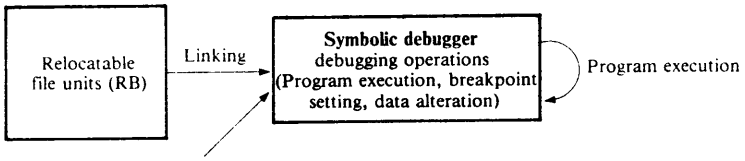
CHAPTER 3
SYMBOLIC DEBUGGER

3.1 OUTLINE OF THE SYMBOLIC DEBUGGER

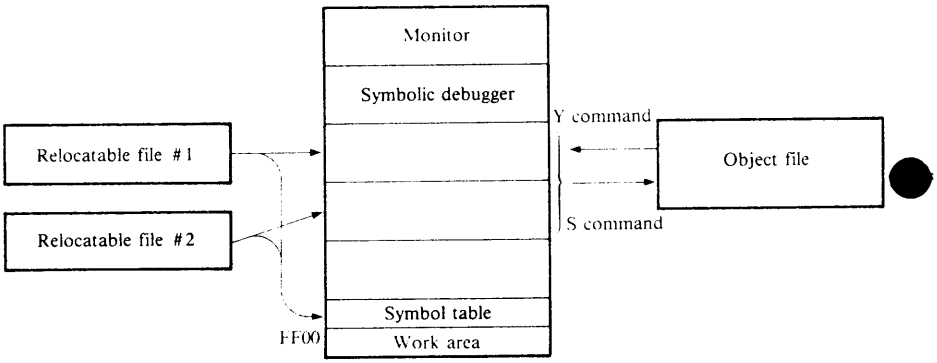
The symbolic debugger loads and links one or more program units from relocatable files to form an object program in memory in an immediately executable form and runs the object program for debugging. It provides the programmer with facilities for taking a memory dump of the object program in the link area, for setting **breakpoints** in the program, for displaying and altering the contents of the CPU internal registers and for starting the execution of the program at a given address with the CPU internal registers set to specified values (indicative start).



Debugging with the symbolic debugger

The debugger is said to be "**symbolic**" since it permits the programmer to reference addresses (e.g., breakpoints) during debugging not only in absolute hexadecimal representation but with global symbols declared as entry symbols in the source program with the ENT assembler directive. This releases the programmer from the burden of remembering the relative addresses in relocatable programs and offset values specified when they are loaded.

When all errors are detected by debugging, the source program is reedit. After debugging of all source programs is completed, the final object program can be obtained by linking them. The method of setting the symbol table is explained in 3.3 Symbol table.



Symbolic debugger file processing

— Symbolic debugger command table —

The symbolic debugger provides the commands listed in the tabl below. Among them, those marked with a dagger " † " permit symbolic operation.

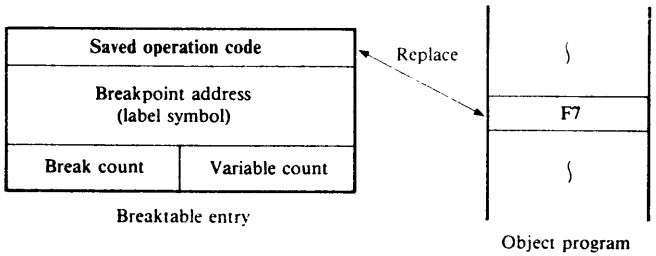
Command type	Command name	Function
Link/load symbol table command	L	Loads a relocatable file output by the assembler into the link area. The program in the relocatable file is loaded to form an object program through relocation at the location designated by the assembly bias and link address (relocate Load).
	N	Appends and links a relocatable file to the end of the preceding program in the link area. (Next file).
	H	Displays the current values of the assembly command bias and link address (Height).
	=	Displays the contents of the symbol table. Label symbol names, their absolute addresses, and their definition status are indicated.
	*	Clears the symbol table and current assembly bias and link address values to 0000H (CLEAR bias and table).
Debugging commands	B †	Displays, sets or alters breakpoints. (Breakpoint)
	&	Clears all breakpoints set. (clear breakpoint)
	T †	Traces the program execution starting at the specified address. (Trace)
	M †	Displays the contents of the specified block in the link area in hexadecimal representation or alters them. (Memory dump)
	D †	Disassembles the machine language program in the specified block. (Disassemble)
	W †	Writes hexadecimal data, starting at the specified address in the link area. (Write)
	G †	Executes the program at the specified address. (Goto)
	? †	Searches the specified block for the specified data. (search)
	F †	Fills the specified block with the specified data. (Fill)
	A	Displays the contents of registers A, F, B, C, D, E, H and L in hexadecimal representation or alters them. (Accumulator)
	C	Displays the contents of complementary registers A', F', B', C', D', E', H' and L' in hexadecimal representation or alters them. (Complementary)
	P	Displays the contents of registers PC, SP, IX, IY and I in hexadecimal representation or alters them. (Program counter)
	R	Displays the contents of all registers in hexadecimal representation. (Register)
	X	Transfers the specified memory block to the specified address. (TRANSfer)
File control commands	\ DEFAULT	Sets the specified external storage device as the default device.
	\ DELETE	Deletes the specified file in the RAM file.
	\ DIR	Displays the contents of the directory.
	\ DIR/P	Prints out the contents of the directory on the printer.
	\ INIT	Initializes the MZ disk.
	\ LOADALL	Loads all files on the MZ disk into the RAM file.
	\ MODE	Specifies the number of characters to be printed on a line by the colour plotter printer or that displayed on a line on the screen.
	\ RENAME	Changes the name of the specified file in the RAM file.
	\ RUN	Executes machine language programs.
	\ SAVEALL	Saves all files in the RAM files on the MZ disk.
File I/O command	S	Saves the object program in the link area in an output file with the specified name.(Save)
	Y	Reads the object program with the specified filename into memory. (Yank)
	V	Compares the file whose filename is specified with the contents of the link area. (Verify)
	I	Displays the value at the specified input port. (Inport)
	O	Outputs the specified data to the specified output port. (Outport)
Special commands	#	Switches the list mode for listing on the printer.
	!	Transfers control to the monitor.

3.2 BREAKPOINTS

A breakpoint is a checkpoint set up in the program at which program execution is stopped and the contents of the CPU registers are saved into the register buffer. At this point, the programmer can examine and alter the memory and register contents. He can also restart the program at this point. Thus, breakpoints facilitate program checking and debugging.

The symbolic debugger allows a maximum of nine breakpoints. When setting a breakpoint, the programmer must specify not only its address but also its count. The count specifies the number of allowable passes through the breakpoint in a looping program before a break actually occurs. The maximum allowable value of the break count is E in hexadecimal (14 in decimal).

When a breakpoint is set in a program, the debugger saves the operation code at that location (address) in the break table and replaces it with code FF. The debugger creates one break table entry for each breakpoint as shown below.



Hexadecimal code FF is the operation code for RST 7, which initiates a break operation. When the RST 7 instruction, which is 1-byte CAll instruction, is executed, the contents of program counter are pushed into the stack and the program counter is loaded with new data 0038H; that is, proram control jumps to address 0038H in the monitor, from which point control is immeidiately passed to the debugger. The debugger displayes the error message "RST 7?" Thus, the RST 7 instruction is used in the system and cannot be used by user programs.

When the debugger finds the required breakpoint in the table, it checks the correnponding count and decrements the variable count (this count is initially set to the break count) by one. If the variable count reaches zero, the symbolic debugger performs break processing; otherwise, it continues the program execution.

3.3 SYMBOL TABLE

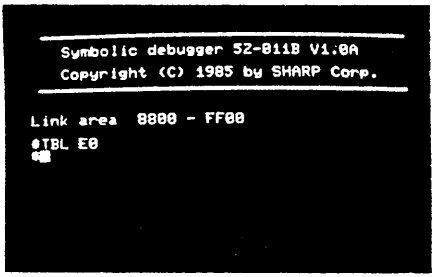
Symbols referred to by the symbolic debugger are label symbols which are declared as global symbols in a source program; that is, label symbols defined with assembler directive ENT or EQU. They are not replaced with the assembler location counter values by the assembler and left in ASCII codes in relocatable files output by the assembler for use during program linkage.

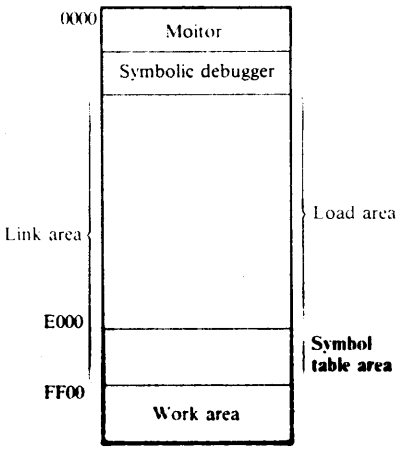
When the symbolic debugger inputs a relocatable file, it stores each label symbols which it encounters into a symbol table. The symbol table is located at the end of the link area. The higher order two digits of the starting address (hexadecimal) of the symbol table must be specified by the user. For example, when the user enter:

*** TBL E0**

the symbol table starting address is set to E000 in hexadecimal.

