

PREFACE

Thank you for buying the Sharp Personal Computer MZ-80 series FDOS.

To make the best use of the FDOS, read the instruction manual thoroughly and perform the operations described correctly; this will enable you to make most effective use of the system.

- The master diskette cannot be replaced after it is purchased; therefore be sure to use the COPY command to create a submaster diskette for normal use.
- It is particularly important to read and understand the explanations of the following commands before using FDOS.
 - **FORMAT command** (page 41 of the **System Command** manual)
Before using FDOS with a new diskette, it must be formatted and initialized for FDOS. The file contents of diskettes initialized for use with other systems (e.g., SB-6510 or SB-6610) will be destroyed if used with this system. Likewise, diskettes initialized with FDOS cannot be used with other systems.
 - **COPY command** (page 35 of the **System Command** manual)
This command allows creation of submaster diskettes from the master diskette and of backup diskettes for slave diskettes.
- Since the FDOS operating instructions are divided into several parts, a guide is included to enable easy reference as needed. Full understanding of FDOS is not a prerequisite to making active use of it, refer to the guide as needed and your knowledge of the system will grow as you use it.

PRODUCT GUIDE

The following materials are included in this group of products.

System Command Instruction Manual

Text Editor Instruction Manual

Z-80 Assembler Instruction Manual

Symbolic Debugger Instruction Manual

Linker Instruction Manual

Programming Utility Instruction Manual

PROM FORMATTER

EXAMPLE OF PLOTTER CONTROL APPLICATION

Library/Package Instruction Manual

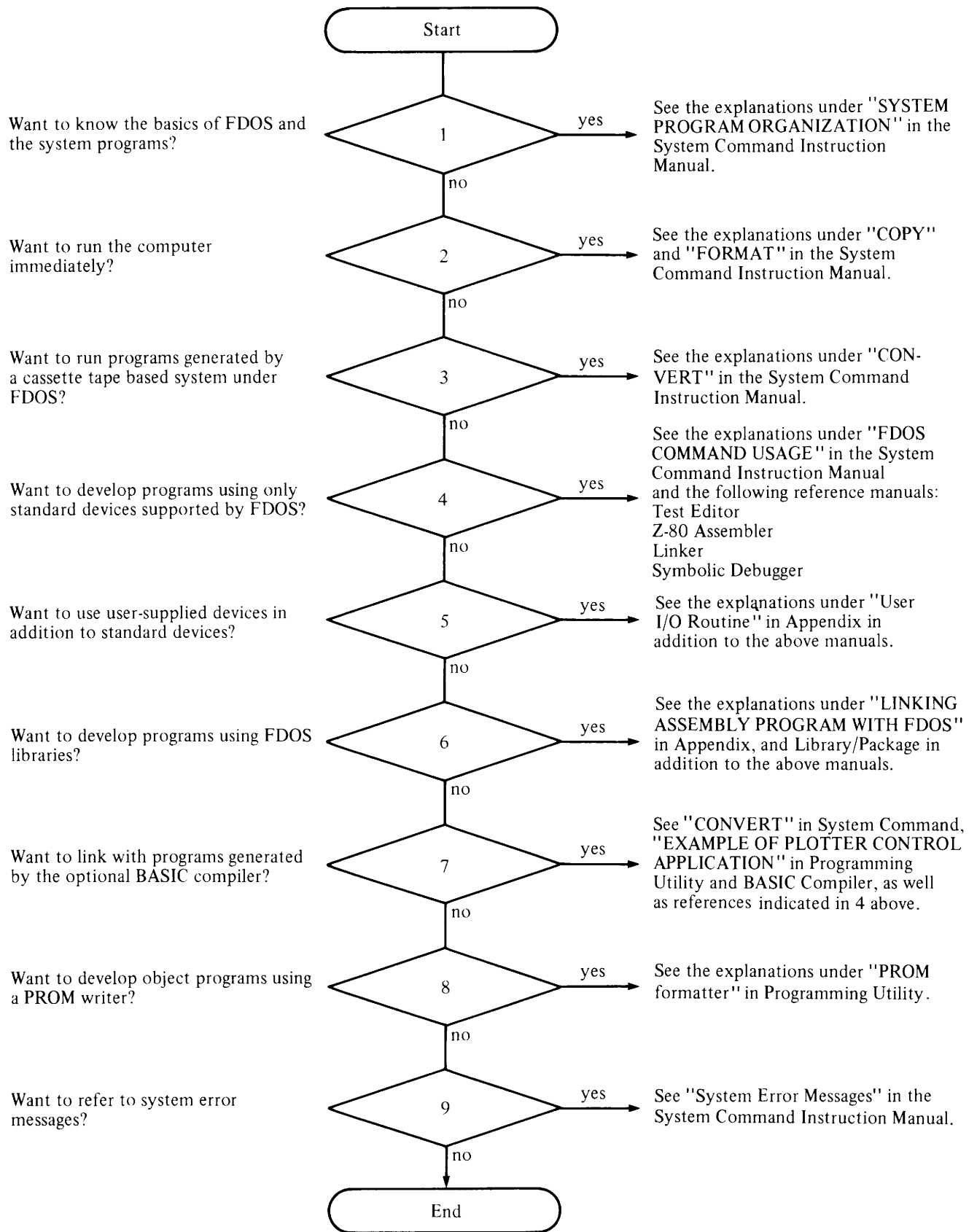
Appendix

FDOS Master Diskette

Also, the following files are included in FDOS Master Diskette. Refer to the various instruction manuals for details.

File name	Applicable command or manual	Function
ASM . SYS	ASM	Z80 Assembler
EDIT . SYS	EDIT	Text editing
LINK . SYS	LINK	Linker
MLINK . SYS	MLINK	Linker
& DEB & . SYS	DEBUG	Symbolic debugger
PROM . SYS	PROM	PROM formatter
BASIC . SYS	BASIC	BASIC compiler (sold separately)
FORMAT . SYS	FORMAT	Formatting diskettes
COPY . SYS	COPY	Copying diskettes
HCOPY . SYS	HCOPY	Copying one frame on CRT
LIMIT . SYS	LIMIT	FDOS management area declaration
LOAD . SYS	LOAD	Loading object files
ASSIGN . SYS	ASSIGN	Device definition
STATUS . SYS	STATUS	Device status control
CONVERT . SYS	CONVERT	File mode conversion
PTRP . ASC	"Appendix"	Paper tape reader/punch control
PTRP . OBJ	"Appendix"	Paper tape reader/punch control
SIO . ASC	"Appendix"	RS 232C control
SIO . OBJ	"Appendix"	RS 232C control
CMT1 . ASC	"Appendix"	MZ-80K cassette tape control
CMT1 . OBJ	"Appendix"	MZ-80K cassette tape control
START-UP . ASC	"System Command"	Key definition
LOADAUX . ASC	"System Command"	Loading auxiliary device controller
MONEQU . AS	"Library/Package"	Monitor library source file
MONEQU . LIB	"Library/Package"	Library file for the above
FDOSQU . ASC	"Library/Package"	FDOS library source file
FDOSQU . LIB	"Library/Package"	Library file for the above
1510QU . ASC		SB-1510 monitor library source file
1510QU . LIB		Library file for the above
RELO . LIB	"Library/Package"	BASIC compiler library file
SB-1511 . RB		SB-1511 monitor relocatable file

—GUIDE TO USE OF THESE PUBLICATIONS—



—OPTIONAL FDOS PROGRAM PRODUCTS—

1. BASIC Compiler SB-7701 (Previously released)

- Requirements: ○ FDOS and 64K bytes of RAM
- Major features: ○ Fast execution.
- FDOS commands can be invoked from BASIC programs.
- Can be linked to assembly language programs.
- Compilation mode: ○ Compiles a source file (source program) and generates a relocatable file (RB file) which can be linked and loaded with the FDOS LINK command.
- Compatibility: ○ Programs developed by the SB-5000 and SB-6000 series must be converted to the FDOS format by the FDOS CONVERT command before compilation. Some BASIC commands (file handling commands) may differ in syntax. Excessively large programs may not be compilable (source programs are limited to about 10K bytes).
- Packaging: The BASIC compiler is available on cassette tape with a reference manual. The compiler should be copied onto the submaster diskette so that it can be run under FDOS control.

2. Serial I/O Ports (to be released in the near future)

- Requirements: ○ FDOS and 64K bytes of RAM
- Major features: ○ One Z-80SIO serial interface LSI device.
- Baud rate is switch-selectable.
- Two RS232C I/O channels. One of which may be used for a current loop circuit.
- Packaging: ○ One interface board, its control programs (on cassette tape) and a reference manual.
- The control programs should be copied onto the submaster diskette so that they can be run under FDOS control.

Personal Computer
mz-80B

SHARP

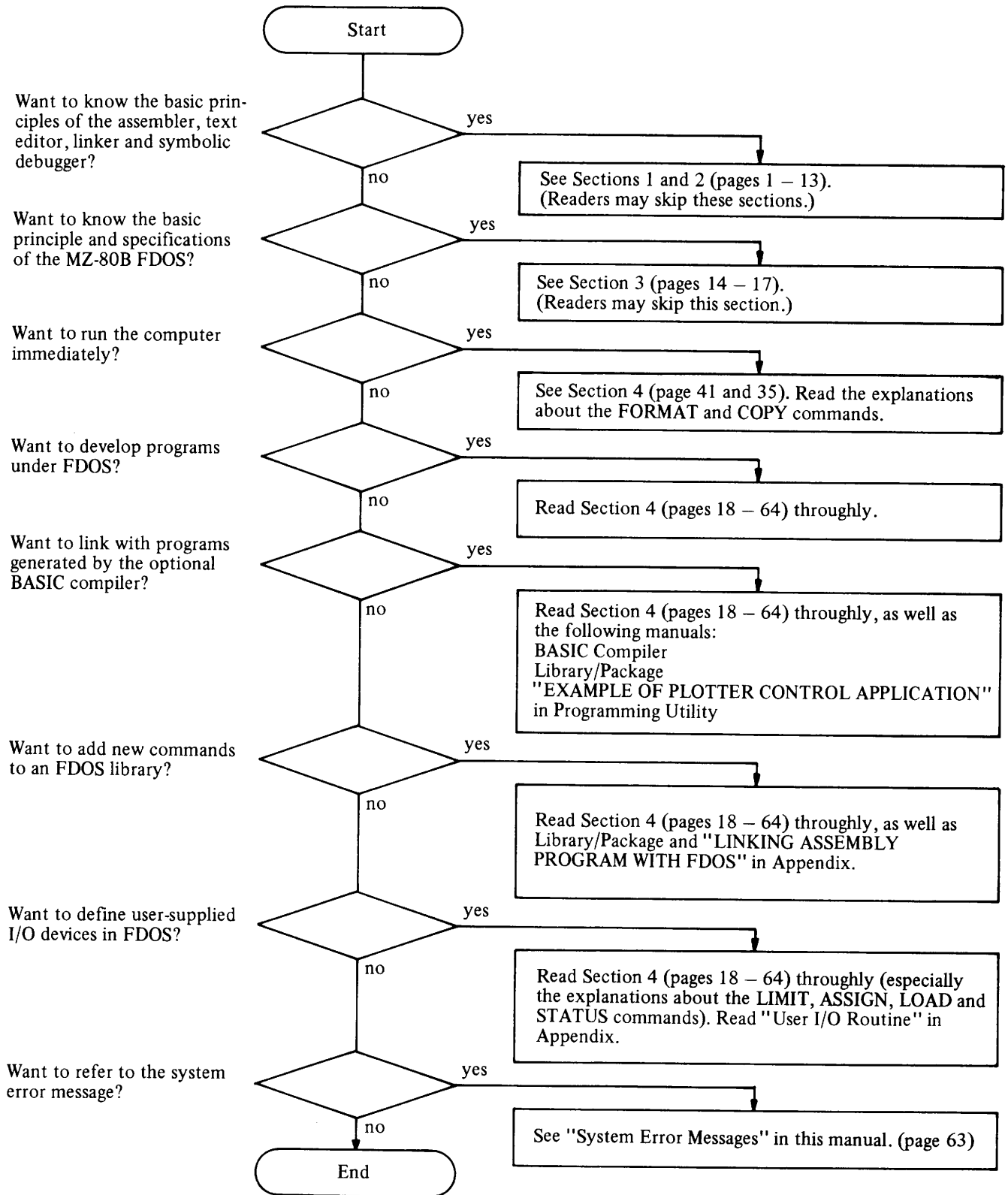
NOTICE

The MZ-80 series of sophisticated personal computers is manufactured by the SHARP CORPORATION. Hardware and software specifications are subject to change without prior notice; therefore, you are requested to pay special attention to version numbers of the monitor and the system software (supplied in the form of cassette tape or mini-floppy disk files).

This manual is for reference only and the SHARP CORPORATION will not be responsible for difficulties arising out of inconsistencies caused by version changes, typographical errors or omissions in the descriptions.

This manual is based on the SB-1500 series monitor and the SB-7000 series FDOS.

GUIDE TO USE OF THIS MANUAL



— CONTENTS —

1. THE MEANING OF "CLEAN COMPUTER"	1
2. SYSTEM PROGRAM ORGANIZATION	3
2.1 Text Editor Functions	4
2.2 Assembly Procedures	5
2.3 Linker	9
2.4 Symbolic Debugger	11
2.5 PROM Formatter	13
3. FDOS ORGANIZATION	14
3.1 Boot Linker	15
3.2 IOCS	15
3.3 Dynamic Segmentation	17
4. FDOS COMMAND USAGE	18
4.1 Program Development Under FDOS	18
4.2 FDOS Control Keys	20
4.2.1 Main keyboard	20
4.2.2 Automatic repeat function	21
4.2.3 Cursor control keys	21
4.2.4 Initial settings	21
4.2.5 Differences between the SB-1511 and SB-1510	22
4.3 FDOS Command Coding Rules	23
4.3.1 Command line format	23
4.3.2 File name	23
4.3.3 File modes	24
4.3.4 File attributes	24
4.3.5 File types	24
4.3.6 Wildcard characters	25
4.3.7 Drive number and volume number	25
4.3.8 Basic device name	25
4.3.9 Auxiliary device name	26
4.3.10 Switches	27
4.3.11 Default assumptions	28
4.3.12 Arguments	29

4.4	Using FDOS Commands	30
4.4.1	ASM	30
4.4.2	ASSIGN	31
4.4.3	BASIC	31
4.4.4	BOOT	32
4.4.5	CHATR	32
4.4.6	CONSOLE	33
4.4.7	CONVERT	34
4.4.8	COPY	35
4.4.9	DATE	36
4.4.10	DEBUG	37
4.4.11	DELETE	38
4.4.12	DIR	38
4.4.13	EDIT	39
4.4.14	EXEC	40
4.4.15	FAST	41
4.4.16	FORMAT	41
4.4.17	FREE	42
4.4.18	HCOPY	43
4.4.19	KEY	43
4.4.20	KLIST	44
4.4.21	LIBRARY	45
4.4.22	LIMIT	45
4.4.23	LINK	46
4.4.24	LOAD	47
4.4.25	MLINK	47
4.4.26	MON	48
4.4.27	PAGE	49
4.4.28	POKE	49
4.4.29	PROM	50
4.4.30	RENAME	50
4.4.31	REW	51

4.4.32 RUN	51
4.4.33 SIGN	52
4.4.34 STATUS	53
4.4.35 TIME	53
4.4.36 TYPE	54
4.4.37 VERIFY	54
4.4.38 XFER	55
4.5 FDOS Command Summary	57
4.6 System Error Messages	63
5. MUTUAL CONVERSION	65

1. THE MEANING OF "CLEAN COMPUTER"

Three important developments accompanied the shift from the boom in microcomputer kits to the entrance of personal computers.

(1) Mass production reduced the cost of RAM and ROM devices so that they became readily available.

This development eliminated the need to devote great amounts of time and effort to compressing system functions to the maximum extent possible to conserve valuable memory for user programs. Now it is more important that system programs be written and managed in a structured manner and that their overall usefulness be raised. It is more and more apparent that what the user comes in contact with is not so much a unit of hardware as a software reinforced computer.

(2) Compact, reliable external memory units with large storage capacities became available.

Floppy disks and fixed disks are currently the basis for system configurations, but sooner or later charge coupled devices and magnetic bubble memories will be used in this capacity. This suggests that there will be increasing stratification of programs culminating in operating systems, and that the efficiency of systems will also increase. From the user's point of view, this means that a wide variety of programs will be readily available for use.

(3) The development of various peripheral circuit LSIs has made possible realization of efficient interfaces with high performance terminals.

This means the main concern of the user in the future will be with how many functions are provided in a system and how useful they are. In terms of the contents of the system, the main concern will be in developing operating systems capable of organically combining terminals and program processing with a minimum of effort on the part of the user. It is even possible that real time processing of multiple tasks and jobs on a level approaching that of minicomputers will become possible with the operating systems of microcomputers.

As is apparent, it is extremely difficult to predict the extent to which computers will evolve as integrated circuit technology and program language theory become widely dispersed. This tends to undermine the belief which some people have that rapid changes in hardware result in good computers.

Although the name "clean computer" has been given to the MZ-80 series, computers are basically clean in principle. As the field of personal computers opens, the concept of embedding a single language, BASIC, in ROM has become a hindrance to use of full computer capacity. Out of consideration for the many different types of service which will be required by users as yet-to-be developed technology comes into use in the future, it will be necessary to preserve the cleanliness of the computer to the maximum degree possible to minimize constraints placed on its use. The ultimate ends to which computers are applied will be determined by the junction of technological possibilities and user requirements; the only other limits imposed are those which are inherent in the fact that the computer is nothing more than a machine. In order for computers and users to get along well together, it is necessary that computers be designed with a minimum of constraints so that they can be suited to user requirements, rather than the other way around. In other words, the usefulness of the computer and the efficiency of the service it provides depends on how clean it is.

The explanations in these publications are intended to show how flexible the MZ-80 series of computers is in terms of system development. A tape-based program development system is provided to enable inexpensive development of small programs; the floppy disk operating system (FDOS) was developed to assist with the creation of large programs which require large quantities of memory. The functions and configuration of FDOS are suited to a range of applications approaching those provided by a low level minicomputer. We think that the software technology and utilization procedures applied in this system will open a new world of possibilities for personal computers.

2. SYSTEM PROGRAM ORGANIZATION

SHARP MZ-80B system programs include an assembler, a text editor, a linker and a symbolic debugger. They are organized to execute a sequence of assembly phases.

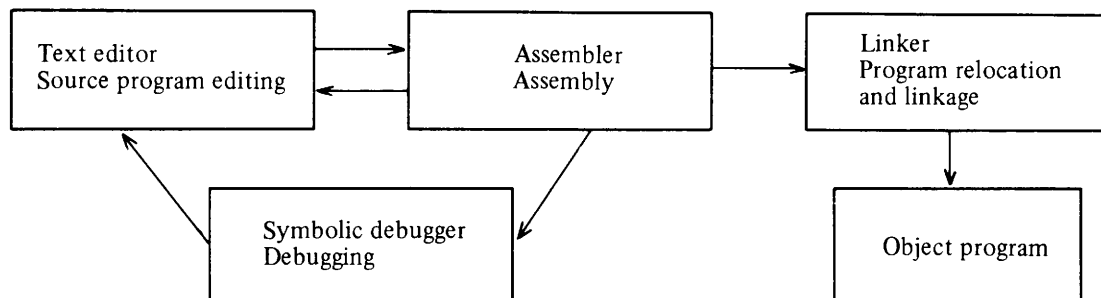


Fig. 2-1 Assembly phases

Figure 2-1 shows the assembly process, which consists of creating source programs, assembling them, relocating and linking the assembly output and debugging them.

One cycle of the phases in the left half of the figure makes up a program creation stage. The programmer prepares a source program with the text editor and creates a source file, then inputs it to the assembler. The assembler analyzes and interprets the syntax of the source program and assembly language instructions into relocatable binary code. When the assembler detects errors, it issues error messages. The programmer then corrects the errors in the source program with the text editor.

After all assembly errors are corrected, the programmer inputs the relocatable object program (the relocatable binary file), output by the assembler to the symbolic debugger. The symbolic debugger reads the object program into the link area in an executable form and runs the program. During the debugging phase, the programmer can set breakpoints in the program to start, interrupt and continue program execution, and to display and alter register and memory contents for debugging purposes. If program logic errors and execution inefficiency are detected during the debugging phases, the programmer reedits the source program using the text editor.

After all bugs are removed from the source program, the programmer loads and links the program unit(s) in the relocatable file(s) and creates an object program in executable form with the linker.

Each system program always generates an output file for use in other system programs. Figure 4-1 shows the interrelationship of the system programs.

As shown above, the program development phases are executed by four independent system programs. By assigning the system functions to separate programs, the MZ-80B can accommodate large-scale, serious application programs, thus enhancing its program development capabilities. "PROM formatter" is provided which punches object programs into paper tape in several formats for use with various PROM writers now on the market.

The system program commands are listed in the last part of Appendix.

2.1 Text Editor Functions

The major functions of a text editor are to insert, delete and modify characters, words and/or lines. If the editor does not allow the programmer to use these functions interactively and easily, he will have to devote more effort to editing and modifying programs than to executing them. To alleviate this problem, SHARP uses a command format which is almost perfectly compatible with that of the NOVA minicomputer series from the Data General Corp.; this has been refined through the support of many uses.

The most important concern of the programmer in conjunction with the text editor is the concept of the character pointer (CP) and its usage. During line-base editing, the CP is situated not on a line but between two consecutive lines, as shown in Figure 2-2. Therefore, the location to/from which a line is to be inserted/deleted can uniquely identified. If the CP was located somewhere on a line, two locations would be possible; that is, before and after the CP. The J and L in CP move commands are representative commands which use this interline pointer concept.

During character-base editing, the CP is situated not on a character but between two consecutive characters. This permits close editing. The programmer will become accustomed to the text editor quickly if he is aware of what commands use the interline CP and what command use the intercharacter CP concept.

During normal editing sessions, several commands are combined to carry out an intended task. Such commands can be placed on a line separated by separators so that the programmer lists them as they come into his head.

