OPERATIONS
MANUAL

Z80 Starter System

SD Systems

Z80 STARTER KIT™

OPERATIONS MANUAL

S.D. Systems and Micro Design Concepts reserves the right to make changes to this publication at any time and without notice. The information furnished herein is believed to be accurate and reliable. However, no responsibility is assumed by S.D. Systems or Micro Design Concepts for its use; nor for any infringements of patents or other rights of third parties resulting from its use.

August 1978
Rev. A January 1979

TABLE OF CONTENTS

TABLE OF CONTENTS

SECTION 1

INTRODUCTION

1-1 GENERAL

Congratulations on your choice of the Z80 STARTER KIT! This kit has been designed to be the best value on the market for the hobbiest/experimenter/student who wants to learn about and work with microcomputers. Its expansion capabilities are limited only by your imagination with the on-board wire wrap area and the two optional S-100 bus connectors. Compatibility with S-100 provides you with a varied selection of modules from several manufacturers to accomplish your task whether it be learning, running Basic, or industrial control. Use of this expansion capability is entirely optional, as the Z80 STARTER KIT enclosed is a fully functional microcomputer with debug Monitor (ZBUG) by the addition of a 5 volt power supply.

The choice of the Z80 Microprocessor as the "brains" of this kit was no accident. The Z80 is the most powerful 8-bit machine available on the market, as its instruction set capability and throughput exceeds that of any other 8-bit machine available. Its vast instruction set of 158 instruction types and clear, easy-to-learn mnemonics make it an ideal processor on which to learn assembly language programming. The 8080A instruction set is a subset of the Z80's, so that programs written for the 8080A will run on the Z80 allowing you access to many programs written for the 8080A and documented in the trade and personal computing magazines. The straightforward hardware architecture (no multiplexing) of the Z80 make it ideal for the experimenter who wants to connect other peripherals or custom circuitry onto the bus. The Z80's indexing capability, 16-bit Op Codes, and 16-bit Arithmetic operations provide features normally found only in a 16-bit minicomputer.

Two forms of permanent storage for your programs have been provided in the Z80 STARTER KIT. The first is a Kansas City Standard audio cassette interface that can be used with inexpensive home audio recorders. Programs can be transferred from RAM to cassette tape using this feature, providing an inexpensive method of saving and reloading your programs. Second, on-line Non-Volatile memory has been provided in the form of two EPROM sockets on the Z80-CPU bus and an EPROM Programmer for 2758/2716 5 volt only EPROMs. With this facility, user programs can be placed in EPROM and on the bus for immediate access by the Z80-CPU.

1-2 USES FOR THE Z80 STARTER KIT

The Z80 STARTER KIT was designed with five major types of user in mind. These are the computer hobbiest, electronics experimenter, amateur radio operator, instructor/student, and industrial OEM evaluation/control.

The computer hobbiest may have experience with other microcomputers or minicomputers, but needs a low-cost method to get "hands on" experience with the Z80. The diagnostic capability of the Z80 STARTER KIT and its mass storage facilities provide the user with the capability to meet this goal.

The expansion capabilities make it possible to add memory and a CRT Interface in order to turn the KIT into a BASIC terminal for high level language experience.

The electronics experimenter is probably familiar with TTL integrated circuits and is looking for a low-cost method to "get into microcomputing". The wire wrap area with the Z80-CPU signals brought to wire wrap pins is intended for those who want to connect additional circuitry onto the Z80 bus for experimentation. With the on-board keyboard and display, no computer peripherals such as Teletypes are required to communicate with and to control the Z80-CPU. The experimenter can easily learn how to substitute Z80 instructions for gates, flip-flops, adders, and shift registers.

The amateur radio operator may be looking for methods to integrate computers into his "shack" in order to more fully automate his station. Areas the power of the Z80 could be used is in automatic keying - the key could be connected to the Z80-PIO and the Z80 Microcomputer programmed to make precise "dits" and "dahs". A Morse code to ASCII conversion program could be written for the Z80 STARTER KIT to allow the operator to type on an ASCII keyboard and view a CRT Display while communicating in Morse code.

An instructor of, or a student in a course in microcomputers, can use the Z80 STARTER KIT to provide a low-cost method to provide lab experience with a microcomputer. Each student could have his own cassette of the program being developed and would load it into the Z80 STARTER KIT(s) during the lab session in order to debug it. Timeshare could be used as a supplement to provide experience in Assembly Language Programming, but "hands on" experience at the hardware level is needed to gain a working knowledge of microcomputers. The timeshare charges from one class would be sufficient to equip an entire lab with Z80 STARTER KITS!

The engineer in industry can use the Z80 STARTER KIT as an evaluation tool in order to determine if microcomputers, or the Z80 in particular, can solve the problem. Short programs or benchmarks can be written and debugged using the diagnostic features of the ZBUG Monitor (Single Step, Breakpoints, etc.). The Restart to EPROM feature of the KIT would allow its use as a dedicated computer in a test fixture or process control system. Whenever power was applied, the KIT would begin to run the Control Program in EPROM. The integral Keyboard/Display and the ZBUG Monitor would also be available should debugging be required in the final installation.

1-3 Z80 OVERVIEW

The Z80 component set is a third generation design based on the Intel 8080A. The Z80 has at its base the entire 8080A instruction set with an additional 80 instruction types added (158 total). The Z80-CPU hardware configuration with no multiplexed signals and a simple to generate TTL compatible single phase clock is straightforward and easy to understand. Enhanced features such as Relative Addressing, two 16-bit Index Registers, bit addressing, full Rotates and Shifts, 22 CPU

Registers, 16-bit Arithmetic capability, Block operations, dynamic memory refresh, and 16-bit Op Codes make the Z80 a more flexible and more powerful Microcomputer than any other 8-bit machine available including: 8080A/8085, 6502, or the 6800/6802. It is this computing power that has made the Z80 STARTER KIT possible.

Included in the Z80 STARTER KIT is the MOSTEK Z80 Micro-Reference Manual. This is a summary of the Z80 Instruction Set and will be used in the following discussion. Further information on the Z80 can be found in Section 1-4. The cover shows the Z80-CPU registers as viewed by the programmer. The main register set is exactly the same as those in an 8080A/8085 and are used by the 8080A compatible instructions. The alternate register set is an exact duplicate of the main set and is a unique feature to the Z80. These registers can be used to handle additional variables/flags, or can be used to preserve the status of the Z80-CPU during an interrupt or subroutine. The Special Purpose Registers contain the two 16-bit Index Registers, the I register used to provide the fast Z80 Mode 2 interrupts, the R register used in dynamic memory refresh, the Stack Pointer, and the Program Counter.

Page 1 is a summary of the Z80 Flag (F) Register and how instruction types affect it. Pages 2 and 3 are a summary of the 8-bit load operations. Those blocks shaded are the instruction Op Codes compatible with the 8080A. Unshaded blocks are the new Z80 instructions. The new instructions deal with Indexed loads, as the 8080A has a rather complete 8-bit load group. Page 3 details each instruction--additional information is provided in the Z80 Programming Manual. The 16-bit loads on pages 4 and 5 show the addition of the load and store of the two Index Registers. BC and DE have additional pointer capability, as they can be loaded directly from memory without having to go through HL. The Exchange group shows the Z80 Block Operations which are complete subroutines implemented in a single Z80 instruction. The Exchange instructions allow the programmer access to the Alternate Registers. Pages 8 and 9 contain the 8-bit arithmetic instructions, most of which are compatible with the 8080A. The indexed operations show up here as new instructions. Pages 10 and 11 contain some miscellaneous instructions. New instructions include the 2's complement (negate) and interrupt mode selection. Mode zero emulates an 8080A, while Mode 2 is used by Z80 Peripherals such as the Z80-PIO and Z80-CTC included in this kit.

The 16-bit arithmetic instructions allow additions with the index registers for pointer address modifications. Two new instructions (ADC and SBC) use the HL register pair as a 16-bit accumulator providing 16-bit arithmetic capability normally found only in minicomputers. Pages 14 and 15 detail all of the new Rotates and Shifts added to the 8080A subset. Note that these instructions can also operate on memory with the HL, IX, and IY registers as pointers much like the 6800. The RLD and RRD instructions allow the packing and unpacking of BCD digits from memory to the accumulator. The Bit Manipulation Group allows the programmer to address any one bit in memory or the CPU registers. Individual bits can be tested, reset, and set. On pages 18 and 19 the Relative Jump additions, which allow two byte jumps rather than the 8080A three byte jumps are described. The hexadecimal arithmetic capability of the ZBUG Monitor make Relative Jump offsets easy to calculate. The Call and Return group is the same with the addition of two new Return instructions (RETI and RETN). RETI is decoded by the Z80 Peripherals to signal the end of an Interrupt Service Routine and RETN is used to exit Non-Maskable Interrupts. Several new instructions have been added to the Input and Output group on pages 22 and 23. These new instructions allow the use of the C register as a pointer to the port, and data can be transferred from all 8-bit CPU registers to the I/O Port. The Block I/O instructions also use C as the port pointer with HL used as the memory buffer pointer. Page 24 is a summary of the Z80-CPU Interrupt Structure which will be useful when programming an interrupt driven system. Pages 25 and 26 are programming summaries for the Z80-PIO and CTC.

1-4 ADDITIONAL INFORMATION

Additional information on the Z80 hardware and software is available from your local dealer or S.D. Sales. References are given below in two catagories:

General information on Microcomputers

1) An Introduction to Microcomputers, Volume 0-
The Beginner's Book, Osborne and Associates.

If you know nothing about computers, then this is the book for you. It introduces computer logic and terminology in language a beginner can understand. Computer software, hardware, and component parts are described, and simple explanations are given for how they work. The text is supplemented with creative illustrations and numerous photographs. Volume 0 prepares the novice for Volume 1. (300 pages)

2) An Introduction to Microcomputers, Volume 1- Basic Concepts, Osborne and Associates.

This best selling text describes hardware and programming concepts common to all microprocessors. These concepts are explained clearly and thoroughly, beginning at an elementary level. (350 pages)

More information on the Z80

1) Z80-CPU Technical Manual, MOSTEK/Zilog.
2) Z80-CTC Technical Manual, MOSTEK/Zilog.
3) Z80-PIO Technical Manual, MOSTEK/Zilog.

These manuals completely describe the three Z80 chips that form the heart of the Z80 STARTER KIT.

4) Z80 Programming Manual, MOSTEK/Zilog.

A complete description of how every Z80 instruction operates--the absolute reference when programming the Z80.

5) Z80 Programming for Logic Design, Osborne and Associates.

Describes programming/logic design tradeoffs. Detailed examples to illustrate effective usage of microprocessors in traditional digital applications.

SECTION 2

CONSTRUCTION

2-1 INTRODUCTION

The Z80 STARTER KIT is intended for those people who have had some prior experience with kit building and digital electronics. If you do not fall into this catagory, it is highly recommended that you find an experienced person to help you in assembly and check out of the board. Appendix I shows the parts list for the Z80 STARTER KIT.

2-2 ASSEMBLY PROCEDURE - Check ( ) when done.

( ) 1. Install the nylon legs and metal screws in the holes spread around the board. Use Figure 2-1 to locate holes for the nylon legs. Do not overtighten the screws (finger tight is OK) or be concerned if the metal screws touch a PC etch run - this is normal.

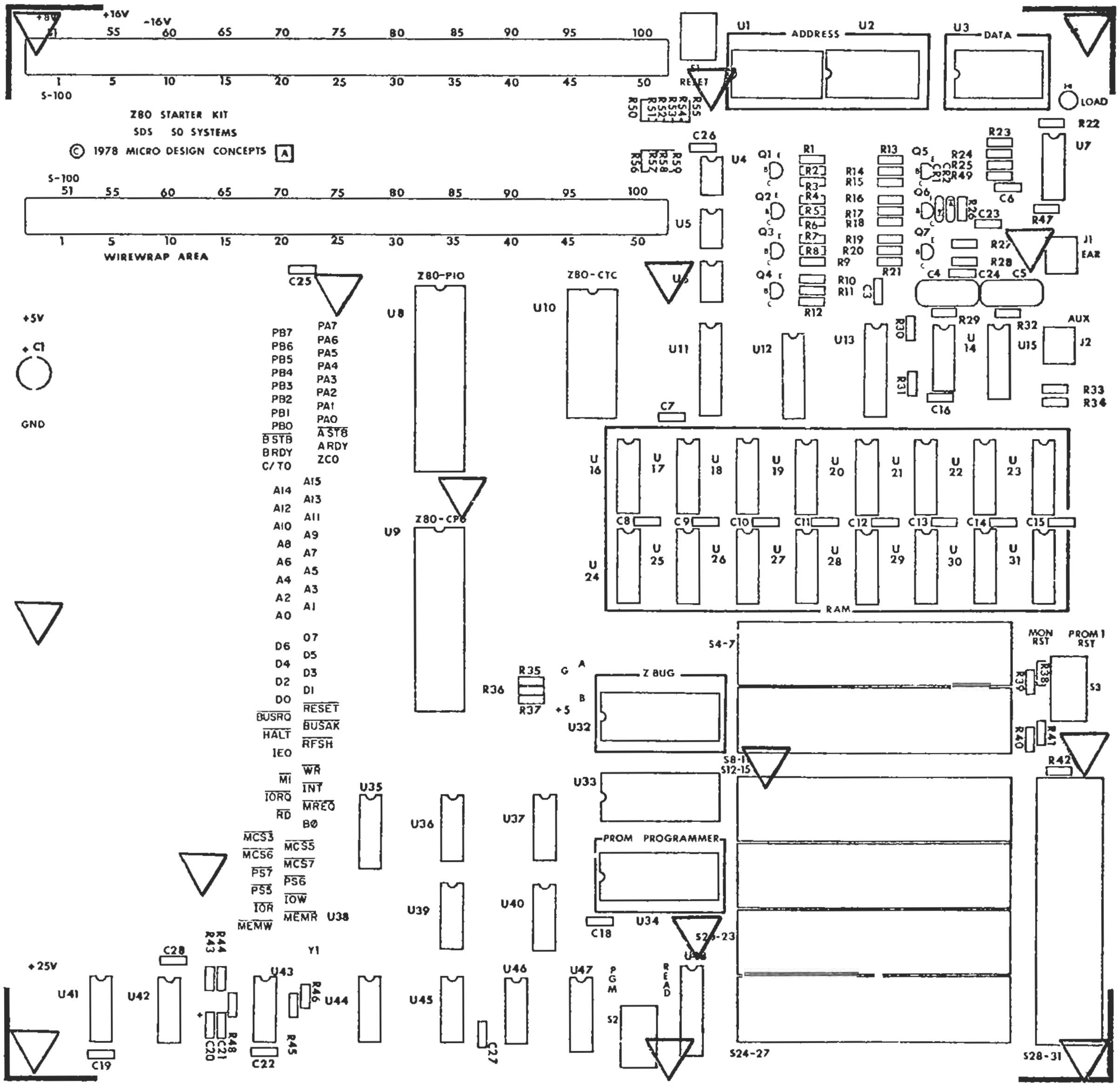
( ) 2. Install the IC sockets in their proper locations as follows: (Pin 1 alignment is shown with a ".")

( ) a. Three 8-pin sockets at U4, U5, and U6.

( ) b. Twelve 14-pin sockets at U7, U14, U36, U37, U39, U40, U41, U42, U43, U44, U45, and U46.

( ) c. Eleven 16-pin sockets at U15, U24, U25, U26, U27, U28, U29, U30, U31, U35, and U47.

( ) d. Four 20-pin sockets at U11, U12, U13, and U48.



▽ = NYLON LEG LOCATION

FIGURE 2-1

NYLON LEG MOUNTING DIAGRAM

( ) e. Three 24-pin sockets at U32, U33, and U34.

( ) f. One 28-pin socket at U10.

( ) g. Two 40-pin sockets at U8 and U9.

( ) 3. Install the resistors as follows:

( ) a. R3, R6, R9, R12, R15, R18, R21 - 68 Ohm, ¼W, 5% (Blue, Grey, Black)

( ) b. R22, R46 - 330 Ohm, ¼W, 5% (Orange, Orange, Brown)

( ) c. R23, R27, R28, R33, R43, R45, R48 - 1K Ohm, ¼W, 5% (Brown, Black, Red)

( ) d. R2, R5, R8, R11, R14, R17, R20 - 4.7K Ohm, ¼W, 5% (Yellow, Violet, Red)

( ) e. R1, R4, R7, R10, R13, R16, R19, R25, R30, R34, R35, R36, R37, R38, R39, R40, R41, R42, R47, R50, R51, R52, R53, R54, R55, R56, R57, R58, R59 - 10K Ohm, ¼W, 5% (Brown, Black, Orange)

( ) f. R26, R31, R49 - 100K Ohm, ¼W, 5% (Brown, Black, Yellow)

( ) g. R24 - 220K Ohm, ¼W, 5% (Red, Red, Yellow)

( ) h. R29, R32 - 470K Ohm, ¼W, 5% (Yellow, Violet, Yellow)

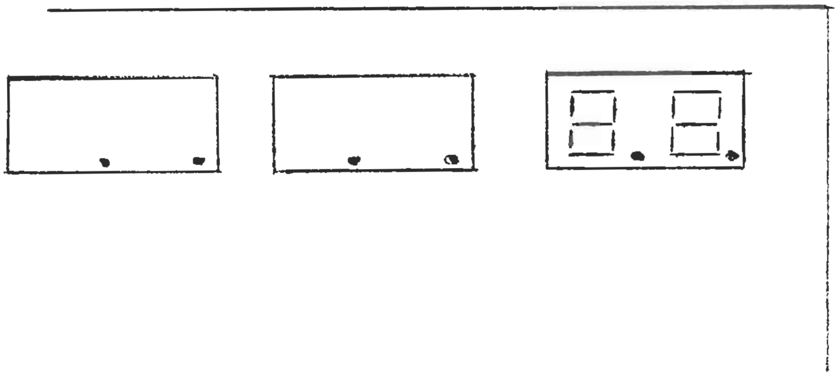
( ) i. R44 - 47K Ohm, ¼W, 5% (Yellow, Violet, Orange)

( ) 4. Install diodes CR1 and CR2 with the banded ends as shown on the PC board.

( ) 5. Install the capacitors as follows:

( )	a.	C1 -	10uF Tantalum (note polarity)
( )	b.	C21 -	10pF Mica 2%
(✓)	c.	C4, C5 -	620pF Mica 2%
(✓)	d.	C16 -	.0047uF Ceramic 20%
(✓)	e.	C22 -	.01uF Ceramic 20%
(✓)	f.	C23 -	.047uF Ceramic 20%
( )	g.	C2, C3, C6, C7, C8, C9, C10, C11, C12, C13, C14, C15, C18, C19, C24, C25, C26, C27, C28 -	.1 Ceramic 20%
(✓)	h.	C20 -	1uF Tantalum 20% (note polarity)

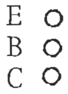
( ) 6. Install the three display modules at U1, U2, and U3. Make sure that the decimal point on the display is oriented away from the edge of the PC board.



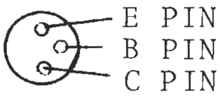
( ) 7. Install the LED at DS1 observing the cathode orientation (the flat portion of the LED housing is the cathode).

( ) 8. Install transistors Q1, Q2, Q3, Q4, Q5, Q6, and Q7 observing the proper orientation marked on the pc board.

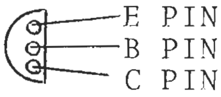
BOARD HOLE PATTERN



TYPE #1



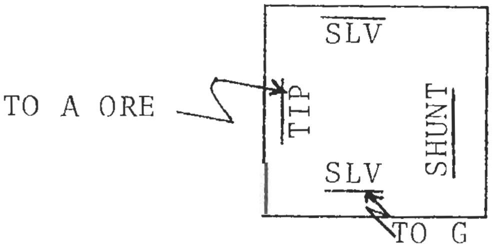
TYPE #2



( ) 9. Install the Crystal at location Y1.

( ) 10. Install the two audio jacks J1 and J2 in 1/4 in. holes. On J2 connect the TIP to the pad with "A" next to it. Connect the SLV terminal to the "G" pad as shown. On J1 connect the TIP to the pad with "E" next to it and the SLV terminal to the "G" pad.

AUDIO JACK-REAR VIEW



( ) 11. Install the Push Button Switch at S1.

( ) 12. Install the two Toggle Switches at S2 and S3.

( ) 13. Install a switch bank at locations S4-S27 and lock down with the plastic nuts provided. Make sure that the switch bank is securely mounted against the pc board before soldering it in. Make sure that the switch leads are coming straight out of the switch body for easier assembly.

( ) 14. Mount the keytops using the layout given in the following sketch:

PROM PROG	CASS LOAD	CASS DUMP	BREAK POINT	
MEM EXAM	PORT EXAM	REG EXAM	REG' EXAM	
7	8	9	A	NEXT
4	5	6	B	MON
1	2	3	C	SINGLE STEP
0	F	E	D	EXEC

2-3 VOLTAGE CHECK

( ) 1. Connect a +5v±5% power supply capable of supplying at least 1 Ampere to the designated points on the left hand side of the board.

( ) 2. Measure the power at socket U35--pin 16 should be +5v with pin 8 being ground. Do not proceed past this point until this voltage reading is correct. Remove the power from the board.

( ) 3. Install the IC's in their sockets observing pin 1 designation (small solder spot on PC board).

(✓)	a.	U4, U5, U6	75452
(✓)	b.	U7	339
( )	c.	U9	MK3880 Z80-CPU
( )	d.	U10	MK3882 Z80-CTC
( )	e.	U8	MK3881 Z80-PIO
(✓)	f.	U11, U12	74LS273
( )	g.	U13, U48	74LS244
(✓)	h.	U14	14013
(✓)	i.	U15	14538
( )	j.	U24, U25, U26, U27, U28, U29, U30, U31	21L02
( )	k.	U32	8316/2316 ZBUG ROM
(✓)	l.	U35, U47	74LS138
(✓)	m.	U36	74LS08
(✓)	n.	U37, U44	74LS04

(✓)	o.	U39, U40, U46	74LS32
(✓)	p.	U41, U42	74LS74
(✓)	q.	U45	74LS02
(✓)	r.	U43	7404

( ) 4. Double check all IC's for proper orientation and location.

( ) 5. Make sure S3 is in the MONITOR RST position. Apply power to the board (5v±5%) and depress the Reset switch (S1). A "–" should appear on the display indicating the kit is alive and well. If this is the case, read Section 3 concerning the ZBUG Monitor commands.

( ) 6. If the prompt "–" does not come up, remove the power and carefully inspect the underside of the board for cold solder joints, unsoldered joints, and solder shorts. Experience has found these problems to be the most likely cause of kit malfunction.

( ) 7. If the kit still fails to operate and you do not have the equipment available to diagnose the problem contact your dealer or S.D. Systems for further instructions.

SECTION 3

ZBUG MONITOR DESCRIPTION

3-1 INTRODUCTION

The ZBUG Monitor program is a 2048 byte program written for the Z80 which allows the user to enter and debug machine level Z80 programs. This program is supplied in a mask programmed Read Only Memory (ROM) in the Z80 STARTER KIT. The ZBUG Monitor uses a Hexadecimal keyboard for data entry and a six digit Hexadecimal display for data readout. Also included in ZBUG are Load and Dump programs which allow inexpensive audio cassette recorders to be used for storage of programs. An EPROM Programmer for 2716/2758 EPROMs is included so that user's programs can be placed in these non-volatile memory devices for on-line use at any time. Advanced diagnostic capability such as multiple Breakpoints, Instruction Single Step, and Z80-CPU Register display/modification provides the user with diagnostic capability normally found only in expensive development equipment. The ZBUG Monitor provides the user with a Hexadecimal arithmetic capability which makes the Z80's relative addressing mode easy to use. In summary, the ZBUG Monitor provides the user of the Z80 STARTER KIT with complete control over the execution and debug of the program being developed and provides for non-volatile data or program storage in either EPROM or cassette tape. The use of ZBUG is described in the following paragraphs.

3-2 RESET PUSH BUTTON

The RESET push button near the top center of the Z80 STARTER KIT forces the Z80-CPU to reset and begin program execution at address 0000H. The ZBUG Monitor is located in the lower 2K bytes of the address space so that a RESET of the Z80-CPU will restart to the ZBUG Monitor. The ZBUG Monitor initializes the user's Stack Pointer to 23C0H, initializes RAM variables, and if switch S3 is set to the MONITOR RESTART position the ZBUG Monitor places the prompt symbol "—" on the left hand display and begins scanning the keyboard for an entry. If switch S3 is in the PROM1 RESTART position, the ZBUG Monitor automatically vectors the program execution after reset to PROM1 at address 0800H, providing a means to do a power-up restart to a user's program in PROM1. This feature can be useful if the Z80 STARTER KIT is being used in a dedicated control application and it is desired to always restart to the control program in PROM1. Interrupts are disabled in the Z80-CPU and a Mode 0 interrupt is selected upon reset. Active interrupts in Mode 2 will normally be selected by instructions in a user's program.

3-3 MONITOR

The purpose of the MONitor key is to suspend program execution and to return control to that portion of ZBUG that scans the keys for a new input or command. While RESET will also return control to ZBUG, the MON key preserves the status of the Z80-CPU registers and ZBUG RAM variables. The MON key is used in one of two basic ways: first it is used to cancel or terminate a previous command or data entry. Depressing of the MON key will terminate a partial or complete data entry or will allow one to exit a command mode such as Port Examine or Memory Examine. Second, the MON key has been designed to produce a Non-Maskable Interrupt to the Z80-CPU whenever it is depressed while the Z80-CPU is executing a user's program. This feature is very useful in trouble shooting; e.g. if the processor has executed a Halt instruction or has gone "out into the weeds" due to a faulty program, the MON key allows the user to return to ZBUG while saving the Z80-CPU registers so that it can be determined where the Z80-CPU was executing at the time the MON key was depressed. Whenever the MON key is depressed the Prompt symbol "–" will be displayed on the left hand display. Make sure this prompt symbol is displayed before a new mode of operation is attempted.

3-4 MEMORY EXAMINE

The MEM EXAM key is used to examine and change memory locations. The first step in using the MEM EXAM key is to enter four Hex digits (0 through F) representing the memory address desired. Enter the address high digit first and it will be registered on the address displays as it is entered. When all four address digits have been entered, press the MEM EXAM key and the data in that memory location will appear on the data displays. If the MEM EXAM key is depressed before four hex digits have been entered, no data will be displayed and ZBUG will wait for the remaining address digit(s). Depressing the NEXT key will cause the memory address to increment by one and the data display will update corresponding to the new address. At any time there are six digits showing (four address, two data) new data can be entered into that memory address by simply entering two more hex digits. The data display will not update until both digits have been entered because ZBUG first writes data into the memory, then reads it back to the data display. This is done in order to display to the user the actual data taken by memory so that attempts to change ROM or non-existant memory will be noticed by the user. Continual depressing of the MEM EXAM key will re-read and re-display the contents of the address displayed.