The photograph below shows the display when the symbolic debugger is started up and the figure at the right is the memory map for the symbolic debugger.





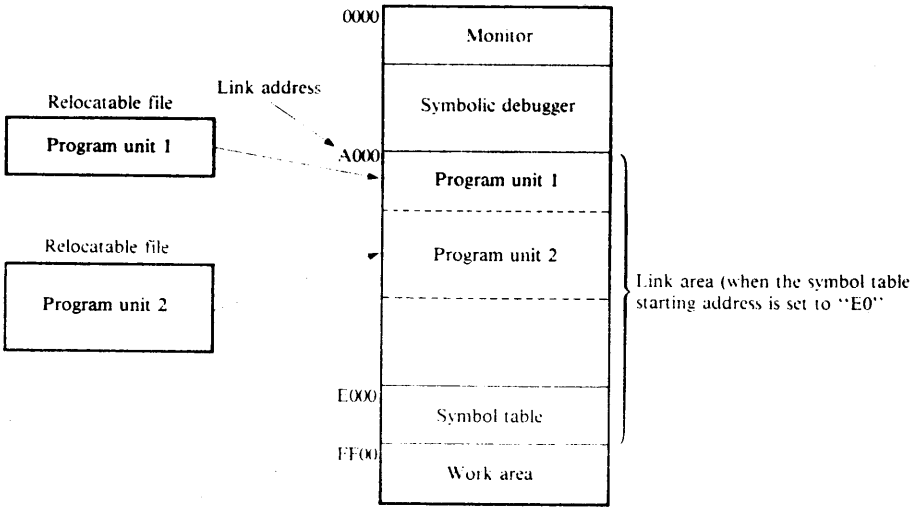
3.4 LINKER BIAS AND ADDRESSES

The user must specify four addresses in addition to the symbol table starting address when using the symbolic debugger. These addresses are assembly bias and the link address which are used when inputting relocatable files, and the execution address and load address which are used when an object files are output.

These addresses determine some of the characteristics of the object program. However, they cannot be determined arbitrarily; attention must be paid to their interrelationship. These addresses are described below.

— Link address —

The link address specifies the starting address of a relocatable file in the link area.
The figure below shows the memory map set up when a link address of A000H is specified.

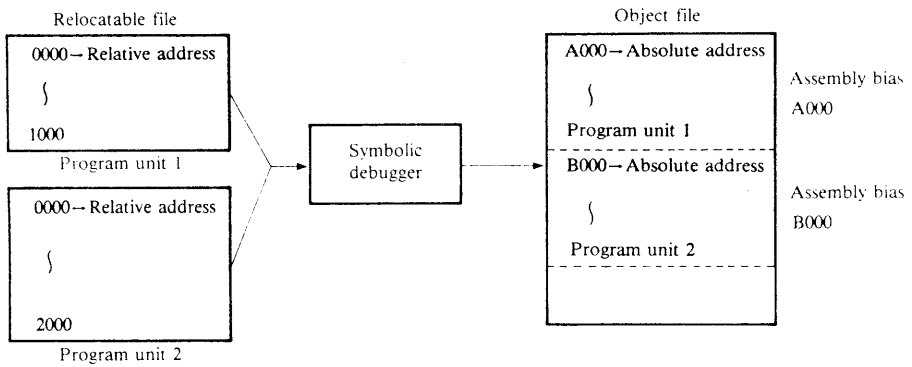


— Assembly bias —

The assembly bias specifies the address at which the object file generated is executed. It is added to all relative address of a relocatable file to form absolute address.

Each relocatable file generated by the assembler uses relative address starting at 0000H. To convert such relocatable file to an object file which starts at A000H, for example, the user must specify assembly bias A000H to the symbolic debugger.

When an object program generated is to be debugged, the assembly bias must be set to the same address as the link address. When debug is not required, the assembly bias can be set arbitrarily.
The function of the assembly bias is illustrated below.



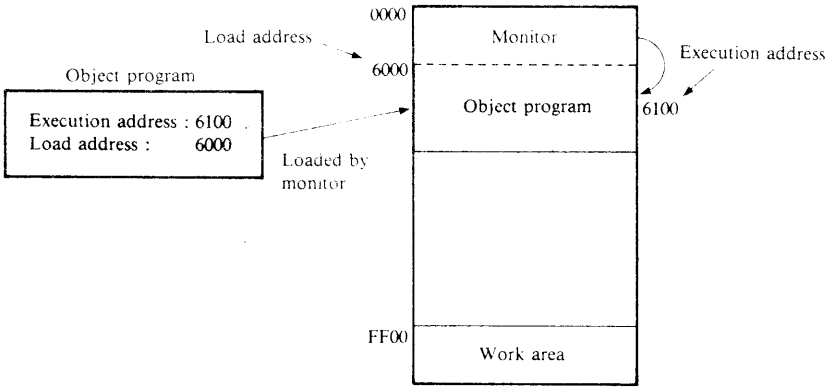
— Execution and load addresses —

The execution and load addresses must be specified when an object file is output by the symbolic debugger. Normally, the load address is set to the same address as the assembly bias.

These addresses are stored as information data in an object file.

The load address specifies the starting address from which an object file is loaded through the monitor. The execution address specifies the value to be set up in the program counter in the CPU after the object program has been loaded.

The figue below shows how an object program is loaded and given control when a load address of 6000H and an execution address of 6100H are specified.



When an object program is loaded by the symbolic debugger or with the Y command in the machine language, or when it is linked to a BASIC program, the execution address is ignored and control is retained by the system program. To execute the object program in this case, it is necessary to transfer control to the program by means of the system program execute command.

— Relationship between the ORG directive and the four addresses —

The load address can be specified with the ORG assembler directive as well as by the symbolic debugger. This subsection describes the relationship between the ORG directive and the four addresses (assembly bias, link address, load address and execution address).

Assume two program TEST1 and TEST2, whose starting addresses and program size are as follow.

TEST1: ORG 6000H specified, occupies 6000H to 6C00H

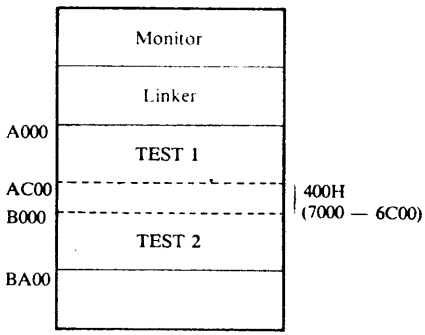
TEST2: ORG 7000H specified, occupies 7000H to 7A00H

When loading TEST1 with a L command, it is necessary to specify the assembly bias and link address. In this example, any assembly bias value specified is ignored and the assembly bias is automatically set to 6000H. The specified link address remanins valid.

When the command

***L A000 7400**

Filename? TEST 1

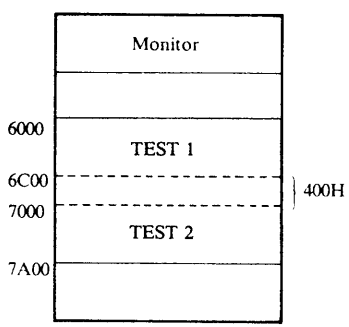


is executed, TEST1 is loaded in the link area from address A000H to AC00H as shown in the figure at right.

When a N command is entered to read in TEST2, TEST2 is loaded in address B000H to BA00H, resulting in an offset of 400H (7000H − 6C00H). Note that the assembly bias for TEST2 is set to 7000H as with TEST1.

The object file can be output with an S command after the two programs have been linked. Since in this case TEST1 and TEST2 have been assembled with assembly bias of 6000H and 7000H, respectively, the load address must be set to 6000H to run the object program properly.

The memory map when the object program is executed is shown at right.



3.5 SYMBOLIC DEBUGGER COMMANDS

— Link-load commands —

L (relocate Load) command

The L command loads relocatable file output by the assembler. Relative addresses in a relocatable file is converted into absolute addresses by specifying the assembly bias in this command.

* L A000 A000	Loads a relocatable file converting relative addresses to absolute addresses. The assembly bias and link address are set to A000.

— Key in L while the system is in the command wait state (*).

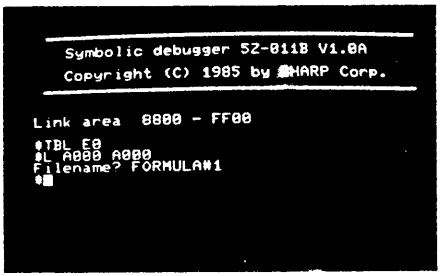
— Key in assembly bias and link address values as 4-digit hexadecimal numbers. An immediately executable program is generated in the link area using the assembly bias as explained in section 3.4. Normally, the assembly bias and link address are set to the **same address** in the link area. When the source program contains an ORG directive, the assembly bias specified is ignored.