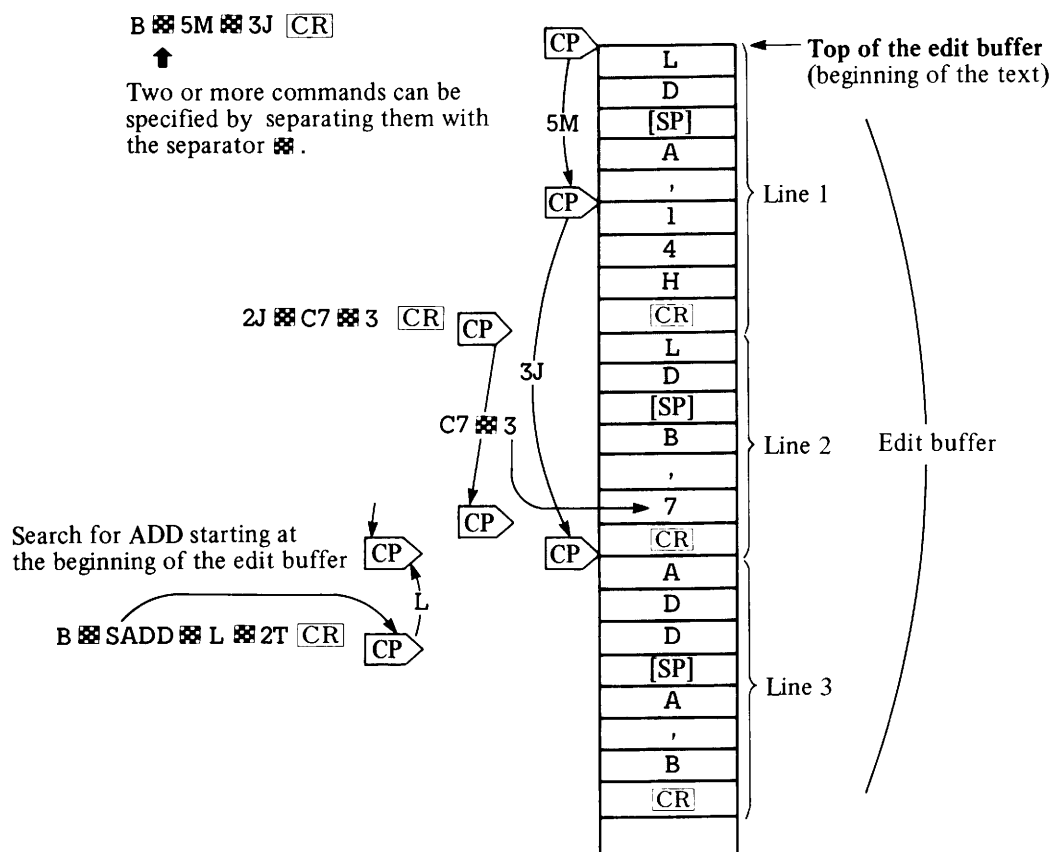


Fig. 2-2 Character pointer movement

2.2 Assembly Procedures

As the programmer becomes familiar with the Z-80 instructions, he is able to construct programs more easily, even though he may feel difficulty in grasping the structure of large programs. At this stage, it is not hard for the programmer to handle other microprocessors such as the M6800 and the F-8 with the help of good reference manuals. One of the major reasons for this is the operating principles and architecture of most computers tend to be alike. It is therefore possible to develop a general purpose assembler for such micro-processors. In this section, the technique employed in the MZ-80 assembler is described. This will serve as a model for designing general-purpose assemblers.

The basic operation of any assembler is the interpretation of statements. It is therefore important to establish a proper statement coding format. **Figure 2-3** shows an example of a coding format, used in the MZ-80 assembler, which is familiar to humans and which is easy for the computer to interpret.

Scanning the statements in this format, the assembler:

- (1) Recognizes labels and stores them into the label table,
- (2) Recognizes fields and assembles object codes,
- (3) Generates an assembly listing, and
- (4) Generates relocatable binary code.

Step (2) differs from one processor to another. The assembler constitutes a general-purpose assembler if it can perform this step flexibly. As the nucleus of the process for step 2, an instruction list (**Figure 2.4**) and a 2-dimensional operation table (**Table 1**) are introduced.

Label	:	Mnemonic	=	Operand 1	,	Operand 2	;	Comment	↵
Field 1		Field 2		Field 3		Field 4		Field 5	

Fig. 2-3 Assembler coding format

The symbol # in the instruction list represents a register and the symbol \$ represents a label or numeric value. The assembler identifies each instruction by matching the read assembly statement with this listing. As a result of this match, the assembler produces the major portion of the op-code, the byte length of the instruction and its atom type. An atom type is one of the numbers identifying the instruction groups of the Z-80 instruction set. As is seen from Table 1, there are 48 atom types; these are sufficient for newly defined instructions.

The operations to be performed for each atom type are designated by a 16-bit flag field. For atom type 01, for example, flag bits 0, 3 and 4 are set, indicating that the operations identified by these bits are to be performed in that order. The control words identified by the set flag bits specify the actual operations to be performed. Flag 3 indicates that this instruction must be a 1-byte instruction, that it must shift the data to the left 3 bits, and that the size of the field must be 3 bits or less. Similarly, flag 4 indicates that this atom type represents the LD r,r' operation.

Let us examine atom type 18. The set flag bits are 0, 1 and A. The control word for flag 1 is all zeros, which means no operation. Flag A indicates that the instruction requires address modification (address procedure) and that the address field must be not longer than 16 bits (size of the field). Thus, atom type 18 represents instructions such as JP nn' and JP NZ, nn'.

The above assembler operating procedure is summarized in Figure 2-5. Most of the assembly operations involve table references. In fact, the assembler uses a register table, a separator table and a label table during the assembly process, in addition to the instruction list and the 2-dimensional operation table. If these tables are redefined to conform to a new instruction set the assembler may also be used as a cross assembler. The MZ-80B assembler is currently being used not only as a Z-80 self-assembler but also as cross assemblers for the Intel 8080A, Fujitsu MB8840 series (4-bit microprocessors), and NEC μ COM-40 series (4-bit microprocessors).

01	0000	:			
02	0000	:	INSTRUCTION LIST		
03	0000	:			
04	0000	:			
05	0000	SYMP :	ENT		
06	0004		DFFM	' LD #, #'	; LIKE LD B, C
07	0006		DFFB	F1H	F delimits the instruction pattern. 1 indicates the length of the instruction in bytes.
08	0007		DFFB	40H	Main portion of the mnemonic code
09	0008		DFFB	01H	Atom type
10	0009		DFFM	' LD #, (IX\$) '	; LIKE LD A, (IX+15)
11	000D				
12	0011				
13	0013		DFFB	F3H	3 indicates the length of the instruction in bytes.
14	0014		DFFW	46DDH	DD4600 is the main portion of the mnemonic code.
15	0016		DFFB	00H	
16	0017		DFFB	03H	Atom type
17	0018		DFFM	' LD #, (IY\$) '	; LIKE LD B, (IY+AFC)
18	001C				
19	0020				
20	0022		DFFB	F3H	
21	0023		DFFW	46FDH	
22	0025		DFFB	00H	
23	0026		DFFB	03H	
24	0027		DFFM	' LD (IX\$), #'	; LIKE LD (IX+23), A
25	002B				
26	002F				
27	0031		DFFB	F3H	
28	0032		DFFW	70DDH	
29	0034		DFFB	00H	
30	0035		DFFB	04H	

Fig. 2.4 Instruction list (part)

Table 1 Two-dimensional operation table

Atom	Description	Flags (analyzed and processed in ascending flag bit number order)															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	Reserved																
01	LD #, #	1			1	1											
02	LD #, \$	1					1			1							
03	LD #, (IX+\$) LD #, (IY+\$)	1					1	1		1							
04	LD (IX+\$), # LD (IY+\$), #	1	1							1	1						
05	LD (IX+\$), \$ LD (IY+\$), \$	1	1						1	1							
06	LD A, (\$)	1	1										1				
07	LD (\$), A	1											1				
08	LD BC, \$ etc.	1	1										1				
09	LD IX, \$ LD IY, \$	1	1										1				
0A	LD HL, (\$)	1	1										1				
0B	LD BC, (\$) etc.	1	1										1				
0C	LD (\$), HL	1											1				
0D	LD (\$), BC etc.	1											1				
0E	ADD A, # etc.	1	1			1											
0F	ADD A, \$ etc.	1	1							1							
10	ADD A, (IX+\$) etc.	1	1					1		1							
11	INC # etc.	1			1												
12	INC (IX+\$) etc.	1	1							1							
13	RLC # etc.	1				1											
14	RLC (IX+\$) etc.	1	1						1								
15	BIT \$, # etc.	1		1		1											
16	BIT \$, (HL) etc.	1		1													
17	BIT \$, (IX+\$) etc.	1		1				1	1								
18	JP NZ, \$ etc.	1	1										1				
19	JR C, \$ etc.	1	1											1			
1A	JR \$ DJNZ \$	1												1			
1B	SUB # etc.	1				1											
1C	SUB \$ etc.	1								1							
1D	SUB (IX+\$) etc.	1						1		1							
1E	RST \$	1		1													
1F	IN A, (\$)	1	1							1							
20	IN #, (C)	1			1												
21	OUT (\$), A	1								1							
22	OUT (C), #	1	1		1												
23																	
24																	
2E																	
2F																	
CONTROL WORD	ADDRESS PROCEDURE			1					1	1		1	1				
	MUST BE SINGLE			1	1	1	1		1	1	1		1				
	MUST BE ADR-2												1				
	LEFT SHIFT POSITION			1	1		1										
				1	1		1										
							1		1		1						
	DON'T CARE																
	EQUATION PROCEDURE			1					1	1							
	SIZE OF FIELD			1	1	1	1				1						
				1	1	1	1				1						
									1	1			1				
												1					

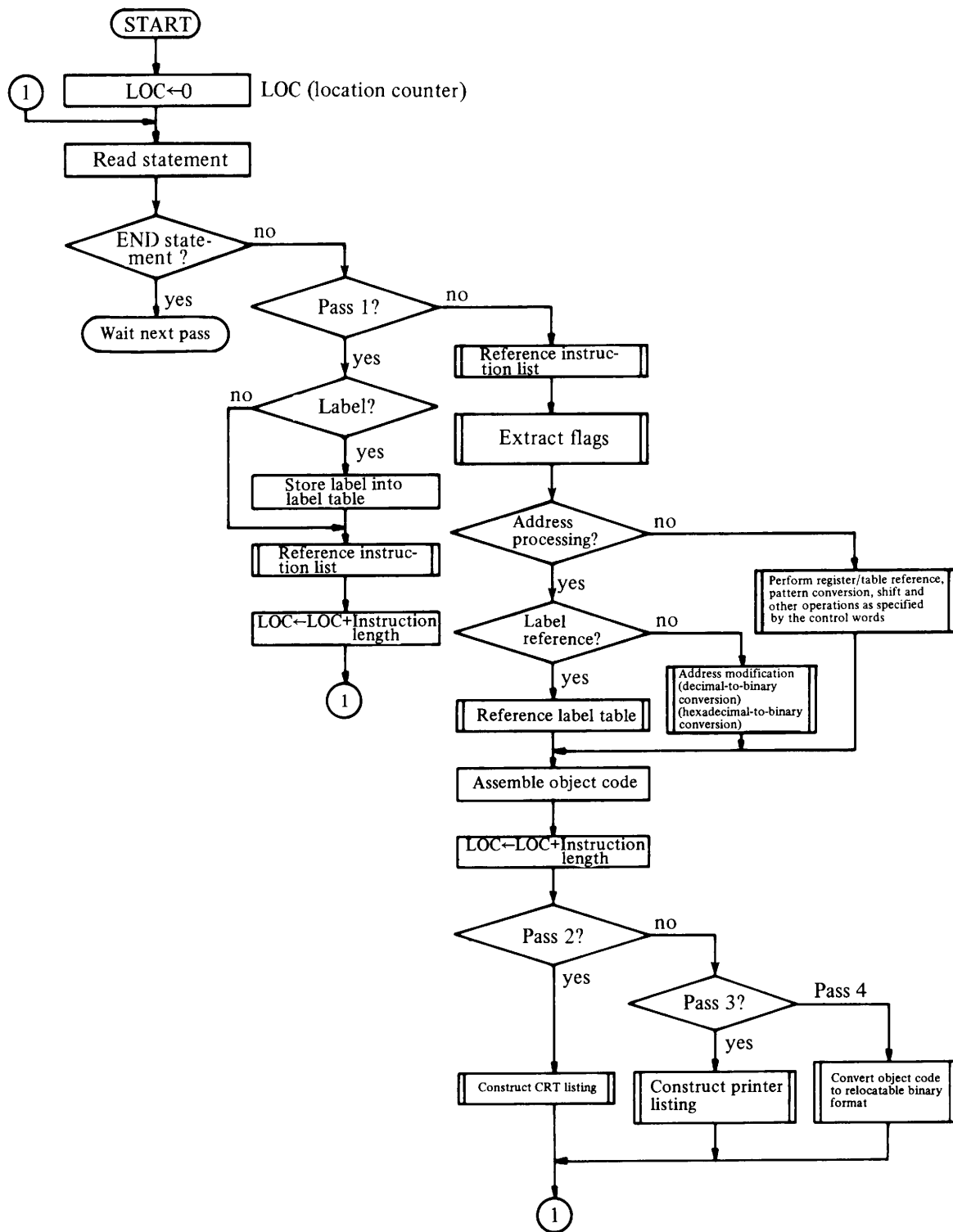


Fig. 2-5 General assembly flow (excluding assembler directive processing)

2.3 Linker

The linker loads and links two or more program units using external symbol referencing instruction from relocatable files and generates absolute binary code in the link area and saves it into an object file. The relocatable files contain control frames and external symbol information. The linker resolves external symbol references and relocates the program units as described below.

(1) External symbol reference resolution

The linker refers to the symbol table when resolving external symbol references (see **Figure 2-6**). The symbol table contains a 9-byte symbol table entry for each external symbol. A symbol table entry consists of a 6-byte field containing the symbol name, a 1-byte field containing the definition status, and a 2-byte field containing an absolute address with which the symbol is defined or a relocation address.

When the linker encounters an external symbol reference while loading the program unit from a relocatable file, it checks to determine whether the symbol has been cataloged in the symbol table.

- (1) If it has not been cataloged, the linker enters it into the symbol table as a new undefined symbol, loads the relocation address into the symbol table entry and loads code FFFFH into the operand address of the instruction in memory.
- (2) If it has been cataloged and defined, the linker loads the defined absolute address into the operand address in memory.
- (3) If it has been cataloged but not defined, the linker moves the old relocation address in the symbol table entry to the operand address in memory and loads the new relocation address into the symbol table entry.

Thus, the linker chains undefined references to each symbol and, when the symbol is defined, replaces all reference addresses with the defined absolute address. In other words, when an external symbol defined by the ENT assembler directive appears in the control frame, the linker enters the symbol into the symbol table as a defined symbol and replaces all preceding operand addresses chained in memory (terminated by FFFFH) with the absolute address defined. The programmer can examine the definition status of the symbols using the table dump command.

An example of external symbol reference resolution follows. Assume that three program units are to be linked and that each unit references subroutine SUB1 in the third program unit (see **Figure 2-8**).

When the first CALL SUB1 instruction is encountered in program unit 1, the linker enters SUB1 into the symbol table as an undefined symbol, loads the operand address (relocation address 5001H in this case) into which the value of the symbol is to be loaded into the 2-byte value field of the symbol table entry and loads the code FFFFH into the operand address in memory (see **Figure 2-8(a)**).

When the CALL SUB1 instruction is encountered twice in program unit 2, the linker chains together their operand addresses which reference SUB1 (see **Figure 2-8(b)**). When SUB1 is defined in program unit 3, the linker designates SUB1 as a defined symbol and loads all operand addresses referencing SUB1 with the defining absolute address. The end of the operand address chain is identified by the code FFFFH. **Figure 2-8(c)** shows that SUB1 is defined by absolute address 5544H. When the linker subsequently encounters a CALL SUB1 instruction, it immediately loads 5544H into the operand address of the instruction since symbol SUB1 has been defined.

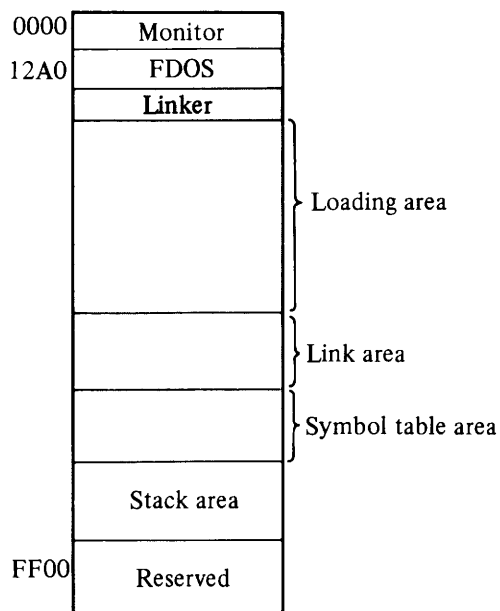


Fig. 2-6 Memory map for the linker

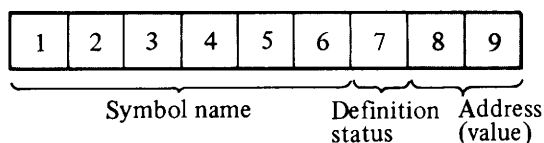


Fig. 2-7 Symbol table entry format

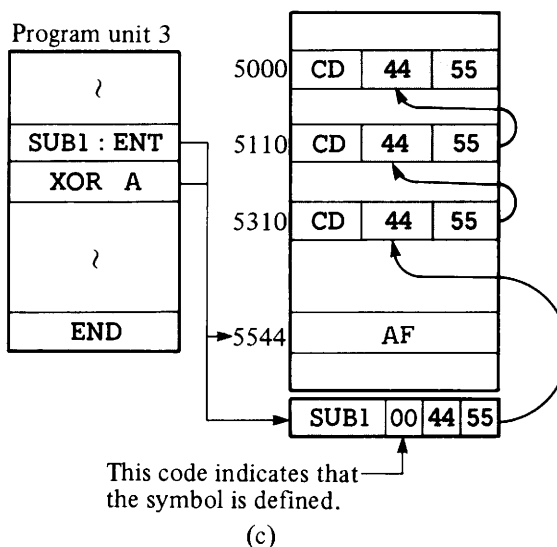
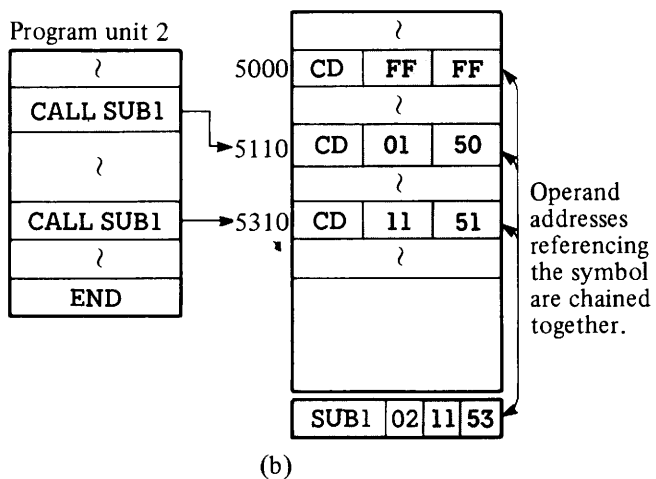
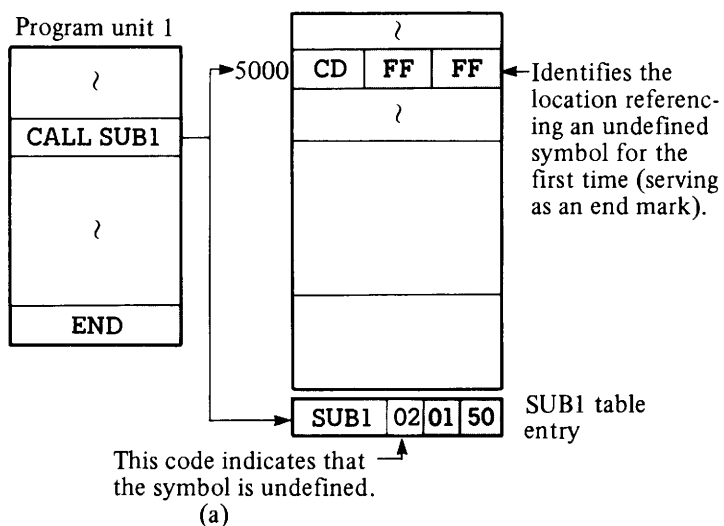


Fig. 2-8 Example of external symbol reference chaining

(2) Program relocation

The linker relocates instructions referencing external symbols while linking the programs. For instructions which reference internal symbols and for which relocation addresses are generated by the assembler, however, the linker produces absolute addresses for the symbols by adding bias to the relocation addresses.

Thus, the linker generates absolute binary code in the link area in an executable format which is dependent on the bias specified by the programmer when the program unit is loaded. When creating an object file, the linker saves the absolute binary code from the link area in the file together with its loading address and execution address.

2.4 Symbolic Debugger

The symbolic debugger inputs relocatable files under the same input conditions as the linker except that it presumes that absolute binary code is loaded into the link area in an immediately executable form. The symbolic debugger permits the programmer to debug his program while running it.

With the symbolic debugger, the programmer can run a program, interrupts its execution at specified locations and check the system status at these points. The programmer specifies the breakpoints at which program execution is interrupted. When a breakpoint is encountered, the symbolic debugger saves the operation code at the address set as the breakpoint in the break table and replaces it with an RST 6 instruction (F7H) (see Figure 2-9).

The RST 6 instruction is a 1-byte call instruction to address 30 in hexadecimal. Its operation is as follows:

$$(SP - 1) \leftarrow PC_H, (SP - 2) \leftarrow PC_L$$
$$PC \leftarrow 0030H$$

Hexadecimal address 30H contains a jump instruction which transfers control to the breakpoint control routine in the debugger.

Each breakpoint is associated with a break counter. A break is actually taken when the breakpoint is reached the number of times specified by the break counter. Before the break count is reached, execution is continued with the original operation code saved.