3-5 PORT EXAMINE

The PORT EXAM key is used to examine and change port locations. The first step in using the PORT EXAM key is to enter two hex digits (there are 256 port addresses in the Z80 architecture) representing the port address desired. Enter the address high digit first and it will be registered on the address displays as it is entered. When both address digits have been entered, press the PORT EXAM key and the data at that port location will appear on the data displays. If the PORT EXAM key is depressed before two address digits have been entered, no data will be displayed and ZBUG will wait for the remaining address digit. Depressing the NEXT key will cause the port address to increment by one and the data display will update corresponding to the new address. Any time there are four digits showing (two address, two data), new data can be entered into that port address by simply entering two more hex digits. Depressing the NEXT key will cause the port address to increment by one and the data display will update corresponding to the new address. Continual depressing of the PORT EXAM key will re-read and re-display the contents of the port address displayed. This can be useful with the Z80-CTC because the down counter can be observed counting by using this technique. The Port Examine mode can be aborted at any time by depressing the MON key.

3-6 REGISTER EXAMINE

The following registers can be examined and changed by use of the REG EXAM key: A, B, C, D, E, F, H, L, I, IFF, PC, IX, and IY. The Stack Pointer can be examined with this key but it cannot be changed. Depress the data key corresponding to the register desired followed by the REG EXAM key. The display will show the register selected and its value. The values displayed come from the "User's Register Map" area of RAM and are unloaded from the Z80-CPU whenever a breakpoint is encountered, a single step is commanded, or the MON key is depressed. To change the value shown for a register simply enter two digits of new data (four digits for IX, IY, and PC). IFF is the state of the Interrupt Flip Flop inside the Z80-CPU (a value of 00 means that interrupts are disabled and a value of 04 means that interrupts are enabled). This mode of operation can be aborted at any time by depressing the MON key. Whenever an execution or a single step is commanded, the Z80-CPU registers will be initialized with the values from the "User's Register Map", allowing the user to modify registers before execution begins.

3-7 ALTERNATE REGISTER EXAMINE

The following registers can be examined and changed by the use of the ALT REG EXAM key: A', B', C', D', E', F', H', and L'. The prime mark is another designation for the alternate register set. Depress the data key corresponding to the register desired followed by the REG EXAM' key. The display will show the register selected and its value. The values displayed come from the "User's Register Map" area of RAM and are unloaded from the Z80-CPU whenever a breakpoint is encountered, a single step is commanded, or the MON key is depressed. to change the value shown for a register simply enter two digits of new data. This mode of operation can be aborted at any time by depressing the MON key. Whenever an execution or a single step is commanded, the Z80-CPU registers will be initialized with the values from the "User's Register Map" allowing the user to modify registers before execution begins.

3-8 BREAKPOINTS

The ZBUG Monitor has the capability to set up to five breakpoints in any user's program that is executing out of

RAM. The method of breakpointing is to exchange the user's Op Code with a RST8 (CFH) and to preserve the user's Op Code in a table of breakpoint addresses and Op Codes (BPTAB). Whenever a breakpoint is set it is entered into the table and then a RST8 inserted into the user's code just before execution of the user's code. Upon encountering a breakpoint, control is returned to ZBUG through the RST8 and all Z80-CPU registers are preserved in the "User's Register Map". All RST8 instructions are removed and replaced by the user's Op Codes which have been saved in BPTAB. This is done so that whenever control is transferred to ZBUG, all user code is intact and can be examined and modified by the MEM EXAM command. Breakpoints are set by entering the four digit address of the Op Code at which the breakpoint is desired, followed by the BREAKPOINT key. When setting a breakpoint at a two byte Op Code, the address of the first byte of the Op Code must be used as the breakpoint address. The display will show the address entered after the BREAKPOINT key is released to indicate that the breakpoint has been accapted. To enter another breakpoint press the MON key to get the prompt sign, followed by a new four digit address and then the BREAKPOINT key. Should an attempt be made to enter more than five breakpoints, ZBUG will notify the user by not displaying the address after the BREAKPOINT key is released and instead will display the prompt sign. The five breakpoints already entered are left intact.

Breakpoints can be canceled at any time in one of three ways: First, if the breakpoint key is depressed before four address digits have been entered, all breakpoints will be canceled or removed. Thus, when the prompt sign is being displayed, depressing the BREAKPOINT key will clear all previous entrys. Second, use of the SINGLE STEP key will cancel or remove all existing breakpoints. Third, depressing the RESET push button will cancel or remove all breakpoints. Use of the MON key has no effect on breakpoints with the following exception: The only way the ZBUG Monitor will restore the user's Op Codes is if it is entered through the RST8 instruction. Should the MON key be used to abort execution of the user's code (because the user's code has a HALT instruction, it is hung in a loop, or just "out in the weeds") the RST8 breakpoint instructions will be left imbedded in the user's code. The addresses of the breakpoints can be determined by examining the BPTAB table (23E4H) - the format of this table is first breakpoint address high byte, first address low byte, first Op Code, second address high byte, etc. The number of breakpoints currently active is contained in BFLG (23F4H).

3-9 SINGLE STEP

The SINGLE STEP command key provides the user with the capability to execute the program under development one instruction at a time - returning to the ZBUG Monitor after each instruction for examination of registers, memory, ports, etc. Single step can be used on programs in RAM, ROM, or EPROM because no modification of user's code is required. One channel of the Z80-CTC is used to produce a pulse on the Non-Maskable interrupt input to the Z80-CPU at the beginning of the first instruction, thereby returning the Z80-CPU to the ZBUG Monitor, As with breakpoints, all Z80-CPU registers are preserved in the "User's Register Map" after the one instruction has been executed.

The instruction that will be executed is the one pointed to by the PC in the "User's Register Map" and can be examined or changed by using the Register Examine Mode. Depressing the SINGLE STEP key will re-load all Z80-CPU registers from the "User's Register Map" and execute one instruction. The Z80-CPU will return to the ZBUG Monitor through 66H (NMI address) and ZBUG will save all Z80-CPU registers in the "User's Register Map" portion of memory. The address of the <u>next</u> instruction to be executed is displayed in the Address displays and the current state of the Accumulator is displayed in the Data displays. By repeatedly depressing the SINGLE STEP key, the user can step through the program under development, viewing the address of the next instruction to be executed (very useful for conditional Jumps and Calls) and the current contents of the Accumulator. Other registers can be examined or changed by depressing the MON key, followed by the appropriate Register Examine keys. Use of the SINGLE STEP key will cancel or remove any breakpoints inserted to date.

3-10 EXECUTE

The EXECute key allows the user to command the Z80-CPU to begin execution of a user's program in either RAM, ROM, or EPROM. Two modes of operation are provided: Proceed from the current address, or Execute from the address entered and shown on the display. The Proceed mode uses the Program Counter saved in the "User's Register Map" as the beginning point of execution. To use this mode simply depress the EXEC key and execution will begin at the address displayed by the PC key in the Register Examine mode. This mode is very useful to resume execution (Proceed) after hitting a Breakpoint or after using the Single Step mode. To execute from the beginning of a program, enter the four digits of the desired starting address followed by the EXEC key. Once execution has been started, control will remain in the user's program until a breakpoint is hit or until the MON abort key is used.

3-11 CASSETTE DUMP

This mode of operation is used to save volatile programs or information in the RAM on inexpensive cassette tape using the Kansas City Standard recording technique. Experience has shown that this mode can be used with most recorders and audio cassette tape on the market. Should you be buying an audio recorder to use with this mode, a Panasonic Model No. RQ-309DS is recommended. Radio Shack Realistic C-30 cassettes (Cat. No. 44-601A) can be used for inexpensive cassette tapes. Connect the recorder to the Z80 STARTER KIT using an audio patch cord (also available at Radio Shack) connecting the "AUX" connector on the Z80 STARTER KIT to the "AUXILIARY" or "MIC" input of the tape recorder. Once the data to be saved is in RAM, set up memory locations 23C0H-23C3H with the starting and ending address of the memory locations to be saved using the following procedure:

1) Place tape to be recorded into tape recorder and rewind fully.

2) Using the Memory Examine mode, enter the starting address of the memory locations to be saved into 23C0H and 23C1H (high byte into 23C0H and low byte into 23C1H).

3) Using the Memory Examine mode, enter the address of the last RAM location to be saved into 23C2H and 23C3H (high byte to 23C2H and low byte to 23C3H).

4) Make sure the prompt symbol is being displayed and depress the CASS DUMP key, followed by turning the recorder on in the record mode. The prompt will disappear.

5) No volume adjustments are required as this is handled by the AGC of the recorder. When the Dump is completed, the prompt sign will reappear, indicating that the Dump is complete and that the recorder can be shut off. At least 30 seconds will be required for a Dump - see the following for more details on the recording format.

The format used to record data on the cassette tape adheres to two standards: the Kansas City Standard for recording "1's" and "0's" and the Intel Hex Format for recording blocks of data - both of these standards will be explained in the following paragraphs.

The Kansas City Standard was formulated on November 7 and 8 of 1975 at a symposium held in Kansas City, Mo. by BYTE Magazine. The purpose of this symposium was to standardize audio cassette recording techniques among the manufacturers of equipment being sold into the hobby market. The following list is a summary of the Kansas City Standard (to which the Z80 STARTER KIT adheres):

1) A mark (logical one) bit consists of eight cycles at a frequency of 2400Hz.
2) A space (logical zero) bit consists of four cycles at a frequency of 1200Hz.
3) A recorded character consists of a space as a start bit, seven or eight data bits, and two or more as stop bits. (The Z80 STARTER KIT uses a seven bit ASCII data character and one stop bit.)
4) The seven ASCII data bits are organized least significant bit first, most significant bit last.
5) There will be at least a 30 second leader and a 5 second trailer on all data blocks.
6) Data rate is 300 baud (3.33 mSec bit width).
7) The contents of a data block are not specified.

Because the Kansas City Standard does not specify the contents of the data blocks recorded, another standard - the Intel Hex Format - has been selected to define the organization of the data blocks. The following is a summary of the Intel Hex Format:

1) Each record within a block of data starts with a colon(:) and ends with a carriage return and line feed.

2) All information is in ASCII (seven bits no parity).

3) Data Record Format

   Byte 1 Colon(:) delimiter

   2-3 Number of binary bytes in this record. The maximum is 16 binary bytes (32 ASCII bytes).

   4-5 Most significant byte of the start address of the data.

   6-7 Least significant byte of the start address of the data.

   8-9 ASCII zeros

   10- Data bytes in ASCII

   Last two bytes - Checksum of all bytes except the delimiter, carriage return, and line feed. The checksum is the negative of the binary sum of all bytes in the record.

   Carriage return, Line feed

4) End-of-file Record

   Byte 1 Colon(:) delimiter

   2-3 ASCII zeros

   4-5 ASCII zeros

6-7	ASCII zeros
8-9	Record type 01 (ASCII 0, ASCII 1)
10-11	Checksum

3-12 CASSETTE LOAD

This mode of operation is used to load programs or information from cassette tape to RAM using the Kansas City Standard as the recording technique on the tape. To load a tape simply follow these steps:

1) Connect the recorder to the Z80 STARTER KIT using an audio patch cord to connect "MONITOR OUT" or "EARPHONE" to the connector marked "EAR" on the Z80 STARTER KIT.
2) Turn the recorder's tone control to maximum treble and minimum bass. Rewind the tape.
3) Turn the recorder's volume control to minimum volume.
4) Make sure the prompt is showing and depress the CASS LOAD key - the prompt will disappear.
5) Increase the volume until the LOAD LED just lights and then increase the volume control about 20% more. The LED should stay lit during the load.
6) If the load is successful (i.e. all checksums have been verified) ZBUG will respond with the prompt symbol and the recorder can be shut off.
7) If a checksum error is detected during loading, the address of the next block of data will be shown on the display. All data up to the previous block of

data had been loaded successfully. Try to load the tape again and verify the volume and tone control settings.

8) The LOAD LED can be used to index into several records on the same cassette, as it will light when data is present on the tape and go off during inter-record gaps. This feature will allow the user to put several programs on the same cassette tape.

3-13 EPROM PROGRAMMER

The EPROM Programmer moves data from RAM at address 2000H to a 2716/2758 five volt only EPROM in socket PROM2 (address 1000H). To program an EPROM requires an auxiliary power supply of +25±1 volts capable of supplying at least 30mA of current. This power supply should be connected to the Z80 STARTER KIT at the designated spot on the left hand edge of the board. The EPROM to be programmed should be erased and then placed in socket PROM2 with the power turned off. Turn on both the +5v and +25v supplies and then load the desired data into RAM using the Memory Examine mode or the Cassette Load mode. Make sure a prompt character is being displayed. If it is not, depress the MON key. Enter a four digit hex number (high digit first) representing the number of bytes to be transferred from RAM to PROM1. Place switch S2 in the PGM position and depress the PROM PROG key, which will cause the display to go dark. After programming, ZBUG will respond with one of two indications. The first and most likely indication is the return of the prompt character which indicates that the EPROM has been programmed and verified to be exactly like RAM. The second indication possible is a four digit address of the first location in EPROM that doesn't agree with RAM and the data in EPROM at that address. By pressing the NEXT key, ZBUG will continue checking the EPROM against RAM and will display the next EPROM address where the data doesn't match. This error indication is caused by an attempt to program an un-erased EPROM or an attempt to program a faulty EPROM. Return switch S2 to the READ position after programming is completed.

The EPROM programmer inserts Wait states of 52.5 mSec in duration (timing pulses generated by the Z80-CTC) and will suspend the Z80-CPU's refresh of dynamic memories during EPROM programming. This has no effect on the Z80 STARTER KIT but is mentioned should the user be experimenting with dynamic memories.

The top 110 bytes of RAM are used for system RAM (see Memory Map discussion - Section 3-18) and cannot be used to hold data to be transferred to the EPROM Programmer. Should an advanced user desire to remove this restriction, a new EPROM Programmer routine could be written based on ZBUG routine CCS12 (see listing in Appendix) and placed into EPROM in the PROM1 socket. By this means an advanced user of the Z80 STARTER KIT could modify the EPROM Programmer to move data from anywhere in RAM to EPROM in the PROM2 socket. The availability of an EPROM Programmer and inexpensive five volt only EPROMs allows an advanced user the capability to easily expand and enhance ZBUG to suit a particular need. (See example 5-6.)

3-14 NEXT KEY

The NEXT key is used in conjunction with three modes of operation: Memory Examine, Port Examine, and next EPROM Programmer error. In the Memory and Port Examine modes the NEXT key selects the next sequential memory or port location and automatically displays its contents. If there should be errors during the programming of an EPROM, the NEXT key can be used to step through the errors. (An error is where the contents of EPROM are different from the RAM locations providing the data - 2000H and above.) Errors during EPROM Programming are caused by an attempt to program an EPROM which hasn't been erased or by an attempt to program a defective EPROM.

3-15 RELATIVE OFFSET CALCULATION

One of the enhanced features of the Z80 Microprocessor is the Relative Addressing Mode. In this mode the next address is determined by the addition of a two's complement offset to the current address. The Relative Jumps (JR) use this addressing mode with a one byte Op Code followed by the two's complement offset. When programming at the machine level (no Assembler is used), the calculation of this offset requires the user to do addition and subtraction in hexadecimal, which can be error prone. In order to make the Relative Addressing Mode easy to use (as it should be), an automatic offset calculation routine has been included in the ZBUG Monitor. This routine will automatically calculate the correct offset for Relative Addressing and place it in the proper location in RAM.

To use this feature of ZBUG proceed as follows: First, using the Register Examine mode, initialize HL to the address of the Op Code at the destination of the Relative Jump (high byte to H). Second, set DE to the address of the Relative Op Code (high byte to D). Third, execute the Relative Offset routine at address 00C0H using the ZBUG's Execute command. Note the similarity between this address and the initial value of the Stack Pointer (23C0H); this was done to make both addresses easier to remember. This display will contain the offset that was calculated and placed into RAM in the low byte of the Address display and either 00 or FF in the high byte of the Address display. Should any other value appear in the high byte of the Address display, it is an indication that the Relative Offset was outside the Z80's legal range and therefore invalid.

3-16 RST INSTRUCTIONS

The Z80 instruction set has eight restart instruction addresses and an NMI vector address. The hardware reset address (0000H) is used as the entry point into ZBUG so that on power-up the Z80 automatically starts executing ZBUG. The NMI vector address (0066H) is used by the MON key's Abort function and provides a method to always gain control of the Z80 from the keyboard. One restart address (0008H) is used to provide the breakpoint capability and as such cannot be used. The other six restart instruction addresses (RST16, RST24, RST32, RST40, RST48, and RST56) are available for use. These one byte instructions, when executed, save the program counter on the stack and then vector to one of the following addresses: 0010H, 0018H, 0020H, 0028H, 0030H, 0038H - same order as above. Since these address locations are in the ZBUG Monitor ROM, jumps have been placed in ZBUG to jump to specific locations in RAM where the user can place another Jump to anywhere desired. The following table is a summary of the mapping to RAM of the free Restart instructions:

	Instruction	Op Code	ZBUG ROM Address	RAM address
- Power-up reset.	RST 0	C7H	0000H	---
Breakpoint.	RST 8	CFH	0008H	---
	RST 16	D7H	0010H	23C4H
	RST 24	DFH	0018H	23C7H
	RST 32	E7H	0020H	23CAH
	RST 40	EFH	0028H	23CDH
	RST 48	F7H	0030H	23D0H
	RST 56	FFH	0038H	23D3H

For example, if an RST 32 (E7H) instruction is used in a program, the Z80-CPU will first save the Program Counter and then go to address 0020H. This address is in the ZBUG Monitor ROM and contains a Jump instruction to RAM location 23CAH. The user may place any one to three byte instruction desired at this location, however, a three byte Jump would probably be used in most cases.

3-17 CTC CHANNEL ZERO INTERRUPTS

Within a user program both the Z80-PIO and Z80-CTC interrupt vectors can be set up as desired. However, should it be desired to use Channel zero of the CTC while the ZBUG Monitor is using the other three channels (Channel 1 is used for Cassette Dump timing, Channel 2 is used for EPROM Programmer timing, and Channel 3 is used for Cassette Load timing) the following provisions have been made. Since the four CTC Channel vectors are related (see Z80-CTC Technical Manual for further details) a provision has been made in ZBUG to map the Channel 0 interrupt into RAM where a Jump to the actual service routine can be placed. When setup by ZBUG, the CTC's Channel 0 will first go to a look up table at address 07F8H where 23D6H has been placed. 23D6H would be the address of the first instruction to be executed after Channel 0 interrupts and is in RAM so that the user can place a jump at this address to the interrupt service routine.

3-18 MEMORY MAP - RAM USAGE

The ZBUG Monitor resides in the bottom 2K of memory (addresses 0000-07FFH) and uses the top 110 bytes of RAM. The memory map of the Z80 STARTER KIT as shipped is defined by the following:

	2800H ↑	UNUSED
	27FFH	OPTIONAL RAM
	2400H	1K BYTES U16-U23
	23FFH	ZBUG SCRATCH RAM
		AND
23Cø	23C1H	BREAKPOINT TABLES
23BF	23C0H	USER'S REGISTER MAP
	23A9H	
	23A8H	ZBUG STACK
	2390H	WORKING AREA
	238FH	RAM AVAILABLE TO USER
	2000H	
	1FFFH	UNUSED
	1800H	
	17FFH	PROM PROGRAMMER
	1000H	PROM2 SOCKET (U34)
	0FFFH	PROM1 SOCKET (U33)
	0800H	
	07FFH	ZBUG MONITOR
	0000H	

The two EPROM sockets reside just above the ZBUG ROM at 0800H and 1000H. In addition, the EPROM socket at address 1000H has the capability of programming EPROMs. The standard RAM memory begins at 2000H and is available to the user up to 2390H. RAM above this value is used by the ZBUG Monitor. From address 2390H to 23A8H is the ZBUG stack working area where return addresses of subroutine calls and interrupts are saved as ZBUG is executed by the Z80-CPU. The User's Register Map resides from address 23A8H to 23C0H. Whenever control is switched to the ZBUG Monitor, the state of the Z80-CPU is saved in this area. The registers can be examined and changed by using the REG EXAM and REG EXAM' keys. Whenever execution is transferred from ZBUG to a user's program, register values from this map are loaded into the CPU. From 23C0H to 23FFH are scratch RAM Variables and a breakpoint table containing the addresses and Op Codes of active breakpoints.

3-19 ZBUG COMMAND SUMMARY

DUMP - Punch to audio cassette in Kansas City Standard Format. Memory block starting address at 23C0H and 23C1H, ending address at 23C2H and 23C3H.

LOAD - Load Kansas City Standard formatted audio cassette tape to memory.

REGISTER DISPLAY - Press key for desired register and then either REG EXAM or REG EXAM' for display of that register on the hex display. Change the register by entering new data.

MEMORY EXAMINE - Press keys for desired memory address (four digits required) and then the MEM EXAM key. The memory data will be displayed in the right hand two digits of the display. By entering two new digits, the memory location may be changed.

PORT EXAMINE - Enter a two digit port number followed by the

PORT EXAM key to display the port data in the right hand two digits of the display. By entering two new digits, the data at that port address will be changed.

BREAKPOINT - Enter a four digit address where the breakpoint is desired, followed by the BREAKPOINT key. Up to five breakpoints are allowed; if the address remains on the display after the BREAKPOINT key is pressed, then the breakpoint was installed. If the breakpoint was not installed, the display address will clear after the BREAKPOINT key is pressed. Clear all breakpoints by pressing BREAKPOINT key with no digit entry or by using single step.

SINGLE STEP - Initialize the Program Counter with a four digit address and press the SINGLE STEP key. The instruction at that address will be executed (after the CPU registers are restored for the register map) and the program counter plus the accumulator will be returned in the display. If no address is entered before the SINGLE STEP key is pressed, the instruction pointed to by the program counter in the register map will be executed. Repeated pressing of the SINGLE STEP key will single step down a user's program.

MONITOR - Pressing this key will force a restart of the ZBUG Monitor through an NMI interrupt and save all CPU registers. This is useful for getting the processor "out of the weeds".

EXECUTE - Entering a four digit address followed by the EXEC key will cause the registers in the register map to be loaded into the CPU. The CPU will start executing the user's program at the entered address. If no address is entered, the CPU will start execution at the address pointed to be the program counter in the register map.

PROGRAM - Enter a four digit hex number to indicate the number of bytes of memory to be moved from RAM (starting at address 2000H) to EPROM (starting at address 1000H). Pressing the PROM PROG key will initiate the transfer. 2716 or 2758 type EPROMs can be programmed.

NEXT - Opens next memory or port location for examination or change.

HEX ARITHMETIC - A hexadecimal arithmetic routine allows easy calculation of the relative offset required for relative jump instructions. Load HL with the address of the Op Code at the destination of the relative jump. Load DE with the address of the relative Op Code. Execute hex arithmetic routine at 00C0H and the hex result will appear in the display.

3-20 SUBROUTINES CALLABLE IN ZBUG

Several general purpose subroutines were written to be used by the different functions of ZBUG. These programs are in the UTILITY section of the ZBUG listing in the Appendix. Several of these subroutines are listed below; an advanced user of the Z80 STARTER KIT can use these subroutines to simplify his programming task. For the inexperienced user, these subroutines provide programming examples which can be analized to learn Z80 programming.

1) UIX3 - Calling address is 0634H. This program adds three to the Index Register (IX) and decrements the B register. Registers affected are IX, B, and F.

2) UFOR1 - Calling address is 063CH. IX points at two locations in memory and A contains two Hex digits to be written into memory (high nibble to (IX), low nibble to (IX+1)). Registers affected are A, B, and F.

3) D20MS - Calling address is 064FH. This subroutine delays 20 mSec before returning to the caller. Registers affected are H, L, and F.

4) UABIN - Calling address is 06B3H. Converts one ASCII character to its equivalent binary value. Registers used: A and F. ASCII character is in A upon calling and binary equivalent is also in A upon return.

5) UBASC - Calling address is 06BBH. Converts one binary character in Accumulator to its equivalent ASCII character in the Accumulator. Registers used are A and F.

SECTION 4

HARDWARE DESCRIPTION

4-1 GENERAL

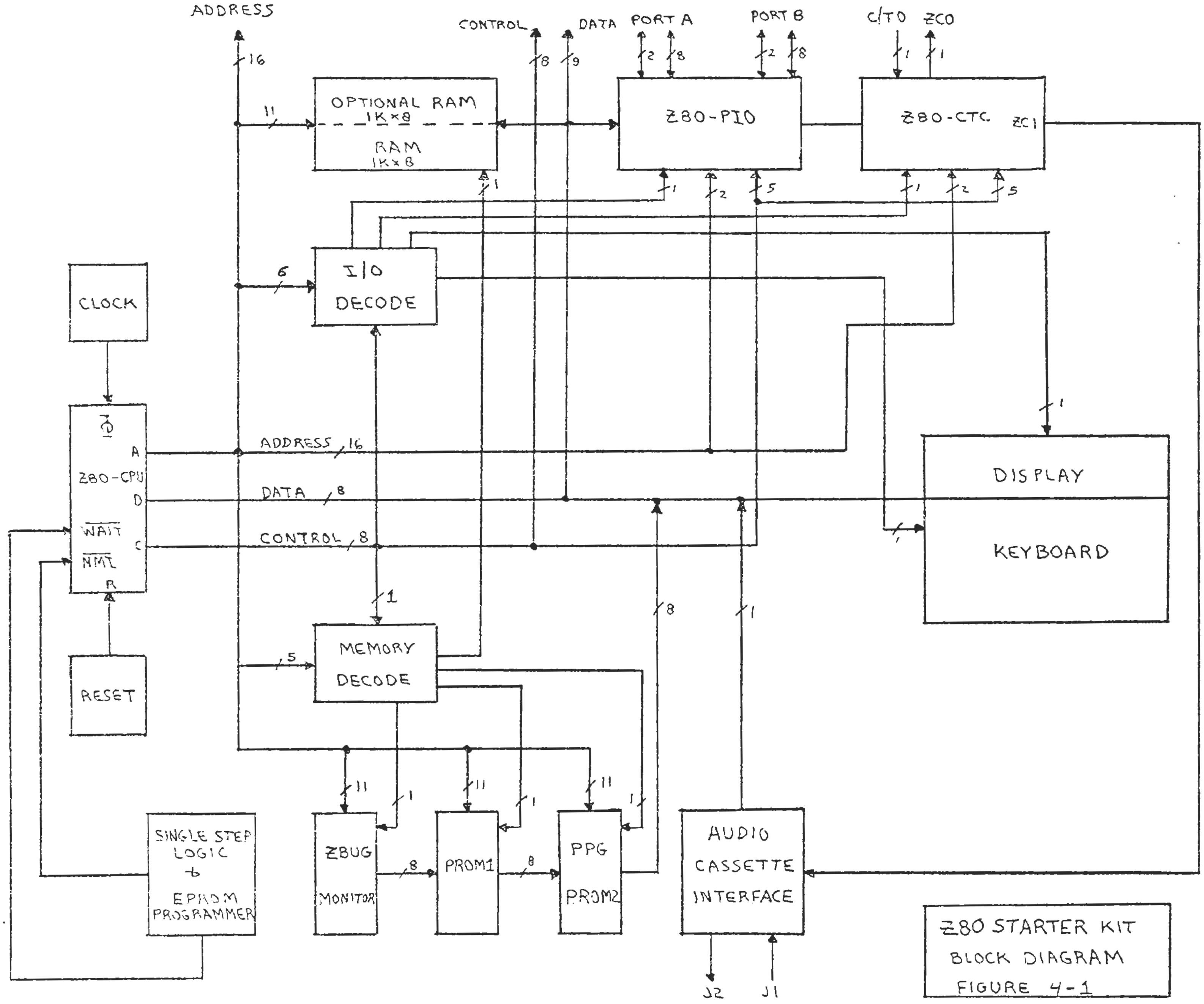
Figure 4-1 is a block diagram of the Z80 STARTER KIT. Refer to this diagram and the schematic diagram in Appendix I during the following discussion.