— The system prompts for the name of the file to be read in with the message "**Filename**?."

— Enter the file name, then press [CR] . The system searches for the specified file and reads it into the link area.

— Press [SHIFT] + [BREAK] to terminate program loading.

— The photograph below shows the operation to load relocatable file FORMULA # 1 into the link area from address A000.



N (Next file) command

The N command link-loads the next relocatable file as specified by the current assembly bias and the link address values (which can be displayed with the H command).

*N	Link-loads the next relocatable file as specified by the current assembly bias and link address values.

— Key in N while in the command wait state (*).
— The system prompts for the name of the file to be read in with the message "**Filename?.**"
— Enter the file name, then press [CR] . The system searches for the specified file and reads it into the link area.
— The system uses the assembly bias and link address values immediately before the N command is executed when loading the relocatable file. If the source program contains an ORG directive, the assembly bias value is ignored.
— The programs are appended and linked during loading.
— Press [SHIFT] + [BREAK] to terminate program loading.

H (Height) command

*H	Displays the current values of the assembly bias and link address (the values cannot be changed).

— Key in H while the system is in the command wait state (*).
— The system displays two 4-digit hexadecimal numbers indicating the current assembly bias and link address. These values cannot be changed.

— Symbol table commands —

= (table dump) command

The table dump (=) command displays the contents of the symbol table. Each symbol table entry consits of a label symbol name, its absolute address, and its definition status.

* =	Displays the contents of the symbol table.

—Key in = while the system is in the command wait state (*) and press CR .
—The system displays the label symbol name, absolute address (in hexadecimal), and definition status for each symbol table entry. Errors in symbol definition can be found by examining the definition status of each symbol.
—Messages pertaining to the symbol definition status are listed in the table below.
Example of the symbol definition status messages output are given at next page.

Message	Definition status
U	Undefined (address or data)
M	Multi-defined (address or data)
X	Cross-defined (address and data)
H	Half-defined (data)
D	EQU-defined (data)

* (clear bias and table) command

* *	Resets the assembly bias and link address to 0000 and clears the symbol table.

—Key in * while the system is in the command wait state (*) and press CR .
—The clear bias and table command (*) is executed to reset the assembly bias and link address to 0000 and to clear the symbol table. However, the symbol table address set with the TBL command is not changed.

— Symbol definition status message examples —

First program unit loaded (UNIT-#1)

```
TMDLYH:  LD      HL,START
COUNT:   ENT
         DEC     HL
         LD      A,H
         CP      COUNT0
         JR      NZ,COUNT
         LD      A,L
         CP      COUNT1
         JR      NZ,COUNT
         CP      COUNT2
         JR      NZ,COUNT
         RET
PEND:    ENT
         DEFM    'TMDLYH'
         DEFB    0DH
COUNT1:  EQU     00H
COUNT0:  EQU     50H
         END
```

Second program unit loaded (UNIT-#2)

```
TMDLYL:  LD      HL,START
LOOP1:   DEC     H
         LD      A,H
         CP      COUNT
         JR      NZ,LOOP1
         RET
PEND:    ENT
         DEFM    'TMDLYL'
         DEFB    0DH
START:   EQU     1000H
COUNT:   EQU     00H
         END
```

Third program unit loaded (UNIT-#3)

```
INPUT:   CALL    0610H
         CALL    TMDLYL
         CALL    0610H
         LD      HL,START
         CP      0DH
         JR      Z,END
         LD      (HL),A
         INC      HL
         JR      INPUT
END:     JP      0000H
COUNT2:  EQU     12
         END
```

"START" X

START is not defined as an address in the first program, but is defined as data in the second or subsequent program with the START: EQU directive.

Note: The EQU directive should be placed at the beginning of the program unit.

"COUNT2" H

COUNT2 is not defined as data in the first program, but is defined as data in the third program with the COUNT2: EQU directive.

"COUNT1" D

COUNTl is defined as data (D indicates no error condition).

"COUNT" X

COUNT is defined as an address in the first program while it is simultaneously defined as data in the second program.

"PEND" M

PEND is defined as an address in the first program while it is simultaneously defined as an address in the second program (duplicated definition).

"TMDLYL" U

TMDLYL is neither defined as an address nor declared with the ENT directive in any other external program unit.

B (Breakpoint) command

The B command sets or changes a breakpoint. Break occurs after instructions immediately preceding the breakpoint have been executed the number of times set in the break counter. Then program execution is interrupted and control is passed to the debugger. The debugger saves the contents of the CPU registers into the register buffer and waits for a debugger command.

Breakpoints can be specified with either an absolute hexadecimal address or a global label symbol. The displacement applied to the label symbol ("+ 5L" in example 3 and "−9" in example 4 below) must be a decimal number from 1 to 65535 in line or from ±1 to ±65535 in byte.

***B** [CR]	Sets breakpoints.
addr count	
1 **A530**␣**2**	The breakpoint is address **A530** and break count is 2.
2 **SORT3**␣**1**	The breakpoint is the address represented by the label symbol **SORT3** and the break count is 1.
3 **SORT3+5L**␣**1**	The breakpoint is the address of the instruction 5 lines away from the address represented by label symbol **SORT3** and the break count is 1.
4 **MAIN0−9**␣**2**	The breakpoint is the address of the instruction 9 bytes before the address represented by label symbol **MAIN0** and the break count is 2.
5 [▩]	(The breakpoint and break count must be separated by at least one space (denoted by ␣).)

— Enter a **B** command while in the command wait state (*).

— The debugger carries out a new line operation and displays "addr count." It then performs a new line operation and displays the breakpoint number followed by a space and the cursor to prompt the programmer to enter a breakpoint address and break count. The programmer may specify a breakpoint address with a 4-digit hexadecimal number or a global symbol (see the example above). In either case, type in a break count following the breakpoint address with a space between the breakpoint and break count. The break count specifies the number of allowable passes through the breakpoint before a break acutally occurs. **The break count can be specified with a hexadecimal value from 1 to E.**

When a break count is specified, the symbolic debugger performs a new line operation and displays the next breakpoint number to prompt for the next breakpoint address.

— **Up to 9 breakpoints can be set.** When nine breakpoints are set, the symbolic debugger displays X on the next line instead of the next breakpoint number. This allows the programmer to clear the breakpoints or change the break counts, and not to set a new breakpoint.

If the programmer attempts to set a new breakpoint, the symbolic debugger will not accept it and prompts for a new command with message "Over."

— Press [SHIFT] + [BREAK] to terminate breakpoint setting and return to the command wait state.

Note: 1. Break count 0 means that the breakpoint is cleared.

2. Breakpoints can be set only at the addresses of op-codes. However, breakpoints cannot be set at the addresses of the CALL and RET instructions when their parameters are specified immediately after them.

```
Example     CALL    CHKACC    ←Breakpoint cannot be set at this address.
            DEFB    2
            DEFM    'AB'      } Parameters of CALL CHKACC
            JR      Z,OK.CHR
            RST     3         ←Breakpoint cannot be set at this address.
            DEFB    5         ←Parameters of RST 3
```

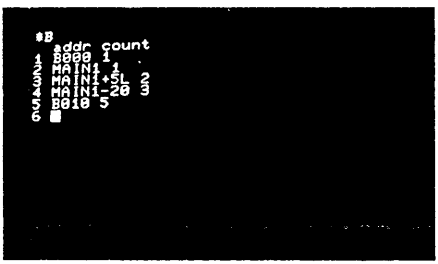
& (clear breakpoint) command

***&**	Clears all breakpoints which have been set.

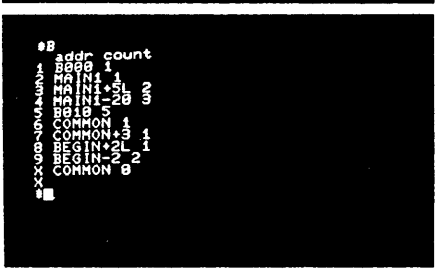
— Enter the **&** command while in the command wait state (*).

— The symbolic debugger clears all breakpoints set and returns to the command wait state.

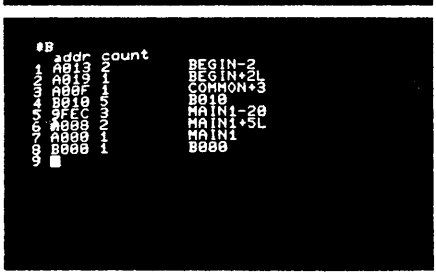
— The photograph at right shows an example of setting breakpoints. The breakpoints are set with a 4-digit hexadecimal number (absolute address), a global label symbol and a global label symbol plus a displacement in lines or bytes.



