODE
0128    07                      RLC                     ;ROTATE ONE OCTAL DIGIT LEFT
0129    07                      RLC
012A    07                      RLC
012B    E6 F8                   ANI     370Q            ;MASK OFF BOTTOM DIGIT
012D    B1                      ORA     C               ;OR NEW DIGIT TO OLD NUMBER
012E    77                      MOV     M,A             ;PUT BACK IN DISPLAY STORAGE
012F    CD 4F 01                CALL    DIS             ;SEND TO DISPLAY
0132    C9                      RET

0133    78              DECOD:  MOV     A,B             ;GET KEY VALUE
0134    17              AGAIN:  RAL                     ;ROTATE INTO CARRY
0135    23                      INX     H               ;BUMP TABLE POINTER
0136    D2 34 01                JNC     AGAIN           ;
0139    4E                      MOV     C,M             ;SAVE KEY CODE

013A    3A C5 07                LDA     OFLAG           ;CHECK HEX/OCT FLAG
013D    B7                      ORA     A               ;SET FLAGS
013E    C2 42 01                JNZ     OCT2            ;IF IN OCTAL MODE JUMP TO CHAR CHECK
0141    C9                      RET

0142    79              OCT2:   MOV     A,C             ;GET KEY VALUE
0143    E6 F8                   ANI     370Q              ;MASK OFF LOWER DIGIT
0145    C8                      RZ                      ;RETURN IF LEGAL OCTAL NUMBER
0146    C3 59 00                JMP     KEY             ;ILLEGEL CHAR GOTO  KEY

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

0149    3A 07 05        READ:   LDA     KEYBD           ;READ KEYBOARD
014C    2F                      CMA                     ;COMPLEMENT
014D    B7                      ORA     A               ;SET FLAGS
014E    C9                      RET

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

014F    3A B7 07        DIS:    LDA     KYTEM           ;GET CURRENT DISPLAY VALUE
0152    4F              DISPLAY: MOV    C,A             ;SAVE A REG
0153    3A C5 07                LDA     OFLAG           ;CHECK HEX/OCT FLAG
0156    B7                      ORA     A               ;SET FLAGS
0157    C2 5E 01                JNZ     OCT             ;SIGN BIT=1 FOR OCT DISPLAY
015A    79              HEX:    MOV     A,C             ;HEX - GET AC
015B    D3 00                   OUT     0               ;SEND TO DISPLAY
015D    C9                      RET
015E    79              OCT:    MOV     A,C             ;GET NUMBER TO DISPLAY
```

```
015F   07                  RLC              ;GET HIGH ORDER DIGIT
0160   07                  RLC              ;ROTATE INTO POSITION
0161   E6 03               ANI    3Q        ;SAVE HIGH ORDER DIGIT
0163   D3 05               OUT    5         ;DISPLAY HIGH ORDER DIGIT
0165   79                  MOV    A,C       ;GET NUMBER AGAIN
0166   17                  RAL              ;MOVE 2ND DIGIT INTO POSTION
0167   E6 70               ANI    160Q      ;SAVE MIDDLE DIGIT
0169   47                  MOV    B,A       ;SAVE MIDDLE DIGIT
016A   79                  MOV    A,C       ;GET NUMBER AGAIN
016B   E6 07               ANI    7Q        ;GET 1ST DIGIT
016D   B0                  ORA    B         ;COMBINE DIGITS 1 & 2
016E   D3 00               OUT    0         ;DISPLAY THEM
0170   C9                  RET

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

0171   06 00      DELAY:   MVI    B,0       ;INITIALIZE COUNTER
0173   04         LOOP:    INR    B         ;BUMP COUNTER
0174   E3                  XTHL             ;EXTRA DELAY IN LOOP
0175   E3                  XTHL
0176   C2 73 01            JNZ    LOOP      ; LOOP UNTIL ZERO
0179   C9                  RET

017A   00         TABLE:   DB     00H       ;NUMBER KEY CODE TABLE
017B   04                  DB     04H
017C   08                  DB     08H
017D   0C                  DB     0CH
017E   01                  DB     01H
017F   05                  DB     05H
0180   09                  DB     09H
0181   0D                  DB     0DH
                           END
NO PROGRAM ERRORS
```

SYMBOL TABLE

* 01

A	0007		AGAIN	0134		B	0000		BACK	00C5	
BKSTO	07BB		EOT	000F	*	BRK	0033	*	BSTOR	07BF	
C	0001		CNTL	0082		COL	006E	*	COL1	00FB	
COL2	0104		D	0002		DECOD	0133		DELAY	0171	
DIS	014F		DISH	0007		DISO	0006		DISP	00B3	
DISPL	0152	*	DSTOR	07C1		E	0003		ENTER	00CE	
EXA	008D		H	0004		HEX	015A	*	HEX1	011B	*
HO	00D9		HOHO	00E9		HSTOR	07C3		HVALU	07BA	
INIT	0000	*	KEY	0059		KEYBD	0507		KYBD1	0501	
KYTEM	07B7		L	0005		LDH	0093		LDKY	0071	
LOOP	0173		LP1	0086		LUT	00EE		LVALU	07B9	
M	0006		NEXT	00C1		OCT	015E		OCT1	0127	
OCT2	0142		OFLAG	07C5		PCSTO	07BB		PSW	0306	
PSWST	07BD		READ	0149		REP	0062		RREAD	0002	*
RUN	009C		SHIFT	010E		SP	0006		ST	000F	
TABLC	0015		TABLE	017A		TOP	00F0	*			

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 $3.50

Page Range	Domestic Price	Page Range	Domestic Price
001-025	$ 5.00	326-250	$18.00
026-050	6.00	351-375	19.00
051-075	7.00	376-400	20.00
076-100	8.00	401-425	21.00
101-125	9.00	426-450	22.00
126-150	10.00	451-475	23.00
151-175	11.00	476-500	24.00
176-200	12.00	501-525	25.00
201-225	13.00	526-550	26.00
226-250	14.00	551-575	27.00
251-275	15.00	576-600	28.00
276-300	16.00	601-up [1]	
301-325	17.00		

[1] Add 2.00 for each additional 25 page increment from 601 pages up.

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