4-2 CLOCK CIRCUITRY

The clock circuitry is set to run the Z80-CPU at slightly below 2 MHz for a 500 nSec T State. This was done to insure maximum compatibility with 8080A peripherals and to allow the use of inexpensive memories. The clock source is a crystal oscillator based on a 74LS04 (U43) running at 3.9936 MHz. This frequency was chosen because it is a multiple of both 1200/2400 Hz and the 300 Baud frequencies required for the Kansas City Standard audio cassette interface. This frequency is divided by two by U41 to form the 1.9968 MHz CPU clock. A 74LS04 gate with a 330 ohm pull-up resistor is used to drive the Φ inputs of all three Z80 devices.

4-3 Z80-CPU

The Z80-CPU is the "brains" behind the Z80 STARTER KIT and provides the major control signals to scan the display and keyboard as well as reading and writing memory. The Z80-CPU generates a 16-bit address bus, an 8-bit bi-directional data bus, and 8 control signals. These signals are all routed



Z80 STARTER KIT
BLOCK DIAGRAM
FIGURE 4-1

to the wire wrap area and marked so that it will be easy for experimenters to add circuitry to the CPU bus. More information on this device can be found in the MOSTEK or Zilog Z80-CPU Technical Manual.

4-4 Z80-PIO

The Z80-PIO is a generalized parallel interface for the Z80 family. It supports fully interrupt driven software with two sets of handshake lines and an integral interrupt controller. Two 8-bit ports plus handshake lines are available on the PIO for interfacing parallel devices. These lines are brought to the wire wrap area for connection to custom circuitry. More information on the PIO can be found in the MOSTEK or Zilog Z80-PIO Technical Manual.

4-5 Z80-CTC

The Z80-CTC is a device which contains four independent 16-bit counters which may be used to divide down the Φ clock, or as an event counter. The ZBUG Monitor heavily uses the CTC to implement Z80 STARTER KIT functions. CTC Ch. 0 is unused by ZBUG and is always available to the user. During ZBUG routines other CTC channels are used as follows:

CTC Ch. 1	Audio Cassette during Dump
CTC Ch. 2	Single Step and EPROM Programmer
CTC Ch. 3	Audio Cassette during Load

See Section 3-17 for a description of how to use CTC Ch. 0 while ZBUG is using the other channels. Whenever one of the

Channel counters counts through zero, a pulse is produced on the Zero Count Output line (ZCO). An input to each channel is the Clock/Trigger (C/TO) which can be used to initiate timing or as an external clock. Both of these signals for Channel 0 are brought to the wire wrap area. More information on the CTC can be found in the MOSTEK or Zilog Z80-CTC Technical Manual.

4-6 KEYBOARD AND DISPLAY

The Hexidecimal display is scanned by the Z80-CPU under control of ZBUG. Data for the display is written to U12 (port address 88H), while the active display is selected by U11 (port address 8CH). Each display is left on for about 1 mSec and then new data is supplied to U12 and the next digit selected by U11. Q1-Q7 provide high current drive capability while U4, U5, and U6 provide high current sinking capability for digit selection. While U11 is scanning the display, it also scans the keyboard. U13 (port address 90H) is the keyboard input to the Z80-CPU data bus. As U11 scans the keyboard, data is input from U13 to determine if a key is closed. S3 (MONITOR/PROM1 RESTART) is sampled by U13 during the Monitor Reset sequence to determine if program execution should be directed to PROM1 or ZBUG.

4-7 AUDIO CASSETTE INTERFACE

The Z80 STARTER KIT has a Kansas City Standard audio cassette interface. The data rate is 300 Baud and a "1" is represented by a 2400Hz tone while a "0" is represented by a 1200Hz tone. There is a 30 second leader and a 5 second trailer of "1"'s on all records.

During a Load from an audio cassette tape player, the data from the player's Earphone jack is connected to J1. U7 is a combination limiting and squaring circuit to provide a non-distorted square wave to U15. Data amplitude from the player must be about 2 volts peak to peak and part of U7 is an LED driver to indicate when correct amplitude data is being received by U15. One-half of U15 and U14 form a frequency detector to discriminate between 1200Hz ("0") and 2400Hz ("1"). Pin 12 of U14 contains the demodulated data stream (similar to Asynchronous data used in data communications) and is gated onto the Z80-CPU bus by U13. The Z80-CPU and Z80-CTC under control of ZBUG become a software UART to receive this serial Asynchronous data and form parallel words which are then written to memory.

During a Dump of data from RAM to the audio cassette recorder, the CTC Channel 1 is set to generate either a 4800Hz pulse train ("1") on CTC-ZC1 or a 2400Hz pulse train ("0"). These two pulse trains are divided in half to form the proper square wave frequencies by one-half of U14. R31 and C16 filter out the high frequency components of this square wave to prevent distortion of the data by the tape recorder. J2 is normally connected to the C16 side of R34 and then to the Auxillary Input of the recorder. The crystal frequency of the system clock oscillator has been selected to be a multiple of

1200Hz (and 2400Hz) so that the CTC can generate these frequencies by simply dividing down the system clock ($\overline{\Phi}$).

4-8 EPROM PROGRAMMER

In order to program 2758/2716 5 volt only EPROMs, correct address and data is applied to the EPROM, the Vpp pin is placed at +25 VDC, $\overline{\text{CS}}$ = 1, and PD/PGM is pulsed high for 50-55 mSec. This is repeated for any address that is to be programmed. (Note: Texas Instruments has a three voltage 2Kx8 EPROM, also called the 2716, which will not work with this kit - the EPROMs must be 5 volt only.)

The technique used in the Z80 STARTER KIT to provide the necessary programming signals is described in the following: A Z80-CPU block move instruction (LDI) is used to move the data from locations in RAM to the EPROM Programming Socket (1000-17FFH). The Z80-CTC Channel 2 is set up to time out every 26 mSec. U42 and one-half of U41 form a synchronous counter clocked by the ZC2 output to time two outputs or 52 mSec. One-fourth of U45 decodes states of this counter to produce the 52 mSec positive pulse required by the EPROMs. This output is also inverted and applied to the $\overline{\text{WAIT}}$ input of the Z80-CPU to force the CPU to hold valid data and addresses during this 52 mSec pulse. The sequence of events to program one location is the following:

1) Synchronous counter enabled by PGM PULSE ENABLE being set to a "1".

2) CTC Channel 2 is set to time out after 26 mSec delay.

3) LDI instruction starts to write into address space decoded by PROM2 (1000-17FFH) causing $\overline{\text{PROM2 SEL}}$ to go low. Address and data are now valid on the PROM2 socket.

4) $\overline{\text{PROM2 SEL}}$ going low clocks one-half of U42 which makes PROM2 $\overline{\text{CS}}$ go high, PD/PGM goes high and Z80-CPU $\overline{\text{WAIT}}$ goes low.

5) Z80-CPU stays suspended in wait state until ZC2 times out twice (52 mSec), causing the timing chain to advance setting PD/PGM low and $\overline{\text{WAIT}}$ high, thereby releasing the Z80-CPU from the wait state. PGM PULSE ENABLE is set low, resetting the timing chain.

Note that when using the EPROM Programmer the Z80-CPU suspends memory refresh during this 52 mSec period, which may affect any dynamic memory circuitry being used with the KIT. The 21L02 memory used on the KIT is static, so this method of implementing the EPROM Programmer does not cause a problem.

Both 2758 and 2716 EPROMs can be programmed without any hardware modifications subject to the restrictions discussed in 3-13. This is because the 2758s available at the writing of this manual specify that A10 should be low (i.e. they were the lower half of a 2716 2Kx8 EPROM). Should 2758s be available with A10 specified as a "1", a jumper provision has been provided to allow strapping the A10 pin of sockets PROM2 (U34) and PROM1 (U33) either high, low, or to A10 of the Z80-CPU. (See schematic.) As shipped, the KIT has the CPU's A10 wired to U33 and U34 so that 2716s will work. 2758s will also work in this configuration if they require A10 tied low.

4-9 MEMORY DECODING

Memory decoding is done by U35, which is a 74LS138 1 of 8 decoder. The lower 16K bytes (0000H-3FFFH) of the Z80's address space is fully decoded into 2K byte blocks. The following tables specify this memory decoding.

Chip Select	Memory Address Space	Connected to
$\overline{\text{CS0}}$	0000H-07FFH	ZBUG MONITOR
$\overline{\text{CS1}}$	0800H-0FFFH	PROM1
$\overline{\text{CS2}}$	1000H-17FFH	PPG-PROM2
$\overline{\text{CS3}}$	1800H-1FFFH	UNUSED
$\overline{\text{CS4}}$	2000H-27FFH	RAM
$\overline{\text{CS5}}$	2800H-2FFFH	UNUSED
$\overline{\text{CS6}}$	3000H-37FFH	UNUSED
$\overline{\text{CS7}}$	3800H-3FFFH	UNUSED

All outputs of the decoder are connected through a 16-pin hole pattern (U38) compatible with a 16-pin socket and header allowing the user to easily modify addressing. Connections to U38 are as follows:

$\overline{\text{CS0}}$ ——9o————o8—— $\overline{\text{MON SEL}}$ (ZBUG MONITOR)
$\overline{\text{CS1}}$ ——10o————o7—— $\overline{\text{PROM1 SEL}}$ (PROM1-U33)
$\overline{\text{CS2}}$ ——11o————o6—— $\overline{\text{PROM2 SEL}}$ (PROM2-U34)

$\overline{CS3}$	12	3	$\overline{MCS3}$	(TO WIRE WRAP AREA)
$\overline{CS4}$	13	4	$\overline{RAM\ SEL}$	(TO RAMs)
$\overline{CS5}$	15	2	$\overline{MCS5}$	
$\overline{CS6}$	16	1	$\overline{MCS6}$	(TO WIRE WRAP AREA)
$\overline{CS7}$	14	3	$\overline{MCS7}$	

16-pin hole pattern-jumpers in pc etch

By cutting the etch on the pc board and installing a 16-pin socket and header at U38, memory addressing can be modified; however, if changes to $\overline{MON\ SEL}$, $\overline{PROM2\ SEL}$, or $\overline{RAM\ SEL}$ are made, then the ZBUG Monitor will not function correctly. $\overline{MCS3}$, $\overline{MCS5}$, $\overline{MCS6}$, and $\overline{MCS7}$ are all brought out to the wire wrap area so that they can be used to select custom circuitry built in the wire wrap area.

4-10 PORT DECODING

I/O Ports are completely decoded into blocks of four by the 1 of 8 decoder at U47. The Z80-PIO and the Z80-CTC further decode the four address blocks into unique addresses. The following table shows this Port address decoding:

Port Select	Port Address Space	Connected to
$\overline{PS0}$	80H-83H	Z80-PIO
$\overline{PS1}$	84H-87H	Z80-CTC
$\overline{PS2}$	88H-8BH	$\overline{SEG\ LATCH}$
$\overline{PS3}$	8CH-8FH	$\overline{DIGIT\ LATCH}$

$\overline{\text{PS4}}$	90H-93H	$\overline{\text{KB SEL}}$	
$\overline{\text{PS5}}$	94H-97H	UNUSED	2nd PIO
$\overline{\text{PS6}}$	98H-9BH	UNUSED	
$\overline{\text{PS7}}$	9CH-9FH	UNUSED	

$\overline{\text{PS5}}$, $\overline{\text{PS6}}$, and $\overline{\text{PS7}}$ are routed to the wire wrap area to be used as I/O decodes for custom circuitry. The following table is a further breakout of the addresses assigned to registers with the PIO and the CTC.

	PORT ADDRESS	USED BY	
94	80H	A Data Register	PIO
95	81H	B Data Register	
96	82H	A Control Register	
97	83H	B ~~Data~~ Control. Register	
	84H	Channel 0	CTC
	85H	Channel 1	
	86H	Channel 2	
	87H	Channel 3	

4-11 SYSTEM RAM

1024 bytes of 21L02-1 RAM located at U24-31 are standard with the KIT. Provision has been made to add an additional 1K bytes of RAM in locations U16-U23 with eight additional 21L02-1 (500 nSec). The use of part of the standard RAM for ZBUG scratch and stack is detailed in Section 3-18. Data from the RAM is gated onto the Z80-CPU bus using a 74LS244 Hex buffer (U48).

4-12 SINGLE STEP LOGIC

The Z80 STARTER KIT has a "hardware single step" which means that the Single Step command operates on programs located either in RAM or EPROM/ROM. Channel 2 of the CTC is used to produce a pulse at the beginning of the first user's instruction after Single Step is commanded. This pulse (ZC2) is routed through gates U44 and U45 to the Non-Maskable Interrupt ($\overline{\text{NMI}}$) input of Z80-CPU. The $\overline{\text{NMI}}$ input is always recognized at the end of the current instruction, and vectors the CPU to address 66H, which is in the ZBUG Monitor. ZBUG preserves all CPU registers and displays the current program counter location and Accumulator contents on the display.

4-13 PROM1 RESTART

When a RESET is applied to the Z80-CPU, the ZBUG Monitor checks the position of S3. If S3 is set to MONITOR RESTART, then ZBUG will place the prompt symbol on the display and scan the keyboard for a command. If S3 is in the PROM1 RESTART position, then the ZBUG Monitor automatically vectors the program execution after Reset to PROM1 at address 0800H. This feature provides a means to Restart to a user's program without having to enter commands through the keyboard.

4-14 INTERRUPT DAISY CHAIN

All Z80 family peripheral devices have built-in interrupt control circuitry, so that no dedicated interrupt controller is required. Priority of interrupts is determined by a daisy chain running between Z80 peripheral devices (IEI-input, IEO-output). On the Z80 STARTER KIT the CTC has been given highest priority, with the PIO next in line. The output of the daisy chain is brought to the wire wrap area (marked IEO) and can be used to continue the daisy chain should additional Z80 peripherals be added. Refer to Z80 Technical Manuals for additional information on the daisy chain interrupt structure.

4-15 S-100 BUS INTERFACE

Provision has been made on the KIT to add two 100 pin connectors, which have been pre-wired to a S-100 configuration. This interface is compatible with general static memory or I/O expansion cards. Specifically this interface is directly compatible with the S.D. Systems' 4K byte Static RAM cards, but not with the EXPANDORAM modules. Interface with modules that require specific 8080A signals such as SYNC, INTA, DBIN, POC, PWR, and PRD may require addition of some logic to the wire wrap area and/or the wiring of additional signals to the S-100 connectors. Refer to the KIT schematic for the exact connections to these S-100 connectors. Power (+8v, ±18v) can be connected to the S-100 connectors via the appropriate pads at the top of the KIT.

4-16 WIRE WRAP AREA

The wire wrap area has room for about 25-30 additional IC's for experimentation, memory expansion, video interface, etc. Power and ground are alternated on the bottom side so that each IC is close to a low impedance power source and ground, thereby reducing noise problems in user's circuitry. Z80-CPU, PIO, and decoded system signals have been placed near the wire wrap area so that by installing wire wrap pins in the holes provided, it will be easy to connect user's circuitry to the Z80 bus signals.

SECTION 5

EXAMPLE PROGRAMS

5-1 USING ZBUG COMMANDS

In the examples that follow, all user's key entries are underlined, while ZBUG responses are not underlined. ZBUG Commands will be used to execute the following program:

```
                  ORG  2000H
     2000   3E AA LD   A,0AAH  ;load A with AA
     2002   06 BB LD   B,0BBH  ;load B with BB
     2004   76    HALT         ;Z80-CPU HALT
```

Display/Keyboard	Explanation
-	Turn on Z80 STARTER KIT, press RESET - S1
2000 MEM EXAM	Open location 2000 with MEM EXAM key - location has random value XX.
2000 XX	
2000 XX 3E NEXT	Enter program using NEXT key to advance to next memory location.
2001 XX AA NEXT	
2002 XX 06 NEXT	
2003 XX BB NEXT	
2004 XX 76 NEXT MON	

- A REG EXAM A XX MON	Examine register A's contents which are undefined.
- B REG EXAM b XX MON	Examine register B's contents which are undefined.
- 2002 BREAKPOINT MON	Set Breakpoint at location 2002.
- 2000 EXEC	Execute program starting at address 2000.
2002 AA	ZBUG responds with address of next instruction and value of Accumulator - Note effect of first instruction.
2002 AA SINGLE STEP	Command execution of one instruction at 2002.
2004 AA MON	Next instruction is at 2004, cancel Single Step Mode with MON.
- B REG EXAM	
b bb MON	Examine B register, it now contains bb due to execution of second instruction.

	Cancel Register Examine Mode with MON.
- PC REG EXAM	Examine current Program Counter value.
1 20 04 MON	PC is at 2004 which is the HALT instruction. Cancel mode with MON.
- 2000 EXEC	Start from beginning of program and execute.
Display dark MON	Program runs down to HALT instruction - Z80-CPU is not in ZBUG Monitor, so display is dark. The one breakpoint was removed by the Single Step operation. Press MON key to abort user's program and regain control.
- 1 REG EXAM 1 2005 MON	Look at PC, it is now pointing to the instruction after the HALT which is outside the program. Cancel mode.

5-2 CALCULATING RELATIVE ADDRESS

The following program requires the calculation of a Relative Address (2's complement Hexadecimal subtraction).

```
                              ORG  2000H
2000   3E 00                  LD   A,00H
2002   06 05                  LD   B,05H
2004   3C     LOOP:           INC  A
2005   10 _                   DJNZ LOOP-$
2007   76                     HALT
```

```
     -     2000 MEM EXAM     Enter the program.
     2000   XX  3E   NEXT
     2001   XX  00   NEXT
     2002   XX  06   NEXT
     2003   XX  05   NEXT
     2004   XX  3C   NEXT
     2005   XX  10   NEXT
     2006   XX  NEXT         Do not enter 2006 as this is
     2007   XX  76   MON          the Relative Offset to
                                  be calculated.
```

```
-   H   REG EXAM
7   XX   20   MON            Destination Op Code is at
                                  2004 which is loaded
                                  into HL.

-   L   REG EXAM
8   XX   04   MON
```

```
-  D  REG EXAM                    Relative Op Code (DJNZ) is
d  XX  20 MON                          at 2005 which is loaded
                                       into DE.

E  XX  05 MON

-  00C0 EXEC                      Execute Relative address
                                       calculation program at
                                       00C0H.

FF  Fd  MON                       FF indicates a valid offset
                                       and Fd is the offset
                                       value.

-  2006  MEM EXAM                 Examine 2006 to see if offset
2006  Fd  MON                          got placed into memory -
                                       it did!  Cancel with MON.

-  1  REG EXAM                    Set Program Counter to begin-
1  XXXX  2000  MON                     ning of program.

-  SINGLE STEP                    Single Step down program.

2002  00  SINGLE STEP
2004  00  SINGLE STEP
2005  01  SINGLE STEP             First INC A
2004  01  SINGLE STEP             Loop back
```

```
2005   02   SINGLE STEP          Second INC A

2004   02   SINGLE STEP          Loop back

2005   03   SINGLE STEP          Third INC A

2004   03   SINGLE STEP          Loop back

2005   04   SINGLE STEP          Fourth INC A

2004   04   SINGLE STEP          Loop back

2005   05   SINGLE STEP MON      Fifth INC A

-   B   REG EXAM

b   01   MON                     B has been decremented down
                                      to 1.

-   SINGLE STEP                  Decrement B and Jump if not
2007   05   MON                       zero - B is now zero so
                                      we fell through to next
                                      instruction at 2007.

-                                ZBUG waits for new command.
```

DELAY   (H) x 2.08ms

```
Ø64F   21,FF,Ø8   ØØ69 D20MS:   LD    HL,Ø8FFH
Ø652   2D,        ØØ7Ø D20MS1:  DEC   L
Ø653   2Ø,FD      ØØ71          JR    NZ,D20MS1-$
Ø655   25,        ØØ72          DEC   H
Ø656   2Ø,FA      ØØ73          JR    NZ,D20MS1-$
Ø658   C9         ØØ74          RET
```

H = Ø8   DMS = 16.4 ms.

5-3 WRITING TO THE DISPLAY

The following program causes the character "8" to move from right to left across the display.

```
2000  3E 00                 LD    A,00H
2002  D3 88                 OUT   (88H),A   ;ACTIVATE ALL SEGMENTS
2004  3E 01                 LD    A,01H
2006  D3 8C           LOOP: OUT   (8CH),A   ;SELECT FIRST DIGIT
2008  CD 4F  06             CALL  D20MS
200B  CD 4F  06             CALL  D20MS
200E  CD 4F  06             CALL  D20MS
2011  CD 4F  06             CALL  D20MS
2014  CD 4F  06             CALL  D20MS     ;DELAY APPROX 100MS
2017  07                    RLCA            ;ROTATE TO NEXT DIGIT
2018  18 EC                 JR    LOOP-$    ;LOOP BACK
```

Enter the program and Execute address 2000. Use MON key to return to ZBUG. This is not a good program to Single Step through, as the Single Step routines use the display which destroys the "8" character loaded in the first two instructions.

5-4 INTERRUPT DRIVEN DELAY

The following program uses Channel zero of the Z80-CTC to interrupt after a fixed delay, rather than the software timing loop (D20MS) used in the previous example. This program uses the Z80 Mode 2 interrupts in which the interrupting device sends in a vector during the Interrupt Acknowledge cycle. The vector is used with the CPU's I register to form a pointer to a table which contains the address of the interrupt service routine. Refer to the Z80-CPU and the Z80-CTC Technical Manuals for a complete description of Z80 interrupts.

Load the program and double check that it is entered correctly. Follow these steps to test the program:

1) Load PC with 2000 and Single Step Down the program. Stop when you get to address 2200 (the interrupt service routine).
2) Single Step a few more times and notice that the Accumulator is rotated left every time address 2201 is executed (remember the address display is the NEXT instruction to be executed).
3) Stop Single Stepping with address 2201 showing. Hit the MON key and look at the Stack Pointer. (SP followed by REG EXAM.) It is at 22FE decremented two from the initial value of 2300. Check the contents of 22FE and 22FF, where you will find 18 and 20, which is the return address.
4) Hit MON and then Single Step until 2018 shows on the display. Now look at the Stack Pointer again - it is now 2300, as we have executed the RET1 instruction and recovered the return address address from the stack.
5) Set a BREAKPOINT at 2200 and Execute from 2000. Continually hit the EXEC key and note that the

Accumulator is rotated left once each time the breakpoint is encountered. Reset using S1 to clear CTC.

```
;INTERRUPT DRIVEN DELAY
                    ORG  2000H
2000 3E 21          LD   A,21H
2002 ED 47          LD   I,A       ;INITIALIZE I=21
2004 31 00 23       LD   SP,2300H ;INITIALIZE STACK POINTER
2007 3E 00          LD   A,00H
2009 D3 84      OUT ~~LD~~ (84H),A   ;CTC VECTOR
200B 3E A5          LD   A,0A5H
200D D3 84          OUT  (84H),A   ;CONFIGURE CTC-SEE PAGE 26
                                   ;OF MICRO REFERENCE MANUAL
200F 3E FF          LD   A,0FFH
2011 D3 84          OUT  (84H),A
2013 3E 01          LD   A,01H
2015 ED 5E          IM   2
2017 FB       LOOP: EI
2018 76             HALT
2019 C3 17 20       JP   LOOP
;TABLE OF INTERRUPT SERVICE ROUTINES
2100 00 22
;INTERRUPT SERVICE ROUTINE
2200 FB             EI             ;ENABLE INTERRUPTS
2201 07             RLCA           ;ROTATE ACCUMULATOR
2202 ED 4D          RETI           ;RETURN FROM INTERRUPT CLEAR
                                   ;CTC
```

5-5 GENERATE AN INTERRUPT FROM THE PIO

Place a 10K ohm pull-up resistor from $\overline{\text{ASTRB}}$ (marked near wire wrap area) to +5 volts. Enter the following program that initializes the PIO to accept an interrupt on the A Port. Set a BREAKPOINT at 2200 and Execute from 2000. The display will go dark as the Z80-CPU has executed the HALT instruction. Take a clip lead and momentarily touch $\overline{\text{ASTRB}}$ to GND, which will cause a PIO interrupt and the breakpoint should be displayed. For more details on the Z80-PIO operation, see the MOSTEK or Zilog PIO Technical Manual.

```
;PIO INTERRUPT TEST
     2000 3E 21    LD    A,21H
     2002 ED 47    LD    I,A         ;INITIALIZE I
     2004 3E 00    LD    A,00
     2006 D3 82    OUT   (82H),A     ;VECTOR
     2008 3E 4F    LD    A,4FH       ;SET UP PIO FOR INPUT
     200A D3 82    OUT   (82H),A     ;MODE - SEE PAGE 25 OF MI-
     200C 3E 87    LD    A,87H       ;CRO REF MANUAL MODE TWO
     200E D3 82    OUT   (82H),A     ;INTERRUPTS
     2010 ED 5E    IM    2
     2012 FB       EI
     2013 76       HALT

     ;INTERRUPT SERVICE ROUTINE TABLE
     2100 00 22
```

```
;INTERRUPT SERVICE ROUTINE
2200 FB      EI
2201 ED 4D   RETI              ;RETURN AND CLEAR PIO
```

5-6 USING THE Z80 STARTER KIT AS AN EPROM PROGRAMMER

Due to limitations in the memory allotted for the ZBUG Monitor, (it had to fit in a 2K byte ROM) some restrictions were placed on the operation of the EPROM programmer. These restrictions are fully documented in Section 3-13 and should not hinder the average user of the Z80 STARTER KIT.

Should a user desire to program the entire contents of a 2758 (1024 bytes) or a 2716 (2048 bytes) EPROM, the above mentioned restrictions need to be removed. The following two example programs give the user the capability of copying any block of memory to any other block of memory, plus allowing any RAM location to be the source of data for the EPROM programmer. These programs can be put on cassette tape to be loaded into RAM when needed, or they can be put into EPROM in the PROM1 socket for on-line use.

The first program is a copy utility based on the Z80 block move instruction. It can be used to copy a block of data from any memory location (source data) to any new memory location (destination data). It can also be used to copy the data from an EPROM in either socket PROM1 or PROM2 into RAM for minor modification before re-programming. To use the following program initialize the Z80 registers using the REG EXAM key as follows:

HL = Address of source data - high byte in H

DE = Address of destination data - high byte in D

BC = Number of bytes to be transferred from source to destination - high byte in B

Execute the following program to move the data:

```
                   ;COPY UTILITY
     2050 ED B0          LDIR         ;BLOCK MOVE
     2052 C3 AE 00       JP   RESTR1 ;RETURN TO ZBUG
```

The second program modifies the ZBUG PROM Programmer to allow source data to come from any memory location. This feature allows the user to use the standard RAM, add-on RAM, or another EPROM as the source of data for the EPROM being programmed. (This last transfer would be used for copying EPROMs.) To use this program press Reset (S1) first, then initialize 23C0H, 23C1H, 23C2H, 23C3H, and HL as follows:

23C0H - High byte of the address of source data to be programmed.

23C1H - Low byte of the address of the source data to be programmed.

23C2H - High byte of the address of the first byte in PROM to be programmed.

23C3H - Low byte of the address of the first byte in PROM to be programmed.

HL - Number of bytes +1 to be programmed in hex (high byte in H). Use REG EXAM mode to initialize HL.