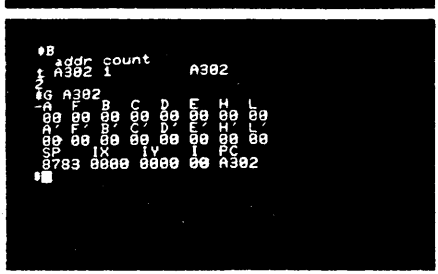
— The photograph at right shows that breakpoint COMMON was cleared on the line identified by X.



— The photograph at right shows an example of displaying previously set breakpoints with a B command. Breakpoints are displayed with hexadecimal absolute addresses shown first, followed by the break counts and the label symbols.



— The photograph at right shows that a break occurred immediately after the program execution was started from address A302 with a G command with a breakpoint at A302 and a break count of 1. As soon as a break occurred, an R command was automatically executed to display the status of the CPU registers.



T (Trace) command

The T command traces the program execution starting at the specified address. After this command is executed, **breakpoints set are automatically cleared.**

***T 1A A000** [CR]	Traces the program execution by **1A** steps starting at address **A000**.
***T F0 START** [CR]	Traces the program execution by **F0** steps starting at the address identified by global symbol **START**.
***T 05** [CR]	Traces the program execution by 5 steps starting at the current address designated by the **PC** (Program Counter).
***T** [CR]	Traces the program execution by 1 step starting at the current address designated by the **PC**.

—Key in **T** while in the command wait state (*).

—The symbolic debugger displays the cursor with a space between the cursor and the letter T, then wait for the programmer to enter the number of steps to be traced. The number of steps must be specified with a 2-digit hexadecimal number. When 00 is specified, the symbolic debugger interprets it as 100. When [CR] is pressed without specifying the number of steps, one is assumed.

—Then specify the starting address with a 4-digit hexadeciaml number or a global symbol and press [CR] ; the symbolic debugger starts trace.

—Each time one step is traced, the contents of all registers and the mnemonic code of the instruction are displayed (or printed). However, **at the time when the information is output, that instruction has not yet been executed.**

—To suspend trace, press [SPACE] . To resume trace, press [SPACE] again. To terminate trace, press [SHIFT] + [BREAK] .

—The photograph at right shows the result when program execution has been traced by stopping it step by step with breakpoints.

The difference between the value to be set in the accumulator by the instruction displayed and the contents of the accumulator indicates that the instruction has not yet been executed.

—The photograph at right shows the next display to that shown in the photograph above.

As displayed with a B command, the breakpoints set were cleared by execution of the T command. Then, a breakpoint was set at AE0E and the program was executed again with a G command. The value in the PC displayed by the last trace is specified as the starting address in the G command because the instruction at the address has not yet been executed.

Note: When parameters are specified immediately after the CALL and RST instructions as described in note 2 for the B command, execution of these instructions cannot be traced.

```
*G A000
-A  F  B  C  D  E  H  L
 12 FD EE EF 46 56 65 76
 A' F' B' C' D' E' H' L'
 12 FD 00 00 00 00 00 00
 SP    IX   IY   I  PC
 8773 0000 0000 00 A006
*T
-A  F  B  C  D  E  H  L
 12 FD EE EF 46 56 65 76
 A' F' B' C' D' E' H' L'
 12 FD 00 00 00 00 00 00
 SP    IX   IY   I  PC
 8773 0000 0000 00 A007  LD   A,B
*T
-A  F  B  C  D  E  H  L
 EE FD EE EF 46 56 65 76
 A' F' B' C' D' E' H' L'
 12 FD 00 00 00 00 00 00
 SP    IX   IY   I  PC
 8773 0000 0000 00 A008  CALL 12H
*
```

```
*T
-A  F  B  C  D  E  H  L
 EE FD EE EF 46 56 65 76
 A' F' B' C' D' E' H' L'
 12 FD 00 00 00 00 00 00
 SP    IX   IY   I  PC
 876D 0000 0000 00 A008  CALL 12H
*B
  addr count
1 A00E 1
2
*G A008

-A  F  B  C  D  E  H  L
 EE FD EE EF 46 56 65 76
 A' F' B' C' D' E' H' L'
 12 FD 00 00 00 00 00 00
 SP    IX   IY   I  PC
 876B 0000 0000 00 A00E
*
```

M (Memory dump) command

The M command displays the contents of the specified memory block in hexadecimal representation. The memory block may be specified with either absolute hexadecimal addresses or global symbols. The M command also permits the programmer to alter data with the cursor.

***M A800␣A850** [CR]	Displays the contents of the memory block from **A800** to **A850.**
***M MAIN7␣MAIN9** [CR]	Displays the contents of the memory block from the address identified by "**MAIN7**" to the address identified by "**MAIN9.**"
***M STEP0−9␣STEP3+15L** [CR]	Displays the contents of the memory block from the address **9** bytes before the address identified by global symbol **STEP0** to the address of the instruction **15** lines after the address indentified by global symbol **STEP3.**

—Key in **M** while in the command wait state (*).

—The symbolic debugger displays the cursor with a space between the cursor and the letter M and waits for the programmer to enter the starting and ending addresses of the memory block to be dumped. The programmer may specify the starting and ending addresses of the memory block with either 4-digit hexadecimal numbers or global symbols.

—**The starting address must be smaller than or equal to the ending address.** Otherwise, the debugger will display the message "?".

—When a memory block in the link area is specified, the symbolic debugger displays the contents of the memory block on the screen with 8-byte of data in hexadecimal and their corresponding ASCII characters on each line.

—If the printer is set in the enable mode, the symbolic debugger prints the memory dump on the printer with 16 bytes on a line. (See the explanation of the # command.)

—The cursor appears on the screen when the memory block dump is completed. The programmer can then alter byte data in the memory block by moving the cursor to the desired byte position on the screen, entering the new byte data in hexadecimal and pressing [CR]. The byte data under the cursor is overwritten with the new data. The debugger displays the message "Error" if the data entered does not match the byte format.

—When [CR] is pressed with the cursor on a memory dump line, the data on that line is reentered into memory. The symbolic debugger is returned to the command mode, however, when [CR] is pressed with the cursor at the beginning of a line containing no data.

—Press [SPACE] to suspend display of the memory dump. To resume display, press [SPACE] again.

—Press [SHIFT] + [BREAK] to force the symbolic debugger to return to the command wait state.

D (Disassemble) command

The D command disassembles the machine language program in the specified memory block. The disassembly list can be output on the screen or printer. The list is also output to the storage device in the format in which it can be edited with the text editor.

***D A000 A300** [CR] **TBL? C000 C800**	Disassembles the machine language program in the memory block from address **A000** to **A300**.
***D START STOP+4L** [CR] **TBL? C000 C800**	Displays the machine language program in the memory block from the address identified by global symbol **START** to the address **4** lines after the address identified by global symbol **STOP**.

—Key in **D** while in the command wait state (*).

—The symbolic debugger displays the cursor with a space between the cursor and the letter D, and waits for the programmer to enter the starting and ending addresses of the memory block. The starting and ending addresses of the memory block can be specified with either 4-digit hexadecimal numbers or global symbols.

—**The starting address must be smaller than or equal to the ending address**. Otherwise, the symbolic debugger will display the message "?".

—After a memory block in the link area is specified, the symbolic debugger then displays "**TBL?**" and wait for the starting address of the symbol table to be entered. When a D command is executed at the first time after the symbolic debugger is started up, the starting address of the symbol table must be specified with a 4-digit hexadecimal number. At the second time on, the address specified at the first time is used when [CR] is pressed without specifying the starting address.

—The symbolic debugger waits for the ending address of the symbol table to be entered. If [CR] is pressed without specifying the ending address when a D command is executed for the first time, a memory area 4K bytes from the starting address is secured for the symbol table. To secure a memory area larger or smaller than 4K bytes for the symbol table, the ending address must be specified with a 4-digit hexadecimal number. At the second time on, the ending address specified at the first time is used if [CR] is pressed without specifying it.

—After the memory area for the symbol table is specified, the symbolic debugger disassembles the machine language program in the specified memory block and outputs the disassembly list on the screen or printer.

—Display can be suspended by pressing [SPACE] and it can be resumed by pressing [SPACE] again. The assembly list can be displayed line by line by pressing [↓] after display has been suspended.

—When output of the assembly list is completed, the symbolic debugger displays "Filename?" and wait for the file name to be entered to save the text obtained in a storage device.
When the text is not to be saved, press [SHIFT] + [BREAK] to return to the command wait state.
When the file name is specified and [CR] is pressed, the text is saved under the specified name. The lable symbols, mnemonic codes, operands and comments are included in the text saved. The machine codes and addresses are not output. Unneeded spaces are also deleted in the text saved. Thus, the text is saved in the format in which it can be read into the edit buffer and edited with the text editor.

—If the text is saved without errors, the symbolic debugger displays the file size in bytes and returns to the command wait state.
When a large machine language program is disassembled, attention must be payed to the size of the text saved. If the size of the text exceeds that of the edit buffer, the machine laguage program must be disassembled by dividing it into smaller blocks.