When a break occurs, the debugger saves the contents of the CPU registers in the register buffer and displays them in the screen. When the program is restarted, the debugger restores the contents of the register buffer to the CPU registers and pops the break address.

The programmer can specify a maximum of nine breakpoints and a maximum break count of 14 in decimal.

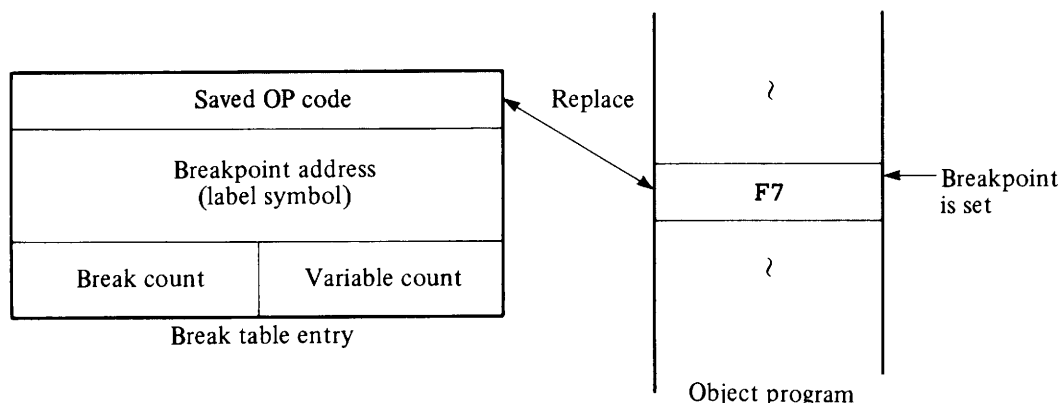


Fig. 2-9 Breakpoint setting and breakpoint table format

The symbolic debugger has indicative start and memory list dump commands in addition to the breakpoint setting command, execution command, memory dump command and register command. The indicative start (I) command displays contents of the CPU registers with which the program is to be executed for confirmation before actually transferring control to the address designated by the program counter (PC) displayed. For example, when an I command is entered, the display shown in **Figure 2-10** appears on the screen. When the programmer pressed **[CR]** after confirming the CPU register contents, the debugger initiates an indicative start as shown in **Figure 2-11**.

```
#DI
A 01 F 23 45 67 89 ABCD EFGH IJKL
PC 01 F 23 45 67 89 ABCD EFGH IJKL
SP 7500 11EF IX 4559 IY 4569 I 00
Start OK?
```

The above display shows that the program is to be started at address 7500 (hex) with the CPU register values shown.

Fig. 2-10 I command example

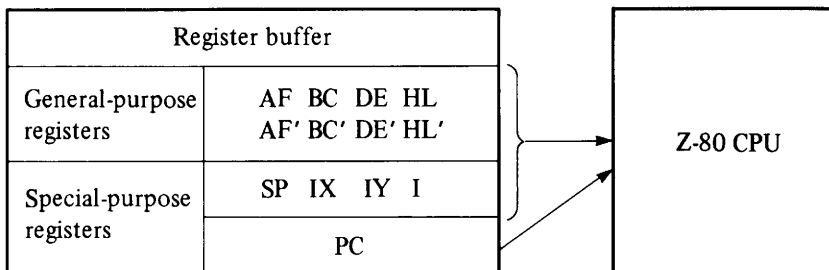


Fig. 2-11 I command operation

The debugger restores the contents of the general-purpose registers and special-purpose registers SP, IX, IY and I, then the value of the PC and initiates program execution.

The memory list dump (D) command displays the machine code in the specified memory block with one instruction on each line.

The symbolic debugger permits the programmer to symbolically specify addresses as shown in **Figure 2-12**. With symbolic addresses, the programmer can specify any addresses in the program wherever the program is located in memory.

The programmer can specify the following types of addresses symbolically:

- (1) Addresses represented by a symbol
- (2) The address of an instruction 1 to 65535₁₀ lines away from the address represented by the symbol
- (3) An address ± 1 to 65535₁₀ bytes away from the address represented by the symbol

Of course, the programmer can also specify memory locations with absolute addresses.

For example, the program unit whose source program is shown at the left of **Figure 2-12** is loaded into memory by the debugger starting at hexadecimal address 7500, execution of a D command will display a dump of the memory block as shown at the right in **Figure 2-12**.

```
START : ENT
        LD     SP, START
        CALL  MSTP
        XOR   A
        LD    (? TABP), A
        LD    B, A
MAIN0 : ENT
        LD    A, 0FH
```

```
*DD START MAIN0
7500 310075
7503 CD4700
7506 AF
7507 32BE75
750A 47
750B 3E0F
*D
```

Fig. 2-12 D Command

2.5 PROM Formatter

The PROM formatter generates formatted absolute binary code and stores it into paper tape under the PTP control. It is the system backup software used to transfer object programs to the PROM writer. Currently, the following paper tape output formats are supported (see Figure 2-13):

- (1) BNPF format: Britronics, Intel and Takeda
- (2) B10F format: Takeda
- (3) Hexadecimal format: Britronics, Takeda, Minato Electronics
- (4) Binary format: Britronics

The variety of tape formats supported by the SHARP PROM formatter extends the application range of programmable ROMs.

```
format ? T
format list
A BNPF (Brightronic RPG-8764)
B hexadecimal
C binary
D BNPF (Intel MDS800)
E BNPF (Takeda T310/28)
F B10F
G hexadecimal
H Minato format
free area 6500 -- D9FF
format ?
```

Fig. 2-13 Paper tape output formats

The PROM formatter is made up of format, the PTP and the PRT controls (See Figure 2-14). These enable the programmer to perform format conversion.

The formatter checks parity in one of three modes (even parity, odd parity or no parity) when reading paper tape. In the formats using ASCII code (BNPF, B10F and hexadecimal), the most significant bit is assigned even or odd parity. When even parity is used, for example, ASCII code "A" (41 hexadecimal) is punched as is, whereas "C" (43 hexadecimal) is converted to C3 in hexadecimal before being punched by setting its MSB. The parity mode can be set using the P command with the desired switch assigned, e.g. *PSPTP/PE/LF.

This PROM formatter assumes that the PTP/PTR interface is compatible with the RP-600 puncher/reader from the Nada Electronics Laboratory. It can control RP-600 directly using the general-purpose I/O card (MZ-80I/O-2). It can also control other models, such as the DPT26A paper tape punch from Anritsu, if I/O conforming to the punch specifications can be implemented on the general-purpose I/O card.

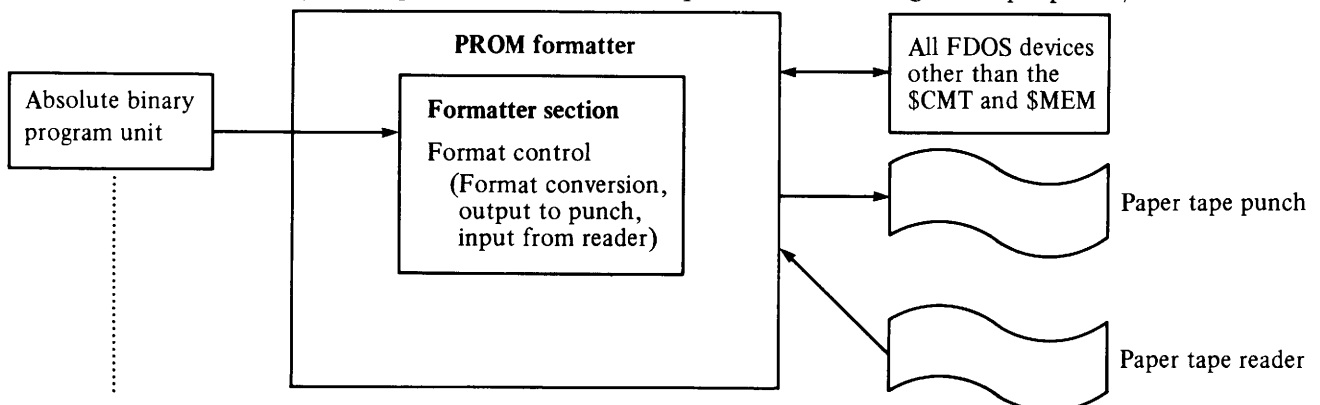


Fig. 2-14 PROM formatter configuration

3. FDOS ORGANIZATION

Figure 3-1 shows the files which are run under control of the SHARP MZ-80B FDOS. The FDOS has the following features:

- (1) Multistatement processing.
- (2) Default argument processing.
- (3) Allows wildcard characters in file references.
- (4) File-oriented processing extended to I/O devices.

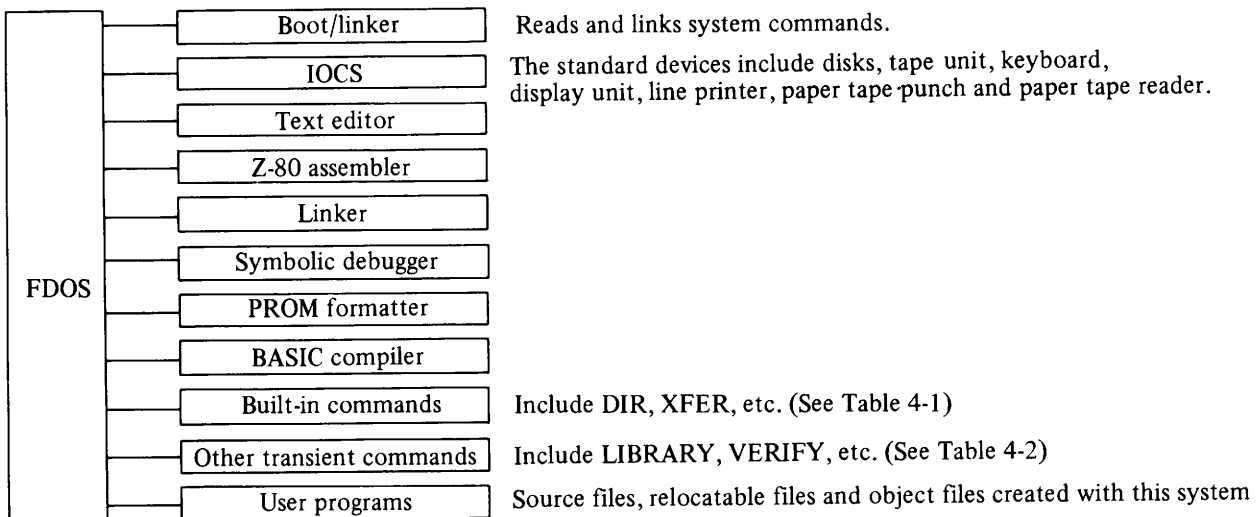


Fig. 3-1 FDOS file organization

Figure 3-2 shows the memory map for the above system resources. FDOS is made up of a resident section and an overlay section. Their resident section includes:

- (1) A command line interpreter which interpretes and executes system commands.
- (2) A boot linker which reads and links command files from the FDOS diskette.
- (3) A supervisor call procedure which manages system resources, including files.
- (4) An I/O control system (IOCS)
- (5) A file management program which manages the diskette allocation map, file table and other information.

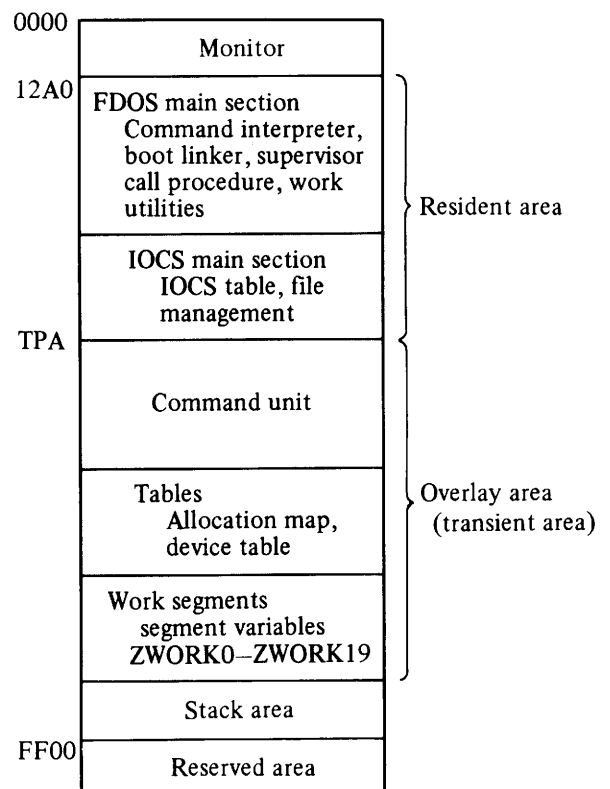


Fig. 3-2 FDOS memory map

3.1 Boot Linker

The FDOS transient commands (whose file mode is .SYS) are not resident in memory, but are stored in relocatable files on the system diskette. These programs exist not in absolute form but in relocatable form. When they are invoked, boot linker relocates them and specifies their loading addresses (see **Figure 3-3**).

These relocatable system files differ from relocatable files generated by the assembler in the way in which they are loaded into memory. The external symbol references of the system files have been resolved; these are just relocated by the boot linker. Accordingly, the control frame associated with each statement of the system programs contains only a field identifying the statement as having a relative address or absolute data and containing the byte count of the statement. When a relative address is indicated in the control frame, the system adds loading bias to the relative address to form an absolute address.

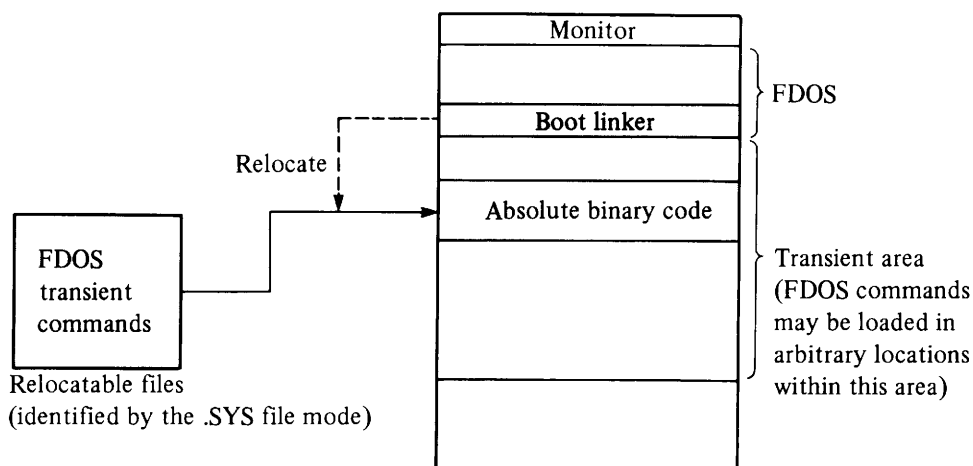


Fig. 3-3 Loading FDOS transient command with the FDOS boot linker

3.2 IOCS

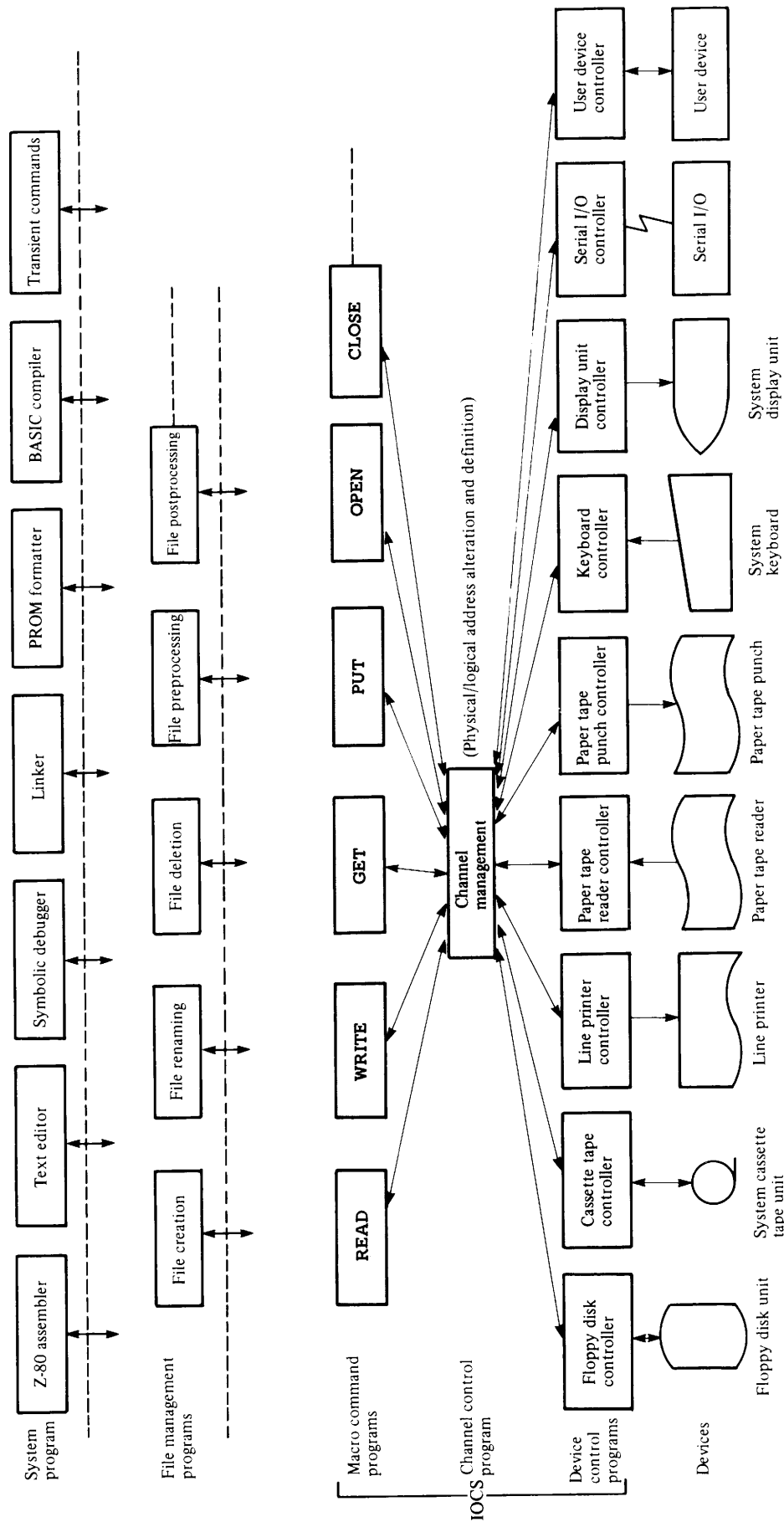
IOCS in FDOS provides control over the display unit, cassette unit, floppy disk unit and printer. The programmer can define other I/O devices using the ASSIGN command.

Control programs for such user I/O devices can be stored in external files and their names can be cataloged in the IOCS table. They are invoked and executed by IOCS as required.

The actual file management programs form a hierarchical structure as shown in **Figure 3-4**. In the MZ-80B system, routines from the macro command programs to the device control programs are collectively called the input/output control system (IOCS). Being of modular construction, these programs are as independent of each other as possible. By hiding controls unique to I/O devices, such as device address management and buffering, IOCS permits the programmer to handle these programs as logical files and to control the I/O devices as general files.

The alternate start/stop feature is enabled during IOCS operations. The system temporarily suspends the read operation when an alternate stop is effected during a data read. At this point, the programmer can switch to the FDOS command mode or continue the suspended IOCS operation by effecting an alternate start.

Fig. 3-4 Hierarchical structure of file management programs



3.3 Dynamic Segmentation

Memory segmentation and relocation can be accomplished easily if a hardware relocation register is used. However, no presently available 8-bit microprocessor has such a register. Consequently, methods of simulating this function are commonly used. The boot linker previously mentioned can be thought of as a variation of such simulations. Here, a method of memory segmentation and assignment which leaves the memory image unchanged is described.

Two subroutines are used for memory segmentation as shown in **Figure 3-5** and **3-6**. These two subroutines and segment variables are maintained in fixed locations in the FDOS main program area. They are accessible to all programs. The 20 segment variables are initialized during preprocessing for each command and assigned values so that no memory segment exists. They are redefined as required during processing of each command, thus creating memory segments.