Because the 2758/2716 EPROMs can be programmed a block or section at a time, the capability is provided by the initialization of 23C2H and 23C3H to start the programming at any address in the EPROM. This feature can be used to program a full 2K byte EPROM with less than 1K of RAM in the standard kit by programming the EPROM a section at a time.

```
                    ;PROGRAM TO MOVE ANY RAM BLOCK TO ANY
                    ;STARTING ADDRESS IN EPROM
2000 3E 01     C12:  LD   A,01H
2002 32 DA 23        LD   (PRFLG),A ;SET PROM PROG FLG
2005 E5              PUSH HL        ;BYTE COUNT IN HL
2006 C1              POP  BC        ;SAVE IT
2007 E5              PUSH HL
2008 3A C0 23        LD   A,(23C0H) ;SOURCE DATA
200B 67              LD   H,A
200C 3A C1 23        LD   A,(23C1H)
200F 6F              LD   L,A
2010 3A C2 23        LD   A,(23C2H) ;DESTINATION DATA
2013 57              LD   D,A
2014 3A C3 23        LD   A,(23C3H)
2017 5F              LD   E,A
2018 3E 25     C12A; LD   A,25H     ;CTC FOR 26 MS
201A D3 86           OUT  (86H),A   ;ZC/TO, NO INTR
201C 3E CB           LD   A,203D
201E D3 86           OUT  (86H),A   ;TIME CONST
2020 3E 80           LD   A,80H     ;CLEAR DISPLAY, SET
```

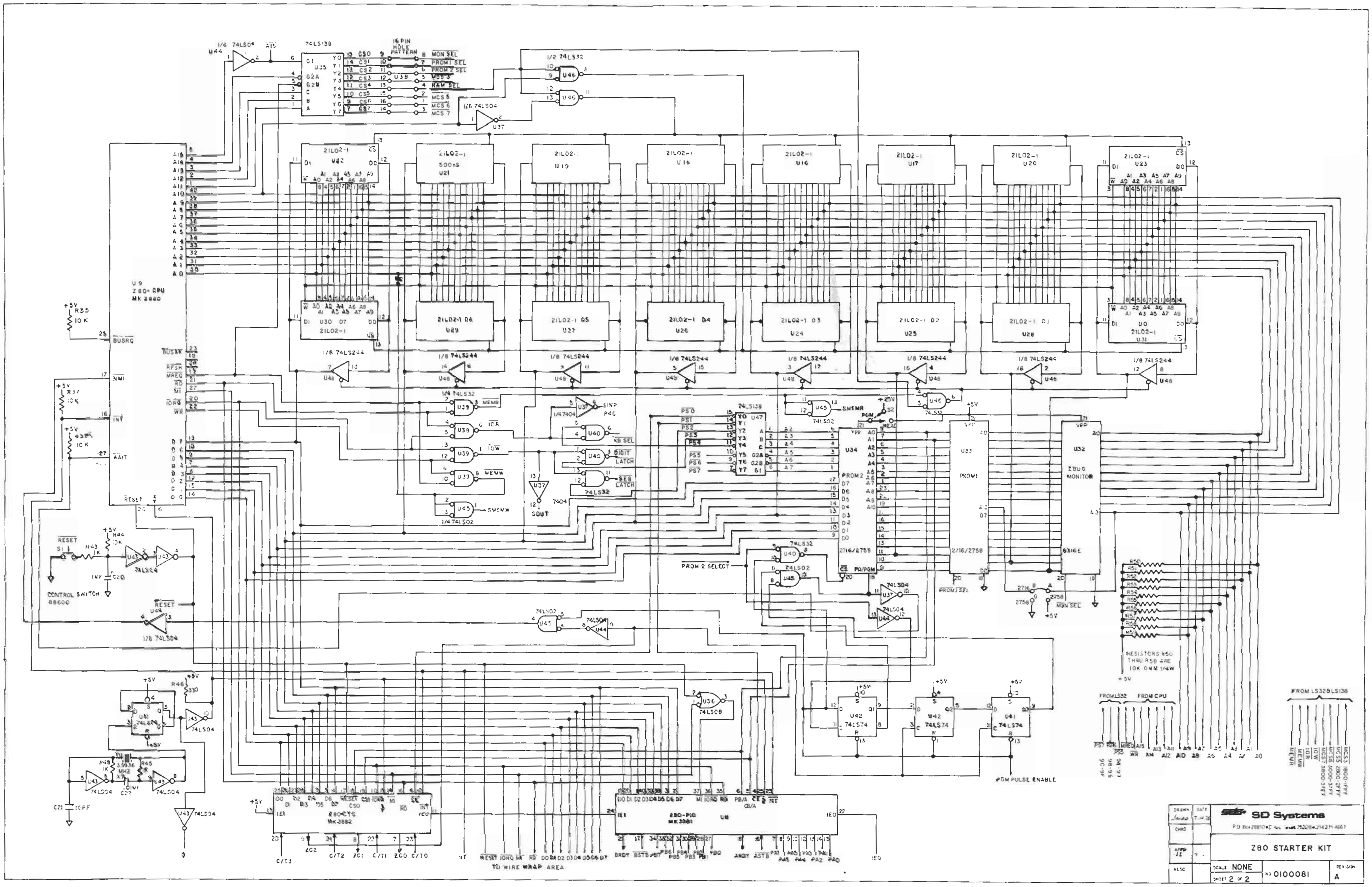
```
2022 D3 8C          OUT  (DIGLH),A ;PROM PROG EN = 1
2024 ED A0          LDI            ;WAIT STATE INSERTED
2026 3E 00          LD   A,00H     ;UNTIL CTC TIMES TWICE
2028 D3 8C          OUT  (DIGLH),A ;CLEAR PROM PROG EN
202A 3E 03          LD   A,03H     ;RESET CTC2
202C D3 86          OUT  (86),A
202E EA 18 20       JP   PE,C12A   ;LOOP BACK IF BC-1 NE 0
2031 C1             POP  BC        ;RESTORE BYTE COUNT
2032 3A C0 23       LD   A,(23C0H)
2035 67             LD   H,A
2036 3A C1 23       LD   A,(23C1H) ;SOURCE DATA
2039 6F             LD   L,A
203A 3A C2 23       LD   A,(23C2H)
203D 57             LD   D,A
203E 3A C3 23       LD   A,(23C3H) ;DEST DATA
2041 5F             LD   E,A
2042 C3 04 06       JP   CCS12B    ;USE ROM CODE
```

APPENDIX I

Schematic

Assembly Drawing

Parts List

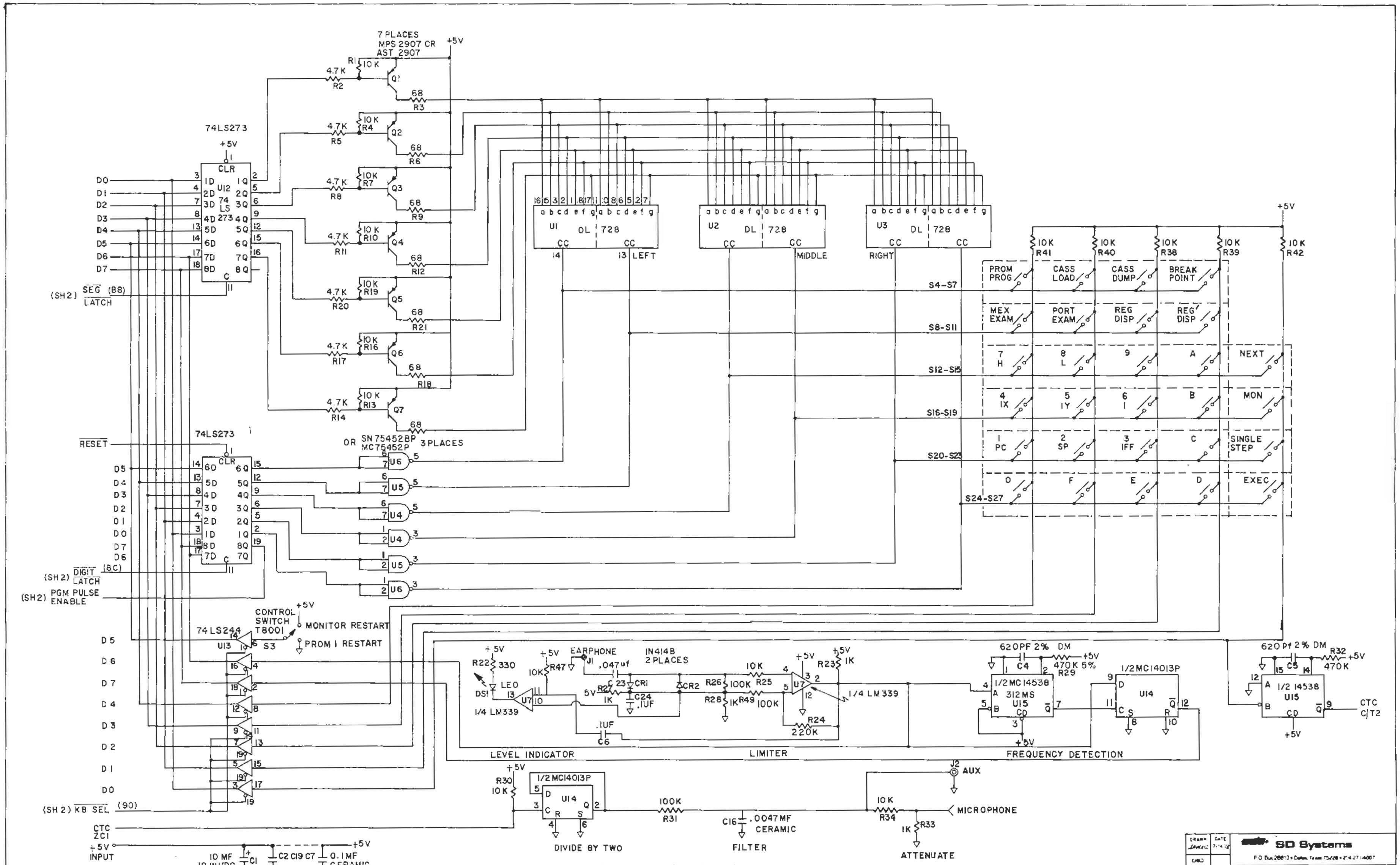


SD Systems

P.O. Box 28810 • Dallas, Texas 75228 • 214-271-4667

Z80 STARTER KIT

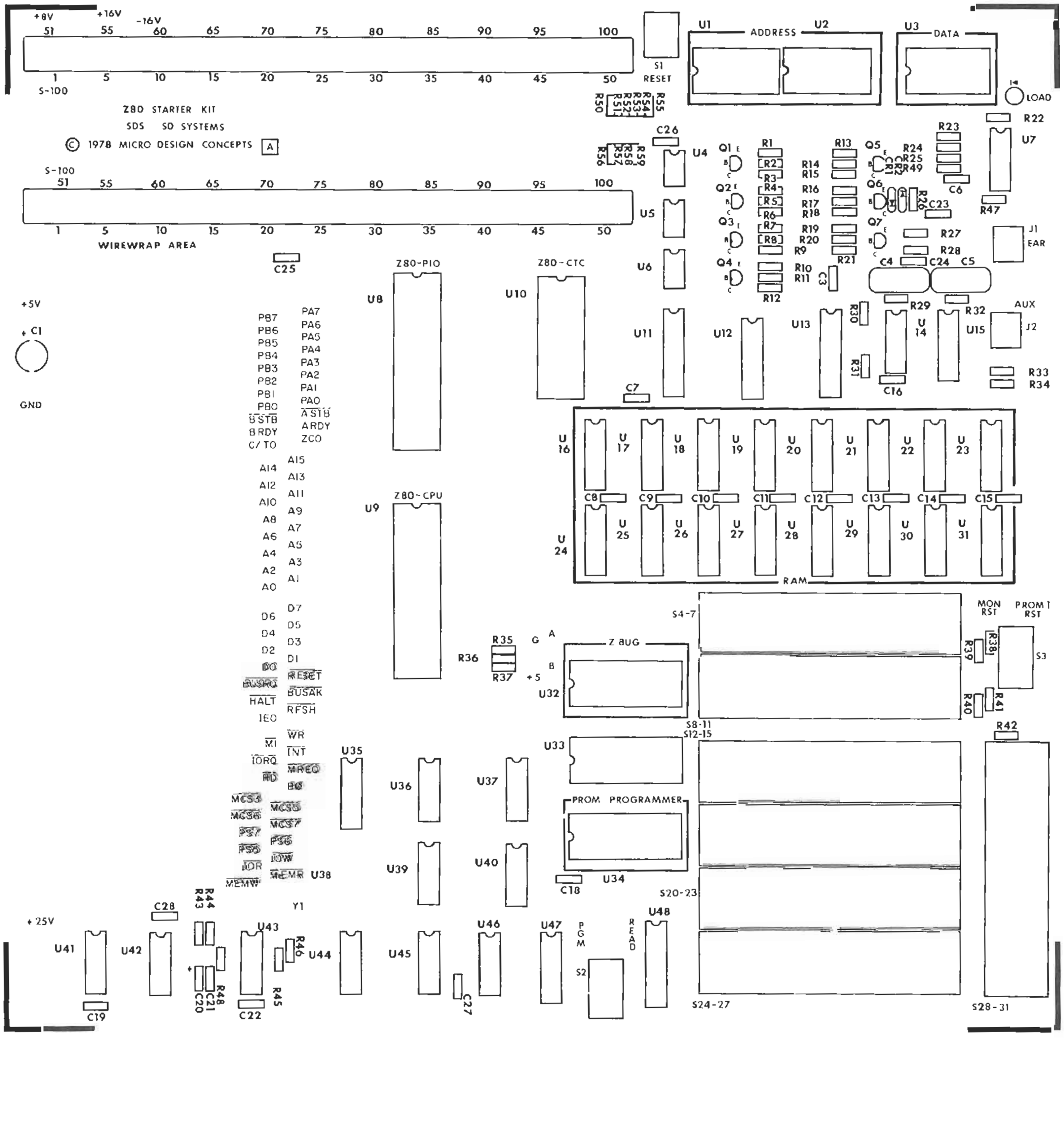
DRAWN	DATE	SCALE	NONE	NO.	0100081	REVISION
CHKD		SHEET	2 of 2			A
APPD						



S-100 CONNECTORS (TWD, BUSSED ONE TO ONE)

PIN NO.	SIGNAL
79	A0
80	A1
81	A2
31	A3
30	A4
29	A5
82	A6
83	A7
84	A8
34	A9
37	A10
87	A11
33	A12
85	A13
86	A14
32	A15
36 95	D0
35 94	D1
41 88	D2
42 89	D3
38 91	D4
39 92	D5
40 93	D6
43 90	D7
47	SMEMR
68	SMEMW
50 100	GND
1 51	8V
49 25	CLK·(BØ)
14	+5V
45	SOUT
46	SPIN
2	+16V
52	−16V

DRAWN	DATE 7-14-78	SD Systems P.O. Box 28810 • Dallas, Texas 75228 • 214-271-4667
CHKD		
	7-14-78	Z80 STARTER KIT
		SCALE NONE SHEET 1 OF 2 · NO 0100081 · REVISION A



Z80 STARTER KIT ASSEMBLY DRAWING

SD Systems

P.O. Box 28810 • Dallas, Texas 75228   214-271-4667

BILL OF MATERIALS

: Z80 STARTER KIT	PL No. 0100080	Rev. A
Date Released: January 15, 1979	Approved:	Sheet 1 Of 3

Item No.	Qty	SD-P/N	Description	Unit Cost	Extension
1	1	7010318	MK3880 Z80-CPU ✓		
2	1	7010319	MK3881 Z80-PIO ✓		
3	1	7010320	MK3882 Z80-CTC ✓		
4	8	7010340	21L02-1 500 NSEC RAM ✓		
5	1	7010350	8316 E ROM-ZBUG ✓		
6	2	7010219	74LS138 DECODER ✓		
7	3	7010181	74LS32 QUAD OR GATE ✓		
8	2	7010164	74LS04 QUAD BUFFER ✓		
9	1	7010166	74LS08 QUAD AND GATE ✓		
10	2	7010264	74LS244 OCTAL BUFFER ✓		
11	2	7010276	74LS273 OCTAL LATCH ✓		
12	2	7010195	74LS74 DUAL LATCH ✓		
13	1	7010162	74LS02 QUAD NOR GATE ✓		
14	3	7010342	75452P PERIPHERAL DRIVER ✓		
15	1	7010351	MC14538BCP CMOS MONOSTABLE ✓		
16	1	7010352	MC14013BCP CMOS LATCH ✓		
17	1	7160004	LM339 QUAD COMPARITOR (MLM339P) ✓		
18	3	7180001	DL728 DUAL DISPLAY		
19	7	7040009	AST2907 PNP TRANSISTOR (MPS2907)		
20	1	7080004	3.9936 MHZ PARALLEL RESONANT CRYSTAL		
21	1	7180002	LED		
22	7	7020089	4.7K OHM CARBON COMP RESISTOR ¼W ± 5%		
23	29	7020097	10K OHM CARBON COMP RESISTOR ¼W ± 5%		
24	7	7020045	68 OHM CARBON COMP RESISTOR ¼W ± 5%		

SD Systems

P.O. Box 28810 • Dallas, Texas 75228   214-271-4667

BILL OF MATERIALS

Title: Z80 STARTER KIT	**PL No.** 0100080	**Rev.** A
Date Released: January 15, 1979	**Approved:**	**Sheet** 2 **Of** 3

Item no	Qty	SD-P/N	Description	Unit Cost	Extension
25	7	7020073	1K OHM CARBON COMP RESISTOR ¼W ± 5%		
26	2	7020137	470K OHM CARBON COMP RESISTOR ¼W ± 5%		
27	1	7020129	220K OHM CARBON COMP RESISTOR ¼W ± 5%		
28	3	7020121	100K OHM CARBON COMP RESISTOR ¼W ± 5%		
29	2	7020061	330 OHM CARBON COMP RESISTOR ¼W ± 5%		
30	1	7020113	47K OHM CARBON COMP RESISTOR ¼W ± 5%		
31	19	7030007	.1 UF CERAMIC CAPACITOR ± 20%		
32	1	7030019	1 UF 10V TANTALUM CAPACITOR ± 20%		
33	1	7030009	10 UF 10V TANTALUM CAPACITOR ± 20%		
34	1	7030012	.0047 UF CERAMIC CAPACITOR ± 20%		
35	1	7030021	10 PF DIPPED MICA CAPACITOR ± 2%		
36	1	7030008	.01 UF CERAMIC CAPACITOR ± 20%		
37	2	7030013	620 PF DIPPED MICA CAPACITOR ± 2%		
38	1	7030014	.047 UF CERAMIC CAPACITOR ± 20%		
39	1	7050004	PUSH BUTTON-CONTROL SWITCH B8600		
40	2	7050005	TOGGLE SWITCH-CONTROL SWITCH T8201		
41	7	7050006	KEYSWITCH STACKPOLE LO-PR05 or CONTROL DEVELOPMENT		
42	1	7050007	KEYTOP - SET		
43	1	7000004	12" x 12" PWB		
44	1	7140020	OPERATIONS MANUAL		
45	1	7140019	MOSTEK Z80 MICRO-REF MANUAL MK78516		
46	2	7090001	AUDIO JACKS SWITCHCRAFT 142A		
47	15	7130011	NYLON SPACER- 1 inch		
48	15	7130010	4-40 x ¼" STEEL SCREW		

SD Systems

P.O. Box 28810 • Dallas, Texas 75228   214-271-4667

BILL OF MATERIALS

Title: Z80 STARTER KIT

PL No. 0100080   Rev. A

Date Released: January 15, 1979

Approved:

Sheet 3 Of 3

Item no	Qty	SD-P/N	Description	Unit Cost	Extension
49	14	7130012	2-56 NYLON NUT		
50	3	7060001	8 PIN SOCKET		
51	12	7060002	14 PIN SOCKET		
52	11	7060003	16 PIN SOCKET		
53	4	7060005	20 PIN SOCKET		
54	3	7060007	24 PIN SOCKET		
55	1	7060008	28 PIN SOCKET		
56	2	7060009	40 PIN SOCKET		
57	2	7040001	IN4148 DIODE		
58	1	7010005	7404 QUAD BUFFER		