Notes:

- When a disassembly list is output on the printer, ASCII characters corresponding to the machine codes are printed in the comment column of the listing.

```
3E06            LD      A,6             ; >.
CDC608          CALL    L08C6           ; ▫▫.
0614            LD      B,14H           ; . .
3E03    L5E07:  LD      A,3             ; >.
```

- When a disassembly list is displayed on the screen, ASCII characters corresponding to the operands of instructions such as PC, LD and so on, are displayed in the comment column of the listing only when they are alphanumerics.

```
FE31            CP      31H             ; 1
3E42            LD      A,42H           ; B
D651            SUB     51H             ; Q
```

- When data, work areas and so on are placed between the machine language programs to be disassembled, the programs cannot be disassembled correctly.

```
314631          LD      SP,L3146        ; 1F1
47              LD      B,A             ; G
314131          LD      SP,L3141        ; 1A1
42              LD      B,D             ; B
312B43  L5E8A:  LD      SP,L432B        ; 1+C
```

- Undefined instructions are assumed as 1-byte constant (the DEFB directive).

```
CB              DEFB    CBH             ; ▫
34              INC     (HL)            ; 4
```

- When the value of nn in the instructions such as "LD BC, nn" is less than 100H, it is disassembled as data because there is little possibility that it is an address.

```
011000          LD      BC,10H          ; ...
```

- A comment line is inserted to leave space between the preceding and following disassembly listings after the following 7 instructions, because program execution discontinues at those instructions.
  RET, HALT, JR e, JP (HL), JP nn, JP (IX), JP (IY)

```
20F5            JR      NZ,-9           ; ◪
C9              RET                     ; ◪
                ;
11000A  L5E60:  LD      DE,L0A00        ; ...
```

W (data Write) command

The W command writes hexadecimal data, starting at the specified memory address. The memory address may be either an absolute hexadecimal address or a global label symbol.

***W A000** [CR]	Writes hexadecimal data, starting at address **A000**.
***W DATA1** [CR]	Writes hexadecimal data, starting at the address indentified by global symbol **DATA1**.

—Key in W while in the command wait state (*).

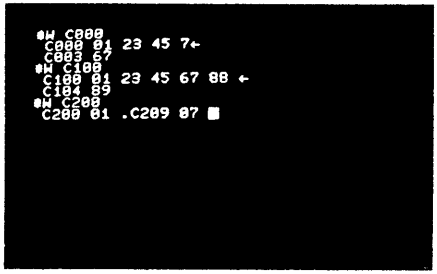
—The symbolic debugger displays the cursor with a space between the cursor and the letter W and waits for the programmer to enter the starting address of the memory area into which data is to be written. The starting address of the memory area can be specified with either a 4-digit hexadecimal numbers or global symbol.

—The memory area must be inside the link area.

—When the programmer press [CR] after specifying an address, the symbolic debugger carries out a new line operation, displays the starting address, then waits for the data to be entered. The data must be entered in two-digit hexadecimal numbers. The symbolic debugger inserts a space each time 2-digit data is entered, and performs a new line operation and displays a new address each time eight bytes of data are entered.

—To correct the data just entered, press [←] to return the cursor back to the byte of data just entered and correct it.

The photograph at right shows an example. As the photograph shows, when [←] is pressed, the cursor is placed on the next line and the address of the byte of data to which the cursor has been moved back is displayed.



—To specify a displacement for a JR, DJNZ or other Z80 relative jump instruction, enter a period "."; the symbolic debugger waits for the programmer to enter an absolute address (no label is allowed) with a 4-digit hexadecimal number as the destination of the jump. When the programmer enters a 4-digit hexadecimal address, the symbolic debugger computes the displacement and stores the 1-byte result in the current byte position. The seventh and eighth lines in the photograph above show an example of specifying a displacement.

—After the necessary data has been written, press [CR]; the symbolic debugger returns to the command wait state.

G (Goto) command

The G command transfers program control to the specified address. This command is also used to restart the program following a break.

***G B7000** [CR]	Executes the program at address **B7000**.
***G START** [CR]	Executes the program at the address identified by global symbol **START**.
*G [CR]	Restarts the program at the breakpoint with the restart address and the **CPU** register data stored in the register buffer.

—Key in **G** while in the command wait state (*).

—The symbolic debugger then waits for entry of an execution address. The programmer can specify the execution address with either a 4-digit hexadecimal number or a global label symbol. When using a label symbol, a displacement in lines or bytes can also be used.

***G MAIN0**	Executes the program at address MAIN0.
***G MAIN0+3L**	Executes the program at the address 3 lines after the address identified by MAIN0.
***G MAIN0−12**	Executes the program at the address 12 bytes before the address identified by MAIN0.

—To restart the program at a breakpoint, enter a G command and press [CR] . If this operation is initiated when no breakpoint is taken, the symbolic debugger returns to the command wait state without executing the program. The contents of the CPU registers restored when the program is restarted are displayed with the R command. The value in the program counter (PC) is used as the restart address. Since the PC value can be changed with the P command, it is possible to restart the program at an address other than the breakpoint.

—To execute the program and returns to the symbolic debugger at a certain address, insert the instruction below.

JP 5603 (4003)

Address 5603 (4003) is the warm start address for the symbolic debugger; at this address, "*" is displayed to prompt for command entry without the contents of the link area being lost. (If a start is made from address 5600 (4000), it is a cold start and the link area, symbol table and bias are cleard.)

—The only methods of stopping program execution are to use a jump instruction to return to the symbolic debugger or to set a breakpoint.

—Press [SHIFT] + [BREAK] to terminate entry of a G command.

Note: Addresses enclosed with the parentheses must be used in the MZ-700 mode.

? (search) command

The ? command searches the specified memory block for the specified data and displayes it. This command also allows the programmer to alter the data displayed with the cursor.

***? A000␣AFFF** [CR]	Searches the memory block from address **A000** to address **AFFF** for ASCII character data ABC and displays them.
Data? ;A␣;B␣;C [CR]	
***? START␣STOP**	Searches the memory block from the address identified by global symbol **START** to that identified by global symbol **STOP** for hexadecimal data **C3,00,00** and displays them.
Data? C3␣00␣00 [CR]	

—Key in ? while in the command wait state (*).

—The symbolic debugger displays the cursor with a space between the cursor and the ?, then wait for the programmer to enter the starting and ending addresses of the memory block to be searched. The starting and ending addresses of the memory block can be specified with either 4-digit hexadecimal numbers or global symbols.

The starting address must be smaller than or equal to **the ending address**.

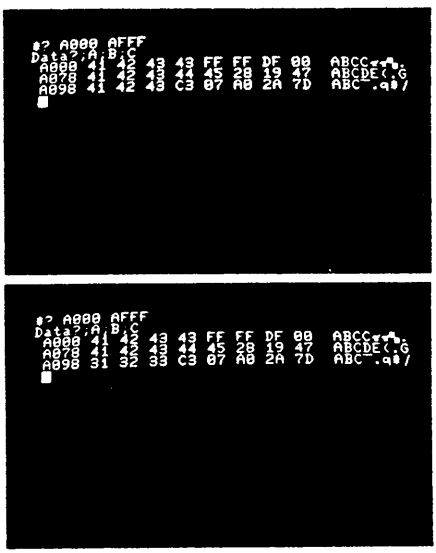
—When a memory block is specified, the symbolic debugger then displays "Data?" and wait for the programmer to enter the data to be searched for. The data must be specified with 2-digit hexadecimal numbers or combinations of ; and a ASCII character.

When the data is specified in combinations of ; and a ASCII character, the ASCII character is converted into the corresponding ASCII code and then searched for. Up to eight 2-digit hexadecimal numbers or combinations of ; and a ASCII character can be entered. They must be separated with a space.

—After specifying the data, press [CR] ; the specified data is searched for and displayed each time it is detected in the same formt as by the M command. The data displayed can be modified with the cursor in the same manner as with the M command.

—The photograph at right shows the state in which three character strings which contain character string ABC are detected and now the modification is possible. The photograph at the bottom right shows the screen after the ABC in the third character string has been altered into 123.

—Press [SHIFT] + [BREAK] to force the symbolic debugger to return to the command wait state.



F (Fill memory) command

The F command fills the specified memory block with the specified data. Data can be specified with up to eight hexacecimal numbers or ASCII characters.

***F A000␣A3FF** [CR] **Data? FE␣;M** [CR]	Fills the memory block from address **A000** to address **A3FF** with the data consisting of a hexadecimal number **FE** and a ASCII character **M**.
***F JUMP␣JUMP + 9L** [CR] **Data? C3␣00␣00** [CR]	Fill the memory block from the address identifed by global symbol **JUMP** to the address **9** instructions after that with hexadecimal numbers **C3**, **00**, and **00**.