Fig. 3-5 Extending a specified segment

```
A ← 2      ; Segment No. (0-19)
BC ← 500    ; 500 bytes
CALL DOPEN  ; DYNAMIC OPEN
```

Segment No.	Segment variables	Results
0	ZWORK 0 : 5000	ZWORK 0 : 5000
1	ZWORK 1 : 5500	ZWORK 1 : 5500
2	ZWORK 2 : 6000+(500)	ZWORK 2 : 6500
3	ZWORK 3 : 6500+(500)	ZWORK 3 : 7000
4	ZWORK 4 : 7000+(500)	ZWORK 4 : 7500
5	ZWORK 5 : 7500+(500)	ZWORK 5 : 8000
6	ZWORK 6 : 8000+(500)	ZWORK 6 : 8500
7	ZWORK 7 : 8500+(500)	ZWORK 7 : 9000
⋮	⋮	⋮
18	ZWORK18 : 29000+(500)	ZWORK18 : 29500
19	ZWORK19 : 29500+(500)	ZWORK19 : 30000

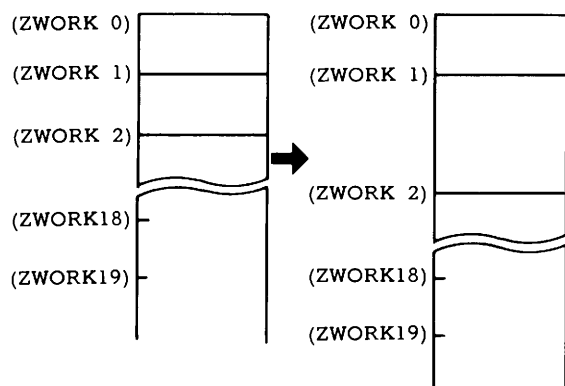
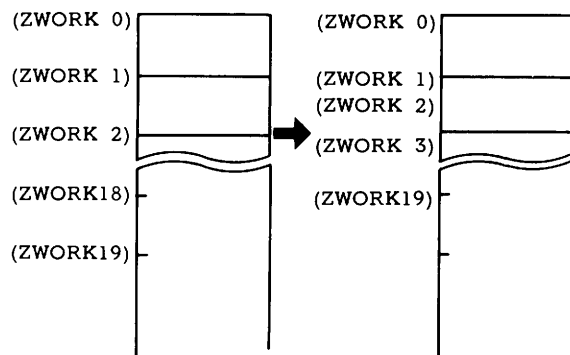


Fig. 3-6 Deleting a specified segment

```
A ← 2      ; Segment No. (0-19)
BC ← 500    ; 500 bytes
CALL DDELET ; DYNAMIC DELETE
```

Segment No.	Segment variables	Results
0	ZWORK 0 : 5000	ZWORK 0 : 5000
1	ZWORK 1 : 5500	ZWORK 1 : 5500
2	ZWORK 2 : 6000-(500)	ZWORK 2 : 5500
3	ZWORK 3 : 6500-(500)	ZWORK 3 : 6000
4	ZWORK 4 : 7000-(500)	ZWORK 4 : 6500
5	ZWORK 5 : 7500-(500)	ZWORK 5 : 7000
6	ZWORK 6 : 8000-(500)	ZWORK 6 : 7500
7	ZWORK 7 : 8500-(500)	ZWORK 7 : 8000
⋮	⋮	⋮
18	ZWORK18 : 29000-(500)	ZWORK18 : 28500
19	ZWORK19 : 29500-(500)	ZWORK19 : 29000



4. FDOS COMMAND USAGE

4.1 Program Development Under FDOS

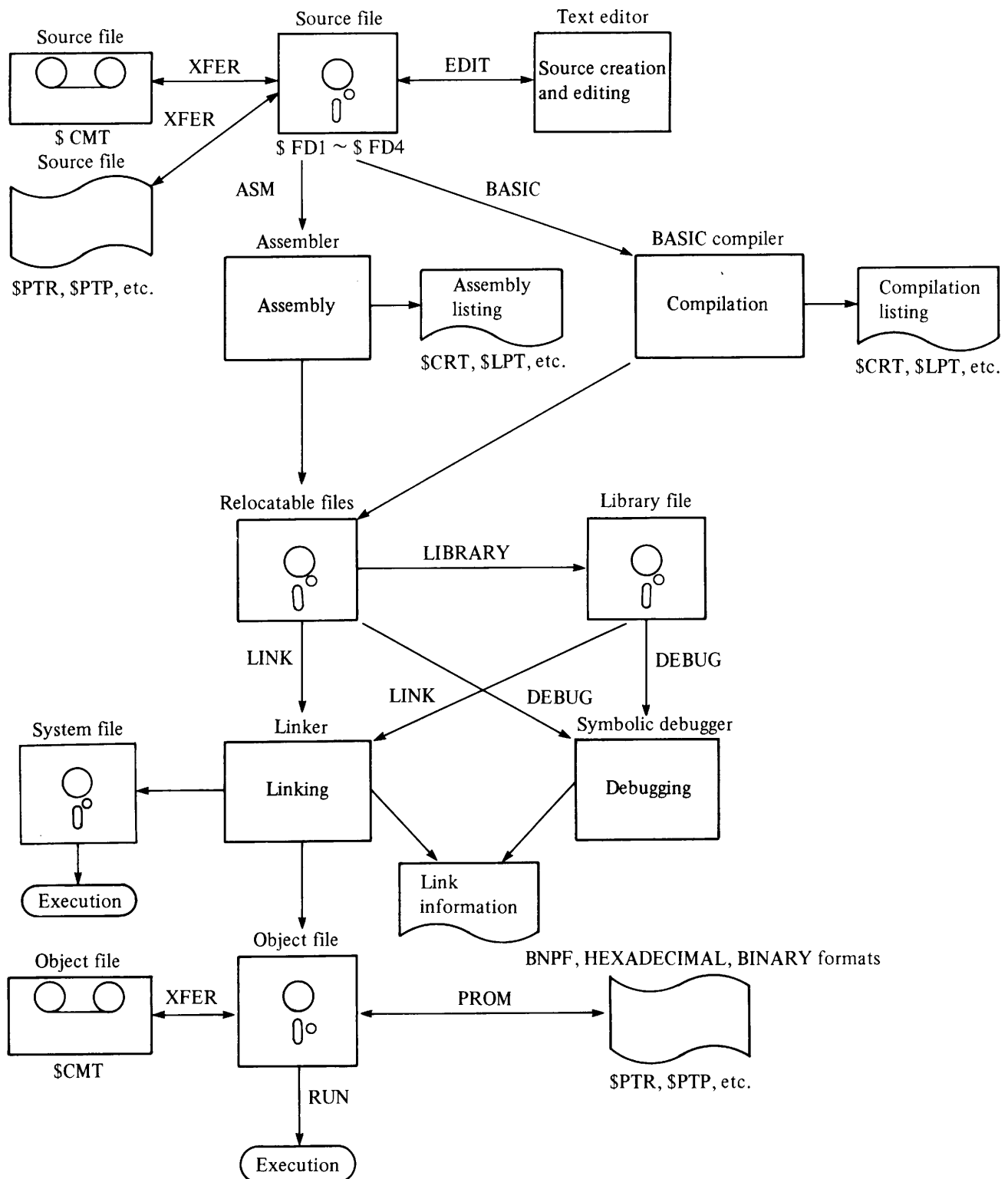
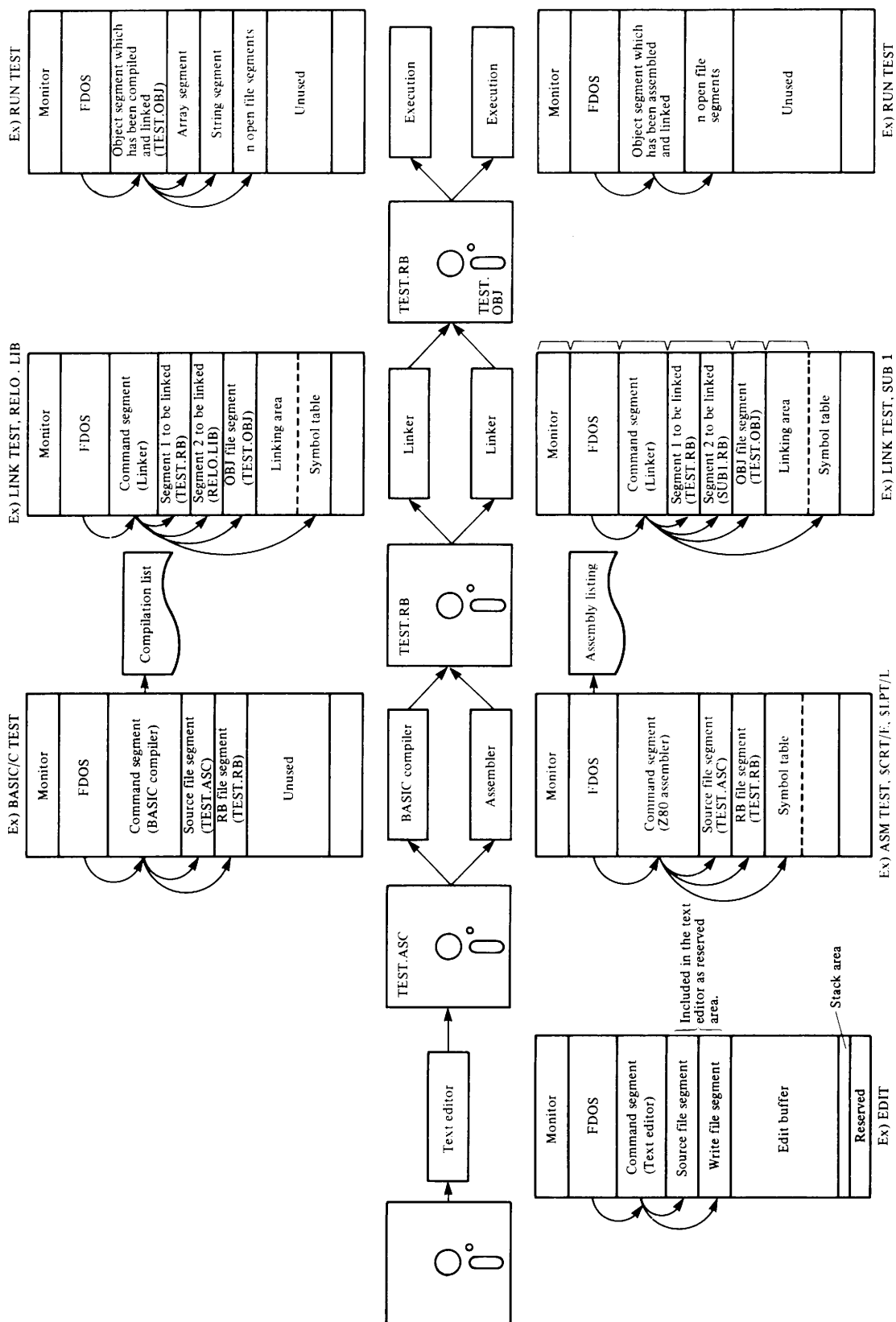


Fig. 4-1

Fig. 4-2 Activation of Segments by FDOS



4.2 FDOS Control Keys

4.2.1 Main keyboard

Except for the following, the control keys on the main keyboard are used in the same manner as under the SB-1510.

SHIFT	The scrolling speed of the display data is maintained at the preset speed while this key is held down. When this key is released, the scrolling speed returns to the maximum speed. The scrolling speed is set with POKE \$000F nn nn = 01 ~ FF The speed slows down as the value of nn is increased. nn = 40 Normal speed
SHIFT + 0	Deletes the portion of the line from the cursor position to the end of the line.
SHIFT + 1	Sets a tab at the cursor position.
SHIFT + 2	Resets the tab at the cursor position.
SHIFT + 3	Resets all tabs set by the above procedure.
SHIFT + 4	Sets the number of characters per line to 40. The screen is cleared and the cursor is returned to the home position.
SHIFT + 5	Reverses the shift mode of the alphabetic keys. Making these entries again resets the reversed shift mode.
SHIFT + 8	Sets the number of characters per line to 80. The screen is cleared and the cursor is returned to the home position.
SHIFT + INST	Enables insertion of an arbitrary number of characters at the cursor position. Pressing the CR key terminates insertion.
BREAK	Terminates the program currently being executed, displays the message "Break" and awaits entry of a new FDOS command. Executing ON BREAK GOTO under the BASIC compiler causes a jump when the BREAK key is pressed.
SPACE	Holding down the space key for a certain period of time suspends current program execution. The time differs according to the operation currently being executed. For example, when the printer is operating, the space key must be held down until a carriage return is performed. After program execution has been suspended, one of the following operations is possible. • Pressing the BREAK key : See the explanation above. • Pressing the SPACE key : Resumes program execution.

The **0** through **8** keys are on the numeric pad.

It is convenient to affix seals on which the following functions are printed to the front of the numeric keys to identify the functions of **SHIFT**+**0**—**5**, **8**.

DELETE, SETTAB, CLRTAB, CLR , CHR40, CHANGE, CHR80
TO EOL ALL TAB

4.2.2 Automatic repeat function

All keys other than the cassette tape control keys are provided with the automatic repeat function: when a key is held down for more than a preset period of time, the key entry is automatically repeated at a preset speed. The period and speed are stored in memory location 000D and can be set with the following BASIC statement.

POKE \$000D sstt

ss = 01 ~ FF : The repetition speed is reduced as the value of ss increases.

tt = 01 ~ FF : The period described above is determined by (ss) * (tt), so it becomes greater as the value of tt is increased.

Example:

POKE \$000D 2010

4.2.3 Cursor control keys

Key entry	Picture character	Code	Function
[GRPH] + 		01H	Moves the cursor down 1 line.
[GRPH] + 		02H	Moves the cursor up 1 line.
[GRPH] + 		03H	Moves the cursor to the right by 1 space.
[GRPH] + 		04H	Moves the cursor to the left by 1 space.
[GRPH] + [HOME]		05H	Moves the cursor to the home position.
[GRPH] + [CLR]		06H	Clears the screen and moves the cursor to the home position.
[SHIFT] + [TAB]		1FH	Delimiter

4.2.4 Initial settings

Various initial values are set when FDOS is activated by MZ-80B system IPL. These values are the initial default values, and they can be updated by the programmer.

- Definable function keys

[F1] RUN	[F11]  \$ F D 1 ;
[F2] X F E R 	[F12]  \$ F D 2 ;
[F3] D E L E T E	[F13]  \$ K B 
[F4] R E N A M E 	[F14]  \$ C R T
[F5] D I R	[F15]  \$ L P T/L
[F6] E D I T 	[F16]  \$ C M T ;
[F7] A S M	[F17] . A S C 
[F8] L I N K	[F18] . R B 
[F9] D E B U G	[F19] . L I B 
[F10] B A S I C	[F20] . O B J 

For [F11] - [F20], press [F1] - [F10] and [SHIFT] simultaneously.

- Scrolling speed: nn=80
- Automatic repeat speed and preset period:
 $(ss) * (tt) = 40 * 0C$
- Tab spacing: 5 characters
- Small letter input mode: Shift position in the normal mode or **SHIFT** + **5**
- Other initial values are the same as those set by BASIC SB-5510.

4.2.5 Differences between the SB-1511 and SB-1510

Item	SB-1510	SB-1511 (monitor)
Automatic repeat function	• Cursor control keys only • Key entry is repeated only when a cursor control key and the SHIFT key are pressed simultaneously.	• All keys other than the cassette tape control keys • The repetition speed and the time required for starting repetition are variable. See page 22.
Definable function keys	• Up to 10 functions can be assigned. F1 ~ F10	• Up to 20 functions can be assigned. F1 ~ F10 and SHIFT + F1 ~ F10
Interrupt	When interruptions are disabled upon entry to a subroutine, they are enabled before the RET instruction is executed.	When interruptions are disabled upon entry to a subroutine, they are enabled or kept disabled according to the condition set just before control was transferred to the subroutine.
RST	RST7 (PANIC) displays the contents of registers AF, BC, DE, HL and PC and awaits entry of a new monitor command.	• RST7 (PANIC) displays the contents of registers AF, BC, DE, HL, PC and SP and awaits entry of a new monitor command. • RST6 is reserved for use by the debugger.

4.3 FDOS Command Coding Rules

This section describes the coding rules for FDOS commands.

4.3.1 Command line format

In the command mode, FDOS prompts for command entry with a number and the symbol ">". Enter a command followed by arguments (described later), if necessary, press `[CR]` key and the FDOS will execute the command.

Example 1:

2 > EDIT TEST `[CR]`

The diagram shows the command line "2 > EDIT TEST [CR]". Arrows point from labels to parts of the command: "Default drive number (described later)" points to "2", "Prompt" points to ">", "Command" points to "EDIT", "Argument" points to "TEST", and a note "denotes a space." points to the space between "TEST" and "[CR]".

The first number (1 ~ 4) indicates the default drive, namely, the currently logged-on disk drive.

Some commands may require two or more arguments.

Example 2:

2 > XFER TEST, \$ CMT `[CR]`

The diagram shows the command line "2 > XFER TEST, \$ CMT [CR]". Arrows point from labels to parts of the command: "Command" points to "XFER", "Argument 1" points to "TEST", "Argument 2" points to "\$ CMT", and "[CR]" is at the end.

The command and arguments must be separated by commas and/or spaces.

(Legal) 2 > XFER TEST \$ CMT `[CR]`

(Legal) 2 > XFER , TEST , \$ CMT `[CR]`

(Illegal) 2 > XF ER TEST , \$ CMT `[CR]`

Only one comma is allowed.
No space is allowed.

Two or more commands may be specified on one logical line by separating them with colons (":"). A line containing two or more commands is called a multistatement line. A logical line may contain any number of commands but it must not exceed 160 characters in length.

Example 3: 2 > DELETE TEST : RENAME AAA, TEST : ASM TEST `[CR]`

Example 4: 2 > Xfer \$ kb, aBc

Either upper or lower case letters may be used for commands and arguments. The FDOS does not distinguish between upper and lower case letters.

4.3.2 File name

All program and data files on a diskette are given file names. The programmer must specify a file name when storing a program or data file on a diskette and when reading it. A file name must be from 1 to 16 alphanumeric characters (including lower case letters) and/or special characters !, #, %, &, ', (,), +, -, <, =, >, @, [,\,], ^ and ←.

No two files on a diskette can have the same file name and file mode (described later). Files with the same file name may exist on a diskette if their file modes are different from one another.

(Files with the same file name and mode may exist on different diskettes).

4.3.3 File modes

The file mode identifies the type of the file. It is usually used with a file name. The MZ-80B file modes are listed below.

File mode

File mode	Meaning
.OBJ	Identifies an object file which contains Z80 machine code.
.ASC	Identifies a source file, such as one created by the text editor, which contains a stream of ASCII characters.
.RB	Identifies a relocatable file which contains pseudo-machine language code (relocatable binary code) generated by the assembler or compiler.
.LIB	Identifies a library file consisting of two or more relocatable files.
.SYS	Identifies a file containing a system program which runs under FDOS, such as the text editor and assembler.

4.3.4 File attributes

File attributes are information pertaining to file protection. There are four file attributes: 0, R, W and P. File attribute 0 indicates that a file is not protected. The other file attributes inhibit the use of specific commands as listed below.

File attribute	R	W	P
Inhibited FDOS Commands	TYPE XFER EDIT ASM LINK DEBUG PROM BASIC	DELETE RENAME	TYPE XFER EDIT ASM LINK DEBUG PROM BASIC DELETE RENAME
Inhibited BASIC Commands	ROPEN # INPUT # ()	PRINT # ()	ROPEN # INPUT # () PRINT # ()

4.3.5 File types

A file type indicates the file access method. There are two file types: sequential (S) and random (X). FDOS normally handles only sequential files. Random files can be accessed only by the DELETE, RENAME and CHATR commands. An optional BASIC compiler is required to create, write to and read from random files.

4.3.6 Wildcard characters

The programmer can specify two or more files at a time by specifying wildcard characters in the file name and file mode. The wildcard characters "?" and "*" are used for file names and ".*" is used for file modes.

Wildcard character "?"

"?" represents any one character. For example, assume that files ABC.ASC, ABC3.ASC, ABCD.RB, XYZ.ASC and ADCN.ASC exist on the currently logged-on disk. When the command.

TYPE A ? C ? .ASC

is entered, the contents of the files ABC3 .ASC and ADCN .ASC will be displayed.

Wildcard character "*"

"*" Represents 0 or more characters.

A*: Represents file names beginning with "A" e.g., A, A2, ABC

*2 : Represents file names ending with "2" e.g., TEST2, SAMPLE2

P*5: Represents file names beginning with "P" and ending with "5" e.g., PROGRAM5, PM5

Wildcard characters ".*"

".*" represents all file modes.

Examples:

DELETE PROG 1.*	Deletes all files whose file name is PROG1
XFER *.ASC, \$PTP	Punches all files whose file mode is .ASC.
DIR A *B* ? 3 .RB	
DELETE *.*	Deletes all files on the diskette.

4.3.7 Drive number and volume number

A drive number refers to the drive number of a floppy disk drive (MZ-80FB or MZ-80FBK). Drive numbers 1 through 4 are assigned device names \$FD1 through \$FD4 respectively.

A volume number (1-255) is a number identifying a diskette. FDOS checks this number for validity each time it accesses a file.

4.3.8 Basic device name

FDOS can handle the following I/O devices:

\$KB :	MZ-80B system keyboard
\$CRT :	MZ-80B system display unit
\$FD1 :	} Floppy disk drives (MZ-80FB or MZ-80FBK)
\$FD2 :	
\$FD3 :	
\$FD4 :	
\$CMT :	System cassette unit
\$LPT :	System printer (MZ-80P4 or MZ-80P5)
\$MEM :	A part of main memory regarded as an I/O resource.

The system automatically reserves an unused area as \$MEM. This area is released by the DELETE \$MEM command or when an error occurs.

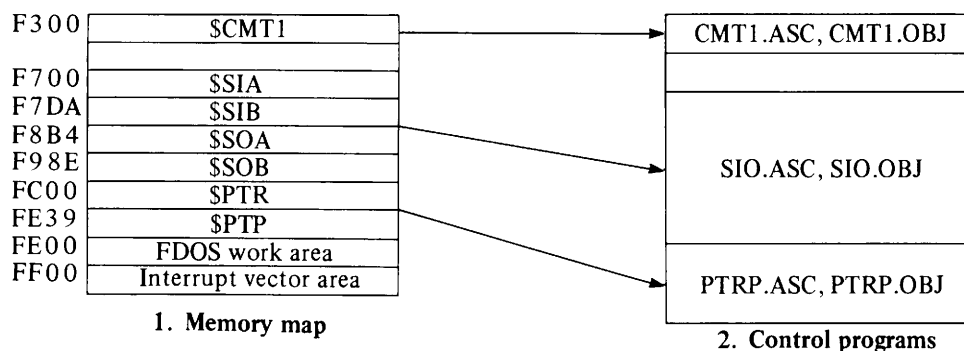
4.3.9 Auxiliary device name

Auxiliary devices are devices whose control programs are not resident in the FDOS area in memory. Their control programs are stored in external files. An auxiliary device name is assigned to an auxiliary device control program using the ASSIGN command to allow IOCS to manage the control program.

\$PTR :	{	Paper tape reader and punch. The user must prepare an interface circuit for these using a universal interface card. The system contains their control programs, however. For details, refer to "PAPER TAPE PUNCH AND READER INTERFACE" in the Appendix.
\$PTP :		
\$SIA :	{	The interface card for these I/O ports is optional.
\$SIB :		
\$SOA :		
\$SOB :		
\$CMT1 :		Cassette tape deck for the MZ-80K
\$USR1 :	{	These device names are provided for user-supplied I/O devices. The control program must be supplied by the user.
\$USR2 :		
\$USR3 :		
\$USR4 :		

To use these device names, prepare a machine language area using the LIMIT command, load the corresponding auxiliary device control program into the area using a LOAD command and link the program with the I/O controller of FDOS using an ASSIGN command. The auxiliary device control programs are supplied in the form of object files and ASCII files. In general, use the object files. If you want to change the loading address, assemble and link the ASCII files with FDOSEQU.LIB from the master diskette.