APPENDIX II

ZBUG Monitor Version 1.0

```
LOAD MAP

DK1:RESTR .OBJE1]     REL     BEG ADDR 0000     END ADDR 00F3
DK1:DISUP .OBJE1]     REL     BEG ADDR 00F4     END ADDR 0122
DK1:DECKY .OBJE1]     REL     BEG ADDR 0123     END ADDR 0633
DK1:UTIL  .OBJE1]     REL     BEG ADDR 0634     END ADDR 07F4
DK1:UTILR .OBJE1]     ABS     BEG ADDR 07F8     END ADDR 07FF


 GLOBAL CROSS REFERENCE TABLE

 SYMBOL ADDR  REFERENCES
 ARFLG  23E0  06A4 03C5 02F0
 BFLG   23F4  0681 04BF 04BB 049E 0275 00A9
 BPTAB  23E4  067E 0250
 CTC1P  07FA  04D5
 CTC3L  07FE  0759
 D20MS  064F  0172 0135
 DECKY  0123  0121
 DIG2   23F5  068F 036C 01A6
 DIG4   23F6  0692 0498 0396 0235 01AC
 DISMEM 23F7  06AA 0687 065B 0617 05BB 0470 0443 03B6 02F4 00F5 00E4
 +            00B4 0080
 DISUP  00F4  062E 05CB 04C8 04B3 03B2 0393 0369 02EB 02D6 01BC 01A3
 +            014D 0132 00F2 00D0 009D
 DSMEM1 23F8  017F 00B9
 DSMEM2 23F9  061F 05C3 0420 0378 0340 0334 00EB 00BC
 DSMEM3 23FA  037B 0187 00C3
 DSMEM4 23FB  0627 03E2 03A4 0385 0357 034A 0327 0315 0301 02E5 00C6
 DSMEM5 23FC  043A 00C9
 DSMEM6 23FD  0363
 DSMEM7 23FE  0434 0360 0192
 FLG24  23D9  0737 072D 0724 0714 0707 04D0
 INCHR  0758  0591
 INCHR4 079D  07FE
 KEYPTR 23DB  068A 03AF 0390 0366 0351 032B 01B9 01B5 01A0 019C 0195
 +            018A 017A
 KYTBL  07B9  0160
 MFLG   23E1  069E 03CC 039C 02A5
 NMIS   01CB  0067
 OTCHR  06F4
 OTCHR1 06F7  050C
 OTCHR6 0732  07FA
 PFLG   23DE  069B 03B9 0373 02AC
 PRFLG  23DA  0698 05D6 02B3
 PUNHEH 23C2  04EE
 PUNHEL 23C3  04F2
 PUNHSH 23C0  0552 0529 04E6
 PUNHSL 23C1  0556 0530 04EA
 REGTB  07D5  047A
 REGTBP 07E5  045D
 REMBP2 007E  0208
 RESTAR 009F
 RESTR1 00AE  060A 05D1 057F 04B0 0482 0465 043F 03D0 02B8 02A2
 RFLG   23DF  06A1 03BF 030B
 RST16  23C4  0011
 RST24  23C7  0019
 RST32  23CA  0021
 RST40  23CD  0029
 RST48  23D0  0031
 RST56  23D3  0039
 SEGPT  07A6  0102
 SSFLG   23F3    0695 027A 0202 01F9
 STKPT   23E2    0485 0468 0346 023F 01F6 01F6 01CD 00A1 0083 004B 0043
 +               000A
 STKPT1  23E3    033C
 UABIN   06B3
 UBASC   06BB
 UFGCR   0686    00AF
 UFOR1   063C    084B 0837 081B 0907 03B4 04E8 03E5 03A7 038B 035B 004B 00E7 009A 0

 UFOR2   0659    05D9 04A3 03E9 03D3 039F 037E 02D9 02BE 023B
 UFOR3   067C    04A6 0248 0205 0062
 UFOR4   06A7    04CB
 UIF     23DD    0437 035A 0295 01F2 00AC 005F
 UIX3    0634    04B6 0260 007A
 ULACC   06DB    05B3 05AC 05A5 05A1 059D 0599
 UPACC   06C4
 UPACCS  06D5    056B 0566 055F 054E 0549 0544 053C 0538 0534 052D 0526


GLOBAL SYMBOL TABLE

ARFLG   23E0     BFLG    23F4     BPTAB   23E4     CTC1P   07FA
CTC3L   07FE     D20MS   064F     DECKY   0123     DIG2    23F5
DIG4    23F6     DISMEM  23F7     DISUP   00F4     DSMEM1  23F8
DSMEM2  23F9     DSMEM3  23FA     DSMEM4  23FB     DSMEM5  23FC
DSMEM6  23FD     DSMEM7  23FE     FLG24   23D9     INCHR   0758
INCHR4  079D     KEYPTR  23DB     KYTBL   07B9     MFLG    23E1
NMIS    01CB     OTCHR   06F4     OTCHR1  06F7     OTCHR6  0732
PFLG    23DE     PRFLG   23DA     PUNHEH  23C2     PUNHEL  23C3
PUNHSH  23C0     PUNHSL  23C1     REGTB   07D5     REGTBP  07E5
REMBP2  007E     RESTAR  009F     RESTR1  00AE     RFLG    23DF
RST16   23C4     RST24   23C7     RST32   23CA     RST40   23CD
RST48   23D0     RST56   23D3     SEGPT   07A6     SSFLG   23F3
STKPT   23E2     STKPT1  23E3     UABIN   06B3     UBASC   06BB
UFGCR   0686     UFOR1   063C     UFOR2   0659     UFOR3   067C
UFOR4   06A7     UIF     23DD     UIX3    0634     ULACC   06DB
UPACC   06C4     UPACCS  06D5
```

```
                   0002          NAME    RESTR
                   0003 ;RST,NMI AND BP ROUTINES
                   0004 ;VERSION 1.7  5/18/78
                   0005          PSECT   REL
                   0006          GLOBAL  DSMEM4
                   0007          GLOBAL  DSMEM1
                   0008          GLOBAL  DISUP
                   0009          GLOBAL  UIF
                   0010          GLOBAL  UIX3
                   0011          GLOBAL  STKPT
                   0012          GLOBAL  RST16
                   0013          GLOBAL  RST24
                   0014          GLOBAL  RST32
                   0015          GLOBAL  RST40
                   0016          GLOBAL  RST48
                   0017          GLOBAL  RST56
                   0018          GLOBAL  DSMEM5
                   0019          GLOBAL  DSMEM2
                   0020          GLOBAL  DISMEM
                   0021          GLOBAL  DSMEM3
                   0022          GLOBAL  UFGCR
                   0023          GLOBAL  UFOR1
                   0024          GLOBAL  NMIS
                   0025          GLOBAL  UFOR3
                   0026          GLOBAL  RESTAR
                   0027          GLOBAL  REMBP2
                   0028          GLOBAL  RESTR1
                   0029          GLOBAL  BFLG
 .0000             0030          ORG     0000H
                   0031 ;****RESTART INSTRUCION HANDLER****
                   0032 ;****RESET ENTRY POINT - RST0 ****
                   0033 ;***RST0 IS RESERVED FOR RESET AND RST8 IS RESERVED
                   0034 ;    FOR BREAKPOINTS.  ALL OTHER RST'S ARE MAPPED
                   0035 ;    TO LOCATIONS IN RAM WHERE THE USER CAN INSERT
                   0036 ;    A JUMP TO THE SERVICE ROUTINE IN RAM FOR
                   0037 ;    RST 8,16,24,32,48, AND 56.  THE USER'S REGISTERS
                   0038 ;    ARE STACKED UPON BREAKPOINT ENTRY AND ALL
                   0039 ;    BREAKPOINTS ARE REMOVED FROM THE USER'S PROGRAM.
                   0040 ;****RESET PUSHBUTTON ENTRY POINT - RST0
'0000  31C023      0041          LD      SP,23C0H
'0003  C39F00'     0042          JP      RESTAR  ;FINISH INITALIZATION
                   0043 ;****BREAKPOINT ENTRY - RST8
 >0008             0044          ORG     0008H   ;START RESTART TABLE
'0008  ED73FFFF    0045 BPENT:   LD      (STKPT),SP      ;SAVE USER'S SP  23E2
'000C  F5          0046          PUSH    AF      ;STACK USER REGISTERS
'000D  C5          0047          PUSH    BC
'000E  1803        0048          JR      BPENT2-$
                   0049 ;****USER ENTRY - RST16
'0010  C3FFFF      0050          JP      RST16   ;MAP INTO RAM  23C4
'0013  ED57        0051 BPENT2:  LD      A,I     ;I INTO A,IFF INTO F
'0015  F3          0052          DI
'0016  1803        0053          JR      BPENT3-$
                   0054 ;****USER ENTRY - RST24
  '18  C3EFFB      0055          JP      RST24   23C7
'001B  D5          0056 BPENT3:  PUSH    DE
'001C  E5          0057          PUSH    HL
'001D  F5          0058          PUSH    AF      ;SAVE I AND IFF
'001E  1803        0059          JR      BPENT4-$
```

```
                    0060 ;****USER ENTRY - RST32
'0020   C3FFFF      0061           JP      RST32    23CA
'0023   08          0062 BPENT4:  EX      AF,AF'   ;GET ALT REGS
'0024   D9          0063           EXX
'0025   F5          0064           PUSH    AF
'0026   1803        0065           JR      BPENT5-$
                    0066 ;USER ENTRY - RST40
'0028   C3FFFF      0067           JP      RST40    23CD
'002B   C5          0068 BPENT5:  PUSH    BC
'002C   D5          0069           PUSH    DE
'002D   E5          0070           PUSH    HL
'002E   1803        0071           JR      BPENT6-$
                    0072 ;****USER ENTRY - RST48
'0030   C3FFFF      0073           JP      RST48    23DØ
'0033   DDE5        0074 BPENT6:  PUSH    IX
'0035   00          0075           NOP               ;SAVE A BYTE
'0036   1803        0076           JR      BPENT7-$
                    0077 ;****USER ENTRY - RST56
'0038   C3FFFF      0078           JP      RST56    23D3
'003B   FDE5        0079 BPENT7:  PUSH    IY
'003D   3E03        0080           LD      A,03H
'003F   D386        0081           OUT     (CTC2),A          ;DISABLE KYBD INTR
'0041   DD2A0A00'   0082           LD      IX,(STKPT)        ;GET USERS SP
'0045   DD23        0083           INC     IX
'0047   DD23        0084           INC     IX
'0049   DD224300'   0085           LD      (STKPT),IX
'004D   DD7EFE      0086           LD      A,(IX-2)          ;TEST PC LOW BYTE
'0050   B7          0087           OR      A    ;SET STATUS
'0051   2003        0088           JR      NZ,BPENT8-$
'0053   DD35FF      0089           DEC     (IX-1)            ;DEC HIGH BYTE
'0056   DD35FE      0090 BPENT8:  DEC     (IX-2)            ;DEC LOW BYTE
'0059   DD7EF4      0091           LD      A,(IX-12)         ;IFF ON STACK
'005C   E604        0092           AND     4                 ;MASK OUT IFF
'005E   32FFFF      0093           LD      (UIF),A           ;SAVE FOR LATER
'0061   CDFFFF      0094           CALL    UFOR3    ;GET TAB ADDR AND BPFLG
'0064   1803        0095           JR      BPNT8A-$
'0066   C3FFFF      0096           JP      NMIS
'0069   2813        0097 BPNT8A:  JR      Z,REMBP2-$        ;NO BKPTS FO DISPLAY REGS
                    0098 ;*****BREAKPOINT REMOVAL*********
                    0099 ;REMOVE BREAKPOINTS WHILE IN ZBUG.THEY WILL BE RESTORED ON
                    0100 ;A EXECUTE OR PROCEED COMMAND.
'006B   DD7E02      0101 REMBP:   LD      A,(IX+2)          ;GET OP CODE TO RESTORE
                    0102 ;CHECK FOR MULTI-DEFINED BREAKPOINTS
'006E   FECF        0103           CP      0CFH
'0070   2807        0104           JR      Z,REMBP1-$        ;BRANCH IF MULTI-DEFINED
'0072   DD6E01      0105 REMBP0:  LD      L,(IX+1)
'0075   DD6600      0106           LD      H,(IX+0)          ;GET ADDR OF BP
'0078   77          0107           LD      (HL),A   ;RESTORE OP CODE
'0079   CDFFFF      0108 REMBP1:  CALL    U1X3     ;GET NEXT POSITION AND DEC B
'007C   20ED        0109           JR      NZ,REMBP-$        ;GO AGAIN
                    0110 ;DISPLAY PC AND A
'007E   DD21FFFF    0111 REMBP2:  LD      IX,DISMEM         ;POINT TO DISMEM
'0082   2A4B00'     0112           LD      HL,(STKPT)        ;POINT TO STACK
'0085   2B          0113           DEC     HL
'0086   7E          0114           LD      A,(HL)   ;GET PCH
'0087   CDFFFF      0115           CALL    UFOR1    ;WRITE TO DISMEM
'008A   DD23        0116           INC     IX
'008C   DD23        0117           INC     IX
```

```
 ADDR   OBJECT     ST # SOURCE STATEMENT          DATASET = DKO:RESTR .SRC

'008E   2B         0118            DEC     HL
'008F   7E         0119            LD      A,(HL)  ;GET PCL
'   90  CD8800'    0120            CALL    UFOR1   ;WRITE TO DISMEM
'0093   DD23       0121            INC     IX
'0095   DD23       0122            INC     IX
'0097   2B         0123            DEC     HL
'0098   7E         0124            LD      A,(HL)  ;GET A
'0099   CD9100'    0125            CALL    UFOR1   ;WRITE TO DISMEM
'009C   C3FFFF     0126            JP      DISUP   ;GO DISPLAY
                   0127 ;*******FINISH RESTART ROUTINE***********
'009F   ED738300'  0128 RESTAR:    LD      (STKPT),SP       ;INIT SP
'00A3   31A823     0129            LD      SP,23A8H         ;INIT SP (ZBUGS)
'00A6   3E00       0130            LD      A,00H
'00A8   32FFFF     0131            LD      (BFLG),A         ;CLEAR BFLG
'00AB   325F00'    0132            LD      (UIF),A          ;CLEAR INTR BIT
'00AE   CDFFFF     0133 RESTR1:    CALL    UFGCR   ;CLR FLAGS
'00B1   3E11       0134            LD      A,11H   ;PROMPT CHARACTER
'00B3   328000'    0135            LD      (DISMEM),A
'00B6   3E10       0136            LD      A,10H   ;B CHARACTER
'00B8   32FFFF     0137            LD      (DSMEM1),A
'00BB   32FFFF     0138            LD      (DSMEM2),A
'00BE   1802       0139            JR      RESTR2-$
'00C0   1813       0140            JR      RESTR3-$         ;RELATIVE OFFSET CALCULATI
'00C2   32FFFF     0141 RESTR2:    LD      (DSMEM3),A
'00C5   32FFFF     0142            LD      (DSMEM4),A
'00C8   32FFFF     0143            LD      (DSMEM5),A
'00CB   DB90       0144            IN      A,(KBSEL)        ;SENSE SWITCH
'' OCD  CB6F       0145            BIT     5,A     ;BIT TEST
    CF  C29D00'    0146            JP      NZ,DISUP         ;GO TO ZBUG
'00D2   C30008     0147            JP      0800H   ;RESTART TO PROM
                   0148 ;RELATIVE OFFSET CALCULATION ROUTINE-AUTOMATICALLY CALCULA
                   0149 ;RELATIVE OFFSET AND PLACES IT IN MEMORY AND ON THE DISPLA
                   0150 ;HL = ADDRESS OF OPCODE AT DESTINATION OF RELATIVE JUMP
                   0151 ;DE = ADDRESS OF THE INSTRUCTION WHICH HAS THE OFFSET
'00D5   13         0152 RESTR3:    INC     DE      ;POINT TO OFFSET
'00D6   D5         0153            PUSH    DE
'00D7   DDE1       0154            POP     IX      ;SAVE DISPLACEMEMT ADDRESS
'00D9   13         0155            INC     DE      ;POINT TO NEXT OPCODE
'00DA   7D         0156            LD      A,L     ;GET LOW BYTE
'00DB   93         0157            SUB     E       ;SUBTRACT
'00DC   6F         0158            LD      L,A     ;;SAVE OFFSET
'00DD   DD7700     0159            LD      (IX+0),A         ;CHANGE MEMORY
'00E0   7C         0160            LD      A,H     ;GET HIGH BYTE
'00E1   9A         0161            SBC     A,D     ;SUBTRACT
'00E2   DD21B400'  0162            LD      IX,DISMEM
'00E6   CD9A00'    0163            CALL    UFOR1   ;WRITE OVERRANGE TO DISPLAY
'00E9   DD21BC00'  0164            LD      IX,DSMEM2
'00ED   7D         0165            LD      A,L
'00EE   CDE700'    0166            CALL    UFOR1   ;WRITE OFFSET TO DISPLAY
'00F1   C3D000'    0167            JP      DISUP   ;GO DISPLAY
 >0086             0168 CTC2:      EQU     86H     ;SS/PROM PGM
 >0090             0169 KBSEL:     EQU     90H     ;MONITOR/PROM1 SENSING
                   0170            END


ERRORS=0000
```

```
DISUP   (C) 1978 MICRO DESIGN CONCERTS   MOSTEK FLP-80 ASSEMBLER V2.0 PAGE 0001
 ADDR   OBJECT    ST # SOURCE STATEMENT         DATASET = DK0:DISUP .SRC

                  0002          NAME    DISUP
                  0003 ;DISPLAY UPDATE ROUTINE
                  0004 ;VERSION 0.3  3/13/78
                  0005          PSECT   REL
                  0006          GLOBAL  DISMEM
                  0007          GLOBAL  SEGPT
                  0008          GLOBAL  DECKY
                  0009          GLOBAL  DISUP
                  0010          GLOBAL  DISUP
 >00F4            0011          ORG     00F4H
                  0012 ;*********DISPLAY UPDATE ROUTINE*********
                  0013 ;**FUNCTION:MOVES DATA FROM DISMEM TO DISPLAYS.
                  0014 ;**ENTRY:DATA TO BE DISPLAYED IN SIX MEMORY LOCATIONS
                  0015 ;           STARTING AT LOCATION DISMEM.
                  0016 ;**EXIT:  ALL SIX DIGITS REFRESHED - REGISTERS DISTROYED AR
                  0017 ;          IX, A, B, D, E, H, L
                  0018 ;**REGISTER USAGE:HL IS POINTER TO CURRENT LOC IN DISMEM
                  0019 ;                 IX IS POINTER TO SEG PATTERN TABLE
                  0020 ;                 B  IS CURRENT DIGIT POINTER
                  0021 ;                 E  HOLDS DIGIT DATA TO BE DISPLAYED
                  0022 ;                 A AND D ARE SCRATCH REGISTERS
'00F4  21FFFF     0023 DISUP:   LD      HL,DISMEM       ;POINT TO START OF BUFFER
'00F7  0620       0024          LD      B,020H  ;POINT TO LEFT DIGIT
'00F9  5E         0025 DISUP1:  LD      E,(HL)  ;GET FIRST BYTE
'00FA  1600       0026          LD      D,0     ;CLR D
'00FC  3E00       0027          LD      A,0
'00FE  D38C       0028          OUT     (DIGLH),A       ;TURN OFF DISPLAY
'0100  DD21FFFF   0029          LD      IX,SEGPT        ;INDEX INTO TABLE
'0104  DD19       0030          ADD     IX,DE           ;FOR SEGMENT PATTERN
'0106  DD7E00     0031          LD      A,(IX+0)        ;GET PATTERN
'0109  D388       0032          OUT     (SEGLH),A       ;OUT TO SEG LATCH
'010B  78         0033          LD      A,B
'010C  D38C       0034          OUT     (DIGLH),A       ;OUT TO DIGIT LATCH
'010E  1E2D       0035          LD      E,45D   ;SET DELAY FOR 1 MS
'0110  1D         0036 DISUP2:  DEC     E
'0111  3E00       0037          LD      A,0     ;DELAY LOOP
'0113  BB         0038          CP      E
'0114  20FA       0039          JR      NZ,DISUP2-$
'0116  3E01       0040          LD      A,01H
'0118  B8         0041          CP      B       ;B=1?
'0119  2805       0042          JR      Z,DISUP3-$      ;YES,RE-INIT & EXIT
'011B  23         0043          INC     HL      ;NO,POINT TO NEXT DIGIT
'011C  CB38       0044          SRL     B       ;SHIFT TO NEXT DIGIT
'011E  18D9       0045          JR      DISUP1-$
'0120  C3FFFF     0046 DISUP3:  JP      DECKY   ;GO DECODE KEYS
 >008C            0047 DIGLH:   EQU     8CH     ;WRITE ONLY - DIGIT SELECT
 >0088            0048 SEGLH:   EQU     88H     ;WRITE ONLY - SEG PATTERN
                  0049          END


ERRORS=0000
```

```
>0123              0002          ORG      0123H
                   0003 ;KEY DECODE AND EXECUTION
                   0004 ;VERSION 2.1   5/17/78
                   0005          PSECT    REL
                   0006          NAME     DECKY
                   0007          GLOBAL   UFOR4
                   0008          GLOBAL   UFOR1
                   0009          GLOBAL   UFOR2
                   0010          GLOBAL   UFOR3
                   0011          GLOBAL   REMBP2
                   0012          GLOBAL   D20MS
                   0013          GLOBAL   KEYPTR
                   0014          GLOBAL   STKPT
                   0015          GLOBAL   STKPT1
                   0016          GLOBAL   SSFLG
                   0017          GLOBAL   UPACCS
                   0018          GLOBAL   OTCHR1
                   0019          GLOBAL   OTHCR6
                   0020          GLOBAL   UFOR1
                   0021          GLOBAL   BPTAB
                   0022          GLOBAL   DIG4
                   0023          GLOBAL   DIG2
                   0024          GLOBAL   DIG8
                   0025          GLOBAL   MFLG
                   0026          GLOBAL   BFLG
                   0027          GLOBAL   RFLG
                   0028          GLOBAL   ARFLG
                   0029          GLOBAL   PFLG
                   0030          GLOBAL   PRFLG
                   0031          GLOBAL   INCHR
                   0032          GLOBAL   PUNHSH
                   0033          GLOBAL   PUNHSL
                   0034          GLOBAL   PUNHEH
                   0035          GLOBAL   PUNHEL
                   0036          GLOBAL   REGTB
                   0037          GLOBAL   RESTAR
                   0038          GLOBAL   RESTR1
                   0039          GLOBAL   DISUP
                   0040          GLOBAL   REGTBP
                   0041          GLOBAL   DISMEM
                   0042          GLOBAL   RFLG
                   0043          GLOBAL   UIF
                   0044          GLOBAL   DSMEM1
                   0045          GLOBAL   DSMEM2
                   0046          GLOBAL   DSMEM3
                   0047          GLOBAL   DSMEM4
                   0048          GLOBAL   DSMEM5
                   0049          GLOBAL   DSMEM6
                   0050          GLOBAL   DSMEM7
                   0051          GLOBAL   ULACC
                   0052          GLOBAL   FLG24
                   0053          GLOBAL   UIX3
                   0054          GLOBAL   KYTBL
                   0055          GLOBAL   CTC1P
                   0056          GLOBAL   NMIS
                   0057          GLOBAL   DECKY
                   0058 ;FUNCTION:BLANKS DISPLAY,SCANS ENTIRE MATRIX FOR CLOSURE.
                   0059 ;          IF CLOSURE IS FOUND A 20 MS DELAY IS INVOKED
```

```
                  0060 ;            TO DEBOUNCE KEYS.  KEY MATRIX IS SCANNED ONE
                  0061 ;            ROW AT A TIME, CHECKING ALL COLUMNS.  KEY VALU
                  0062 ;            IS FOUND AND IF IT IS A HEX DIGIT IT IS PLACE
                  0063 ;            IN THE NEXT EMPTY LOCATION OF DISMEM (POINTER=
                  0064 ;            KEYPTR).  FLAGS ARE SET WHEN 2 DIGITS (DIG2)
                  0065 ;            AND 4 DIGITS HAVE BEEN ENTERED (DIG4).  WHEN
                  0066 ;            8 DIGITS HAVE BEEN ENTERED A MEMORY CHANGE IS
                  0067 ;            CALLED. IF A COMMAND KEY IS DECODED THE CORRECT
                  0068 ;            SERVICE ROUTINE IS FOUND IN A JUMP TABLE (JPTAB)
                  0069 ;            COMMAND KEY EXECUTION ROUTINES ARE ALSO INCLUDED
                  0070 ;            IN THIS MODULE.
                  0071 ;ENTRY:     NONE REQUIRED, DONE AFTER DISP UPDATE (DISUP)
                  0072 ;EXIT:      IF CMND KEY, CMND HAS BEEN EXECUTED. IF HEX
                  0073 ;            KEY, DATA HAS BEEN ENTERED INTO DISMEM.
                  0074 ;REGISTERS USED:
                  0075 ;            A-SCRATCH-KEYSL VALUE
                  0076 ;            B-DIGLH VALUE
                  0077 ;            C-SCRATCH-UNIQUE WORD FROM ROW AND CLMN DATA
                  0078 ;            HL-POINTER INTO KYTBL
                  0079 ;            HL-POINTER INTO DISMEM (KYPTR)
                  0080 ;            BC-SCRATCH-USED TO COMPUTE OFFSET INTO DISMEM
                  0081 ;            EXX-USED BY PROM PROGRAMMER TO HOLD POINTERS
                  0082 ;            DURING ERROR CHECKING AND DISP USING NEXT KEY.
'0123  3E7F       0083 DECKY:  LD      A,7FH
'0125  D388       0084         OUT     (SEGLH),A       ;TURN OFF DISPLAY
'0127  3E3F       0085         LD      A,3FH
'0129  D38C       0086         OUT     (DIGLH),A       ;OUTPUT ALL ROWS LOW
'012B  DB90       0087         IN      A,(KBSEL)       ;INPUT COLUMN DATA
'012D  E61F       0088         AND     1FH             ;MASK OUT OTHER INPUTS
'012F  FE1F       0089         CP      1FH             ;ANY KEY DOWN?
'0131  CAFFFF     0090         JP      Z,DISUP ;NO, RETRUN TO DISPLAY
'0134  CDFFFF     0091         CALL    D20MS           ;YES,WAIT FOR DEBOUNCE
'0137  0E8C       0092         LD      C,DIGLH
'0139  0601       0093         LD      B,01H
'013B  ED41       0094 KEYDN1: OUT     (C),B   ;PLACE SELECTED ROW LOW
'013D  DB90       0095         IN      A,(KBSEL)       ;INPUT COLUMN DATA
'013F  E61F       0096         AND     1FH     ;MASK OUT D5,D6,D7
'0141  FE1F       0097         CP      1FH     ;KEY DOWN?
'0143  200A       0098         JR      NZ,KEYDN2-$     ;YES,DECODE KEY
'0145  CB20       0099         SLA     B       ;NO,SELECT NEXT ROW
'0147  3E40       0100         LD      A,40H
'0149  B8         0101         CP      B       ;ALL DONE?
'014A  20EF       0102         JR      NZ,KEYDN1-$     ;NO LOOP BACK
'014C  C33201'    0103         JP      DISUP   ;YES EXIT
                  0104 ;**ENTRY CONDITIONS TO KEYDECODE ARE ROW IN B AND
                  0105 ;   COLUMN (MASKED BY 1F) IN A.
'014F  0E00       0106 KEYDN2: LD      C,00H
'0151  0D         0107 KEYDN3: DEC     C
'0152  CB38       0108         SRL     B
'0154  20FB       0109         JR      NZ,KEYDN3-$     ;FALL THROUGH WHEN B=0
'0156  CB21       0110         SLA     C
'0158  CB21       0111         SLA     C
'015A  CB21       0112         SLA     C
'015C  CB21       0113         SLA     C       ;GET VALUE TO HIGH NIBBLE
'015E  81         0114         ADD     A,C     ;COMBINE WITH A
'015F  21FFFF     0115         LD      HL,KYTBL        ;SETUP POINTER TO NEXT DIG
'0162  BE         0116 KEYDN4: CP      (HL)    ;A=TABLE FOUND
'0163  2804       0117         JR      Z,KEYDN5-$      ;YES,FOUND IT
```

```
'0165  23         0118            INC     HL
'0166  04         0119            INC     B        ;ADJ POINTERS
'   57 18F9       0120            JR      KEYDN4-$         ;LOOP BACK
'0169  DB90       0121 KEYDN5:    IN      A,(KBSEL)        ;CK FOR KEY RELEASE
'016B  E61F       0122            AND     01FH     ;MASK OTHER INPUTS
'016D  FE1F       0123            CP      01FH
'016F  20F8       0124            JR      NZ,KEYDN5-$      ;LOOP BACK TILL KEY UP
'0171  CD3501'    0125            CALL    D20MS    ;DEBOUNCE
'0174  78         0126            LD      A,B      ;GET KEY VALUE
'0175  FE10       0127            CP      10H
'0177  3045       0128            JR      NC,KEYDN6-$      ;COMMAND KEY GO DECODE
'0179  2AFFFF     0129            LD      HL,(KEYPTR)      ;HEX KEY, SAVE IN DISMEM
'017C  70         0130            LD      (HL),B
'017D  B7         0131            OR      A
'017E  01FFFF     0132            LD      BC,DSMEM1
'0181  ED42       0133            SBC     HL,BC    ;ENTERED DIGIT 2?
'0183  2820       0134            JR      Z,KEYDNA-$
'0185  B7         0135            OR      A        ;CLEAR CARRY
'0186  01FFFF     0136            LD      BC,DSMEM3        ;CLEAR CARRY
'0189  2A7A01'    0137            LD      HL,(KEYPTR)
'018C  ED42       0138            SBC     HL,BC    ;ENTERED DIGIT 4?
'018E  281B       0139            JR      Z,KEYDN8-$       ;YES, INC FLAG
'0190  B7         0140            OR      A
'0191  01FFFF     0141            LD      BC,DSMEM7
'0194  2A8A01'    0142            LD      HL,(KEYPTR)      ;HEX KEY, SAVE IN DISMEM
'0197  ED42       0143            SBC     HL,BC    ;8 DIGITS IN?
'0199  2816       0144            JR      Z,KEYDN9-$       ;YES, INC FLAG
'   9B 2A9501'    0145 KEYDN7:    LD      HL,(KEYPTR)      ;GET KEY POINTER
'019E  23         0146            INC     HL       ;INCREMENT IT
'019F  229C01'    0147            LD      (KEYPTR),HL      ;SAVE HL
'01A2  C34D01'    0148            JP      DISUP    ;EXIT
'01A5  21FFFF     0149 KEYDNA:    LD      HL,DIG2
'01A8  34         0150            INC     (HL)
'01A9  18F0       0151            JR      KEYDN7-$
'01AB  21FFFF     0152 KEYDN8:    LD      HL,DIG4
'01AE  34         0153            INC     (HL)
'01AF  18EA       0154            JR      KEYDN7-$
'01B1  CDB403'    0155 KEYDN9:    CALL    ALTER    ;ENTER DATA TO MEM,PORT OR REG
'01B4  2AA001'    0156            LD      HL,(KEYPTR)
'01B7  2B         0157            DEC     HL       ;BACKUP POINTER TO 6 DIGITS IN
'01B8  22B501'    0158            LD      (KEYPTR),HL      ;SAVE HL
'01BB  C3A301'    0159            JP      DISUP
                  0160 ;FIND ADDRESS OF COMMAND KEY HANDLER-KEYVALUE IN A
'01BE  D610       0161 KEYDN6:    SUB     10H      ;KEY 16=0 KEY 17=3
'01C0  4F         0162            LD      C,A
'01C1  81         0163            ADD     A,C      ;DOUBLE OFFSET
'01C2  81         0164            ADD     A,C      ;TRIPLE OFFSET
'01C3  4F         0165            LD      C,A
'01C4  0600       0166            LD      B,00H
'01C6  210C02'    0167            LD      HL,JPTAB         ;GET TABLE ADDR
'01C9  09         0168            ADD     HL,BC    ;ADD OFFSET
'01CA  E9         0169            JP      (HL)
                  0170 ;SERVICE ROUTINE FOR NMI INTERRUPTS, SINGLE STEP OR ABORT
   .CB ED73FFFF   0171 NMIS:      LD      (STKPT),SP       ;PRESERVE USER'S SP
'01CF  F5         0172            PUSH    AF
'01D0  3E03       0173            LD      A,03H
'01D2  D386       0174            OUT     (CTC2),A         ;SHUT DOWN CTC
'01D4  ED57       0175            LD      A,I      ;GET I INTO A, IFF INTO F
```

```
'01D6  C5        0176         PUSH    BC
'01D7  D5        0177         PUSH    DE      ;SAVE REGISTERS
'01D8  E5        0178         PUSH    HL
'01D9  F5        0179         PUSH    AF
'01DA  08        0180         EX      AF,AF'
'01DB  D9        0181         EXX
'01DC  F5        0182         PUSH    AF
'01DD  C5        0183         PUSH    BC
'01DE  D5        0184         PUSH    DE
'01DF  E5        0185         PUSH    HL
'01E0  DDE5      0186         PUSH    IX
'01E2  FDE5      0187         PUSH    IY
'01E4  DD2ACD01' 0188         LD      IX,(STKPT)      ;GET USER'S SP
'01E8  DD23      0189         INC     IX
'01EA  DD23      0190         INC     IX      ;ADJUST TO USER'S REAL SP
'01EC  DD7EF4    0191         LD      A,(IX-12)       ;IFF ON STACK
'01EF  E604      0192         AND     4       ;MASK IFF
'01F1  32FFFF    0193         LD      (UIF),A ;SAVE
'01F4  DD22E601' 0194         LD      (STKPT),IX      ;SAVE USER'S SP
'01F8  3AFFFF    0195 NMIS1:  LD      A,(SSFLG)       ;IN SS MODE?
'01FB  B7        0196         OR      A       ;SET STATUS
'01FC  CA2301'   0197         JP      Z,DECKY ;NO, ITS A KB INTR
'01FF  3E00      0198 NMIS2:  LD      A,00H
'0201  32F901'   0199         LD      (SSFLG),A       ;CLEAR FLAG
'0204  CDFFFF    0200         CALL    UFOR3   ;GET BP DATA
'0207  CAFFFF    0201         JP      Z,REMBP2        ;NO BP, DISP PC AND A
'020A  1842      0202         JR      CCS1C-$ ;INSTALL BP,ACTIVATE KB AND RET
                 0203 ;JUMP TABLE FOR COMMAND KEYS
'020C  C33002'   0204 JPTAB:  JP      CCS1    ;EXEC
'020F  C37202'   0205         JP      CCS2    ;SS
'0212  C3A102'   0206         JP      CCS3    ;MON
'0215  C3A402'   0207         JP      CCS4    ;NEXT
'0218  C3ED02'   0208         JP      CCS5    ;REG' DISP
'021B  C30803'   0209         JP      CCS6    ;REG DISP
'021E  C36B03'   0210         JP      CCS7    ;PORT EXAM
'0221  C39503'   0211         JP      CCS8    ;MEM EXAM
'0224  C39704'   0212         JP      CCS9    ;BP
'0227  C3CA04'   0213         JP      CCS10   ;PUNCH TAPE
'022A  C38105'   0214         JP      CCS11   ;LOAD TAPE
'022D  C3D305'   0215         JP      CCS12   ;PROG PROM
                 0216 ;EXEC KEY HANDLER
'0230  3E00      0217 CCS1:   LD      A,00H