— Key in **F** while in the command wait state (*).

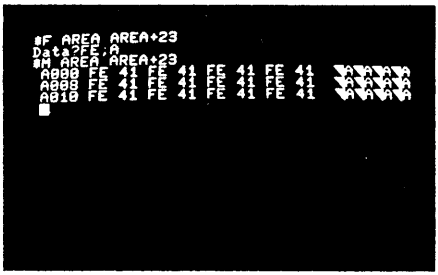
— The symbolic debugger displays the cursor with a space between the cursor and the letter F, then wait for the programmer to enter the starting and ending addresses of the memroy block to be filled. The starting and ending addresses can be specified with either hexadecimal numbers or global symbols.

— The starting address must be smaller than or equal to the ending address and the memory block specified must be located with the link area.

— After the memory block is specified, the symbolic debugger displays "**Data?.**" to prompt for the programmer to enter the data with which the memroy block is to be filled. The data can be specified with up to eight bytes of 2-digit hexadecimal numbers or ASCII characters (each ASCII character must be preceded by a semicolon ";"). ASCII charactes are converted into ASCII codes when they are stored. Hexadecimal numbers or ACII characters must be entered with a space between them.

— Press [CR] after specifying the data; the symbolic debugger returns to the command wait state.

— In the example shown in the photograph at right, a 24-byte memory block from the address identified by global symbol AREA to the address 23 instructions after that is filled with the data consisting of a hexadecimal number FE and ASCII character, then the contents of the memory block is displayed with a M command.



A (Accumulator) command

The contents of the Z80 CPU registers are saved in the register buffer when a breakpoint is taken; the contents of the primary general registers saved can be displayed with the A command. The buffer contents can also be altered using the cursor manipulation.

***A** **— A F B C D E H L** **01 23 45 67 89 AB CD EF**	Displays the contents of primary register pairs **AF**, **BC**, **DE** and **HL**.

—Enter an **A** command in response to the prompt (*).

— The debugger displays the contents of accumulator A, flag register F, and general register pairs BC, DE and HL with 2-digit hexadecimal numbers. These values represent **the contents of the primary CPU registers set up when a break occurs at a breakpoint**. They are saved in the register buffer for use in subsequent restart operations at the breakpoint (see the G command description).

—If necessary, the programmer can alter the register contents. To change a register value, place the cursor on the desired register value, overwrite it with a new value, and press CR .

The register values displayed with the A command are restored to the CPU internal registers on a restart.

—Press CR ; the symbolic debugger returns to the command wait state.

C (Complementary) command

The C command displays the contents of the complementary general-purpose registers set up on the last break. The programmer can alter their contents through cursor mainpulation.

***C** **A' F' B' C' D' E' H' L'** **01 23 45 67 89 AB CD EF**	Displays the contents of complementary register pairs **AF', BC', DE'** and **HL'**.

—Enter a **C** command while in the command wait state (*).

—The debugger displays the contents of accumulator A', flag register F' and general-purpose register pairs BC', DE' and HL' with 2-digit hexadecimal numbers. They are used for restart at a breakpoint.

—Press CR ; the symbolic dybugger then returns to the command wait state.

P (Program counter) command

The P command displays the contents of the special-purpose registers set up on the last break. The programmer can alter their contents through cursor mainpulation.

```
*P
SP   IX   IY   I    PC
EFEA 5F70 5F50 00   78AB
```

Displays the contents of special-purpose registers **SP, IX, IY, I** and **PC**.

—Enter a **P** command while in the command wait state (*).

—The symbolic debugger displays the contents of special-purpose registers SP, IX, IY, I and PC and I with 2-and/or 4-digit hexadecimal numbers. Register values displayed or altered through cursor mainpulation are restored to the pertinent registers upon restart at a breakpoint. The program does not have to restart at the breakpoint; **the programmer can specify another restart address by altering the PC value.**

R (Register) command

The R command displays the contents of all CPU internal registers set up on the last break or altered with the A,C or P commands. The programmer cannot alter their contents.

```
*R
A   F   B   C   D   E   H   L
01  23  45  67  89  AB  CD  EF
A'  F'  B'  C'  D'  E'  H'  L'
01  23  45  67  89  AB  CD  EF
SP      IX      IY      I   PC
EFEA    5F70    5F50    00  78AB
```

Displays the contents of all CPU registers.

—Enter a **R** command while in the command wait state (*).

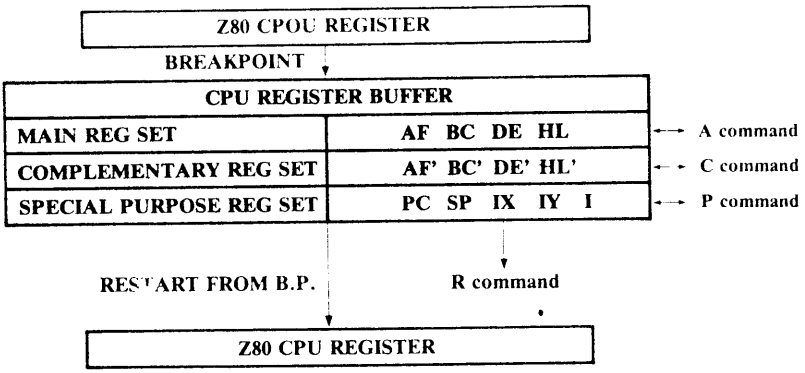
—The symbolic debugger displays the contents of all CPU registers with 2-and/or 4-digit hexadecimal numbers. The cursor does not appear and the values displayed cannot be altered.
The same data as displayed with a R command is automatically displayed when a break occurs.

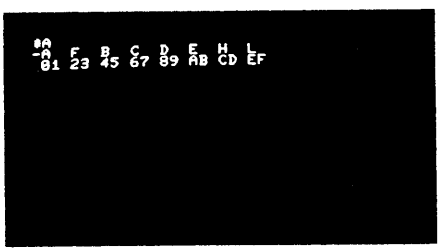
—The symbolic debugger enters the command wait state after displaying all the register contents.

Using register commands (A, C, P and R)

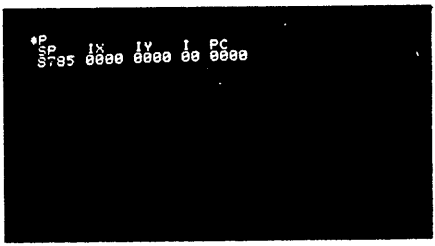
Values displayed with register commands (A, C, P and R) are the actual contents of the register buffer in the symbolic debugger. The values in the register buffer are the contents of the CPU registers saved at the last break or those changed through cursor mainpulation with the A, C or P command. These values are restored to the CPU registers when a restart is made.

The figure below shows the relationship between the CPU registers and the register commands; the photographs show examples of use of the register commands.

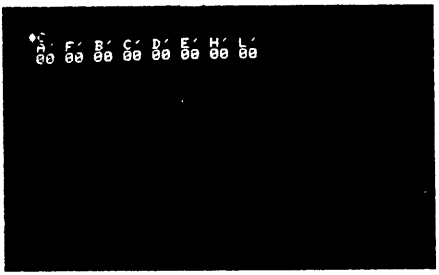




A command



P command



C command

R command

X (data TRANSfer) command

The X command transfers the contents of the specified memory block to the specified memory area.

***X**
From?A000 To? A11F Top?C000 to C11F

Transfers the Contents of the memory block from address **A000** to address **A11F** to the memory area starting at address **C000.**

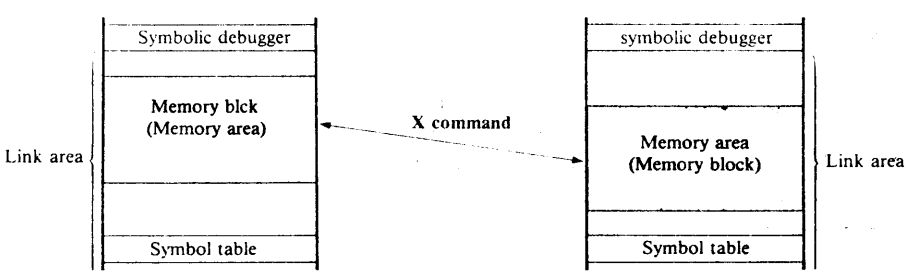
— Enter an **X** command while in the command wait state (*).

— The symbolic debugger displays the message "From?" and waits for the programmer to enter the starting address of the source memory block with a 4-digit hexadecimal number. When the starting address is entered, the debugger displays the message "To?" to prompt for the programmer to enter the ending address with a 4-digit hexadecimal number. When the ending address is entered, the symbolic debugger displays the message "Top?" to prompt for the programmer to enter the starting address of the destination memory area with a 4-digit hexadecimal number (No global symbol can be used to specify these addresses.)