The loading address of each auxiliary device control program is shown below.



Example 5: 1 > LIMIT \$F300
 1 > LOAD CMT1 SIO PTRP
 1 > ASSIGN \$CMT1 \$F300 \$SIA \$F700 \$SIB \$F7DA \$SOA \$F8B4 \$SOB
 \$F98E \$PTR \$FC00 \$PTP \$FE39

Example 6: EXEC \$FD1 ; LOADAUX
 All the auxiliary device control programs are loaded since file LOADAUX.ASC contains the above programs.

Notes:

1. Any file input from the keyboard (\$KB) is terminated by pressing the **BREAK** key. For example, execution of the command

1 > XFER \$KB, XYZ

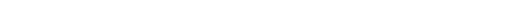
is terminated when the programmer presses the **BREAK** key.

- ### 4.3.10 Switches

Global switches

Example 7: $1 > \text{DATE}/P$ $/P$ denotes LPT.

Invalid: 1 > LINK_/P_/ _ T _ TEST

 No space may appear in these positions.

Local switches

Example 9: 1 > ASM TEST, \$ LPT/L, XYZ/O

/L specifies the device on which the assembly listing is
to be output.

/O specifies the relocatable output file.

Device switches

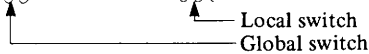
The meanings of the device switches are listed below.

Note: An error is generated (il data) if the MSB of the data is set to 1 from the beginning (e.g., graphic characters).

Note:

Any switch following the first argument of the RUN command is treated as a global switch.

Example 10: 1 > RUN ASM48/P TEST, XYZ/O

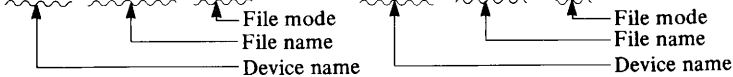


The meanings of the individual global switches are described in the related command descriptions.

4.3.11 Default assumptions

The general format of a file specification (valid for \$FD1–\$FD4 and \$CMT) is given below.

Example 11: \$ FD2 ; PROG2 . ASC \$ CMT ; TEST2 . OBJ



The programmer can omit portions of the complete file specification as explained below.

Default drive

The device name may be omitted as exemplified below.

Example 12: 2 > LINK TEST1, \$FD3 ; TEST2, TEST3

In the above example, the system assumes the name of the currently logged-on disk drive (identified by "2>") before TEST1 and TEST3. Consequently, the above command line is equivalent to the following:

2 > LINK \$FD2 ; TEST1, \$FD3; TEST2, \$FD2; TEST3

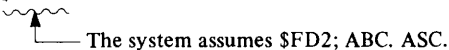
The default drive can be changed by:

1. Executing the DIR command or
2. Moving the cursor to the left of the prompt ">" and changing the drive number (e.g., changing "2>" to "1>").

Default file name

The file name may be omitted when reading files from the cassette tape unit (\$CMT). When a file name is omitted in the XFER command or other similar command (See example 10), the system assumes an appropriate file name.

Example 13 : XFER \$ FD1 ; ABC . ASC, \$ FD2



Default file mode

When the file mode is omitted, the system makes an appropriate default assumption according to the command. See the individual command descriptions.

Notes:

1. Both device name and file name cannot be omitted simultaneously.
2. No file name can be assigned to devices other than \$FD1 through \$FD4 and \$CMT.

4.3.12 Arguments

There are several argument formats.

1. Device name + File name + File mode

Examples : \$ FD1 ; ABC . ASC \$ CMT ; XYZ . OBJ \$ FD2 ; * . *

2. Device name + File name. The file mode is omitted.

Examples : \$ FD1 ; ABC \$ FD2 ; A * \$ CMT ; TEST

3. File name + File mode. The device name is omitted (default drive).

Examples : TEST3 . RB *. ASC PROG? . RB

4. Device name

a. When the file name and mode are omitted or when the device name proper is to be specified.

Examples : \$ FD1 \$ CMT

b. When neither file name nor mode can be specified.

Examples : \$ PTR \$ CRT \$ LPT

5. Hexadecimal constant

Examples : \$ 1200 \$ C000

6. Special arguments

Examples : TIME 9 : 30 : 00
 ↑ ↑
 Argument Argument
 ↑ ↑
 Command Command

 LIMIT MAX
 ↑ ↑
 Argument Argument
 ↑ ↑
 Command Command

4.4 Using FDOS Commands

4.4.1 ASM

Transient

Format

ASM filename

Function

The ASM command assembles the source program in the source file specified by the argument, outputs the result to a relocatable file and outputs an assembly listing to the specified file or device.

Default file mode

.RB when local switch /O is specified; otherwise, .ASC.

Switches

Global switches:

- None: A relocatable file is generated.
- /N: No relocatable file is generated.

Local switches:

- None: Specifies that the specified source file is to be assembled.
- /O: Specifies that the relocatable code is to be output to a file under the selected name.
- /E: Specifies that only error statements are to be output to the selected file or device.
- /L: Specifies that the assembly listing is to be output to the selected file or device.

Wildcard characters

Not allowed

Examples

(1) ASM TEST

Assembles source file TEST.ASC and generates relocatable file TEST.RB.

(2) ASM TEST, \$ LPT/L, XYZ/O

Assembles source file TEST.ASC, generates relocatable file XYZ.RB and outputs the assembly listing to LPT.

(3) ASM/N TEST, \$CRT/E, \$ SOA/L

Assembles source file TEST.ASC while displaying error statements (including external symbol references) and outputting the assembly listing to SOA. No relocatable file is generated.

(4) ASM TEST, \$ FD2 ; TEST1/L, \$FD2 ; TEST1 .RB/O

Assembles source file TEST.ASC and saves relocatable file TEST1.RB and assembly listing TEST1.ASC on FD2.

(5) ASM TEST, \$ LPT/L, \$ 2000

Assembles source file TEST.ASC, generates relocatable file TEST.RB and outputs the assembly listing to LPT with a bias of 2000H added.

4.4.2 ASSIGN

Transient

Format

ASSIGN devicename1, \$nnnn,, devicenameN, \$nnnn

Function

The ASSIGN command assigns logical device names to user-supplied I/O control routines.

Switches

None.

Wildcard characters

Not allowed.

Examples

(1) **LIMIT \$F000**

ASSIGN \$USR1, \$F000

Assigns device name \$USR1 to the user I/O control routine at address \$F000.

(2) **ASSIGN \$USR2, \$F200, \$USR3, \$F400**

Assigns \$USR2 to the routine at address \$F200 and \$USR3 to the routine at address \$F400.

(3) **ASSIGN \$PTP, \$F600**

Assigns \$PTP to the new PTP routine at address \$F600 in place of the PTP control routine in FDOS.

Programming notes

(1) When a device name is assigned more than once, the last assignment is taken.

(2) To cancel an assignment, set the address operand to \$FFFF.

Example : ASSIGN \$USR1, \$FFFF This command cancels \$USR1.

(3) When an I/O control routine is destroyed by execution of a new LIMIT or LOAD command it is necessary to cancel the device assignment for that routine using the above procedure.

4.4.3 BASIC

Transient

Format

BASIC filename

Function

The BASIC command compiles the source program written in BASIC language identified by the argument and outputs the BASIC listing.

Default file mode

.RB when local switch /O is specified; .ABC otherwise.

Switches

Global switches

- /N: Specifies that no relocatable file is to be generated.
 - /C: Specifies that the BASIC listing is to be displayed on CRT.
 - /P: Specifies that the BASIC listing is to be printed on LPT.
- (Note that switches /C and /P cannot be specified simultaneously.)

Local switches

- None: Specifies that the specified source file is to be compiled.
- /O: Specifies that the relocatable file is to be output to the selected file.

Wildcard characters

Not allowed.

Examples

(1) BASIC TEST

Compiles source file TEST.ASC and generates relocatable file TEST.RB.

(2) BASIC/C TEST, XYZ/O

Compiles source file TEST.ASC, generates relocatable file XYZ.RB and displays the BASIC listing on CRT.

(3) BASIC/N/P TEST

Compiles source file TEST.ASC and prints the BASIC listing on LPT. No relocatable file is generated.

Programming notes

- (1) The compiler terminates generation of the relocatable file when it detects an error during compilation.
- (2) The BASIC compiler is available as an option.

4.4.4 BOOT

Built-in

Format

BOOT

Function

Terminates execution of FDOS and activates the MZ-80B system IPL (Initial Program Loader).

Programming notes

The system program is loaded into memory when IPL is activated. Therefore, former memory contents (such as FDOS, monitor and user programs) are cleared.

4.4.5 CHATR

Built-in

Format

CHATR sign, filename1, attribute1,, filenameN, attributeN

Function

The CHATR command changes the attributes of a specified file.

Default file mode

.ASC

Switches

None.

Wildcard characters

Only .* can be used to specify the file mode.

File attributes

0 : None.

R : Read-protected file

W : Write-protected file

P : Permanent file

Examples

(1) CHATR KEY, TEST, R

Assigns the password "KEY" to file TEST.ASC and declares the file as a read-protected file.

(2) CHATR SECRET, TEST.OBJ, 0

Deletes the file attributes of file TEST.OBJ. The specified password, "SECRET", matches with the password specified for the file before the command is actually executed.

(3) CHATR

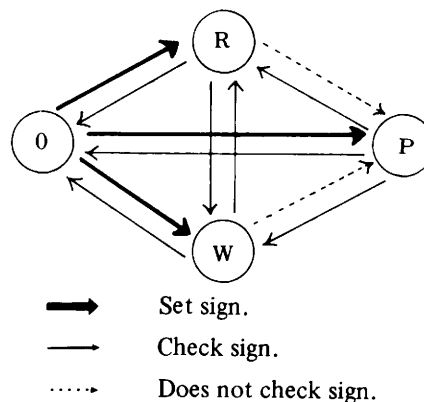
Allows the programmer to interactively specify the sign, file name and attribute in that order.

(4) CHATR sign

Allows the programmer to interactively specify the file name and attribute in that order.

Programming note

The interrelationship of the file attributes is shown below.



4.4.6 CONSOLE

Built-in

Format

CONSOLE Sscrolling-start-line, endline, Ccharacter-number, R, N

Function

Sets the scrolling area on the CRT screen, sets the number of characters per line to 40 or 80 and/or reverses the picture.

Default file mode

None.

Switches

None.

Wildcard characters

None.

Examples:

(1) **CONSOLE 2,10**

Sets the scrolling area to the area from the 2nd line through the 10th line.

(2) **CONSOLE R, C80**

Reverses the characters and graphic display on the screen and sets the number of characters per line to 80.

Programming notes

The arguments of the **CONSOLE** commands can be written in any order. The modes set are effective until they are set again.

4.4.7 CONVERT

Transient

Format

CONVERT

Function

Converts a file generated with the SB-5000 series BASIC interpreter or the D-BASIC SB-6000 series into a file usable under FDOS, or converts a file generated with FDOS into a file usable under the SB-5000 series or SB-6000 series. The relationship between file modes handled by this command is as follows.

BASIC		FDOS
BTX	↔	ASC
BSD	↔	ASC
OBJ	↔	OBJ

Default file mode

None.

Switches

None.

Wildcard characters

Not allowed.

Example

2 > **CONVERT**

Choose one from:

1 : BTX → ASC

2 : BTX ← ASC

3 : BSD ↔ ASC

4 : OBJ ↔ OBJ

(1 ~ 4) ?

Source drive No. (1 ~ 4, CMT = 0) ?2

Enter 1 ~ 4 for the \$FD and 0 for the \$CMT.

Source file name ? SAMPLE

Destination drive No. (1 ~ 4, CMT = 0) ?3

Destination file name ? SAMPLE

End of convert

Programming notes

- (1) Never intermix D-BASIC format diskettes and FDOS format diskettes. Otherwise, disk contents may be destroyed.
- (2) Since the syntax of D-BASIC and that of the BASIC compiler differ slightly, there are some cases in which programs converted with the CONVERT command cannot be compiled by the BASIC compiler without some modification. Use the text editor to modify such programs before compiling them with the BASIC compiler.
- (3) A BRD file cannot be converted. First convert it into a BSD file, then execute the CONVERT command.

4.4.8 COPY

Transient

Format

COPY

Function

The COPY command copies the contents of the source diskette to the destination diskette. The programmer can specify only predetermined types of diskettes as the destination and source diskettes as summarized in the table below.

Source	Destination	Allowed/disallowed	Remarks
(Any diskette)	Master	Disallowed	The destination diskette becomes a submaster diskette.
Master	Submaster	Allowed	
Master	Slave	Allowed	
Submaster	Submaster	Disallowed	
Submaster	Slave	Disallowed	The destination diskette becomes a slave diskette.
Slave	Submaster	Allowed	
Slave	Slave	Allowed	

It is desirable to create a submaster diskette from the master diskette using the COPY command and to use this submaster diskette during normal operation. It is also desirable to make copies at appropriate times when the original diskette is updated to prevent errors due to physical defects in the disk or software errors or inadvertent use of the DELETE command.

Default file mode

None.

Switches

None.

Wildcard characters

None.

Examples

- (1) FDOS always copies from \$FD1 to \$FD2 when the system has two or more floppy disk units.

2 > COPY

Destination diskette's sign ?BACKUP ← Proceeds to the next step if the passwords match.

Insert source into \$FD1 ← Insert the source diskette in drive FD1.

Destination into \$FD2, ↵ space key ← Insert the destination diskette in drive FD2, then press the **[SP]** key.

2 > Copying is completed.

4.4.9 DATE

Built-in

Format

DATE mm/dd/yy

Function

The DATE command sets or displays the system calendar date in the month/date/year format.

This information is assigned to each file when it is saved on a diskette. The date is not automatically updated, however.

Default file mode

None.

Switches

Global switch/P: Specifies that the date is to be printed on LPT.

Wildcard characters

Not allowed.

Examples

(1) DATE 11/20/81

Sets the system calendar date to November 20th, 1981

(2) DATE

Displays the current date on CRT.

(3) DATE/P

Prints the current date on LPT.

Format

DEBUG filename1,, filenameN

Function

The DEBUG command links and loads relocatable files specified by the arguments to form an object program in memory for debugging.

Default file mode

.OBJ when local switch /O is specified; .RB otherwise.

Switches

Global switches

- None: Specifies that only the link information is to be displayed on CRT.
- /T: Specifies that the symbol table information is to be output (on CRT unless global switch /P is specified).
- /P: Specifies that the link and symbol table information is to be printed on LPT when global switch /T is specified.

Local switch

- /O: Specifies that the object file is to be created under the selected file name.

Wildcard characters

Not allowed.

Examples**(1) DEBUG TEST1, TEST2**

Links and loads relocatable files TEST1.RB and TEST2.RB and waits for a debugger command. The link information is displayed on CRT.

(2) DEBUG /T/P TEST, TEST/O

Loads relocatable file TEST.RB, prints the link and symbol table information on LPT and generates object file TEST.OBJ.

(3) DEBUG TEST1, \$1000, TEST2, TBL \$20

Links and loads relocatable files TEST1.RB and TEST2.RB and reserves \$1000 bytes of free area in memory between them. The symbol table size is set to \$2000 (approximately 8K bytes).

When the table size is not specified, the debugger automatically allocates 6K bytes for it.

(4) DEBUG

Invokes the symbolic debugger and enters the command mode.

4.4.11 DELETE

Built-in

Format

DELETE filename1,, filenameN

Function

The DELETE command deletes the files specified by the arguments except those with the W or P file attribute.

Default file mode

.ASC

Switches

Global switches /C : When this switch is specified, the system displays each file on CRT for confirmation. The file is deleted when the programmer presses the **[Y]** key and skipped when he presses the **[N]** key.

/N : Specifies that no deleted file is to be displayed. (The programmer must not specify /N and /C simultaneously.)

Wildcard characters

Allowed.

Examples

(1) DELETE TEST . *

Deletes all files whose file name is TEST.

(2) DELETE /C * .OBJ

Displays all files with a file mode of .OBJ on CRT for confirmation before deleting them.

(3) DELETE \$FD2 ; * . *

Deletes all files on FD2 except those with the file attribute P or W. To delete file-protected file, it is necessary to cancel the file protect attributes with the CHATR command.

(4) DELETE \$ MEM

Deletes file \$ MEM.

4.4.12 DIR

Built-in

Format

DIR devicename (filename)

Function

Displays the contents of the directory specified by devicename of filename. "devicename" must refer to a floppy disk unit.

Default file mode

*

Switches

Global switch /P : Specifies that the directory is to be printed on LPT.

Wildcard characters

Allowed.

Examples

(1) DIR \$FD2

Displays the file information of all files on the diskette in FD2 on CRT. FD2 is designated as the default drive.

(2) DIR/P

Prints the file information of all files on the deskette in the current default drive on LPT. The directory device remains unchanged.

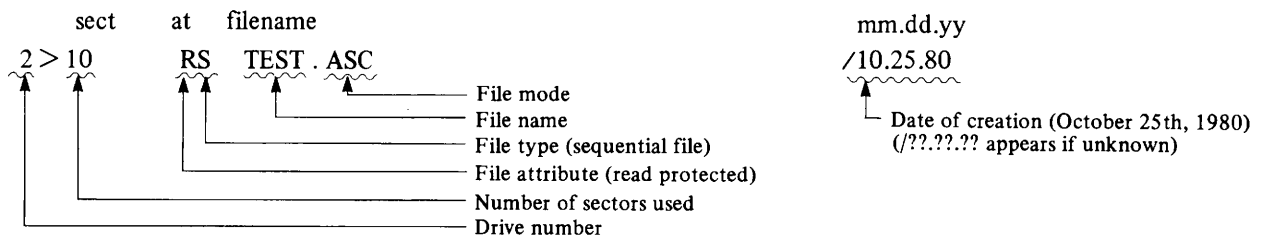
(3) DIR TEST

Displays on CRT the file information of all files on the diskette in the current default drive whose file name is TEST.

(4) DIR \$FD2 ; * . ASC

Displays the file information of all source files on the diskette in FD2 on CRT. FD2 is designated as the default drive.

Programming notes



4.4.13 EDIT

Transient

Format

EDIT filename

Function

The EDIT command invokes the text editor to create a new source file or edit an existing source file.

Default file mode

.ASC

Switches

None.

Wildcard characters

Not allowed.

Examples

(1) EDIT

Invokes the text editor and enters the command mode.

(2) EDIT TEST

Invokes the text editor, reads source file TEST. ASC and enters the command mode.

(3) EDIT \$FD2 ; TEST

Invokes the text editor, reads source file TEST. ASC from the \$FD2 and enters the command mode.

Format

EXEC filename

Function

The EXEC command executes the contents of the file specified by the argument as FDOS commands. A device name may be specified in place of filename. Files containing FDOS command are called EXEC files.

Default file mode

.ASC

Switches

None.

Wildcard characters

Not allowed.

Examples**(1) EXEC MACRO**

Executes the contents of source file MACRO.ASC assuming that the file consists of FDOS commands. When the file MACRO.ASC contains the command lines shown below, the system executes the commands in sequence from the top to the bottom.

ASM \$FD2 ; TEST

LINK /T/P \$FD2 ; TEST

CHATR KEY, \$FD2 ; TEST.OBJ, W

RUN \$FD2 ; TEST

3 > FREE ← Display the number of used sectors on the diskette in FD3.

DIR/P \$FD2 ← Print the contents of the FD2 directory on LPT and designate FD2 as the current default drive.

(2) EXEC MYDEVICE

Sequentially executes the command lines contained in source file MYDEVICE.

LIMIT \$F000 ← Limit the FDOS area to \$F000.

LOAD MYPRINTER ← Set the loading and execution addresses to \$F000.

LOAD MYLIGHTPEN ← Set the loading and execution addresses to \$F800.

ASSIGN \$USR1, \$F000, \$USR2, \$F800 ← Assign user I/O names to user programs.

ASM TEST, \$USR1 /L ← Assemble source file TEST and output the assembly listing on the user-defined printer.

XFER \$USR2, XYZ ← Transfer data obtained with a light pen to file XYZ.

(3) EXEC ABC

Executes the routine in file DEF repeatedly if file ABC.ASC contains the following routine.

RUN DEF

EXEC ABC

Programming notes

- (1) Since the EXEC command executes the commands specified in a file as macro commands, it cannot be specified on a multistatement line as shown below.

EXEC MACRO : TYPE MACRO

- (2) The specified file may have the file attribute R, W or P. However, execution of files with the attribute R or P is not displayed.
- (3) When an error occurs during execution of an EXEC file, the system immediately terminates processing and waits for entry of a new FDOS command from the keyboard.
- (4) When the file name START-UP is assigned to an EXEC file, that file will be automatically executed when FDOS is activated.

4.4.15 FAST

Built-in

Format

FAST

Function

Fast-forwards the cassette tape. Control is given to the next command as soon as the fast-forward operation has been started.

4.4.16 FORMAT

Transient

Format

FORMAT \$FDn

Function

The FORMAT command formats (initializes) a new diskette.

The user must always format new diskettes before using them.

Default file mode

None.

Switches

None.

Wildcard characters

Not allowed.

Examples

(1) FORMAT \$FD2

FDOS diskette formatting

Insert diskette into \$FD2, ↵ space key

New sign ? **SHARP**

Volume No. ? **50**

END

Insert diskette into \$FD2, ↵ space key

Break ← Press the **BREAK** Key to return to FDOS.

The above interaction shows an example of formatting a completely new diskette.

"sign" prompts for a password to be given at the diskette. When this diskette is resubmitted for formatting, the system checks for a password match before actually reformatting the diskette. "Volume No." prompts for a volume number to be assigned to the diskette. The programmer can specify any number from 1 to 255. The volume number should be unique.

(2) FORMAT

FDOS diskette formatting

Insert diskette into \$FD1, ↵ space key

Old sign ? **SHARP** ← The system matches the password entered with that stored on the diskette and proceeds to the next step if they match.

New sign ? **MZ-80** ← Set a new password.

Volume No. ? **128**

END

Insert diskette into \$FD1, ↵ space key

Break ← Press the **BREAK** key to return to FDOS.

The above interaction shows an example of reformatting a previously formatted diskette. The meanings of "sign" and "Volume No." are identical to those in example (1).

Programming notes

The following message will be displayed if a diskette cannot be initialized because of defects, etc.