'0232  D38C      0218         OUT     (DIGLH),A       ;CLEAR DISPLAY
'0234  3AAC01'   0219         LD      A,(DIG4)
'0237  B7        0220         OR      A       ;SET STATUS
'0238  280D      0221         JR      Z,CCS1A-$       ;4 DIGITS NOT IN PROCEED
'023A  CDFFFF    0222         CALL    UFOR2   ;GET STARTING ADDR IN HL
'023D  DD2AF601' 0223         LD      IX,(STKPT)      ;SET POINTER TO USER'S SP
'0241  DD75FE    0224         LD      (IX-2),L
'0244  DD74FF    0225         LD      (IX-1),H        ;CHANGE PC TO NEW ONE
                 0226 ;PROCEED MODE
'0247  CD0502'   0227 CCS1A:  CALL    UFOR3   ;CHECK IF BP'S ACTIVE
'024A  202B      0228         JR      NZ,CCS2A-$      ;YES SS ONCE
'024C  1816      0229         JR      CCS1D-$ ;NO, ENABLE KB AND RET
                 0230 ;INSTALL BREAKPOINTS AND SAVE USERS OP CODES
'024E  DD21FFFF  0231 CCS1C:  LD      IX,BPTAB        ;POINTER TO BP TAB
'0252  DD6600    0232 CCS1CA: LD      H,(IX+0)
'0255  DD6E01    0233         LD      L,(IX+1)        ;GET ADDR OF OP CODE
```

```
'0258  7E         0234           LD      A,(HL)   ;GET OP CODE
'0259  0ECF       0235           LD      C,0CFH   ;INSTALL RST7 OP CODE
'025B  71         0236           LD      (HL),C   ;AS A BP
'025C  DD7702     0237           LD      (IX+2),A          ;SAVE USERS OP CODE
'025F  CDFFFF     0238           CALL    UIX3     ;POINT TO NEXT LOCATION
'0262  20EE       0239           JR      NZ,CCS1CA-$       ;LOOP BACK FOR MORE
                  0240 ;ENABLE KEYBOARD FOR ABORT
'0264  3E04       0241 CCS1D:    LD      A,04H
'0266  D38C       0242           OUT     (DIGLH),A         ;ENABLE NMI INTERRUPT
'0268  3E45       0243           LD      A,45H
'026A  D386       0244           OUT     (CTC2),A          ;CTC TO MAKE A ZC/TO
'026C  3E01       0245           LD      A,01H
'026E  D386       0246           OUT     (CTC2),A          ;ON ONE NEG PULSE IN
'0270  1812       0247           JR      CCS2B-$ ;RESTORE USER'S REGISTERS
                  0248 ;SINGLE STEP KEY HANDLER
'0272  3E00       0249 CCS2:     LD      A,00H
'0274  32FFFF     0250           LD      (BFLG),A          ;CLEAR OUT ANY BP
'0277  3E01       0251 CCS2A:    LD      A,01H
'0279  320202'    0252           LD      (SSFLG),A         ;SET SS FLG
'027C  3E07       0253           LD      A,07H
'027E  D386       0254           OUT     (CTC2),A          ;NO INTR RESET CTC2
'0280  3E0B       0255           LD      A,11D    ;INIT CTC FOR 176 COUNTS
'0282  D386       0256           OUT     (CTC2),A          ;UNTIL NMI INTR
                  0257 ;RESTORE REGISTERS
'0284  FDE1       0258 CCS2B:    POP     IY
'0286  DDE1       0259           POP     IX
'0288  E1         0260           POP     HL
'0289  D1         0261           POP     DE
'028A  C1         0262           POP     BC      ;ALTERNATE REGISTERS
'028B  F1         0263           POP     AF
'028C  08         0264           EX      AF,AF'
'028D  D9         0265           EXX
'028E  F1         0266           POP     AF      ;I AND IFF
'028F  ED47       0267           LD      I,A     ;RESTORE I
'0291  E1         0268           POP     HL
'0292  D1         0269           POP     DE
'0293  C1         0270           POP     BC
'0294  3AF201'    0271           LD      A,(UIF) ;GET IFF
'0297  B7         0272           OR      A       ;SET STATUS
'0298  C29E02'    0273           JP      NZ,CCS2C          ;GO ENABLE INTR
'029B  F1         0274           POP     AF
'029C  F3         0275           DI              ;DISABLE INTERRUPTS
'029D  C9         0276           RET             ;TO USER'S PROGRAM
'029E  F1         0277 CCS2C:    POP     AF
'029F  FB         0278           EI              ;ENABLE INTERRUPTS
'02A0  C9         0279           RET             ;RETURN TO USER PROGRAM
                  0280 ;MON KEY HANDLER - FORCES RESTART OF MONITOR
                  0281 ;AND SAVES USER'S REGISTERS
'02A1  C3FFFF     0282 CCS3:     JP      RESTR1  ;CLEAR FLAGS, DISP HDR
                  0283 ;NEXT KEY HANDLER
                  0284 ;USED TO SELECT NEXT MEMORY, NEXT PORT OR NEXT ERROR
                  0285 ;IN FROM PROGRAMMING ROUTINE
'02A4  3AFFFF     0286 CCS4:     LD      A,(MFLG)
'02A7  FE01       0287           CP      01H      ;MEMORY EXAM MODE ACTIVE
'02A9  2812       0288           JR      Z,CCS4B-$
'02AB  3AFFFF     0289           LD      A,(PFLG)
'02AE  FE01       0290           CP      01H      ;PORT EXAM MODE ACTIVE?
'02B0  2826       0291           JR      Z,CCS4C-$
```

```
 ADDR   OBJECT     ST # SOURCE STATEMENT          DATASET = DK0:DECKY  .SRC

'02B2  3AFFFF     0292           LD      A,(PRFLG)
'02B5  FE01       0293           CP      01H      ;PROM PROG MODE ACTIVE?
'02B7  C2A202'    0294 CCS4A:    JP      NZ,RESTR1        ;KEY HAD NO MEANING
'02BA  C33006'    0295           JP      CCS12D   ;YES, PROM PROG MODE ACTIVE
                  0296 ;                          GO DISPLAY NEXT ERROR
'02BD  CD3B02'    0297 CCS4B:    CALL    UFOR2    ;FORMAT TO FOUR BYTES IN HL
'02C0  23         0298           INC     HL       ;SELECT NEXT MEMORY ADDR
'02C1  7C         0299           LD      A,H
'02C2  CDFFFF     0300           CALL    UFOR1    ;UPDATE FIRST TWO DIGITS
'02C5  DD23       0301           INC     IX       ;IX IS ALREADY PTING TO DISMEM
'02C7  DD23       0302           INC     IX
'02C9  7D         0303           LD      A,L
'02CA  CDC302'    0304           CALL    UFOR1    ;UPDATE SECOND TWO DIGITS
'02CD  DD23       0305           INC     IX
'02CF  DD23       0306           INC     IX
'02D1  7E         0307           LD      A,(HL)   ;READ NEW MEMORY
'02D2  CDCB02'    0308           CALL    UFOR1    ;UPDATE THIRD TWO DIGITS
'02D5  C3BC01'    0309           JP      DISUP    ;GO DISPLAY
'02D8  CDBE02'    0310 CCS4C:    CALL    UFOR2    ;FORMAT TO FOUR BYTES IN HL
'02DB  4C         0311           LD      C,H      ;;ONLY H HAS MEANING
'02DC  0C         0312           INC     C        ;SELECT NEXT PORT
'02DD  ED78       0313           IN      A,(C)    ;READ PORT
'02DF  79         0314           LD      A,C      ;IX ALREADY POINTING TO DISMEM
'02E0  CDD302'    0315           CALL    UFOR1
'02E3  DD21FFFF   0316           LD      IX,DSMEM4        ;POINT TO LAST TWO DIGITS
'02E7  CDE102'    0317           CALL    UFOR1    ;WRITE INTO DISMEM
'02EA  C3D602'    0318           JP      DISUP    ;GO DISPLAY
                  0319 ;ALTERNATE REGISTER DISPLAY
'02ED  3E01       0320 CCS5:     LD      A,01
'02EF  32FFFF     0321           LD      (ARFLG),A        ;SET ARFLG
'02F2  DD21FFFF   0322           LD      IX,DISMEM        ;POINT TO REG
'02F6  3E12       0323           LD      A,12H    ;PRIME CHAR
'02F8  DD7701     0324           LD      (IX+1),A         ;WRITE TO DISPLAY
'02FB  CD5504'    0325           CALL    ALTER5
'02FE  7E         0326           LD      A,(HL)   ;GET INTO A
'02FF  DD21E502'  0327           LD      IX,DSMEM4        ;PT TO LAST 2 DIGITS
'0303  CDE802'    0328           CALL    UFOR1    ;WRITE A TO DISMEM
'0306  185A       0329           JR      CCS6C-$
                  0330 ;MAIN REGISTER DISPLAY
'0308  3E01       0331 CCS6:     LD      A,01H
'030A  32FFFF     0332           LD      (RFLG),A         ;SET RFLG
'030D  CD6E04'    0333           CALL    ALTER6
'0310  380A       0334           JR      C,CCS6A-$        ;SPEC REG LT 6?
'0312  7E         0335           LD      A,(HL)   ;GET INTO A
'0313  DD210103'  0336           LD      IX,DSMEM4        ;PT TO LAST 2 DIGITS
'0317  CD0403'    0337           CALL    UFOR1    ;WRITE A TO DSMEM4,5
'031A  1846       0338           JR      CCS6C-$
                  0339 ;HANDLE PC,SP,IX,IY
'031C  FE03       0340 CCS6A:    CP      3        ;IS IT KEY 3
'031E  2835       0341           JR      Z,CCS6B-$        ;YES,IFF
'0320  FE02       0342           CP      2        ;KEY 2?
'0322  2817       0343           JR      Z,CCS6A1-$       ;YES, STACK POINTER
'0324  7E         0344           LD      A,(HL)   ;LOW BYTE INTO A
'0325  DD211503'  0345           LD      IX,DSMEM4
'0329  DD22B901'  0346           LD      (KEYPTR),IX      ;PREPARE FOR FOUR DIGITS I
'032D  CD1803'    0347           CALL    UFOR1    ;FIRST BYTE TO DSMEM2,3
'0330  23         0348           INC     HL
'0331  7E         0349           LD      A,(HL)   ;HIGH BYTE TO A
```

```
 ADDR  OBJECT    ST # SOURCE STATEMENT          DATASET = DKO:DECKY .SRC

'0332  DD21FFFF  0350           LD      IX,DSMEM2        ;POINT TO FIRST TWO
'0336  CD2E03'   0351           CALL    UFOR1   ;SECOND BYTE TO DSMEM4,5
'   9  182D      0352           JR      CCS6D-$ ;GO DISPLAY
'033B  3AFFFF    0353 CCS6A1:   LD      A,(STKPT1)
'033E  DD213403' 0354           LD      IX,DSMEM2
'0342  CD3703'   0355           CALL    UFOR1   ;FIRST BYTE TO DISMEM
'0345  3A3F02'   0356           LD      A,(STKPT)
'0348  DD212703' 0357           LD      IX,DSMEM4
'034C  CD4303'   0358           CALL    UFOR1   ;SECOND BYTE TO DISMEM
'034F  DD222B03' 0359           LD      (KEYPTR),IX      ;PREPARE FOR 4 DIGITS IN
'0353  1813      0360           JR      CCS6D-$
                 0361 ;HANDLE IFF
'0355  DD214A03' 0362 CCS6B:    LD      IX,DSMEM4
'0359  3A9502'   0363           LD      A,(UIF) ;GET VALUE 4 OR 0
'035C  CD4D03'   0364           CALL    UFOR1   ;WRITE TO DISMEM
'035F  219201'   0365           LD      HL,DSMEM7
'0362  21FFFF    0366 CCS6C:    LD      HL,DSMEM6        ;SETUP FOR REG CHG
'0365  225103'   0367           LD      (KEYPTR),HL
'0368  C3EB03'   0368 CCS6D:    JP      DISUP
                 0369 ;EXAMINE PORTS
'036B  3AA601'   0370 CCS7:     LD      A,(DIG2)
'036E  FE01      0371           CP      1
'0370  2020      0372           JR      NZ,CCS7A-$       ;VERIFY TWO DIGITS IN
'0372  32AC02'   0373           LD      (PFLG),A         ;SET PFLG
'0375  3E10      0374           LD      A,10H
'0377  324003'   0375           LD      (DSMEM2),A
'037A  328701'   0376           LD      (DSMEM3),A       ;BLANK DIGITS 2 AND 3
'   7D  CDD902'  0377           CALL    UFOR2   ;GET PORT ADDR INTO HL
'  80  4C        0378           LD      C,H     ;MOVE FIRST TWO DIGITS TO C
'0381  ED78      0379           IN      A,(C)   ;READ THE PORT
'0383  DD215703' 0380           LD      IX,DSMEM4        ;PT TO DATA DIGITS
'0387  CD5D03'   0381           CALL    UFOR1   ;WRITE INTO DISMEM
'038A  DD23      0382           INC     IX
'038C  DD23      0383           INC     IX      ;SETUP TO CHANGE PORTS
'038E  DD226603' 0384           LD      (KEYPTR),IX      ;SETUP TO CHG PORT
'0392  C36903'   0385 CCS7A:    JP      DISUP   ;GO DISPLAY
                 0386 ;EXAMINE MEMORY
'0395  3A3502'   0387 CCS8:     LD      A,(DIG4)
'0398  B7        0388           OR      A       ;VERIFY FOUR DIGITS IN
'0399  2816      0389           JR      Z,CCS8A-$
'039B  32A502'   0390           LD      (MFLG),A         ;SET MFLG
'039E  CD7E03'   0391           CALL    UFOR2   ;GET FOUR BYTE ADDR IN HL
'03A1  7E        0392           LD      A,(HL)  ;GET MEMORY DATA
'03A2  DD218503' 0393           LD      IX,DSMEM4        ;POINT TO CORRECT DIGITS
'03A6  CD8803'   0394           CALL    UFOR1   ;WRITE INTO DISMEM
'03A9  DD23      0395           INC     IX
'03AB  DD23      0396           INC     IX
'03AD  DD229003' 0397           LD      (KEYPTR),IX      ;SETUP FOR DATA CHG
'03B1  C39303'   0398 CCS8A:    JP      DISUP
'03B4  DD21F402' 0399 ALTER:    LD      IX,DISMEM
'03B8  3A7303'   0400           LD      A,(PFLG)
'03BB  B7        0401           OR      A
'03BC  202A      0402           JR      NZ,ALTER2-$      ;PORT CHNG
   BE  3A0B03'   0403           LD      A,(RFLG)
'03C1  B7        0404           OR      A
'03C2  2035      0405           JR      NZ,ALTR3-$       ;REG CHANGE
'03C4  3AF002'   0406           LD      A,(ARFLG)
'03C7  B7        0407           OR      A
```

```
 ADDR  OBJECT    ST # SOURCE STATEMENT          DATASET = DK0:DECKY .SRC

'03C8  C24104'   0408         JP     NZ,ALTR4         ;ALT REG CHANGE
'03CB  3A9C03'   0409         LD     A,(MFLG)
'03CE  B7        0410         OR     A
'03CF  CAB802'   0411         JP     Z,RESTR1         ;FALSE ALARM RESTART
'03D2  CD9F03'   0412         CALL   UFOR2  ;GET ADDR INTO HL
'03D5  DD7E06    0413         LD     A,(IX+6)         ;GET HIGH NIBBLE
'03D8  CD8E04'   0414         CALL   ALTER7
'03DB  DDB607    0415         OR     (IX+7)  ;OR WITH LOW NIB
'03DE  77        0416         LD     (HL),A  ;WRITE TO MEMORY
'03DF  7E        0417         LD     A,(HL)  ;READ IT BACK
'03E0  DD21A403' 0418 ALTER1: LD     IX,DSMEM4
'03E4  CDA703'   0419         CALL   UFOR1   ;WRITE NEW DATA INTO DISMEM
'03E7  C9        0420         RET
'03E8  CDD303'   0421 ALTER2: CALL   UFOR2   ;GET ADDRESS INTO HL
'03EB  DD7E06    0422         LD     A,(IX+6)         ;GET ONE NIB
'03EE  CD8E04'   0423         CALL   ALTER7
'03F1  DDB607    0424         OR     (IX+7)  ;OR 1WTH LOW NIB
'03F4  4C        0425         LD     C,H
'03F5  ED79      0426         OUT    (C),A   ;WRITE TO PORT
'03F7  18E7      0427         JR     ALTER1-$
                 0428 ;MAIN REGISTER CHANGE
'03F9  CD6E04'   0429 ALTR3:  CALL   ALTER6
'03FC  380D      0430         JR     C,ALTR3A-$       ;SPEC REG LT 6?
'03FE  DD7E06    0431         LD     A,(IX+6)         ;GET HIGH NIBBLE
'0401  CD8E04'   0432         CALL   ALTER7
'0404  DDB607    0433         OR     (IX+7)  ;OR WITH HIGH NIBBLE
'0407  77        0434         LD     (HL),A  ;MODIFY REGS ON STACK
'0408  C3E003'   0435         JP     ALTER1
'040B  FE03      0436 ALTR3A: CP     3       ;IS IT KEY 3?
'040D  2824      0437         JR     Z,ALTR3B-$       ;YES, IFF
'040F  FE02      0438         CP     2       ;IS IT KEY 2?
'0411  282A      0439         JR     Z,ALTR3C-$       ;YES, SP
'0413  DD7E04    0440         LD     A,(IX+4)         ;GET HIGH NIBBLE
'0416  CD8E04'   0441         CALL   ALTER7
'0419  DDB605    0442         OR     (IX+5)  ;OR WITH LOW NIBBLE
'041C  23        0443         INC    HL
'041D  77        0444         LD     (HL),A  ;MODIFY HGIH BYTE
'041E  DD217803' 0445         LD     IX,DSMEM2
'0422  CDE503'   0446         CALL   UFOR1   ;WRITE DATA TO DISPLAY
'0425  DD7E04    0447         LD     A,(IX+4)         ;GET HIGH NIBBLE
'0428  CD8E04'   0448         CALL   ALTER7
'042B  DDB605    0449         OR     (IX+5)  ;OR WITH LOW NIBBLE
'042E  2B        0450         DEC    HL
'042F  77        0451         LD     (HL),A  ;MODIFY HIGH BYTE
'0430  C3E003'   0452         JP     ALTER1
'0433  3A6003'   0453 ALTR3B: LD     A,(DSMEM7)       ;GET NEW VALUE
'0436  325A03'   0454         LD     (UIF),A ;MODIFY UIF
'0439  32FFFF    0455         LD     (DSMEM5),A       ;WRITE TO DISPLAY
'043C  C9        0456         RET
                 0457 ;SP CHANGE IS ILLEGAL
'043D  E1        0458 ALTR3C: POP    HL      ;DUMMY TO SIM RET
'043E  C3D003'   0459         JP     RESTR1
                 0460 ;ALTERNATE REGISTER CHANGE
'0441  DD21B603' 0461 ALTR4:  LD     IX,DISMEM        ;POINT TO REG
'0445  CD5504'   0462         CALL   ALTER5
'0448  DD7E06    0463         LD     A,(IX+6)         ,GET HIGH NIBBLE
'044B  CD8E04'   0464         CALL   ALTER7
'044E  DDB607    0465         OR     (IX+7)  ;OR WITH LOW NIBBLE
```

```
 ADDR  OBJECT    ST # SOURCE STATEMENT         DATASET = DK0:DECKY .SRC

'0451  77        0466          LD     (HL),A   ;MODIFY REG ON STK
'0452  C3E003'   0467          JP     ALTER1   ;GO DISPLAY
'0455  DD5E00    0468 ALTER5:  LD     E,(IX+0)         ;GET REGISTER
'0458  0600      0469          LD     B,00H
'045A  1600      0470          LD     D,00H
'045C  21FFFF    0471          LD     HL,REGTBP        ;POINT TO ALT TABLE
'045F  19        0472          ADD    HL,DE    ;ADD KEYVALUE OFFSET
'0460  4E        0473          LD     C,(HL)   ;GET VALUE
'0461  79        0474          LD     A,C
'0462  FE19      0475          CP     25D      ;ILLEGAL VALUE?
'0464  CA3F04'   0476          JP     Z,RESTR1         ;YES, RESTART MONITOR
'0467  2A4603'   0477          LD     HL,(STKPT)       ;POINT TO USER'S STACK
'046A  B7        0478          OR     A        ;CLEAR CARRY
'046B  ED42      0479          SBC    HL,BC    ;SUB OFFSET FROM SP
'046D  C9        0480          RET
'046E  DD214304' 0481 ALTER6:  LD     IX,DISMEM        ;POINT TO REGISTER
'0472  DD5E00    0482          LD     E,(IX)   ;GET REGISTER
'0475  0600      0483          LD     B,00H
'0477  1600      0484          LD     D,00H
'0479  21FFFF    0485          LD     HL,REGTB         ;POINT TO MAIN TABLE
'047C  19        0486          ADD    HL,DE    ;ADD KEYVALUE OFFSET
'047D  4E        0487          LD     C,(HL)   ;GET VALUE
'047E  79        0488          LD     A,C
'047F  FE19      0489          CP     25D      ;ILLEGAL VALUE?
'0481  CA6504'   0490          JP     Z,RESTR1         ;IF SO RESTART
'0484  2A6804'   0491          LD     HL,(STKPT)       ;POINT TO USER'S STACK
'0487  7B        0492          LD     A,E      ;GET KEYVALUE AGAIN
'0488  B7        0493          OR     A        ;CLEAR CARRY
'0489  ED42      0494          SBC    HL,BC    ;SUB OFFSET FROM SP
'048B  FE06      0495          CP     6        ;SPEC REG?
'048D  C9        0496          RET
'048E  CB27      0497 ALTER7:  SLA    A
'0490  CB27      0498          SLA    A
'0492  CB27      0499          SLA    A
'0494  CB27      0500          SLA    A
'0496  C9        0501          RET
                 0502 ;BREAKPOINT INSTALLATION
                 0503 ;MAKES AN ENTRY INTO BREAKPOINT TABLE IF ENOUGH SPACE
                 0504 ;EXISTS.  THE ACTUAL BREAKPOINTS ARE PUT INTO RAM
                 0505 ;ON THE EXECUTE COMMAND.
'0497  3A9603'   0506 CCS9:    LD     A,(DIG4)         ;GET FLAG
'049A  B7        0507          OR     A        ;SET STATUS
'049B  2005      0508          JR     NZ,CCS90-$       ;FOUR DIGITS IN?
'049D  327502'   0509          LD     (BFLG),A         ;;NO CLEAR ALL BP'S
'04A0  1810      0510          JR     CCS91-$
'04A2  CDE903'   0511 CCS90:   CALL   UFOR2    ;GET BP ADDR INTO HL
'04A5  CD4802'   0512          CALL   UFOR3    ;GET BP STATUS
'04A8  2810      0513          JR     Z,CCS9B-$        ;NO BP, INSTALL ONE
'04AA  78        0514          LD     A,B
'04AB  FE05      0515          CP     05       ;TABLE FULL?
'04AD  2006      0516          JR     NZ,CCS9A-$       ;NO GO ON
'04AF  C38204'   0517          JP     RESTR1   ;YES, CLEAR DISPLAY
'04B2  C3B203'   0518 CCS91:   JP     DISUP
                 0519 ;FIND FIRST FREE SPACE IN TABLE
'04B5  CD6002'   0520 CCS9A:   CALL   UIX3     ;IX+3 AND B-1
'04B8  20FB      0521          JR     NZ,CCS9A-$       ;LOOP TIL FREE SPC
                 0522 ;INSERT NEW BREAKPOINTS IN TABLE
'04BA  3A9E04'   0523 CCS9B:   LD     A,(BFLG)
```

```
 ADDR  OBJECT    ST # SOURCE STATEMENT          DATASET = DK0:DECKY .SRC

'04BD  3C        0524         INC     A
'04BE  32BB04'   0525         LD      (BFLG),A         ;INCREMENT FLAG
'04C1  DD7400    0526         LD      (IX),H   ;WRITE TO TABLE
'04C4  DD7501    0527         LD      (IX+1),L
'04C7  C3B304'   0528         JP      DISUP
                 0529 ;PUNCH DATA TO CASSETTE INTERFACE IN KC FORMAT
                 0530 ;STARTING ADDRESS IN DE', ENDING ADDRRESS IN HL'.
                 0531 ;INTERRUPTS ARE LIVE DURING THIS ROUTINE.
'04CA  CDFFFF    0532 CCS10:  CALL    UFOR4    ;CLEAR DISPLAY
'04CD  3E01      0533         LD      A,1
'04CF  32FFFF    0534         LD      (FLG24),A        ;SET FLG24 FOR MARKS
'04D2  ED5E      0535         IM      2
'04D4  01FFFF    0536         LD      BC,CTC1P
'04D7  78        0537         LD      A,B
'04D8  ED47      0538         LD      I,A
'04DA  79        0539         LD      A,C
'04DB  D384      0540         OUT     (CTC0),A         ;INTERRUPT VECTOR
'04DD  3E85      0541         LD      A,85H
'04DF  D385      0542         OUT     (CTC1),A         ;CTC1 TO INTR
'04E1  3E1A      0543         LD      A,26D
'04E3  D385      0544         OUT     (CTC1),A         ;WITH THIS TIME CONST
'04E5  3AFFFF    0545         LD      A,(PUNHSH)
'04E8  57        0546         LD      D,A
'04E9  3AFFFF    0547         LD      A,(PUNHSL)       ;GET STARTING ADDR
'04EC  5F        0548         LD      E,A
'04ED  3AFFFF    0549         LD      A,(PUNHEH)
'04F0  67        0550         LD      H,A
'04F1  3AFFFF    0551         LD      A,(PUNHEL)       ;GET ENDING ADDE
'04F4  6F        0552         LD      L,A
'04F5  AF        0553         XOR     A        ;CLEAR CARRY
'04F6  ED52      0554         SBC     HL,DE    ;CALCULATE BLOCK SIZE
'04F8  23        0555         INC     HL
'04F9  E5        0556         PUSH    HL       ;SAVE IT
'04FA  210000    0557         LD      HL,0000H
'04FD  0603      0558         LD      B,3D     ;SETUP FOR 40 SEC DELAY
'04FF  FB        0559         EI
'0500  76        0560 CCS10A  HALT             ;WAIT FOR CTC1 INTR
                 0561 ;*****************
'0501  2D        0562         DEC     L
'0502  20FC      0563         JR      NZ,CCS10A-$
'0504  25        0564         DEC     H
'0505  20F9      0565         JR      NZ,CCS10A-$      ;DELAY LOOP
'0507  10F7      0566         DJNZ    CCS10A-$         ;40 SEC FOR LEADER
'0509  3E3A      0567 CCS10B: LD      A,03AH   ;START WITH A COLON
'050B  CDFFFF    0568         CALL    OTCHR1   ;OUTPUT COLON
'050E  AF        0569         XOR     A        ;CLEAR CARRY,INTRPTS DISABLED
'050F  011000    0570         LD      BC,10H
'0512  E1        0571         POP     HL
'0513  ED42      0572         SBC     HL,BC    ;COMPARE HL WITH 10
'0515  3009      0573         JR      NC,CCS10C-$      ;NC=>HL>10H
'0517  09        0574         ADD     HL,BC    ;RESTORE BLOCK SIZE<10H
'0518  85        0575         ADD     A,L      ;SET STATUS IF L=0
'0519  47        0576         LD      B,A
'051A  2E00      0577         LD      L,0
'051C  283C      0578         JR      Z,CCS10F-$
'051E  1801      0579         JR      CCS10D-$
'0520  41        0580 CCS10C: LD      B,C      ;B=BLOCK SIZE
'0521  E5        0581 CCS10D: PUSH    HL       ;SAVE
```

```
 ADDR   OBJECT      ST # SOURCE STATEMENT          DATASET = DKO:DECKY .SRC

'0522   0E00        0582           LD     C,0     ;CLEAR CHECKSUM
'0524   78          0583           LD     A,B     ;PUNCH RECORD LENGTH
'0525   CDFFFF      0584           CALL   UPACCS
'0528   3AE604'     0585           LD     A,(PUNHSH)
'052B   67          0586           LD     H,A
'052C   CD2605'     0587           CALL   UPACCS
'052F   3AEA04'     0588           LD     A,(PUNHSL)        ;PUNCH HIGH BYTE FIRST
'0532   6F          0589           LD     L,A
'0533   CD2D05'     0590           CALL   UPACCS
'0536   AF          0591           XOR    A       ;ACC=0,PUNCH RECORD TYPE
'0537   CD3405'     0592           CALL   UPACCS
'053A   7E          0593 CCS10E:   LD     A,(HL)  ;PUNCH DATA
'053B   CD3805'     0594           CALL   UPACCS
'053E   23          0595           INC    HL
'053F   10F9        0596           DJNZ   CCS10E-$
'0541   97          0597           SUB    A
'0542   91          0598           SUB    C
'0543   CD3C05'     0599           CALL   UPACCS  ;PUNCH CKECKSUM
'0546   3E0D        0600           LD     A,0DH   ;CR
'0548   CD4405'     0601           CALL   UPACCS
'054B   3E0A        0602           LD     A,0AH   ;LF
'054D   CD4905'     0603           CALL   UPACCS
'0550   7C          0604           LD     A,H
'0551   322905'     0605           LD     (PUNHSH),A        ;SAVE HIGH BYTE
'0554   7D          0606           LD     A,L
'0555   323005'     0607           LD     (PUNHSL),A        ;SAVE LOW BYTE
'0558   18AF        0608           JR     CCS10B-$          ;PUNCH SOME MORE
'055A   0603        0609 CCS10F:   LD     B,3
'055C   AF          0610 CCS10G:   XOR    A
'055D   4F          0611           LD     C,A
'055E   CD4E05'     0612           CALL   UPACCS  ;PUNCH EOF
'0561   10F9        0613           DJNZ   CCS10G-$
'0563   3E01        0614           LD     A,1
'0565   CD5F05'     0615           CALL   UPACCS
'0568   97          0616           SUB    A
'0569   91          0617           SUB    C
'056A   CD6605'     0618           CALL   UPACCS  ;CKSUM FOR EOF
'056D   21FFFF      0619 CCS10H:   LD     HL,0FFFFH         ;SETUP 5 SEC TRAILER
'0570   FB          0620           EI             ;ENABLE INTERRUPTS
'0571   76          0621           HALT
                    0622 ;*************
'0572   2D          0623 CCS10J:   DEC    L
'0573   20FD        0624           JR     NZ,CCS10J-$
'0575   25          0625           DEC    H
'0576   20FA        0626           JR     NZ,CCS10J-$
'0578   F3          0627           DI
'0579   3E03        0628           LD     A,03H
'057B   D385        0629           OUT    (CTC1),A          ;DISABLE CTC KILL TONE
'057D   F3          0630           DI             ;DISABLE INTERRUPTS
'057E   C3B004'     0631           JP     RESTR1  ;RESTART MONITOR
                    0632 ;LOAD DATA FROM CASSETTE TAPE TO MEMORY (KC FORMAT)
'0581   ED5E        0633 CCS11:    IM     2       ;SELECT MODE TWO INTERRUPTS
'0583   21FF0F      0634           LD     HL,0FFFH          ;START 15 SEC DELAY
'0586   0620        0635           LD     B,20H
'0588   2D          0636 CCS11A:   DEC    L
'0589   20FD        0637           JR     NZ,CCS11A-$
'058B   25          0638           DEC    H
'058C   20FA        0639           JR     NZ,CCS11A-$
```

```
'058E  10F8      0640         DJNZ    CCS11A-$
                 0641 ;NOW INTO LEADER - START MAIN LOOP
'0590  CDFFFF    0642 CCS11G: CALL    INCHR   ;GET FIRST CHARS
'0593  D63A      0643         SUB     03AH    ;CHECK FOR COLON
'0595  20F9      0644         JR      NZ,CCS11G-$     ;LOOP BACK
'0597  4F        0645         LD      C,A     ;ZERO CHECKSUM
'0598  CDFFFF    0646         CALL    ULACC   ;GET RECORD LENGTH AND CKSUM
'059B  47        0647         LD      B,A
'059C  CD9905'   0648         CALL    ULACC   ;GET HIGH BYTE OF ADDR
'059F  57        0649         LD      D,A
'05A0  CD9D05'   0650         CALL    ULACC   ;GET LOW BYTE OF ADDR
'05A3  5F        0651         LD      E,A
'05A4  CDA105'   0652         CALL    ULACC   ;GET RECORD TYPE
'05A7  3D        0653         DEC     A       ;RECORD TYPE=EOF?
'05A8  F5        0654         PUSH    AF
'05A9  2807      0655         JR      Z,CCS11J-$      ;JP IF YES
'05AB  CDA505'   0656 CCS11H: CALL    ULACC   ;GET DATA
'05AE  12        0657         LD      (DE),A  ;STORE IT
'05AF  13        0658         INC     DE
'05B0  10F9      0659         DJNZ    CCS11H-$        ;LOOP FOR MORE
'05B2  CDAC05'   0660 CCS11J: CALL    ULACC   ;GET CHECKSUM
'05B5  AF        0661         XOR     A       ;CLEAR ACC
'05B6  81        0662         ADD     A,C
'05B7  2814      0663         JR      Z,CCS11K-$      ;CKSUM OK
                 0664 ;ERROR REPORTING
'05B9  DD217004' 0665         LD      IX,DISMEM       ;POINT TO BUFFER
'05BD  7A        0666         LD      A,D
'05BE  CD2304'   0667         CALL    UFOR1   ;DISPLAY HIGH BYTE
'05C1  DD212004' 0668         LD      IX,DSMEM2
'05C5  7B        0669         LD      A,E
'05C6  CDBF05'   0670         CALL    UFOR1   ;DISPLAY LOW BYTE
'05C9  F1        0671         POP     AF      ;RESTORE SP
'05CA  C3C804'   0672         JP      DISUP   ;GO DISPLAY
'05CD  F1        0673 CCS11K: POP     AF      ;EST FOR EOF
'05CE  20C0      0674         JR      NZ,CCS11G-$     ;GO LOOK OF NEXT RECORD
'05D0  C37F05'   0675         JP      RESTR1  ;RESTART MONITOR
                 0676 ;PROM PROGRAMMER FOR 2758 AND 2716 PROMS
                 0677 ;MOVES DATA FOR RAM STARTING AT 2000H TO PROM
                 0678 ;STARTING AND 1000H.  NUMBER OF BYTES TO BE
                 0679 ;TRANSFERED (EXPRESSED IN FOUR HEX DIGITS) IS
                 0680 ;IN DISPLAY BUFFER (DISMEM).
'05D3  3E01      0681 CCS12:  LD      A,01H
'05D5  32B302'   0682         LD      (PRFLG),A       ;SET PROM PROG FLG
'05D8  CDA304'   0683         CALL    UFOR2           ;GET BYTE COUNT
'05DB  E5        0684         PUSH    HL      ;SAVE IT
'05DC  C1        0685         POP     BC
'05DD  E5        0686         PUSH    HL
'05DE  210020    0687         LD      HL,2000H        ;RAM SOURCE DATA
'05E1  110010    0688         LD      DE,1000H        ;PROM DEST ADDR
'05E4  3E25      0689 CCS12A: LD      A,25H   ;CTC FOR 26MS ZC/TO2
'05E6  D386      0690         OUT     (CTC2),A        ;NO INTR
'05E8  3ECB      0691         LD      A,203D
'05EA  D386      0692         OUT     (CTC2),A        ;TIME CONST
'05EC  3E80      0693         LD      A,80H   ;CLEAR DISPLAY SET
'05EE  D38C      0694         OUT     (DIGLH),A       ;PROM PROG EN=1
'05F0  EDA0      0695         LDI             ;WAIT STATE INSERTED UNTIL
'05F2  3E00      0696         LD      A,00H   ;CTC2 TIMES OUT TWICE
'05F4  D38C      0697         OUT     (DIGLH),A       ;CLEAR PROM PROG EN
```

```
 ADDR   OBJECT     ST # SOURCE STATEMENT          DATASET = DKO:DECKY .SRC

'05F6   3E03       0698           LD      A,03H    ;RESET CTC2
'05F8   D386       0699           OUT     (CTC2),A
'05FA   EAE405'    0700           JP      PE,CCS12A        ;LOOPBACK IF BC-1 NE 0
                   0701 ;VERIFY PROM IS LIKE RAM
'05FD   C1         0702           POP     BC       ;RESTORE BYTE COUNT
'05FE   210020     0703           LD      HL,2000H         ;HL PTS TO RAM
'0601   110010     0704           LD      DE,1000H         ;DE PTS TO PROM
'0604   1A         0705 CCS12B:   LD      A,(DE)   ;GET PROM DATA
'0605   EDA1       0706           CPI              ;COMPARE WITH RAM
'0607   2006       0707           JR      NZ,CCS12C-$      ;ERROR REPORT IT