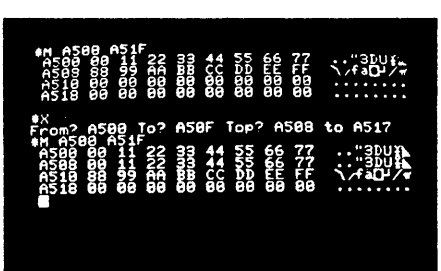
— After all addresses are specified, the system computes and displays the ending address of the destination area and starts transferring data. The system returns to the command wait state after the data transfer is completed.

— The source and destination memory blocks must be located within the link area.

— Data transfer is accomplished successfully even if the source and destination memory blocks overlap as shown below. The memory block shown in the figure at left may be transferred to the memory block shown in the figure at right and vice versa.



— The photograph at right shows data transfer from the memory block from address A500 to address A5UF to the memory block starting at A508. The data in the source block before transfer and that in the destination block transferred are displayed with a M command. The data has been successfully transferred.



— File control commands —

\ DEFAULT command

This command sets the specified storage device as the default device. The I/O commands write or read the specified file to/from the default device when a device name is omitted. The default device is also effective for the DIR command.

* \ **DEFAULT QD** [CR]	Sets the MZ disk as the default device.
* \ **DEFAULT CMT** [CR]	Sets the cassette tape as the default device.
* \ **DEFAULT RAM** [CR]	Sets the RAM file as the default device.

—Type in \ **DEFAULT** while in the command wait state (*).
—Type in a device name with a space between the command and device name.
—Press the [CR] key; the symbolic debugger sets the specified device as the default device.

Note: When the editor-assembler is started up, the storage device from which it was read is set as the default device.

\ DIR command

This command displays the contents of the directory, that is, a list of the names of files stored in the media in the specified storage device.

* \ **DIR** [CR]	Displays the contents of the directory of the current storage device.
* \ **DIR QD** [CR]	Displays the contents of the directory of the MZ disk.
* \ **DIR RAM** [CR]	Displays the contents of directory of the RAM file.

—Type in \ **DIR** while in the command wait state (*).
—Specify a device name with a space between the command and device name. (The device name may be omitted when the default device is to be specified.)
—Press the [CR] key; the system displays the contents of the directory of the specified device.

Note: This command cannot be used with the cassette tape.

\ DIR/P command

This command prints out the contents of the directory on the printer.

* \ **DIR/P** [CR]	Prints out the contents of the directory of the current device on the printer.
* \ **DIR/P QD** [CR]	Prints out the contentes of the directory of MZ disk on the printer.
* \ **DIR/P RAM** [CR]	Prints out the contentes of the directory of the RAM file on the printer.

—Type in \ **DIR/P** while in the command wait state (*).
—Specify a device name with a space between the command and device name. (The device name may be omitted when the default device is to be specified.)
—Press the [CR] key; the contents of the directory of the specified device are printed out on the printer.

Note: This command cannot be used with the cassette tape.

\ INIT command

This command initializes the MZ disk or RAM file. Refer to the MZ-800 Owner's manual for the other functions of this command.

* \ **INIT** CR	**Initializes the MZ disk.**
* \ **INIT QD** CR	Initializes the MZ disk.
* \ **INIT "RAM:$FFFF"** CR	Initializes the RAM file.
* \ **INIT "LPT:S2"** CR	Sets the listing device to a centronics standard printer.

—Type in \ **INIT** while the symbolic debugger is in the command wait state (*).

—Type in QD or RAM:$FFFF with a space between the command and device name. (When the device name is omitted, the MZ disk is assumed.)

—Press the CR key; confirmation message "OK? [Y/N]" is displayed on the creen.

—Press Y to execute initialization and N to cancel it in response to the message. When you press N, the symbolic debugger returns to the command wait state again.

\ MODE command

This command sets the number of characters printed on a line by the color plotter printer in its text mode or that displayed on the CRT screen. Both the printer and CRT are set in 40-character mode when the power is turned on.

* \ **MODE TN** CR	Sets the line length for printout to 40 characters per line.
* \ **MODE TL** CR	Sets the line length for printout to 26 characters per line.
* \ **MODE TS** CR	Sets the line length for printout to 80 characters per line.
* \ **MODE DL** CR	Sets the line length for display to 40 characters per line.
* \ **MODE DS** CR	Sets the line length for display to 80 characters per line.

—Key in \ **MODE** while the text editor is from the command wait state (*).

—Specify TN, TL, TS, DL or DS.

—Press the CR key.

Note: \ MODE DS and \ MODE DL cannot be used in the MZ-700 mode.

\ RUN command

This command executes the specified machine language program.

* \ **RUN"TRANS"** CR	Loads machine language program TRANS from the current device and executes it.
* \ **RUN"TEST",R** CR	Sets the memory in the same state as when IPL (Initial Program Loading), loads machine language program TEST, and executes it.

—Type in \ **RUN** while the text editor is in the command wait state (*).

—Specify the file name.

When you execute machine language programs created on the MZ-80K series computers, R must be specified following the file name. Type in a comma "," after the file name when specifying R.

Note: This command cannot be executed with the cassette tape.

—Press the CR key.

When R is not specified, the specified program is loaded without changing the current memory state and executed. When R is specified, the memory is set in the same state as when IPL and the specified program is loaded and executed.

Note: When the RUN command is executed, control is transferred to the specified program. In some cases, control is not returned to the symbolic debugger. If the specified machine language program is to be executed in a memory area overlapping the area in which the symbolic debugger is stored, the program is loaded over the symbolic deubgger and the symbolic debugger will be destroyed.

\ DELETE command

* \ **DELETE"RAM:SAMPLE"** CR	Deletes file "SAMPLE" in the RAM file.

—Key in \ **DELETE** while in the command wait state.
—Type in RAM and file name.
—Press CR ; the specified file is deleted.

Note: This command cannot be used in the MZ-700 mode.

\ RENAME command

* \ **RENAME"RAM:OLDPROG","NEWPROG"** CR	Changes filename OLDPROG of the file in the RAM file to NEWPROG.

—Type in \ **RENAME** while in the command wait state (*).
—Specify the current file name to be changed and a new filename.
—Press CR ; the current file name is changed to the new one.

Note: 1. When a file with the same name as the new name specified exists on the RAM file, execution of this command results in an error.
2. This command cannot be used in the MZ-700 mode.

\ LOADALL command

* \ **LOADALL** CR	Reads the entire contents of the MZ disk into the RAM file.

—Type in \ **LOADALL** while in the command wait state (*).
—Press CR ; the entire contents of the MZ disk is read into the RAM file.

Note: This command cannot be used in the MZ-700 mode.

\ SAVEALL command

* \ **SAVEALL** CR	Saves the entire contents of the RAM file on the MZ disk.

—Key in \ **SAVEALL** while in the command wait state (*).
—Press CR ; the entire contents of the RAM file is saved on the MZ disk.

Note: 1. This command cannot be used in the MZ-700 mode.
2. When two or more files are stored in the RAM file, all files in the RAM file cannot always be saved on the MZ disk even if the total file size does not exceed the capacity of the MZ disk, because files are recorded on the MZ disk with blank spaces for separation.

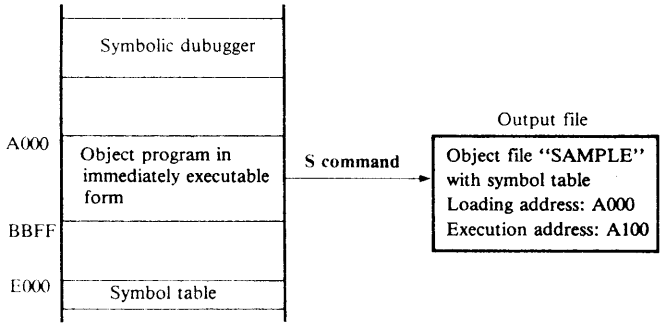
— Object file I/O commands —

S (Save) command

The S command saves a specified block of the object program in the symbolic debugger link area into a named output file. The contents of this file can be restored to the link area with the Y command.

***S** **Filename? SAMPLE** **From?A000 To?BBFF** **Load?A000 Execute?A100**	Saves the object program in the link area starting at address **A000** and ending at address **BBFF** in the default device under file name **SAMPLE.**

—Enter an S command while in the command wate state (*).
—The system displays the message "**Filename?**" on the next line and waits for the file name to be specified.
—Specify the filename and press CR .
—The system displays the message "**From?**" on the next line and waits for the starting address of the block to be entered.
—Enter the starting address in a 4-digit hexadecimal number.
—The system then displays message "**To?**" and waits for the ending address of the block to be enterd.
—Enter the ending address in a 4-digit hexadecimal number.
—The system then displays the message "**Load?**" on the next line and waits for the load address to be specified.
—Enter the load address in a 4-digit hexadecimal number.
—Finally, the system displays message "**Execute?**" and waits for the execution address to be specified.
—Enter the execution address in the the same manner.
—After the four addresses have been specified, the specified memory block is output.
—In the figure below, the block starting at A000 and ending at BBFF is output with file name SAMPLE, load address A000 and execution address A100.