(1) bad track #nn

When this message is displayed, the XFER command can be executed for the diskette but the COPY command cannot.

(2) no usable diskette

When this message is displayed, neither the XFER nor COPY command can be executed for the diskette.

4.4.17 FREE

Built-in

Format

FREE \$FDn

Function

The FREE command displays the number of used sectors, the number of unused sectors, and/or the volume number of the diskette in the specified floppy disk unit.

Default file mode

None.

Switches

Global/P : Specifies that the disk usage information is to be printed on LPT.

Wildcard characters

Not allowed.

Examples

(1) FREE \$FD2

\$ FD2 vol : 128 left : 1072 used : 48

(2) FREE/P

Prints the same information as given in example (1) on LPT, except that the information pertains to the diskette in the default drive.

Programming note

A diskette is comprised of 1120 sectors (each consisting of 256 bytes). Of these 1120 sectors, however, 48 sectors are reserved by the system as FDOS areas. Consequently, used: 48 is indicated for new diskettes.

4.4.18 HCOPY

Transient

Format

HCOPY display page

Function

Copies one frame from the CRT screen on the LPT.

Default file mode

None.

Switches

None.

Wildcard characters

Not allowed.

Examples:

HCOPY 1 Copies character data from the CRT screen on the LPT.

HCOPY 2 Copies the display data in graphic area 1 on the LPT.

HCOPY 3 Copies the display data in graphic area 2 on the LPT.

HCOPY 4 Copies the display data onto the LPT when the contents of graphic areas 1 and 2 are displayed simultaneously.

4.4.19 KEY

Built-in

Format

KEY keynumber = "S"

Function

Assigns a function to the definable function key indicated by a key number from 1 through 20. The function is specified by writing a string or command name enclosed in double quotation marks.

Default file mode

None.

Switches

None.

Wildcard character

None.

Examples:

```
KEY 1  = "XFER"  
KEY 7  = "DELETE"  
KEY 13 = "$KB"  
KLIST  
KEY 1  = "XFER"  
:  
KEY 7  = "DELETE"  
:  
KEY 13 = "$KB"  
:  
KEY 20 = "  "
```

Programming notes

Definable function keys 11 through 20 are activated by pressing the **SHIFT** key and one of keys **F1** through **F10** simultaneously.

4.4.20 KLIST

Built-in

Format

KLIST

Function

Lists the definition status of the definable function keys.

Default file mode

None.

Switches

None.

Wildcard character

None.

Examples

```
KLIST  
KEY 1 = "RUN"  
KEY 2 = "XFER"  
KEY 3 = "DELETE"  
KEY 4 = "RENAME"  
KEY 5 = "DIR"  
KEY 6 = "EDIT"  
KEY 7 = "ASM"  
KEY 8 = "LINK"  
KEY 9 = "DEBUG"  
KEY 10 = "BASIC"  
KEY 11 = "$FD1;"  
KEY 12 = " "  
KEY 13 = " "  
KEY 14 = " "  
KEY 15 = " "  
KEY 16 = " "  
KEY 17 = " "  
KEY 18 = " "  
KEY 19 = " "  
KEY 20 = " "
```

4.4.21 LIBRARY

Transient

Format

LIBRARY filename1,, filenameN

Function

The **LIBRARY** command reads the relocatable files specified by the arguments to form a library file.

Default file mode

.LIB when local switch /O is specified; .RB otherwise.

Switches

Global switches

None: Link information pertaining to the relocatable files is displayed on CRT.

/P: Specifies that the link information is to be printed on LPT.

Local switches

None: The first filename specified is used as the name of the library file.

/O: Specifies that the library file is to be created with the selected file name.

Wildcard characters

Not allowed.

Examples

(1) **LIBRARY TEST1, TEST2**

Reads relocatable files TEST1.RB and TEST2.RB to generate library file TEST1.LIB. The link information is displayed on CRT.

(2) **LIBRARY /P TEST1.LIB, TEST2, XYZ /O**

Reads relocatable files TEST1.LIB and TEST2.RB and generates a library file named XYZ.LIB. The link information is printed on LPT.

4.4.22 LIMIT

Transient

Format

LIMIT \$nnnn

Function

The **LIMIT** command sets the FDOS area boundary at address \$nnnn.

Default file mode

None.

Switches

None.

Wildcard characters

None.

Examples

(1) **LIMIT \$F000**

Limits the FDOS area to \$F000 and frees the higher area.

(2) **LIMIT MAX**

Sets the FDOS area to the maximum available address.

Programming note

The **LIMIT** command cannot be specified in a multistatement as shown below.

Illegal: **LIMIT \$E000 : DIR \$FD2**

4.4.23 LINK

Transient

Format

LINK filename1,, filenameN

Function

The **LINK** command links the relocatable files specified by the arguments to generate an object or system file.

Default file mode

.OBJ when local switch /O is specified; .RB otherwise.

Switches

Global switches

None: Only the link information is displayed on CRT.

/T: Specifies that the symbol table is to be output (on CRT unless global switch /P is specified).

/P: Specifies that the link and symbol table information is to be output to LPT (when global switch /T is specified).

/S: Specifies that a system file is to be generated.

Local switches

None: The first filename specified is used as the name of the object file.

/O: Specifies that the object file is to be created under the specified file name. If global switch /S is specified, specifies that the system file is to be created under the specified file name.

Wildcard characters

Not allowed.

Examples

(1) **LINK TEST1, TEST2**

Links relocatable files TEST1.RB and TEST2.RB and generates an object file named TEST1.OBJ. The loading and execution addresses of the object file are automatically set to the beginning address managed by FDOS. The link information is displayed on CRT.

(2) **LINK/T/P TEST1, TEST2, XYZ/O**

Links relocatable files TEST1.RB and TEST2.RB and generates object file XYZ.OBJ. The loading and execution addresses of the object file are set to the beginning address managed by FDOS. The link and symbol table information is output to LPT.

(3) LINK \$C000, TEST, FDOSEQU.LIB, EXEC\$C100

Links TEST.RB and FDOSEQU.LIB and generates object file TEST.OBJ, specifying \$C000 as the loading address. The execution address of the object file is \$C100.

(4) LINK TEST1, \$1000, TEST2, TBL \$20

Links file TEST1.RB (specifying the beginning of the FDOS area as the loading address), then links and loads file TEST2.RB, reserving \$1000 bytes of free area between the two files. The symbol table size is set to 8K (\$2000) bytes.

4.4.24 LOAD

Transient

Format

LOAD filename1,, filenameN

Function

The LOAD command loads the object files specified by the arguments in areas outside the area managed by FDOS.

Default file mode

.OBJ

Switches

None.

Wildcard characters

None.

Example

(1) LOAD TEST1, TEST2

Loads object files TEST1.OBJ and TEST2.OBJ into memory areas outside the area managed by FDOS. The programmer must create object files so that they are to be loaded in appropriate addresses.

4.4.25 MLINK

Transient

Format

MLINK filename1,, filenameN

Function

The MLINK command links the relocatable files specified by the arguments to generate an object file.

Default file mode

.OBJ when local switch /O is specified; .RB otherwise.

Switches

Global switches

None: Only the link information is displayed on the CRT.

/T: Specifies that the symbol table is to be output (on the CRT unless global switch /P is specified).

/P: Specifies that the link and symbol table information is to be output to the LPT (when global switch /T is specified).

Local switches

None: The first file name specified is used as the name of the object file.

/O: Specifies that the object file is to be created under the selected file name.

Wildcard characters

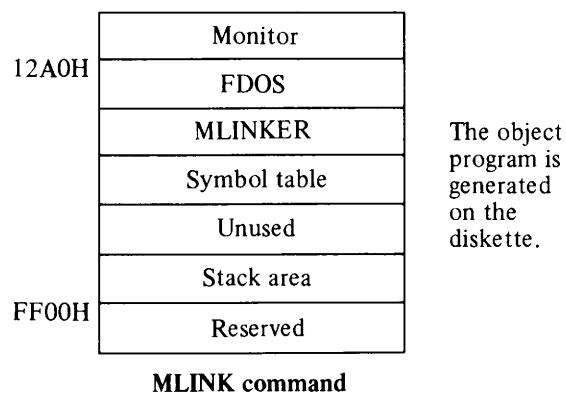
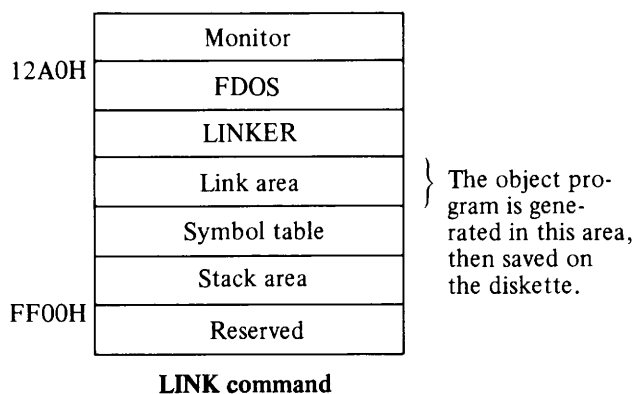
Not allowed.

Examples

2 > MLINK STARTREC

Programming notes

- (1) The MLINK command can be used in the same manner as the LINK command except that it cannot specify the table size (TBL\$hh).
- (2) The LINK command can generate an object file of up to approx. 36K bytes. The MLINK command is used when the file exceeds this size to generate object files of up to approx. 46K bytes. However, the MLINK command takes twice as long as the LINK command to generate an object file because the MLINK command links relocatable programs using a 2-pass system. The following diagrams show memory maps applicable to execution of the LINK and MLINK commands.



4.4.26 MON

Built-in

Format

MON

Function

The MON command returns control to the monitor.

Programming notes

Control is transferred to FDOS from the monitor with the following monitor command.

* J

J-adr.\$12A0

4.4.27 PAGE

Transient

Format

PAGE output-device or PAGE n

Function

The PAGE command carries out a paging operation on the output device specified by output-device, or sets the number of lines per page on LPT.

Default file mode

None.

Switches

None.

Wildcard characters

None.

Examples

(1) PAGE or PAGE \$ LPT

Carries out a form feed on LPT.

(2) PAGE \$ PTP, \$ SOA, \$ USR1

Produces a feeder tape on PTP and outputs the code defined with the STATUS command to SOA and USR1.

(3) PAGE 22

Sets the number of lines per page on the LPT to 22. The print form is fed to the top of the next page when a page feed code is issued or the TOP OF FORM button is pressed.

4.4.28 POKE

Built-in

Format

POKE \$nnnn, data1,, \$uuuu, dataN

Function

Stores data1 consisting of an even number of digits in and from address \$nnnn (4-digit hexadecimal number) on, and stores dataN consisting of an even number of digits in and from address \$uuuu on. Any address is accessible. The maximum length from POKE to dataN is 160 characters including 0DH, space, etc.

Default file mode

None.

Switches

None.

Wildcard characters

None.

Examples

POKE \$000D, 2010, \$000F, 40

Stores 20 in address \$000D, 10 in \$000E and 40 in \$000F.

POKE \$000D, 1235678, 12, \$000F, 40

Not allowed

4.4.29 PROM

Transient

Format

PROM

Function

The PROM command converts the format of the object file to an appropriate PROM writer format.

Default file mode

None.

Switches

None.

Wildcard characters

None.

Example

(1) PROM

Invokes the PROM formatter program and enters the command mode. Refer to the "PROM Formatter" manual for further information.

4.4.30 RENAME

Built-in

Format

RENAME oldname1, newname1,, oldnameN, newnameN

Function

The RENAME command renames specified files.

Default file mode

.ASC

Switches

None.

Wildcard characters

An asterisk may be used to specify the file mode (. *).

Examples

(1) RENAME TEST1, TEST2

renames TEST1.ASC to TEST2.ASC.

(2) RENAME \$FD2 ; TEST1 . OBJ, TEST2, TEST3 . RB, TEST4

Renames TEST1.OBJ on the diskette in FD2 to TEST2.OBJ and TEST3.RB on the diskette in the default drive to TEST4.RB.

Programming notes

- (1) Files with the file attribute W or P cannot be renamed.
- (2) The command `RENAME $FD2;TEST1, $FD2;TEST2` cannot be executed since \$FDn specified for the old name applies to the new name, which is illegal.
- (3) The command `RENAME TEST1.LIB, TEST2.RB` cannot be executed since the file modes of the old and new names disagree.
- (4) The command `RENAME TEST.LIB, TEST2` can be executed normally. The new name is assigned the file mode of the old name.

4.4.31 REW

Built-in

Format

REW

Function

Rewinds the cassette tape. Control is transferred to the next command as soon as the rewind operation has been started.

The next command is executed when the cassette tape is fully wound.

4.4.32 RUN

Built-in

Format

RUN filename or file name

Function

The RUN command executes the program in the object file specified by the argument.

Default file mode

.OBJ, .SYS

Switches

None.

Wildcard characters

None.

Example

(1) RUN TEST

Executes the program TEST.OBJ. When its loading address is such that it overwrites the FDOS area, the system issues the message

destroy FDOS ?

on the CRT. When the programmer press the **Y** key, the system loads the program, overwriting the FDOS area and executing it. When the programmer presses the **N** key, the system issues the error message "memory protection" and waits for a new FDOS command.

(2) 1 > TEST

Accesses the drive 1 to seek .SYS mode file and executes it if found. If not found, error occurs.

(3) 2] TEST

Accesses drive 2 to seek program TEST.SYS and executes it if found. If not found, it seeks TEST.OBJ and executes it if found. If not found, error occurs.

Programming notes

The meanings of the prompt symbols (> and]) are shown below.

Command	filename	RUN filename	RUN \$FDn filename	RUN \$nnnn
File mode	. SYS . OBJ	. OBJ	. OBJ	
Prompt >	Accesses the drive 1 to seek . SYS mode file and executes it if found. If not found, error occurs.	Accesses the default drive to seek . OBJ mode file and executes it if found. If not found, error occurs.	Accesses \$FDn to seek . OBJ mode file and executes it if found. If not found, error occurs.	Calls address \$nnnn.
Prompt)	Accesses the default drive to seek . SYS mode file and executes it if found. If not found, it seeks . OBJ mode file and executes it if found. If not found, error occurs.	Same as above.	Same as above.	Same as above.

4.4.33 SIGN

Transient

Format

SIGN \$FDn

Function

The SIGN command defines or changes the password and/or volume number of the diskette in the specified drive.

Default file mode

None.

Switches

None.

Wildcard characters

None.

Example

(1) SIGN

Old sign ? **SHARP** ← Proceeds to the next step if the password entered matches the old password.

New sign ? **MZ-80**

New volume No ? **79**

The above interaction changes the password from "SHARP" to "MZ-80" and defines the volume number as 79.

4.4.34 STATUS

Transient

Format

STATUS devicename, Snnnn

Function

The STATUS command displays or sets the control status of the specified device. The control status information is used to initialize the I/O controllers. Refer to "User I/O Routine" in Appendix for details.

Default file mode

None.

Switches

None.

Wildcard characters

None.

Examples

(1) STATUS \$SOA, \$ 1234

Sets the SOA control status to 1234 (hexadecimal).

(2) STATUS \$USR1

Displays the control status of USR1 on CRT.

(3) STATUS \$LPT, \$0000

00 normal mode
12 double-size mode
14 reduced mode

Programming note

This command is available for the serial I/O devices (\$SIA, \$SIB, \$SOA and \$SOB), \$LPT and user devices (\$USR1 to \$USR4). Any STATUS command set for \$CMT1, \$PTR, \$KB, \$CRT, \$FD1 to \$FD4, \$CMT, \$MEM or \$PTP will be invalid.

4.4.35 TIME

Built-in

Format

TIME mm ; dd ; ss

Function

The TIME command sets or displays the time of the system clock.

Default file mode

None.

Switches

Global switch / P: Specifies that the time is to be printed on LPT.

Wildcard characters

None.

Examples

(1) **TIME 20 : 30 : 40**

Sets the system clock to 20 hours, 30 minutes and 40 seconds.

(2) **TIME**

Displays the current time on CRT.

(3) **TIME/P**

Prints the current time on LPT

4.4.36 TYPE

Built-in

Format

TYPE filename1,, filenameN

Function

The TYPE command outputs contents of the files specified by the arguments on the CRT or LPT device.

Default file mode

.ASC

Switches

Global switch /P: Specifies that the file contents are to be printed on the LPT device.

Wildcard characters

Allowed.

Examples

(1) **TYPE TEST**

Displays the contents of source file TEST .ASC on CRT.

(2) **TYPE/P TEST1, TEST2**

Prints the contents of source files TEST1 .ASC and TEST2 .ASC on LPT.

4.4.37 VERIFY

Transient

Format

VERIFY sourcefile 1, destinationfile 1,, sourcefileN, destinationfileN

Function

The VERIFY command compares the contents of the source and destination files specified by the arguments and displays any mismatching contents on a line basis (if their file mode is .ASC) or on a byte basis (if the file mode is other than .ASC).

Default file mode

.ASC

Switches

Global switch /P: Specifies that the matching results are to be printed on LPT.

Wildcard characters

Allowed for source files (see example (4) below).

Examples

(1) **VERIFY TEST1, TEST2**

Matches source files TEST1.ASC and TEST2.ASC and displays mismatching lines on CRT.

(2) **VERIFY /P \$CMT ; XYZ, \$FD2 ; TEST**

Matches source file XYZ.ASC on CMT with source file TEST.ASC on the diskette in FD2 and prints the results on LPT.

(3) **VERIFY \$CMT, \$FD2**

Matches the first file on CMT with the file on the diskette in FD2 which has the same name as the file on CMT. An error is generated if file on CMT has no file name.

(4) **VERIFY \$CMT ; TEST*, \$FD2**

Matches the first file on CMT whose name matches TEST* with the file that name on the diskette in FD2. Note that only the first file whose file name matches TEST* is taken.

4.4.38 XFER

Built-in

Format

XFER sourcefile1, destinationfile1,, sourcefileN, destinationfileN

Function

The XFER command transfers the contents of source files to destination files.

Default file mode

.ASC

Switches

None.

Wildcard characters

Allowed for the source files (see example (5) below).

Examples

(1) **XFER TEST1, TEST2**

Transfers the contents of source file TEST1.ASC to TEST2.ASC.

(2) **XFER \$PTR, \$LPT**

Reads the file on PTR and prints it on LPT.

(3) **XFER \$CMT ; XYZ.OBJ, \$FD2**

Reads object file XYZ.OBJ from CMT and creates object file XYZ.OBJ on \$FD2.

(4) **XFER \$CMT, \$FD2**

Reads in the first file on CMT and creates a file with that file name on the diskette in FD2. An error is generated if file on CMT has no file name.

(5) **XFER \$CMT ; TEXT*, \$FD2**

Reads in the first file on CMT whose file name matches file name TEXT* and creates a file with the same name on the diskette in FD2. Note that only the first source file on CMT whose file name matches TEXT* is taken.

(6) XFER \$KB, TEST

Reads a file from the system keyboard and creates source file TEST.ASC. The file read from the keyboard is terminated by pressing the BREAK key.

(7) XFER \$FD2 ; * . ASC, \$FD3

Transfers all source files on the diskette in FD2 to that in FD3. The source drive must not contain files with the file attribute R or P.

(8) XFER * . * , FD2

Transfers all files on the diskette in the current default drive to that in FD2. The source drive must not contain files which have the file attribute R or P.

Table 4-1 Built-in commands cont.

DIR [\$FDn] or [filename]	(? ,*)
Displays file information in the directory specified by \$FDn or of the file specified by filename on the screen. Global switch /P : Specifies that the file information is to be output to LPT. The file information is displayed on the screen when this switch is not specified. Examples: DIR ↵ : Displays all file information in the current directory on the screen. DIR/P \$FD2 ↵ : Outputs all FD2 file names to LPT and switches the currently logged disk to FD2. DIR \$FD2 ; ABC. * ↵ : Displays the file information of files in FD2 identified by ABC. * .	
EXEC filename	
Executes the contents of the file identified by filename as FDOS commands. Example: EXEC ABC . ASC ↵ : Sequentially executes the FDOS commands in file ABC.	
FAST	
Fast forwards the cassette tape. Example: FAST ↵	
FREE [\$FDn]	
Lists statistical information about the disk identified by \$FDn on the screen or on the LPT. Example: FREE \$FD2 ↵ \$FD2 master left : XXXX used : YYYY Indicates that the diskette on FD2 is a master diskette, that the number of unused sectors is XXXX and that the number of used sectors is YYYY.	
KEY keynumber = "S"	
Assigns a function to the definable function key indicated by a keynumber from 1 through 20. The function is specified by writing a string or command name enclosed in double quotation marks. Example: KEY 1 = "RUN↵" ↵ : Assigns the function of the RUN command to key 1.	
KLIST	
Lists the definition status of the definable function keys on the screen. Example: KLIST ↵	
MON	
Terminates FDOS processing and returns control to the monitor. Example: MON ↵	
POKE \$nnnn, data [, ..., \$uuuu, dataN]	
Stores data in the specified addresses in memory. Example: POKE \$000D, 2010, \$000F, 40 ↵	
RENAME oldname1, newname1 [, ..., oldnameN, newnameN]	
Renames the file specified by oldname to newname. Examples: RENAME ABC, XYZ ↵ : Renames file ABC to XYZ. RENAME ABC, DEF, UVW, XYZ ↵ : Renames file ABC to DEF and UVW to XYZ.	

Table 4-1 Built-in commands cont.

REW	
Rewinds the cassette tape. Example: REW ↵	
RUN filename	
Executes the program in the object file identified by filename. Example: RUN ABC ↵ : Executes the program in file ABC, assuming it ot be ABC.OBJ.	
TIME [HH : MM : SS]	
Displays the current time or sets specified time in hour, minute, second format. This information is used as file information when new files are created. The current time is set to 00 : 00 : 00 upon system start. Global switch /P : Specifies that the current time is to be listed on the LPT. Examples: TIME /P ↵ : Lists the current time on the LPT. TIME 16 : 30 : 30 ↵ : Sets the current time to 16 : 30 : 30	
TYPE filename1 [, ..., filenameN]	(? , *)
Lists the contents of the file(s) identified by filename(s) on the screen or on LPT. Global switch /P : Lists the file contents on LPT. Examples: TYPE ABC, DEF ↵ : Displays the contents of files ABC and DEF on the screen. TYPE /P \$FD3 ; XYZ ↵ : Lists the contents of file XYZ in FD3 on LPT. TYPE \$PTR ↵ : Reads paper tape data from PTR and displays it on the screen.	
XFER sourcefile1, destinationfile2 [, ..., sourcefileN, destinationfileN]	(sourcefile only ? , *)
Transfers the source file(s) to the destination file(s). Examples: XFER ABC, XYZ ↵ : Copies file ABC to XYZ. XFER \$PTR, DEF ↵ : Transfers the file at the PTR to file DEF. XFER XYZ, \$PTP / PE ↵ : Transfers file XYZ to the PTP with even partiy in ASCII code.	