'0609   E2D105'    0708           JP      PO,RESTR1        ;NO ERRORS - RETURN
'060C   13         0709           INC     DE       ;UPDATE DE
'060D   18F5       0710           JR      CCS12B-$         ;CHECK NEXT BYTE
                   0711 ;ERROR HANDLER
'060F   F5         0712 CCS12C:   PUSH    AF       ;PRESERVE ERROR DATA
'0610   C5         0713           PUSH    BC       ;PRESERVE BYTE COUNT
'0611   D5         0714           PUSH    DE       ;PRESERVE ERROR ADDR
'0612   D9         0715           EXX              ;PRESERVE MAIN REGISTERS
'0613   D1         0716           POP     DE       ;DE=ERROR ADDR IN PROM
'0614   C1         0717           POP     BC       ;BC=BYTE COUNT
'0615   DD21BB05'  0718 CCS12E    LD      IX,DISMEM
'0619   7A         0719           LD      A,D
'061A   CDC705'    0720           CALL    UFOR1    ;HIGH BYTE TO DISMEM
'061D   DD21C305'  0721           LD      IX,DSMEM2
'0621   7B         0722           LD      A,E
'0622   CD1B06'    0723           CALL    UFOR1    ;LOW BYTE TO DISMEM
'0625   DD21E203'  0724           LD      IX,DSMEM4
'0629   F1         0725           POP     AF       ;RETRIEVE ERROR DATA
'062A   CD2306'    0726           CALL    UFOR1    ;ERROR DATA TO DISMEM
'062D   C3CB05'    0727           JP      DISUP    ;DISPLAY IT
                   0728 ;NEXT KEY INPUT DURING ERROR DISPLAY
'0630   D9         0729 CCS12D:   EXX              ;GET BACK POINTERS
'0631   13         0730           INC     DE       ;SETUP FOR NEXT ERROR
'0632   18D0       0731           JR      CCS12B-$         ;LOOP FOR MORE ERRS
                   0732 ;*****EQUATES
 >008C             0733 DIGLH     EQU     8CH      ;WRITE ONLY DIGIT SEL
 >0090             0734 KBSEL     EQU     90H      ;READ ONLY KB SEL
 >0084             0735 CTC0:     EQU     84H      ; INTRPT VECTOR
 >0085             0736 CTC1:     EQU     85H      ;CASSETTE INTERFACE-PUNCH
 >0086             0737 CTC2:     EQU     86H      ;SS/PROM PROGRAMMER
 >0087             0738 CTC3:     EQU     87H      ;CASSETTE INTERFACE-LOAD
 >0088             0739 SEGLH:    EQU     88H      ;WRITE ONLY SEGMENT LATCH
                   0740           END


ERRORS=0000
```

```
                  0002            NAME     UTIL
                  0003 ;UTILITY PACKAGE
                  0004 ;VERSION 1.5  5/13/78
 >0634            0005            ORG      0634H
                  0006            PSECT    REL
                  0007            GLOBAL   DIG2
                  0008            GLOBAL   DIG4
                  0009            GLOBAL   DIG8
                  0010            GLOBAL   UIX3
                  0011            GLOBAL   UFOR1
                  0012            GLOBAL   D20MS
                  0013            GLOBAL   UFOR2
                  0014            GLOBAL   UFOR3
                  0015            GLOBAL   UFOR4
                  0016            GLOBAL   BPTAB
                  0017            GLOBAL   SSFLG
                  0018            GLOBAL   UABIN
                  0019            GLOBAL   UBASC
                  0020            GLOBAL   UPACC
                  0021            GLOBAL   UPACCS
                  0022            GLOBAL   ULACC
                  0023            GLOBAL   OTCHR
                  0024            GLOBAL   KYTBL
                  0025            GLOBAL   REGTB
                  0026            GLOBAL   REGTBP
                  0027            GLOBAL   DISMEM
                  0028            GLOBAL   STKPT
                  0029            GLOBAL   SEGPT
                  0030            GLOBAL   KEYPTR
                  0031            GLOBAL   OTCHR
                  0032            GLOBAL   OTCHR1
                  0033            GLOBAL   INCHR
                  0034            GLOBAL   FLG24
                  0035            GLOBAL   RFLG
                  0036            GLOBAL   ARFLG
                  0037            GLOBAL   BFLG
                  0038            GLOBAL   PFLG
                  0039            GLOBAL   CTC1P
                  0040            GLOBAL   MFLG
                  0041            GLOBAL   PRFLG
                  0042            GLOBAL   CTC3L
                  0043            GLOBAL   CTC1P
                  0044            GLOBAL   UFGCR
                  0045            GLOBAL   OTCHR6
                  0046            GLOBAL   INCHR4
                  0047 ;ADDS THREE TO IX AND DECREMENTS B
                  0048 ;USED IN MANIPULATING TNE TABLE HOLDING
                  0049 ;BREAKPOINT ADDRESSES AND USER OP CODES
'0634  DD23       0050 UIX3:      INC      IX
'0636  DD23       0051            INC      IX
'0638  DD23       0052            INC      IX
'063A  05         0053            DEC      B
'063B  C9         0054            RET               ;USER TO CHECK FOR BP OVRFLW
                  0055 ;IX POINTS TO NEXT TWO LOCATIONS IN DISMEM TO BE
                  0056 ;WRITTEN INTO.  A CONTAINS THE DATA AND UFOR1 DOES
                  0057 ;THE WRITING.
'063C  47         0058 UFOR1:     LD       B,A      ;SAVE A IN TEMP REG
'063D  E60F       0059            AND      0FH      ;MASK OUT HIGH NIBBLE
```

```
'063F  DD7701     0060           LD      (IX+1),A         ;PUT IN LOW NIBBLE
'0642  78         0061           LD      A,B      ;GET FULL BYTE
'0643  CB3F       0062           SRL     A
'0645  CB3F       0063           SRL     A
'0647  CB3F       0064           SRL     A
'0649  CB3F       0065           SRL     A        ;HIGH NIBBLE TO LOW
'064B  DD7700     0066           LD      (IX+0),A         ;PUT HIGH NIBBLE IN DISMEM
'064E  C9         0067           RET
                  0068 ;DELAY 20 MILLISECONDS AND RETURN
'064F  21FF08     0069 D20MS:    LD      HL,08FFH
'0652  2D         0070 D20MS1:   DEC     L
'0653  20FD       0071           JR      NZ,D20MS1-$
'0655  25         0072           DEC     H
'0656  20FA       0073           JR      NZ,D20MS1-$
'0658  C9         0074           RET
                  0075 ;FORMATS FIRST FOUR DIGITS IN DISMEM INTO AN ADDRESS IN
                  0076 ;HL, USED BY MEMORY AND PORT UPDATE COMMANDS.
'0659  DD21FFFF   0077 UFOR2:    LD      IX,DISMEM        ;POINTER TO FIRST DIGIT
'065D  DD7E00     0078           LD      A,(IX)   ;GET FIRST DIGIT
'0660  CB27       0079           SLA     A
'0662  CB27       0080           SLA     A
'0664  CB27       0081           SLA     A
'0666  CB27       0082           SLA     A        ;MOVE TO HIGH NIBBLE IN A
'0668  DDB601     0083           OR      (IX+1)   ;BRING IN LOW NIBBLE
'066B  67         0084           LD      H,A      ;SAVE
'066C  DD7E02     0085           LD      A,(IX+2)         ;SECOND BYTE HIGH NIBBLE
'066F  CB27       0086           SLA     A
'0671  CB27       0087           SLA     A
'0673  CB27       0088           SLA     A
'0675  CB27       0089           SLA     A
'0677  DDB603     0090           OR      (IX+3)   ;BRING IN LOW NIBBLE
'067A  6F         0091           LD      L,A      ;COMPLETE POINTER
'067B  C9         0092           RET
                  0093 ;GETS TABLE ADDRESS INTO IX AND BFLG INTO B
'067C  DD21FFFF   0094 UFOR3:    LD      IX,BPTAB         ;POINT IX TO START OF TABL
'0680  3AFFFF     0095           LD      A,(BFLG)         ;NO. OF BP ACTIVE
'0683  B7         0096           OR      A        ;SET STATUS
'0684  47         0097           LD      B,A
'0685  C9         0098           RET              ;USER TESTS STATUS
                  0099 ;FLAG INITALAZATION ROUTINE
'0686  215B06'    0100 UFGCR:    LD      HL,DISMEM
'0689  22FFFF     0101           LD      (KEYPTR),HL      ;POINT INTO DISMEM
'068C  3E00       0102           LD      A,00H
'068E  32FFFF     0103           LD      (DIG2),A         ;ZERO FLAGS
'0691  32FFFF     0104           LD      (DIG4),A
'0694  32FFFF     0105           LD      (SSFLG),A
'0697  32FFFF     0106           LD      (PRFLG),A
'069A  32FFFF     0107           LD      (PFLG),A
'069D  32FFFF     0108           LD      (MFLG),A
'06A0  32FFFF     0109           LD      (RFLG),A
'06A3  32FFFF     0110           LD      (ARFLG),A
'06A6  C9         0111           RET
                  0112 ;PUTS BLANKS INTO DISPLAY MEMORY - DISMEM
'06A7  0608       0113 UFOR4:    LD      B,8
'06A9  218706'    0114           LD      HL,DISMEM
'06AC  3E10       0115           LD      A,10H    ;BLANK CODE
'06AE  77         0116 UFUR4A:   LD      (HL),A
'06AF  23         0117           INC     HL
```

```
'06B0  10FC      0118          DJNZ    UFOR4A-$         ;LOOP TILL DONE
'06B2  C9        0119          RET
                 0120 ;CONVERTS ONE ASCII DIGIT IN ACC TO BINARY
'06B3  D630      0121 UABIN:   SUB     030H
'06B5  FE0A      0122          CP      10
'06B7  F8        0123          RET     M
'06B8  D607      0124          SUB     7
'06BA  C9        0125          RET
                 0126 ;CONVERTS ONE DIGIT OF BINARY IN ACC TO ASCII
'06BB  E60F      0127 UBASC:   AND     0FH      ;MASK OUT HIGH DIGIT
'06BD  C690      0128          ADD     A,90H
'06BF  27        0129          DAA
'06C0  CE40      0130          ADC     A,40H
'06C2  27        0131          DAA
'06C3  C9        0132          RET
                 0133 ;PUNCHS TWO CHARACTERS IN ACCUMULATOR-HIGH NIBBLE FIRST
'06C4  D9        0134 UPACC:   EXX              ;USE ALT REGS IS OTCHR
'06C5  F5        0135          PUSH    AF       ;SAVE ACC
'06C6  0F        0136          RRCA
'06C7  0F        0137          RRCA
'06C8  0F        0138          RRCA
'06C9  0F        0139          RRCA             ;GET UPPER DIGIT
'06CA  CDF406'   0140          CALL    OTCHR
'06CD  F1        0141          POP     AF
'06CE  E60F      0142          AND     0FH      ;MASK OUT HIGH NIBBLE
'06D0  CDF406'   0143          CALL    OTCHR
'06D3  D9        0144          EXX              ;RESTORE REGISTERS
'06D4  C9        0145          RET
                 0146 ;PUNCHS TWO CHARACTERS IN ACCUMULATOR AND ADDS CHECKSUM
'06D5  F5        0147 UPACCS:  PUSH    AF       ;SAVE ACC
'06D6  81        0148          ADD     A,C      ;SUM CHECKSUM
'06D7  4F        0149          LD      C,A      ;SAVE IN C
'06D8  F1        0150          POP     AF
'06D9  18E9      0151          JR      UPACC-$ ;PUNCH ACC .
                 0152 ;READS TWO CHARACTERS FROM TAPE AND CONVERTS TO BINARY
                 0153 ;DATA RETURNED IN ACC AND CHECKSUM KEPT IN C
'06DB  C5        0154 ULACC    PUSH    BC       ;SAVE C REG (CHECKSUM)
'06DC  CD5807'   0155          CALL    INCHR    ;READ CHAR FIRST DIGIT
'06DF  CDB306'   0156          CALL    UABIN    ;CONVERT TO BINARY
'06E2  07        0157          RLCA
'06E3  07        0158          RLCA
'06E4  07        0159          RLCA
'06E5  07        0160          RLCA             ;SHIFT TO HIGH NIBBLE
'06E6  4F        0161          LD      C,A      ;SAVE FIRST DIGIT
'06E7  CD5807'   0162          CALL    INCHR    ;GET SECOND DIGIT
'06EA  CDB306'   0163          CALL    UABIN    ;CONVERT
'06ED  B1        0164          OR      C        ;MERGE NIBBLES
'06EE  C1        0165          POP     BC       ;RESTORE CHECKSUMS
'06EF  F5        0166          PUSH    AF       ;SAVE ACC
'06F0  81        0167          ADD     A,C      ADD TWO DIGITS TO CKSUM
'06F1  4F        0168          LD      C,A      ;SAVE IN C
'06F2  F1        0169          POP     AF       ;RESTORE ACC
'06F3  C9        0170          RET
                 0171 ;ROUTINE USES CTC CHANNEL 1 AS BASIC TIME BASE (4800HZ
                 0172 ;OR 2400HZ).  ENTRY IS WITH ONE BINARY CHARACTER IN THE
                 0173 ;LOW NIBBLE OF A.  THIS ROUTINE WILL CONVERT TO ASCII AND
                 0174 ;THE SHIFT THE CHARACTER OUT WITH ONE START AND TWO STOP
                 0175 ;BITS.  OUTPUTS ARE PULSES (ZC/TO) FROM CTC1 AT TWICE THE
```

```
                  0176 ;KANSAS CITY STANDARD RATES (1=4800HZ, 0=2400HZ).
'04F4  CDBB06'    0177 OTCHR:   CALL    UBASC   ;CONVERT TO ASCII
'   7  FB         0178 OTCHR1:  EI
'06F8  57         0179          LD      D,A      ;A TO D
'06F9  3E10       0180          LD      A,10H    ;INIT A FOR CTC INTR
'06FB  2E0A       0181          LD      L,0AH    ;BIT COUNT OF WORD
'06FD  CB12       0182          RL      D        ;CARRY TO D0
'06FF  FE01       0183 OTCHR2:  CP      01D      ;A=1? WAIT UNTIL
'0701  20FC       0184          JR      NZ,OTCHR2-$       ;CTC ON NEXT TO LAST CNT
'0703  47         0185          LD      B,A      ;SAVE A
'0704  3E00       0186          LD      A,0
'0706  32FFFF     0187          LD      (FLG24),A         ;CLEAR FLG24-START BIT
'0709  78         0188          LD      A,B
'070A  76         0189 OTCHR3:  HALT
                  0190 ;*********
'070B  37         0191          SCF              ;SET CARRY
'070C  CB1A       0192          RR      D        ;SHIFT D ONE RIGHT
'070E  2D         0193         .DEC     L
'070F  2007       0194          JR      NZ,OTCHR4-$
'0711  3E01       0195          LD      A,1
'0713  320707'    0196          LD      (FLG24),A         ;WORD OUT-MARK LINE
'0716  F3         0197          DI               ;EXIT WITH INTERRUPTS DISABLED
'0717  C9         0198          RET
'0718  FE01       0199 OTCHR4:  CP      1        ;NEXT TO LAST COUNT
'071A  20FC       0200          JR      NZ,OTCHR4-$       ;NO WAIT
'071C  CB42       0201          BIT     0,D      ;TEST NEXT BIT
'071E  2009       0202          JR      NZ,OTCHR5-$       ;NEXT BIT IS A ONE
'  20  47         0203          LD      B,A
'  21  3E00       0204          LD      A,0      ;NEXT BIT IS A ZERO
'0723  321407'    0205          LD      (FLG24),A         ;CLEAR FLG24
'0726  78         0206          LD      A,B
'0727  18E1       0207          JR      OTCHR3-$
'0729  47         0208 OTCHR5:  LD      B,A
'072A  3E01       0209          LD      A,1
'072C  322407'    0210          LD      (FLG24),A         ;SET FLG24
'072F  78         0211          LD      A,B               ;RESTORE A
'0730  18D8       0212          JR      OTCHR3-$          ;WAIT FOR END OF CHAR
                  0213 ;CTC1 INTERRUPT SERVICE ROUTINE DURING PUNCH
'0732  3D         0214 OTCHR6:  DEC     A        ;A IS CYCLE COUNTER
'0733  2020       0215          JR      NZ,OTCHR8-$       ;NOT LDAT COUNT,RETURN
'0735  DD212D07'  0216          LD      IX,FLG24
'0739  DDCB0046   0217          BIT     0,(IX+0)          ;TEST FLG24
'073D  200C       0218          JR      NZ,OTCHR7-$
'073F  3E85       0219          LD      A,85H    ;FLG24 IS CLR=ZERO
'0741  D385       0220          OUT     (CTC1),A          ;NEW CONTROL WORD
'0743  3E34       0221          LD      A,52D    ; INTERRUPTS LIVE
'0745  D385       0222          OUT     (CTC1),A          ;TIME CONST FOR 1200HZ
'0747  3E08       0223          LD      A,8      ;COUNT 8 CYCLES
'0749  180A       0224          JR      OTCHR8-$          ;RETURN
'074B  3E85       0225 OTCHR7:  LD      A,85H
'074D  D385       0226          OUT     (CTC1),A          ;NEW CONTROL WORD
'074F  3E1A       0227          LD      A,26D
'0751  D385       0228          OUT     (CTC1),A          ;TIME CONST FOR 2400HZ
'  53  3E10       0229          LD      A,16D    ;SETUP FOR 16 COUNTS
'0/55  FB         0230 OTCHR8:  EI
'0756  ED4D       0231          RETI
                  0232 ;INPUT BIT RATE HAS BEEN AVERAGED AND IS IN (BITRT)
                  0233 ;ON EXIT CHARACTER IS IN A
```

```
                      0234 ;REGISTERS USED ARE A,B,H
         FE Ø7        0235 ;INTERRUPTS ARE ENABLED AND CTC1 INTERRUPTS AT THE BIT RAT
'0758  21FFFF         0236 INCHR:   LD      HL,CTC3L
'075B  7C             0237          LD      A,H
'075C  ED47           0238          LD      I,A     ;SETUP I REGISTER
'075E  7D             0239          LD      A,L
'075F  D384           0240          OUT     (CTC0),A        ;CTC INTERRUPT VECTOR
'0761  0608           0241 INCHR1:  LD      B,8D    ;BIT COUNT FOR A WORD
'0763  FB             0242          EI
'0764  3E00           0243          LD      A,0
'0766  2600           0244          LD      H,0     ;CLEAR WORDS
'0768  DB90           0245 INCH1A:  IN      A,(KBSEL)       ;GET INPUT DATA
'076A  CB7F           0246          BIT     7,A
'076C  20FA           0247          JR      NZ,INCH1A-$     ;LOOP BACK FOR START BIT
'076E  3EA5           0248          LD      A,0A5H  ;INTR-256 PRESCALER
'0770  D387           0249          OUT     (CTC3),A        ;CTC3 CONTROL WORD
'0772  3E0D           0250          LD      A,0DH   ;DIVIDE BY 2 FOR MID OF BIT
'0774  D387           0251          OUT     (CTC3),A        ;CTC3 TIME CONSTANT
'0776  3EA5           0252          LD      A,0A5H
'0778  D387           0253          OUT     (CTC3),A
'077A  3E1A           0254          LD      A,01AH
'077C  D387           0255          OUT     (CTC3),A        ;NEXT ONE FULL BIT WIDTH
'077E  76             0256          HALT
                      0257 ;***************
'077F  CB7F           0258          BIT     7,A
'0781  2014           0259          JR      NZ,INCHR3-$     ;START BIT GONE-FALSE ST
'0783  76             0260 INCHR2:  HALT            ;WAIT FOR FIRST BIT
                      0261 ;***************
'0784  E680           0262          AND     80H     ;MASK OUT OTHER INPUTS
'0786  B4             0263          OR      H
'0787  67             0264          LD      H,A     ;SAVE
'0788  1018           0265          DJNZ INCHR5-$
'078A  CB7F           0266          BIT     7,A
'078C  2809           0267          JR      Z,INCHR3-$      ;FRAMING ERROR,RESTART
'078E  F3             0268          DI
'078F  3E03           0269          LD      A,03H
'0791  D387           0270          OUT     (CTC3),A        ;RESET CTC RTN WITH DATA
'0793  7C             0271          LD      A,H
'0794  E67F           0272          AND     7FH     ;MASK OUT START BIT
'0796  C9             0273          RET
'0797  3E03           0274 INCHR3:  LD      A,03H
'0799  D387           0275          OUT     (CTC3),A        ;RESET CTC3,FRAMING ERROR
'079B  18C4           0276          JR      INCHR1-$        ;GO LOOK FOR ANOTHER CHAR
                      0277 ;CTC INTERRUPT SERVICE ROUTINE DURING LOAD
'079D  DB90           0278 INCHR4:  IN      A,(KBSEL)       ;GET DATA
'079F  FB             0279          EI
'07A0  ED4D           0280          RETI
'07A2  CB0C           0281 INCHR5:  RRC     H
'07A4  18DD           0282          JR      INCHR2-$        ;WAIT FOR NEXT BIT
                      0283 ;**********EQUATES AND TABLES**********
                      0284 ;CTC READ ACCESSES DOWN COUNTER,WRITE SETS UP COUNTER
 >0084                0285 CTC0:    EQU     84H     ;RESERVED FOR USER
 >0085                0286 CTC1:    EQU     85H     ;AUDIO CASSETTE-PUNCH
 >0086                0287 CTC2:    EQU     86H     ;SINGLE STEP AND FROM PROGRAMME
 >0087                0288 CTC3:    EQU     87H     ;AUDIO CASSETTE-LOAD
 >0090                0289 KBSEL:   EQU     90H     ;CASSETTE DATA
                      0290 ;***SEVEN SEGMENT DISPLAY PATTERNS
'07A6  40             0291 SEGPT:   DEFB    40H     ;0
```

```
'07A7  79          0292            DEFB     79H       ;1
'07A8  24          0293            DEFB     24H       ;2
'07A9  30          0294            DEFB     30H       ;3
'07AA  19          0295            DEFB     19H       ;4
'07AB  12          0296            DEFB     12H       ;5
'07AC  02          0297            DEFB     02H       ;6
'07AD  78          0298            DEFB     78H       ;7
'07AE  00          0299            DEFB     00H       ;8
'07AF  18          0300            DEFB     18H       ;9
'07B0  08          0301            DEFB     08H       ;A
'07B1  03          0302            DEFB     03H       ;B LOWER CASE
'07B2  46          0303            DEFB     46H       ;C
'07B3  21          0304            DEFB     21H       ;D LOWER CASE
'07B4  06          0305            DEFB     06H       ;E
'07B5  0E          0306            DEFB     0EH       ;F
'07B6  7F          0307            DEFB     7FH       ;BLANK
'07B7  3F          0308            DEFB     3FH       ;PROMPT
'07B8  7D          0309            DEFB     7DH       ;PRIME MARK
                   0310 ;***KEY VALUE LOOKUP TABLE
'07B9  FF          0311 KYTBL:     DEFB     0FFH      ;0 B=01,A=0F
'07BA  EF          0312            DEFB     0EFH      ;1 B=02,A=0F
'07BB  F7          0313            DEFB     0F7H      ;2 B=02,A=17
'07BC  FB          0314            DEFB     0FBH      ;3 B=02,A=1B
'07BD  DF          0315            DEFB     0DFH      ;4 B=04,A=0F
'07BE  E7          0316            DEFB     0E7H      ;5 B=04,A=17
'07BF  EB          0317            DEFB     0EBH      ;6 B=04,A=1B
'07C0  CF          0318            DEFB     0CFH      ;7 B=08,A=0F
'07C1  D7          0319            DEFB     0D7H      ;8 B=08,A=17
'07C2  DB          0320            DEFB     0DBH      ;9 B=08,A=1B
'07C3  DD          0321            DEFB     0DDH      ;A B=08,A=1D
'07C4  ED          0322            DEFB     0EDH      ;B B=04,A=1D
'07C5  FD          0323            DEFB     0FDH      ;C B=02,A=1D
'07C6  0D          0324            DEFB     00DH      ;D B=01,A=1D
'07C7  0B          0325            DEFB     00BH      ;E B=01,A=1B
'07C8  07          0326            DEFB     07H       ;F B=01,A=17
'07C9  0E          0327            DEFB     0EH       ;EXEC B=01,A=1E
'07CA  FE          0328            DEFB     0FEH      ;SS B=02,A=1E
'07CB  EE          0329            DEFB     0EEH      ;MON    B=04,A=1E
'07CC  DE          0330            DEFB     0DEH      ;NEXT B=08,A=1E
'07CD  CD          0331            DEFB     0CDH      ;REG' DISP B=10,A=1D
'07CE  CB          0332            DEFB     0CBH      ;REG DISP B=10,A=1B
'07CF  C7          0333            DEFB     0C7H      ;PORT EXAM B=10,A=17
'07D0  BF          0334            DEFB     0BFH      ;MEM EXAM B=10,A=0F
'07D1  BD          0335            DEFB     0BDH      ;BP B=20,A=1D
'07D2  BB          0336            DEFB     0BBH      ;PUNCH B=20,A=1B
'07D3  B7          0337            DEFB     0B7H      ;LOAD B=20H,A=17
'07D4  AF          0338            DEFB     0AFH      ;PROG B=20,A=0F
                   0339 ;***REGTAB-RELATES KEYVALUE TO POSITION OF REGISTER ON STK
'07D5  19          0340 REGTB:     DEFB     25D       ;KEY 0 NU NO DISPLAY
'07D6  02          0341            DEFB     02D       ;KEY 1=PC
'07D7  02          0342            DEFB     2D        ;KEY 2=SP
'07D8  0C          0343            DEFB     12D       ;KEY 3=1FF DISPLAY UIF
'07D9  16          0344            DEFB     22D       ;KEY 4=IX
'07DA  18          0345            DEFB     24D       ;KEY 5=IY
'07DB  0B          0346            DEFB     11D       ;KEY 6=I
'07DC  09          0347            DEFB     9D        ;KEY 7=H
'07DD  0A          0348            DEFB     10D       ;KEY 8=L
'07DE  19          0349            DEFB     25D       ;KEY 9=NU NO DISP
```

```
'07DF  03         0350          DEFB    3D      ;KEY A=A
'07E0  05         0351          DEFB    5D      ;KEY B=B
'07E1  06         0352          DEFB    6D      ;KEY C=C
'07E2  07         0353          DEFB    7D      ;KEY D=D
'07E3  08         0354          DEFB    8D      ;KEY E=E
'07E4  04         0355          DEFB    4D      ;KEY F=F
                  0356 ;***ALTERNATE REGISTER SET
'07E5  19         0357 REGTBP:  DEFB    25D     ;KEY 0 NU NO DISPLAY
'07E6  19         0358          DEFB    25D     ;KEY 1 NU NO DISPLAY
'07E7  19         0359          DEFB    25D     ;KEY 2 NU NO DISPLAY
'07E8  19         0360          DEFB    25D     ;KEY 3 NU NO DISPLAY
'07E9  19         0361          DEFB    25D     ;KEY 4 NU NO DISPLAY
'07EA  19         0362          DEFB    25D     ;KEY 5 NU NO DISPLAY
'07EB  19         0363          DEFB    25D     ;KEY 6 NU NO DISPLAY
'07EC  13         0364          DEFB    19D     ;KEY 7=H'
'07ED  14         0365          DEFB    20D     ;KEY 8=L'
'07EE  19         0366          DEFB    25D     ;KEY 9=NU NO DISPLAY
'07EF  0D         0367          DEFB    13D     ;KEY A=A'
'07F0  0F         0368          DEFB    15D     ;KEY B=B'
'07F1  10         0369          DEFB    16D     ;KEY C=C'
'07F2  11         0370          DEFB    17D     ;KEY D=D'
'07F3  12         0371          DEFB    18D     ;KEY E=E'
'07F4  0E         0372          DEFB    14D     ;KEY F=F'
                  0373          END


ERRORS=0000
```

```
 ADDR  OBJECT    ST # SOURCE STATEMENT         DATASET = DKO:UTILR .SRC

                 0002          NAME    UTILR
                 0003 ;UTILTIY RAM AND CONSTANTS
                 0004 ;VERSION 1.2  5/13/78
>07F8            0005          ORG     07F8H
                 0006          PSECT ABS
                 0007          GLOBAL  CTC3L
                 0008          GLOBAL  CTC1P
                 0009          GLOBAL  OTCHR6
                 0010          GLOBAL  INCHR4
                 0011 ;CTC INTERRUPT VECTOR TABLE
 07F8  D623      0012 CTC0:    DEFW    CTC0V   ;MAP TO RAM
 07FA  FFFF      0013 CTC1P:   DEFW    OTCHR6  ;PUNCH VECTOR FA
>07FC            0014 CTC2:    DEFS    2       ;NOT USED FOR INTR
 07FE  FFFF-Ø79D 0015 CTC3L:   DEFW    INCHR4  ;LOAD VECTOR FE
>23C0            0016          ORG     23C0H
                 0017          GLOBAL  DIG2
                 0018          GLOBAL  DIG4
                 0019          GLOBAL  DIG8
                 0020          GLOBAL  BPTAB
                 0021          GLOBAL  SSFLG
                 0022          GLOBAL  UIF
                 0023          GLOBAL  DISMEM
                 0024          GLOBAL  DSMEM1
                 0025          GLOBAL  DSMEM2
                 0026          GLOBAL  DSMEM3
                 0027          GLOBAL  DSMEM4
                 0028          GLOBAL  DSMEM5
                 0029          GLOBAL  DSMEM6
                 0030          GLOBAL  DSMEM7
                 0031          GLOBAL  STKPT
                 0032          GLOBAL  STKPT1
                 0033          GLOBAL  RST16
                 0034          GLOBAL  RST24
                 0035          GLOBAL  RST32
                 0036          GLOBAL  RST40
                 0037          GLOBAL  RST48
                 0038          GLOBAL  RST56
                 0039          GLOBAL  KEYPTR
                 0040          GLOBAL  FLG24
                 0041          GLOBAL  RFLG
                 0042          GLOBAL  ARFLG
                 0043          GLOBAL  BFLG
                 0044          GLOBAL  PFLG
                 0045          GLOBAL  MFLG
                 0046          GLOBAL  PRFLG
                 0047          GLOBAL  PUNHSH
                 0048          GLOBAL  PUNHSL
                 0049          GLOBAL  PUNHEH
                 0050          GLOBAL  PUNHEL
                 0051 ;***RAM VARIABLES
>23C0            0052 PUNHSH:  DEFS     1      ;DUMP STARTING ADDR-HIGH BYTE
>23C1            0053 PUNHSL:  DEFS     1      ;DUMP STARTING ADDR-LOW BYTE
>23C2            0054 PUNHEH:  DEFS     1      ;DUMP ENDING ADDR-LOW BYTE
 23C3            0055 PUNHEL:  DEFS     1      ;DUMP ENDING ADDR-LOW BYTE
>23C4            0056 RST16:   DEFS     3      ;USER INSERTS JUMPS TO
>23C7            0057 RST24:   DEFS     3      ;HANDLE RESTART 16-56
>23CA            0058 RST32:   DEFS     3
>23CD            0059 RST40:   DEFS     3
```

```
UTILR  (C) 1978 MICRO DESIGN CONCEPTS   MOSTEK FLP-80 ASSEMBLER V2.0 PAGE 0002
 ADDR   OBJECT     ST # SOURCE STATEMENT        DATASET = DK0:UTILR .SRC

 >23D0             0060 RST48:  DEFS    3
 >23D3             0061 RST56:  DEFS    3
                   0062 ;***USER INSERTED JUMP FOR CTC0 INTERRUPT
 >23D6             0063 CTC0V:  DEFS    3        ;CTC0 INTR WILL BE VECTORED HERE
                   0064 ;***USER INSERTED JUMP FOR CTC3 INTERRUPT
 >23D9             0065 FLG24:  DEFS    1        ;FLAG FOR MARK (1) OUT (PUNCH)
 >23DA             0066 PRFLG:  DEFS    1        ;PROM PROGRAMMER FLAG
 >23DB             0067 KEYPTR: DEFS    2        ;PTR FOR NEXT WRITE INTO DISMEM
 >23DD             0068 UIF     DEFS    1        ;USERS IFF2
 >23DE             0069 PFLG    DEFS    1        ;PORT EXAMINE FLAG
 >23DF             0070 RFLG    DEFS    1        ;REGISTER EXAMINE FLAG
 >23E0             0071 ARFLG   DEFS    1        ;REGISTER EXAMINE FLAG (ALT)
 >23E1             0072 MFLG:   DEFS    1        ;MEMORY EXAMINE FLAG
 >23E2             0073 STKPT:  DEFS    1        ;USERS STACK POINTER-HIGH BYTE
 >23E3             0074 STKPT1: DEFS    1        ;USERS STACK POINTER-LOW BYTE
                   0075 ;**BREAKPOINT TABLE ORGANIZED AS TWO BYTES OF ADDR (H,L)
                   0076 ;**WHERE BREAKPOINT IS INSTALLED FOLLOWED BY THE ONE BYTE
                   0077 ;**OF THE OP CODE REPLACED BY THE RST 8 INSTRUCTION
 >23E4             0078 BPTAB:  DEFS    15       ;BP ADDR AND OP CODE REMOVED
 >23F3             0079 SSFLG   DEFS    1        ;SINGLE STEP MODE FLAG
 >23F4             0080 BFLG    DEFS    1        ;NO OF BREAKPOINTS INSTALLED
 >23F5             0081 DIG2    DEFS    1        ;2 DIGITS ENTERED FLAG
 >23F6             0082 DIG4    DEFS    1        ;4 DIGITS ENTERED FLAG
 >23F7             0083 DISMEM  DEFS    1        ;DISPLAY MEMORY BUFFER
 >23F8             0084 DSMEM1  DEFS    1
 >23F9             0085 DSMEM2  DEFS    1
 >23FA             0086 DSMEM3  DEFS    1
 >23FB             0087 DSMEM4  DEFS    1
 >23FC             0088 DSMEM5  DEFS    1
 >23FD             0089 DSMEM6  DEFS    1
 >23FE             0090 DSMEM7  DEFS    1
                   0091         END

ERRORS=0000
```