Y (Yank) command

The Y command reads the object file specified with the file name into the link area.

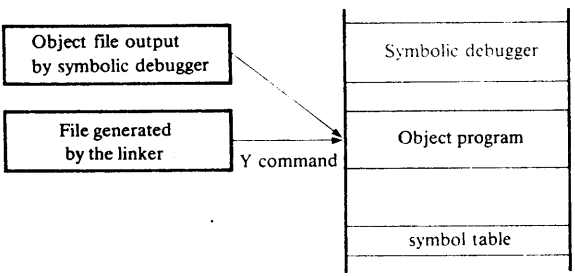
***Y** [CR] **Filename?SAMPLE**	Clears the link area and reads the object file SAMPLE into the link area.

—Enter a **Y** command while in the command wait state (*).

—The system displays the message "**Filename?**" and waits for the file name to be specified.

—Specify the file name and press [CR]; the system searches for the specified file and displays the starting address, ending address and execution address specified when the file was saved with a S command. When the file to be read is the first file on the cassette tape, the file name may be omitted.

—The systm then prompts for the loading address with the message "**From?**". When a 4-digit hexadecimal number is entered in response to the message, the object file is read into the link area starting at the address specified. The load address specified when the object file was saved is ignored. When [CR] is pressed without specifying the loading address, the object file is read into the link area starting at the load address specified when it was saved with a S command.



V (Verify) command

This command verifies the contents of the specified block in the link area with the specified object file. This command is effective only with the cassette tape.

*V [CR] **Filename?SAMPLE**	Verifies the contents of the link area with object file **SAMPLE**.

—Enter a **V** command while in the command wait state (*).

—The system displays message "**Filename?**" and prompts for the object file name to be verified.

—Specify the file name and press [CR] ; the system searches for the object file and displays its starting address, ending address and execution address specified when the object file was saved with an S command.

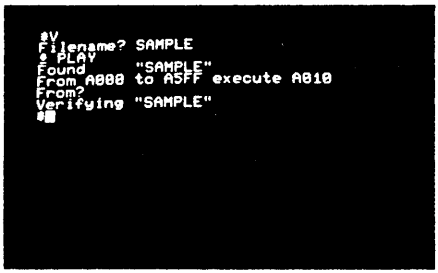
After displaying those addresses, the system then displays the message "**From?**" and waits for the programmer to specify the address in the link area at which verifcation is to be started.

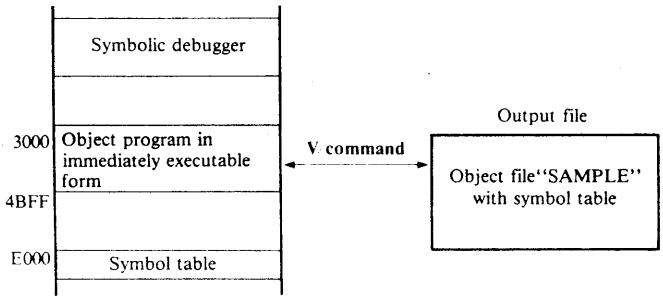
—Enter the address in a 4-digit hexadecimal number. When [CR] is pressed without entering the address, the same address as the starting address specified when the file was saved with the S command is assumed.

—When the starting address for verification is specified, the symtem starts comparing the contents of the link area and object file. If the contentes of the link area and object file match, the symbolic debugger returns to the command wait state.

—To terminate verification, press [SHIFT] + [BREAK] .

—The photograph at right shows the operation which saved the contents of the link area under file name SAMPLE with an S command and then verified the file with the contents of the link area with a V command. After the V command was executed, the symbolic debugger has returned to the command wait state. This indicates that the contents of the link area were saved without errors.





I (Inport) command

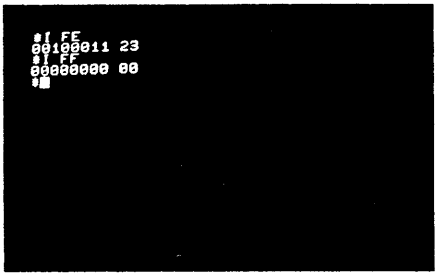
The I command displays the value at the specified input port.

***I 80** CR	**Displays the value at input port 80.**

— Key in **I** while in the command wait state (*).

— The system displays the cursor with a space between the letter I and the cursor and waits for the input port number to be entered.

— Specify the input port number with a 2-digit hexadecimal number; then the value at the specified input port is displayed on the next line both in a 8-digit binary number and in a 2-digit hexadecimal number.



O (Outport) command

Outputs the specified data to the specified output port.

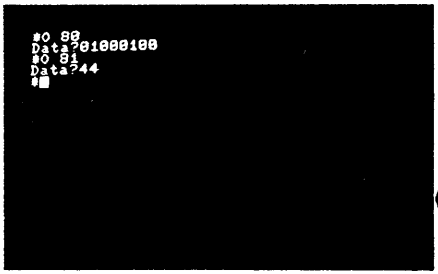
***O 81** CR **Data?01001001** CR	**Outputs 01001001 to output port 81.**

— Key in **O** while in the command wait state (*).

— The system displays the cursor with a space between the cursur and the letter O to prompt for the output port number.

— Specify the output port number with a 2-digit hexadecimal number.

— When the port number is specified, the system displays message "Data?" on the next line and waits for the data to be entered. The data can be specified with either a 2-digit hexadecimal number or an 8-digit binary number.



(sharp mark) command

*# CR	Switches the list mode for printout on the printer.

—Enter a # command while in the command wait state (*).

—The symbolic debugger then switches the list mode. When the symbolic bugger is invoked, the printer list mode is set to the disable mode.

The mode alternates between enable and disable each time a # command is entered. In the enable mode, all output is directed to both the screen and the printer (except with the M command).

! (exclamation mark) command

*! CR	Transfers control to the monitor.

—Enter an ! command while in the command wait state (*).

—The system displays the following message:

"M)onitor B)oot C)ancel?"

Enter M to transfer control to the monitor.

Enter B to transfer control to the IPL program.

Enter C to cancel the # command and return the symbolic debugger to the command wait state.

—There are three methods of returning from the nomitor to the symbolic debugger:

Jump to address 5600 (4000): **The link area is cleared. (cold start)**

Jump to address 5603 (4003): **The link area is not cleared. (warm start)**

Monitor command R: **Same as the warm start above.**

Note: Addresses in the parentheses must be used in the MZ-700 mode.

3.6 ERROR MESSAGES

The monitor and symbolic debugger detect errors and display error messages. The error messages displayed by the monitor and symbbolic debugger are listed in the table below.

3.6.1 Monitor error messages

Error message	Meaning
System id	The type of the system disk is wrong.
File not found	The specified file was not found.
Hardware	An error occurred in the device's hardware.
Already exist	A file with the same name already exists.
Already open	The file is already opened.
Not open	An attempt was made to referece a file not yet opened.
Write protect	The file or device is write-protected.
Not ready	The disk drive is not ready.
Too many files	The number of files exceeds 32.
No file space	The disk space is insufficient to store the file.
Unformat	The disk is not formatted (initialized).
Dev. name	The device name is wrong.
Can't execute	An attempt was made to make the device perform impossible operation.
Illegal filename	The format of the entered file name is incorrect.
Illegal filemode	The file mode is wrong.
Illegal data	The data read is erroneous or comparison with the V command resulted in mismatch.
LPT: not ready	The printer is not connected.
Check sum	A check sum error occurred. (Casstte tape read error)

3.6.2 Symbolic debugger error messages

Error message	Meaning	Relevant commands
???	○ An attempt was made to access a location outside the link area. ○ An attempt was made to set the symbol table outside the link area.	B, W, X, S, V, D
Error	○ An incorrect number of digits was specified or a digit other than a hexadecimal digit was entered during execution of a register (or memory) change command.	V, A, C, P
RST7?	○ An attempt was made to set a breakpoint at a RST7 instruction. ○ A RST7 instruction cannot be traced.	B T
Over	○ An attemp was made to set more than 9 breakpoints. ○ Too many label symbols are defined and all symbols cannot be stored in the symbol table.	B D
Invalid	○ The format of the entered command is incorrect.	F, ?, X, S, V, I, O
?	○ An invalid symbol (undefined label symbol or nonlabel symbol) was specified. ○ An attempt was made to clear a breakpoint which was not set. ○ An attempt was made to set a break count to greater than 14 (E in hexadecimal) times. ○ The format of the specified address is incorrect (not a 4-digit hexadecimal number). ○ The starting address is not smaller than or equal to the ending address. ○ The source and destination block are same.	B, W, D, F, T, ? B B M, D, G, F, T,? M, D, W, F, ? X
Bad command	○ The format of the file control command entered is incorrect.	Backslash (\ ) preceded command