Table 4-2 Transient commands

ASM filename	
Assembles the source file identified by filename and produces a relocatable file and an assembly listing.	
Global switch (none)	: Specifies that the relocatable file is to be output.
Global switch/N	: Suppresses generation of the relocatable file.
Local switch/O	: Specifies that the relocatable file is to be output with the specified file name.
Local switch/E	: Specifies that error statements are to be output to the specified file.
Local switch/L	: Specifies that the listing is to be directed to the specified file.
Examples: ASM ABC ↵	: Assembles source file ABC and generates relocatable file ABC.RB.
ASM/N ABC, \$CRT/E ↵	: Assembles source file ABC and displays error statements on the screen (no relocatable file is created).
ASM ABC, XYZ/O, \$LPT/L ↵	: Assembles source file ABC and generates relocatable file XYZ.RB and an assembly listing on the LPT.
ASM ABC, \$FD2; XYZ/L, \$LPT/E ↵	: Assembles source file ABC outputs the assembly listing to file XYZ.ASC in FD2 and outputs error statements on the LPT.
ASSIGN devicename, address	
Sets the address of a user device drive routine.	
Examples: ASSIGN \$USR1, \$B000 ↵	: Sets the drive routine address of user device \$USR1 to B000 (hexadecimal).
BASIC filename	
Invokes the BASIC compiler to compile the source program identified by filename.	
Example: BASIC XYZ ↵	: Invokes the BASIC compiler, compiles source file XYZ.ASC and generates relocatable file XYZ.RB.
CONVERT	
Converts a file generated with the SB-5000 series BASIC interpreter or the D-BASIC SB-6000 series into a file which can be used under FDOS, or converts a file generated with FDOS into a file which can be used under the SB-5000 series BASIC interpreter or the D-BASIC SB-6000 series.	
Example: CONVERT ↵	
COPY	
Copies the files on the diskette in drive 1 to the diskette in drive 2. The system matches the passwords in these diskettes before carrying out a copy operation.	
Example: COPY ↵	
DEBUG filename [, ..., filenameN]	
Invokes the symbolic debugger and links and loads relocatable file(s).	
Global switch /T	: Specifies that the symbol table information is to be output.
Global switch /P	: Specifies that the listing is to be directed to the LPT (the listing is displayed on the screen if omitted).
Local switch /O	: Specifies that the object file is to be generated with the specified file name.
Example: DEBUG ABC, DEF ↵	: Invokes the symbolic debugger, links and loads relocatable files ABC and DEF and waits for a symbolic debugger command.
EDIT [filename]	
Loads the text editor and reads in the file (if specified). The file must be an ASC mode file.	
Examples: EDIT ↵	: Loads the text editor and waits for an editor command.
EDIT \$FD2; ABC ↵	: Loads the text editor and reads in file ABC from FD2.

Table 4-2 Transient commands cont.

FORMAT [\$FDn]	
Initializes the diskette in \$FDn in the system format. The password set by the SIGN command is checked before execution.	
Examples: FORMAT ↵	: Initializes the currently logged-on diskette.
FORMAT \$FD2 ↵	: Initializes the diskette in FD2.
HCOPY n	
Copies a data frame from the CRT screen to the LPT.	
Examples: HCOPY 4 ↵	: Copies a data frame from the CRT where the contents of graphic areas 1 and 2 are displayed simultaneously.
LIBRARY filename1 [, ..., filenameN]	
Links specified file(s) into a library file.	
Global switch (none)	: Specifies that the link information is to be displayed on the screen.
Global switch /P	: Specifies that the link information is to be printed on the LPT.
Examples: LIBRARY ABC, DEF, ↵	: Links relocatable files ABC and DEF and stores their contents into library file ABC.LIB
LIBRARY ABC, DEF, XYZ /O ↵	: Links relocatable files ABC and DEF and stores their contents into library file XYZ.LIB.
LIMIT address	
Sets or changes the end address of the memory area managed by FDOS.	
Examples: LIMIT \$B000 ↵	: Sets the FDOS area to B000 (hexadecimal).
LIMIT MAX ↵	: Sets the FDOS area to the maximum available address.
LINK filename1 [, ..., filenameN]	
Links relocatable files identified by filename1 through filenameN and outputs an object file with a link table listing.	
Global switch /T	: Specifies that the symbol table information is to be listed.
Global switch /P	: Specifies that the listing is to be directed to the LPT (the listing is displayed on the screen if the switch is omitted).
Global switch /S	: Specifies that a system file is to be generated.
Examples: LINK ABC, DEF ↵	: Links relocatable files ABC and DEF and outputs object file ABC.OBJ
LINK /T/P ABC, DEF, XYZ /O ↵	: Links relocatable files ABC and DEF and outputs object file XYZ.OBJ with the link table information on the LPT.
LOAD filename	
Loads the object file identified by filename into the area immediately following the area established by the LIMIT command.	
Example: LOAD ABC.OBJ ↵	: Loads object file ABC.OBJ into memory.
MLINK filename1 [, ..., filenameN]	
Links relocatable files identified by filename1 through filenameN and outputs an object file with a link table listing. This command can link files to generate an object file of up to 46K bytes, although the LINK command can only deal with up to 36K bytes.	
Global switch /T	: Specifies that the symbol table information is to be listed.
Global switch /P	: Specifies that the listing is to be output on the LPT (the listing is displayed on the screen if this switch is omitted).
Example: MLINK ABC, DEF ↵	: Links relocatable files ABC and DEF and outputs object file ABC.OBJ.

Table 4-2 Transient commands cont.

PAGE [output-device] or nn	
Performs a form feed operation on the output device identified by output-device, or sets the number of lines per page on the LPT. Examples: PAGE ↵ : Moves the print position to the home position of the printer form. PAGE 22 ↵ : Sets the number of lines per page on the LPT to 22. The print form is fed to the top of the next page when a page feed code is issued or the TOP OF FORM button is pressed.	
PROM	
Generates formatted code on the paper tape punch from an object file. Applicable PROM writers are those which are supplied by Britronics, Intel, Takeda and Minato Electronics. Example: PROM ↵	
SIGN [\$FDn]	
Changes the password of the diskette in \$FDn. During a diskette copy or formatting operation, the system checks the programmer-specified password with that stored in the diskette directory for a match and carries out the specified operation only when a match occurs. Example: SIGN ↵ : Changes the password of the diskette currently logged on.	
STATUS devicename, status	
Sets the status of the I/O device identified by devicename to status. Example: STATUS \$SIA, \$1234 ↵ : Sets the control status of serial input port A to 1234 (hexadecimal).	
VERIFY filename1, filename2 [, ..., filenameN-1, filenameN] (? , * only for filename1, ..., filenameN-1)	
Compares the contents of files filename1 through filenameN. Global switch /P : Specifies that the results of the comparison are to be listed on the LPT. Example: VERIFY \$CMT, \$FD2 ; ABC ↵ : Compares the first file on the cassette tape with source file ABC in FD2.	

4.6 System Error Messages

There are four system error message formats.

— ERR: error message

Pertains mainly to coding errors. Issued when invalid commands are detected.

— ERR filename (device name) : error message

Indicates errors pertaining to file or device specifications.

— ERR logical number: error message

Indicates errors pertaining to logical number specifications.

— ERR logical number file name (device name): error message

Indicates errors pertaining to logical number specifications and file (or device) specifications.

The system error messages are listed below. The error numbers are not output.

ERR- 1	syntax	
2	il command	
3	il argument	
4	il global switch	
5	il data	
6	il attribute	; Illegal file attribute found
7	different file mode	
8	il local switch	
9	il device switch	
10		
11	no usable device	; Device unavailable
12	double device	
13	directory in use	
14		
15		
16	not enough arguments	
17	too many argument	
18		
19		
20	no memory space	
21	memory protection	
22	END ?	
37	Break	
38	system id	; Diskette not conforming to FDOS format.
39	System error	; System malfunction, user program error, diskette replaced improperly, etc.

50	not found	
51	too long file	; File size exceeds 65535 bytes
52	already exist	
53	already opened	; The file has been already opened or
54	not opened	the logical number is already used.
55	read protected	
56	write protected	
57	permanent	
58	end of file	
59	no byte file	
60	not ready	
61	too many files	; Number of files exceeds 96
62	disk volume	; Diskette replaced improperly
63	no file space	
64	unformat	; Diskette unformatted
65	FD hard error	; Hardware related disk error
66	il data	
67	no usable diskette	
68	(sub)master diskette	
69	mismatch sign	
70	il file name	; Invalid file name
71	il file attribute	; Invalid file attribute
72	il file type	; Invalid file type
73	il file mode	; Invalid file mode
74	il lu#	; Invalid logical number
75	not ready	} ; Printer error
76	alarm	
77	paper empty	} ; Paper tape reader or punch error
78	time out	
79	parity	
80	check sum	} ; Serial I/O errors (to be implemented later)
81	flaming	
82	over run	
83	interconnect	
84	full buffer	} ; IEEE-488 related errors (to be implemented later)
85	uncontrollable	
86	interface	
87	less data	
88	much data	
89	lu table overflow	; Attempt made to open too many files
90	source ?	
91	destination ?	
92	can't xopen	
93	too long line	; Line exceeding 128 bytes
94	end of record	
95	diff record length	

5. MUTUAL CONVERSION

Mutual conversion between files generated by different system programs are possible for the following combinations of files using the conversion procedure shown:

Possible Combinations of Files

File 1				File 2			Procedure
System Program	File	Mode		System Program	File	Mode	
BASIC	FD/CMT	BTX	↔	FDOS	FD/CMT	ASC	use FDOS CONVERT command
BASIC	FD	BSD	↔	FDOS	FD	ASC	use FDOS CONVERT command
BASIC	FD	BSD	↔	FDOS	CMT	ASC	use FDOS CONVERT command
BASIC	CMT	BSD	↔	FDOS	FD	ASC	use FDOS XFER command
BASIC	CMT	BSD	↔	FDOS	CMT	ASC	fully compatible
BASIC	FD/CMT	OBJ	↔	FDOS	FD/CMT	OBJ	use FDOS CONVERT command
K	CMT	BTX	→	BASIC	CMT	BTX	use convert-tape (MZ-80T10C)
K	CMT	ASC	↔	FDOS	FD	ASC	use FDOS XFER command with SCMT1
K	CMT	OBJ	↔	FDOS	FD	OBJ	use FDOS XFER command with SCMT1

BASIC : MZ-80B BASIC interpreter, Versions SB-5510, -5610, -6510, and -6610.

FDOS : MZ-80 FDOS or BASIC compiler SB-7xxx

K : MZ-80K .

FD : Floppy disk.

CMT : Cassette tape.

BTX : BASIC interpreter text file.

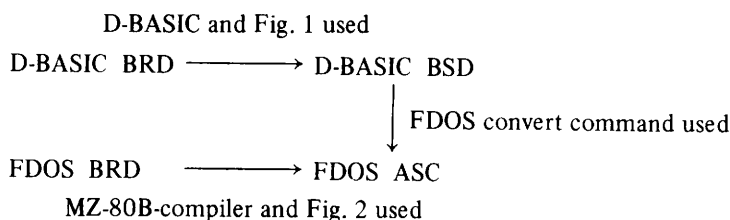
BSD : BASIC interpreter sequential data file.

ASC : ASCII file.

OBJ : Object file.

Example

When converting BRD generated by D-BASIC to File of a form acceptable by FDOS:



```

10 REM BRD → BSD sample conversion program.
20 INPUT "RND FILE ? ";R$
30 INPUT "SEQ FILE ? ";S$
40 XOPEN #1,R$: WOPEN #2,S$
50 I=1
60 INPUT #1(I),A$: IF EOF(#1) THEN CLOSE : END
70 PRINT #2,A$: I=I+1: GOTO 60
  
```

Fig. 1

```

10 REM BSD → BRD sample conversion program.
20 INPUT "SEQ FILE ? ";S$
30 INPUT "RND FILE ? ";R$
40 ROPEN #1,S$: XOPEN #2,R$
50 I=1: D$=CHR$(#0D)
60 A$=""
70 INPUT #1,B$: IF EOF(#1) THEN CLOSE : END
80 A$=A$+B$: L=LEN(A$)
90 IF L>32 THEN PRINT "ERROR": CLOSE : END
100 IF L<32 THEN A$=A$+D$: L=L+1
110 IF L<32 THEN 70
120 PRINT #2(I),A$: I=I+1: GOTO 60
  
```

Fig. 2



Personal Computer
mz-80B

Text Editor

SHARP

NOTICE

The MZ-80 series of sophisticated personal computers is manufactured by the SHARP CORPORATION. Hardware and software specifications are subject to change without prior notice; therefore, you are requested to pay special attention to version numbers of the monitor and the system software (supplied in the form of cassette tape or mini-floppy disk files).

This manual is for reference only and the SHARP CORPORATION will not be responsible for difficulties arising out of inconsistencies caused by version changes, typographical errors of omissions in the descriptions.

This manual is based on the SB-1500 series monitor and the SB-7000 series FDOS.

OUTLINE	1
Activating the Editor	1
Editor Command Table	2
 CHARACTER POINTER AND DELIMITER	3
 TEXT EDITOR COMMANDS	4
Input Commands	4
R (Read file) Command	4
A (Append file) Command	5
Output Command	6
W (Write) Command	6
Page Processing Commands	7
Type Command	10
T (Type) Command	10
CP Positioning Commands	11
B (Begin) Command	11
Z Command	11
J (Jump) Command	11
L (Line) Command	12
M (Move) Command	12
Correction Commands	13
C (Change) Command	13
Q (Queue) Command	13
I (Insert) Command	14
K (Kill) Command	15
D (Delete) Command	16
Search Command	17
S (Search) Command	17
Special Commands	18
\ Command	18
= Command	18
. Command	18
& Command	18
# Command	19
! Command	19
 ERROR MESSAGES	20



OUTLINE

The text editor generates, corrects and edits source files such as assembler source files and BASIC text files.

Data correction and edition are performed conversationally between the editor and programmer.

—Activating the Editor—

The editor is activated by the following FDOS commands. (See the photograph at right.)

1. EDIT

The editor is activated and entry of an editor command is awaited when this FDOS command is executed.

2. EDIT <filename>

The editor is activated and the specified source file in the active disk drive is read when this FDOS command is executed, then the editor waits for entry of an editor command.

3. EDIT \$FD3; <filename>

The editor is activated and the specified source file in drive 3 is read when this FDOS command is executed.

The editor has the following correction and editing functions.

1. Insertion

2. Deletion

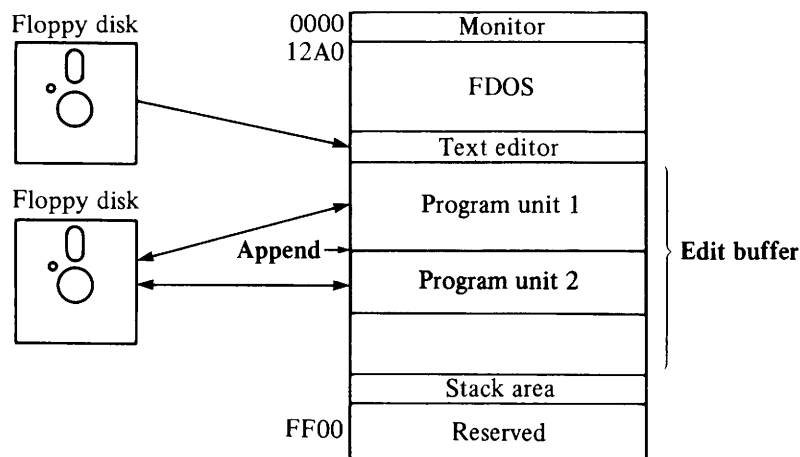
3. Alteration

Each line of source text read into the edit buffer is assigned a sequential **line number**.

Positions to be corrected are specified with the **character pointer (CP)**.

Correction and edition can be performed in a units of lines or characters. Search and replacement can be performed in character string units.

The memory map is as shown below when the text editor is used.



Editor commands are listed in the following table. Commands separated from each other with the delimiter "⌘" and are executed when CR is entered.

Command type	Command name	Function
Input command	R	Clears the edit buffer and loads it with the input file indicated by the filename. The CP is positioned at the beginning of the edit buffer after execution of this command.
	A	Appends the input file indicated by the filename to the contents of the edit buffer. The CP position is not changed.
Output command	W	Writes the edit buffer contents to the output file specified by the filename in ASCII code.
Page processing command	PR	Same as the R command, except that the maximum amount of data read is 1 page.
	PA	Same as the A command, except that the maximum amount of data read is the unused area of the edit buffer.
	PW	Same as the W command, except that the maximum amount of data output is 1 page.
	PC	Terminates execution of the processing command. This command is required whenever a PR, PA or PW command is executed.
	PK	Kills the file opened by a page processing command.
Type command	T	Displays the entire contents of the edit buffer. The CP position is not changed.
	nT	Displays n lines starting at the CP position.
CP positioning command	B	Positions the CP at the beginning of the edit buffer.
	nJ	Positions the CP at the beginning of the line indicated by n.
	nL	Moves the CP to the beginning of the line n lines after the current CP position.
	L	Moves the CP to the beginning of the current line. This is the same as when n = 0 in the nL command.
	nM	Changes the CP position by n characters.
	M	Does not move the CP. This is the same as when n = 0 in the nM command.
Correction command	Z	Moves the CP to the end of the text in the edit buffer.
	C	Searches for the specified character string and replaces it with another character string; the search starts at the current CP position and proceeds to the end of the edit buffer. The CP is repositioned to the end of the character string replaced.
	Q	Repeats the C command each time the specified character string is found until the end of the edit buffer is reached. The CP is repositioned to the end of the character string last replaced.
	I	Inserts the specified character string at the position of the CP. The CP is repositioned to the end of the character string inserted. Line numbers are updated when a line is inserted with this command.
	nK	Deletes the n lines following the CP. The CP position is not changed.
	K	Deletes all characters preceding the CP position until a CR code is detected. The CR code is not deleted.
	nD	Deletes the n characters following the CP.
Search command	D	No operation
	S	Searches for the specified character string, starting at the CP position and proceeding to the end of the buffer. The CP is repositioned to the end of the character string when it is found.
Special command	\	Executes the specified built-in command.
	=	Displays the number of characters stored in the edit buffer (including spaces and CRs).
	.	Displays the number of the line at which the CP is located.
	&	Deletes the entire contents of the edit buffer.
	#	Changes the list mode for listing to the printer.
	!	Transfers control to the FDOS.

Most of the above commands are compatible with those used in the NOVA editor program manufactured by the Data General Corporation.

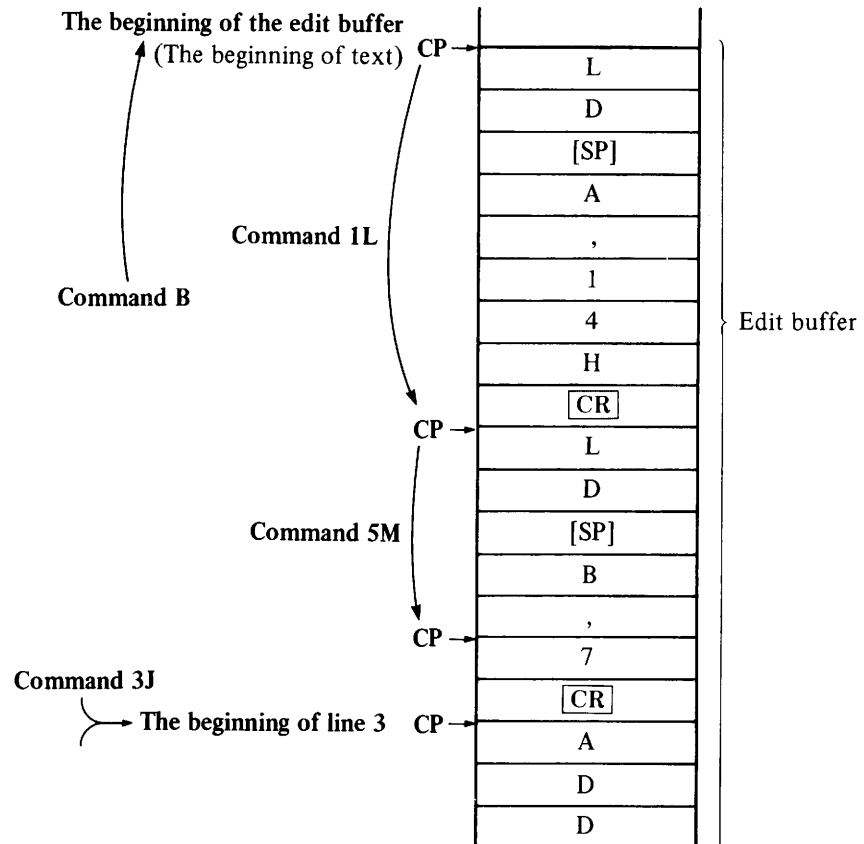
CHARACTER POINTER AND DELIMITER

The character pointer (CP) is positioned at the boundary between two adjacent characters or the beginning or end of the text. It does not point directly at any character.

Movement of the CP is explained below based on the assumption that the following text is stored in the edit buffer.

```
1 LD A, 14H
2 LD B, 7
3 ADD A, B
4 DAA
```

(Line numbers are not stored in the edit buffer.)



The B command moves the CP to the beginning of the edit buffer, the J command moves it to the top of the specified line and the L command to the beginning of the nth line from the line in which the CP is currently located; the top of the specified line is the boundary following the `CR` code of a preceding line.