APPENDIX III

Soldering

and

Assembly Techniques

SOLDERING TECHNIQUE

THE NEED

The assembly of electronic components is essentially the exercise of the art of soldering. If the many connections are soldered properly, the resulting assembly will normally operate properly right from the first application of power. A hasty job here can mean endless hours trying to locate short circuits or intermittent connections.

THE SOLDER

Use a #20 gauge resin or rosin core solder with a ratio of 63% tin and 37% lead. A 60/40 ratio is acceptable. "Kester" and "Ersin" are two dependable brands of solder. Acid core solders or acid flux must not be used as they will corrode electronic joints and will damage printed circuit boards.

THE SOLDERING IRON

Use a small, 30 watt maximum iron with a small, chisel shaped tip. Too much heat will damage both components and boards. Soldering guns are too hot and should not be used. Heat the iron, wipe its tip quickly on the damp sponge, and apply a tiny amount of solder to the tip - just enough to make it silver in color but not so much that it will drip off. This cleaning procedure should be repeated whenever the solder of the tip of the soldering iron begins to thicken or take of a brownish color.

REMOVAL OF MULTI-PIN SOLDERED-IN PARTS

CAUTION

If for any reason, it becomes necessary to remove a soldered-in part having more than just two leads, do not try to remove the part intact. It can be done but only with great risk of damaging the printed circuit board in the process.

Hold the printed circuit board in well padded jaws of a bench vice to avoid damage.

REMOVAL OF SOLDERED-IN IC SOCKETS

Crush the plastic body with a pair of pliers to pull the pins from the body. Gently remove the pins from the top of the board with needle nosed pliers while touching the joint on the other side of the board with the tip of the iron. Do not use force. The pin will come out quite easily once the solder melts.

Clear the holes of any excess solder by rapidly inserting any removing a piece of wire while very briefly holding the soldering iron to the hole at the back of the board.

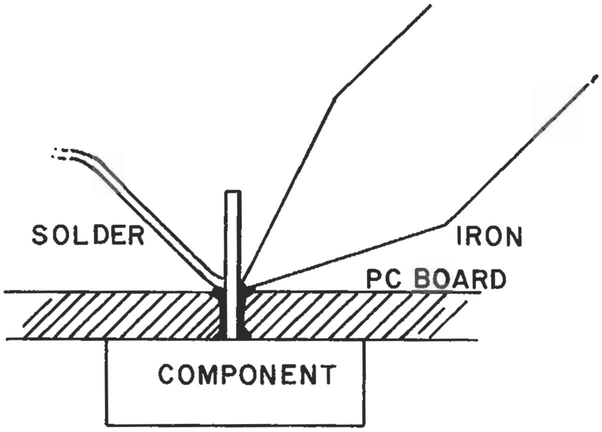
REMOVAL OF SOLDER-IN INTEGRATED CIRCUIT CHIPS

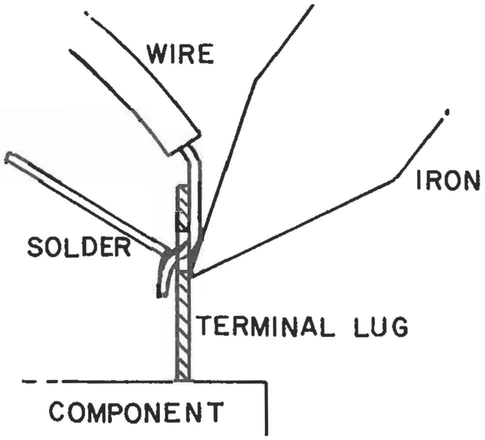
Cut each pin with a pair of diagonal cutters at a point between the chip and the printed circuit board which is as close to the chip as possible so that there is enough of the pin showing above the board to be grasped by needle nosed pliers while removing as described above.

THE PROCEDURE

The entire soldering operation should take little more than two seconds per joint. The sequence is as follows:

Touch the tip of the soldering iron to the joint, as shown below, so that both the conductors to be joined are simultaneously heated sufficiently to melt the solder.





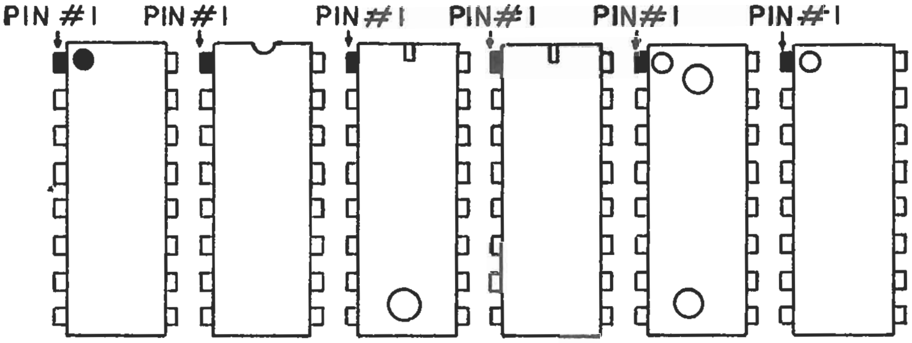
Touch the end of the solder roll to the joint, as shown above, just long enough to let no more than a 1/8" length melt into the joint. Too much solder will short circuit the bottom of the board or flow through the holes and short circuit the top of the board. The melted solder will appear wet and shiny. It will quickly flow completely around the wire and over the surface to which the wire is attached.

Remove the soldering iron as soon as both surfaces have been completely wetted. Remember, the total time from application to removal of the soldering iron should be only two or three seconds. Removal of the soldering iron too soon will result in an incomplete bond between the metals, but leaving the soldering iron at the joint too long will cause heat damage to both components and board.

ORIENTATION OF INTEGRATED CIRCUIT CHIPS

Extreme care must be taken to insure that each integrated circuit chip is so oriented, prior to insertion in its socket, that pin #1 is at the location so designated on the printed circuit board or in the individual assembly instructions for the kit.

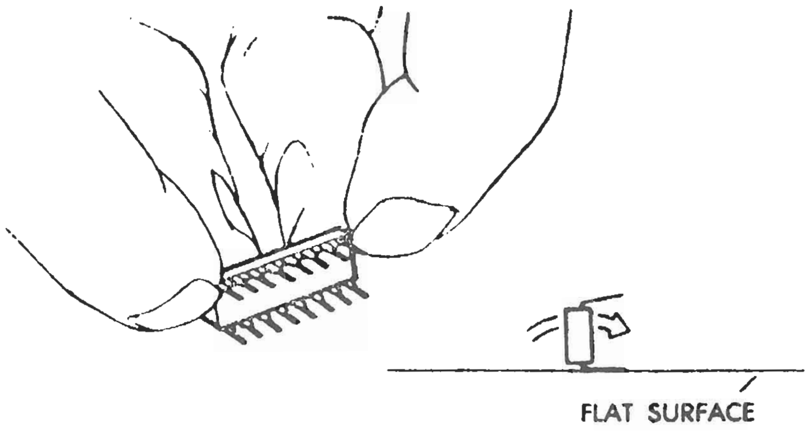
Pin #1 is, unfortunately, designated in a variety of ways depending upon the integrated circuit manufacturer. Several methods are indicated in the chart below. With the leads of the chip pointing away from the viewer, Pin #1 is in the position indicated with respect to the various end notches or tiny circular markings or depressions in one corner.



INSERTION OF INTEGRATED CIRCUIT CHIPS

Be sure all leads are straight and parallel. If not, gently straighten and align the bent pins with needle nosed pliers.

Integrated circuit chips usually come from the manufacturer with their rows of leads spread wider than the distance between rows of holes in the socket into which they are to be inserted. To slightly close the rows of pins in a uniform manner so they are aligned with the socket holes, place the chip on its side on a flat surface so that one row of pins is flat against the surface as shown on the following page.



HOLDING THE SIDE OF THE CHIP FIRMLY AGAINST THE FLAT SURFACE WITH BOTH HANDS, ROTATE IT A SHORT DISTANCE TOWARD ITS PINS UNTIL IT IS IN A FULL VERTICAL POSITION. THIS WILL PUT ITS BODY AT A RIGHT ANGLE TO THAT ROW OF PINS. PLACE THE OTHER ROW OF PINS ON THE FLAT SURFACE AND REPEAT THE PROCESS AS ABOVE.

PARTIALLY INSERT ALL ICS WITH THE PIN #1 ORIENTED AS SHOWN ON THE ASSEMBLY LAYOUT WHICH IS SILK SCREENED ON THE FRONT OF THE BOARD. THE LAYOUT SYMBOL FOR IC PIN #1 IS DESIGNATED BY A WHITE DOT ADJACENT TO THE UPPER LEFT HAND CORNER OF EACH RECTANGULAR IC CHIP LOCATION SYMBOL. RECHECK TO INSURE THAT EACH PIN IS IN ITS HOLE AND HAS NOT BEEN FOLDED UNDER THE CHIP OR BENT OUTSIDE THE SOCKET. COMPLETE INSERTION EVENLY AND FIRMLY.

UNPLUGGING INTEGRATED CIRCUIT CHIPS

UNPLUGGING AND INTEGRATED CIRCUIT CHIP MUST BE DONE EVENLY FROM BOTH ENDS SIMULTANEOUSLY SO THAT THE PINS WILL NOT BE BENT DURING REMOVAL. GENTLY PRYING WITH A SCREWDRIVER A LITTLE BIT AT A TIME FIRST AT ONE END, THEN AT THE OTHER IS RECOMMENDED. IF ACCESS IS POSSIBLE ONLY FROM ONE END, BE SURE THE SCREWDRIVER IS PUSHED AS FAR IN AS POSSIBLE SO AS TO GIVE A UNIFORM LIFTING ACTION OVER THE FULL LENGTH OF THE CHIP.

POWER ON

PLUG THE BOARD INTO YOUR COMPUTER AND CHECK IT OUT IN ACCORDANCE WITH THE USERS MANUAL PRECEDING THESE ASSEMBLY INSTRUCTION.