Delimiters are used as separators between editor commands. Entering several editor commands and separating them with delimiters allows them to be executed consecutively by pressing the `CR` key.

The **I (Insert)** command must be followed by a delimiter because it uses `CR` codes as character codes for the source text.

The following example replaces ADD on line 3 in the above program with ADC.

`3J2M1D1C CR` or `B2CADDADC CR`

—Screen Edit—

The line on which the cursor is located can be modified by entering new data from the keyboard and pressing the `CR` key after data has been displayed by executing the T, C, Q or S command. The CP is positioned at the end of the updated line.

Note that the line number may be changed when the I, D or K command is executed.

TEXT EDITOR COMMANDS

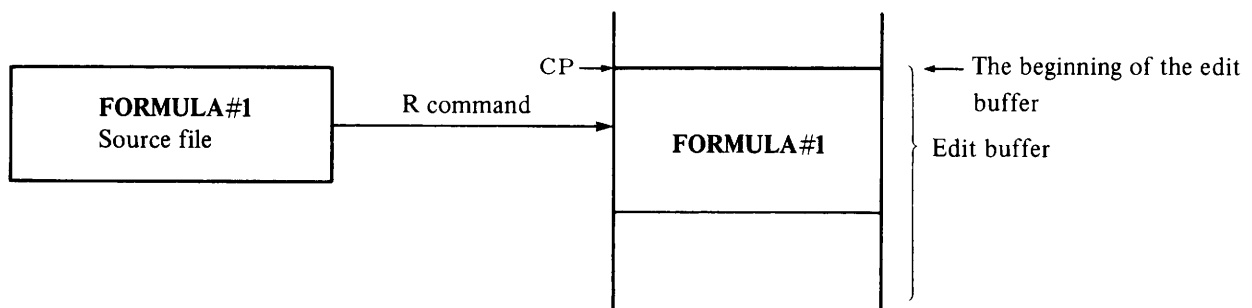
— Input Commands —

R (Read file) Command

This command clears the edit buffer area, then loads it with the source file (ASCII file) specified by the filename; loading starts at the beginning of the edit buffer. The CP is positioned at the beginning of the edit buffer after execution of this command.

- | | |
|---|---|
| * R FORMULA#1 CR | Reads source file FORMULA#1 into the edit buffer. |
| * R\$FD1; FORMULA#1 CR | Reads source file FORMULA#1 from the diskette in drive 1 into the edit buffer. |

- Enter R followed by the file name while in the command wait state.
- The editor searches for the file and reads it.
- The file is stored starting at the beginning of the edit buffer as shown below.
- The CP is positioned to the beginning of the edit buffer after reading.



- The message "Full buffer" is displayed when the edit buffer becomes full. In this case, only part of the input file is stored in the edit buffer.
- The page processing mode is entered automatically when the buffer becomes full. Therefore, the remainder of the input file can be read by the PR command.

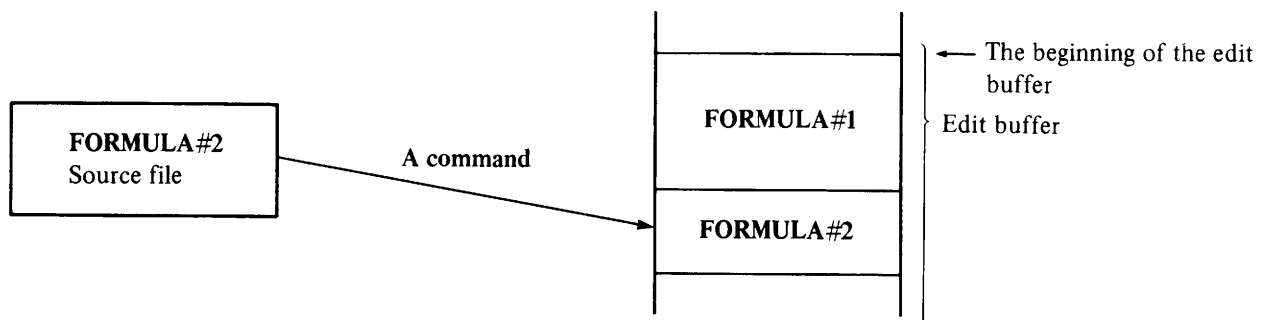
Note: "*" is displayed to indicate that the system is in the command wait state.

A (Append file) Command

This command appends the file specified by the filename to the contents of the edit buffer. The CP position is not changed.

- | | |
|--------------------------------------|---|
| * A FORMULA#2 CR | Appends source file FORMULA#2 to the contents of the edit buffer. |
| * A\$FD2; FORMULA#2 CR | Appends source file FORMULA#2 from the diskette in drive 2 to the contents of the edit buffer. |

- Enter A followed by the file name while in the command wait state.
- The editor searches for the specified file and reads it.
- The file is stored in the area following the end of the last text in the edit buffer. The figure below shows a case in which the file FORMULA#2 is appended to the file FORMULA#1.



- The message "Full buffer" is displayed when the edit buffer becomes full. In this case, only part of the specified file is stored in the edit buffer and the contents of the edit buffer must be reedited to store the entire file.
- The page processing mode is entered automatically when the buffer becomes full. Therefore, the remainder of the input file can be read by the PR command.

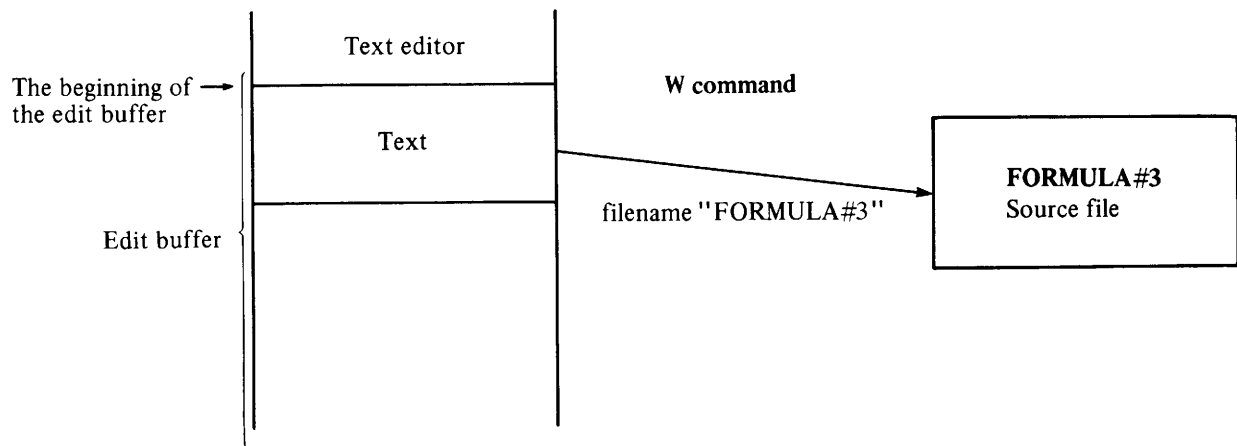
—Output Command—

W (Write) Command

This command outputs the entire contents of the edit buffer to the file specified by the filename regardless of the CP position.

- | | |
|--------------------------------|--|
| * W FORMULA#3 CR | Outputs the contents of the edit buffer to file FORMULA#3 in the active drive. |
| * W\$FD2; FORMULA#3 CR | Outputs the contents of the edit buffer to file FORMULA#3 in floppy disk drive 2 (\$FD2). |

- Enter **W** followed by the filename while in the command wait state.
- The editor waits for entry of another command after the edit buffer contents have been output. The file generated is a source file.



- The CP position is not changed by execution of the **W** command.

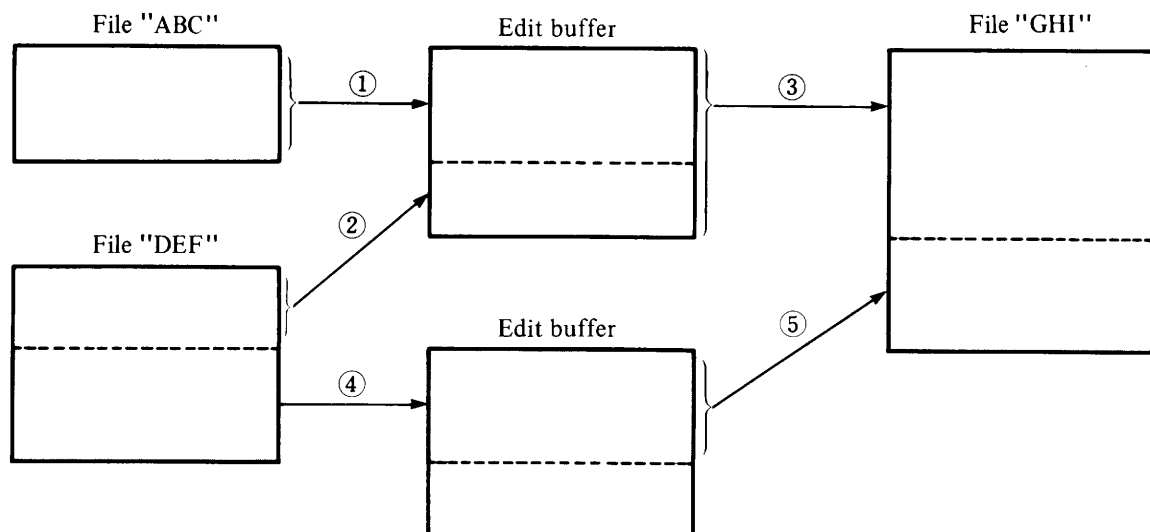
—Page Processing Commands—

These commands are used in cases where the total size of files to be edited exceeds the size of the edit buffer, as shown in the following examples.

If the diskette is replaced with a new one during page processing, the contents of the diskette may be destroyed. Be sure to terminate page processing before replacing the diskette.

1. When several files are to be edited and the resulting file is larger than the edit buffer:

- ① * PRABC CR Reads source file "ABC" into the edit buffer until the buffer is full.
- ② * PADEF CR Reads source file "DEF" and appends it to the contents of the edit buffer until the buffer is full. Message "Full buffer" is displayed.
Note: If * PRDEF CR is entered, the contents of file "ABC" stored in the edit buffer are cleared and file "DEF" is loaded in the edit buffer from its beginning.
- ③ * PWGHI CR Outputs the contents of the edit buffer to file "GHI" after editing is completed.
- ④ * PR CR Reads the remainder of a file specified by a preceding PR command into the edit buffer. In this example, the command reads the remainder of file "DEF" into the edit buffer.
Note: File name "DEF" specified in step ② remains valid.
- ⑤ * PW CR Outputs the contents of the edit buffer and appends it to the file specified by the preceding PW command after editing is completed. In this example, the command appends the contents of the edit buffer to file "GHI."
- ⑥ * PC CR Terminates page processing. (This command is mandatory.)



2. When the file to be edited is larger than the edit buffer:

① * PRABC CR Reads source file "ABC" into the edit buffer until it is full. Message "Full buffer" is displayed.

Note: Omission of the file name in the first page processing command will result in error.

② * PWDEF CR Outputs the contents of the edit buffer to file "DEF" after editing is completed.

Note: *PW CR results in an error. An error results when editing increases the size of file in the edit buffer so that it exceeds the size of the edit buffer.

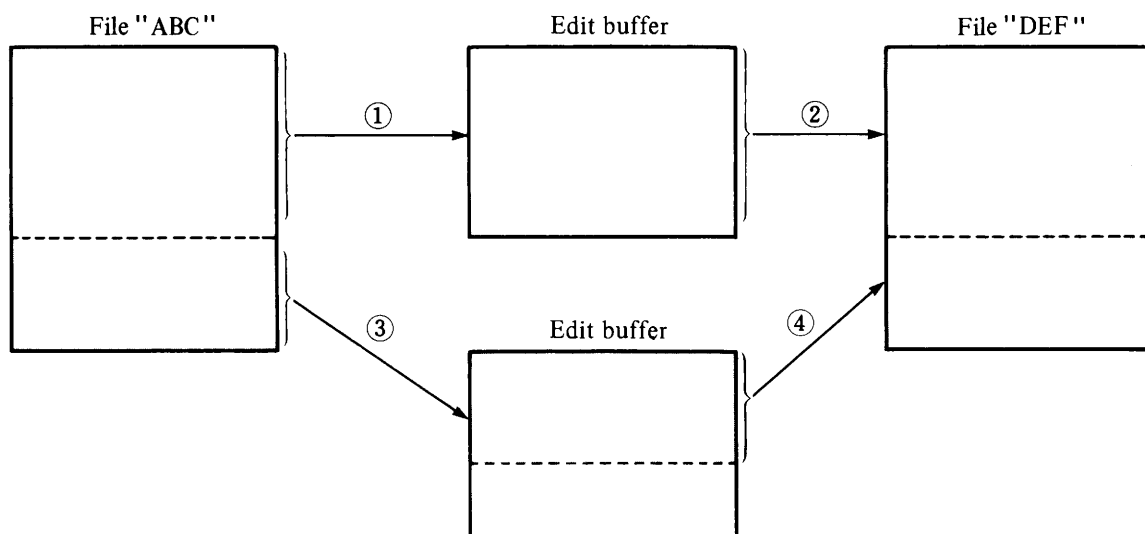
③ * PR CR Reads the remainder of file "ABC" into the edit buffer.

Note: A file name specified in step ① remains valid until a new file name is specified.

④ * PW CR Appends the contents of the edit buffer to file "DEF" after editing is completed.

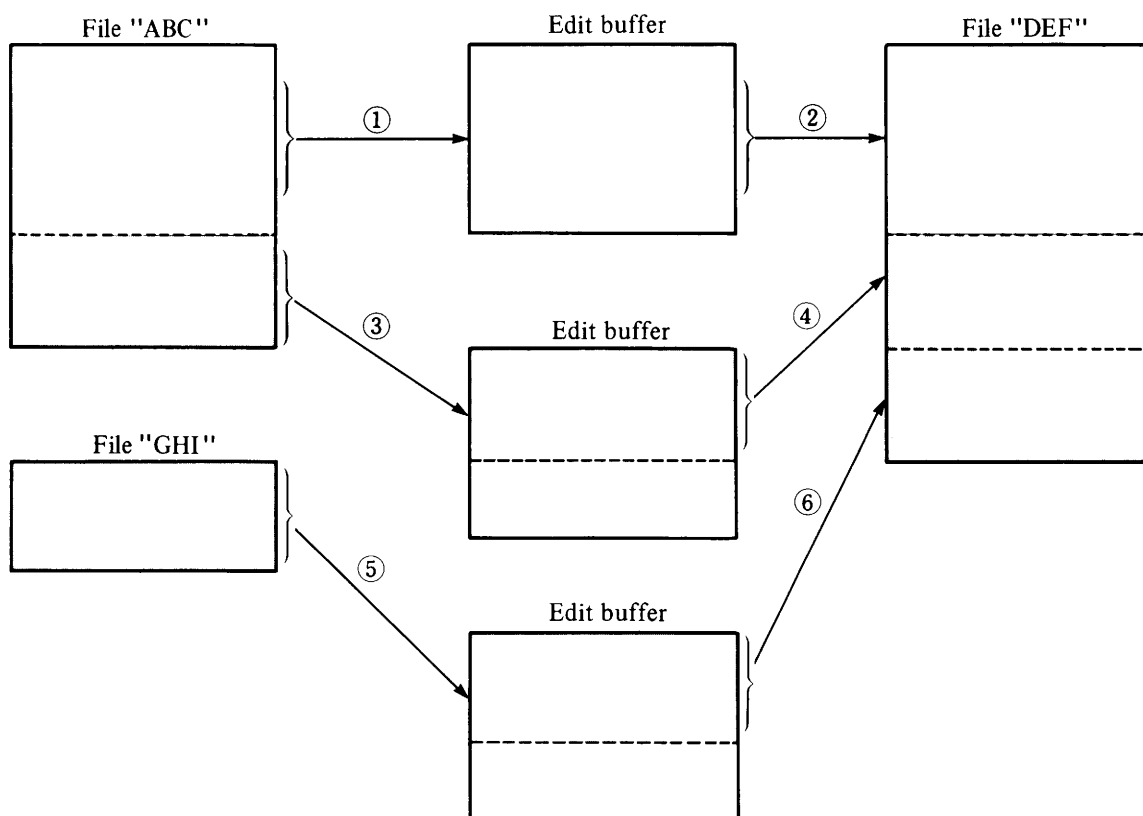
⑤ * PC CR Terminates page processing. (This command is mandatory.)

⑥ Repeat steps ③ and ④ when file "ABC" is too large to be processed by performing steps ③ and ④ once.



3. When the file to be edited first is larger than the edit buffer and another file is to be edited and appended to the first edited file:

- ① * PR ABC CR Reads file "ABC" into the edit buffer until the buffer is full.
 - ② * PW DEF CR Outputs the contents of the edit buffer to file "DEF" after editing is completed.
 - ③ * PR CR Reads the remainder of file "ABC" into the edit buffer.
 - ④ * PW CR Appends the contents of the edit buffer to file "DEF" after editing is completed.
 - ⑤ * PR GHI CR Reads file "GHI" into the edit buffer.
- Note: In this case, specifying *PR CR will not be valid if the end of file "ABC" has been reached.
- ⑥ * PW CR Appends the contents of the edit buffer to file "DEF" after editing is completed.
 - ⑦ * PC CR Terminates page processing. (This command is mandatory.)



4. When a file which was opened by a page processing command is to be killed:

- ① * PK CR Kills the file opened by a page processing command.

T (Type) Command

This command displays all part of the contents of the edit buffer. The CP position is not changed.

- * **T** CR Displays all of the contents of the edit buffer with line numbers attached.
- * **nT** CR Displays the n lines following the CP. (Same as the above when n = 0.)

- Key in the number of lines, **n** followed by **T** (Type) while in the command wait state.
- Press **CR** to display the contents of the edit buffer.

- The following are special cases of nT.

$n = 0$: the same as T

$n < 0$: results in the error message "???"

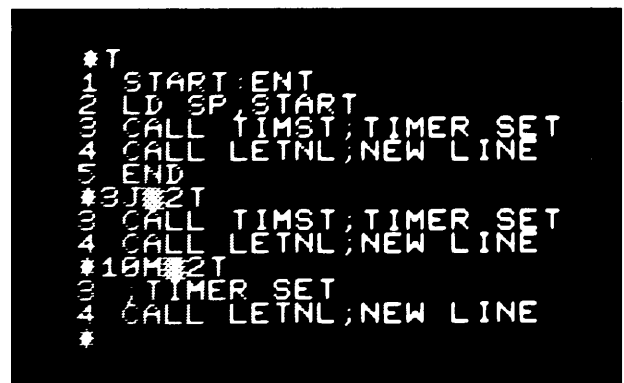
$n \geq m$ (where m is the number of lines from the one at which the CP is located to the end of the buffer contents): only m lines are displayed.

- The current CP position can be determined with the nT command, since display starts with the character following the boundary at which the CP is located.
- Press the **BREAK** key to terminate T command execution.
Press the **SPACE** key to suspend T command execution, and press it again to resume it.
- The photograph at right shows the relationship between the type command and the CP for the following text.

- ```

1. START : ENT
2. LD SP, START
3. CALL TIMST ;TIMER SET
4. CALL LETNL ;NEW LINE
5. END

```



- The error message "Large" is displayed when n exceeds 65535.

## —CP Positioning Commands—

### B (Begin) Command

\* **B** **CR**      Positions the CP to the beginning of the edit buffer.

- Key in **B** while in the command wait state.
- Press **CR**.
- The B command is executed to position the CP to the beginning of the edit buffer.
- nB performs the same function.

### Z Command

\* **Z** **CR**      Positions the CP to the end of the contents of the edit buffer.

- Key in **Z** while in the command wait state.
- Press **CR**.
- The Z command is executed to position the CP to the end of the contents of the edit buffer.
- nZ performs the same function.

### J (Jump) Command

\* **nJ** **CR**      Positions the CP to the beginning of line n.

- Key in line number n and **J** while in the command wait state.
- Press **CR**.
- The nJ command is executed to position the CP to the beginning of line n.
- The following are special cases.

n = 0 or 1 or n is omitted: the command performs the same function as the B command.

n < 0 : results in the error message "???".

n ≥ m (where m is the number of lines of the edit buffer contents): the command performs the same function as the Z command.

## L (Line) Command

This command moves the CP forward or backward the specified number of lines. The CP is positioned at the beginning of the specified line after execution.

- |                       |                                                                                               |
|-----------------------|-----------------------------------------------------------------------------------------------|
| * <b>nL</b> <b>CR</b> | Moves the CP to the beginning of the nth line from the line at which it is currently located. |
| * <b>L</b> <b>CR</b>  | Moves the CP to the beginning of the line at which it is currently located.                   |

- Key in number of lines, **n** and **L** while in the command wait state.
- Press **CR**.
- The CP is positioned at the beginning of the specified line when the **nL** command is executed.
- The following are special cases:

$n = 0$  : the command functions in the same manner as the **L** command.

$n \geq m$  (where  $m$  is the number of lines from the line on which the CP is located to the end of the edit buffer contents): the command functions in the same manner as the **Z** command.

$n < 0$ : the CP is moved  $|n|$  lines toward the beginning of the edit buffer.

$|n| \geq \ell - 1$  (where  $\ell$  is the number of the line at which the CP is currently located): the command functions in the same manner as the **B** command.

## M (Move) Command

This command moves the CP forward or backward by the specified number of characters. Spaces and carriage returns are counted as characters, but line numbers are not.

- |                       |                                                                                      |
|-----------------------|--------------------------------------------------------------------------------------|
| * <b>nM</b> <b>CR</b> | Moves the CP to the position which is <b>n</b> characters from its current position. |
|-----------------------|--------------------------------------------------------------------------------------|

- Key in number of characters, **n** and **M** while in the command wait state.
- Press **CR**.
- Executing the **nM** command moves the CP to the specified boundary between characters.
- When  $n < 0$ , the CP is moved backward by  $|n|$  characters.
- The CP position is not changed when  $n = 0$  or if it is omitted.

## —Correction Commands—

### C (Change) Command

This command replaces a string in the edit buffer with another string. The search for the specified string starts at the current CP position and proceeds toward the end of the edit buffer; the string is replaced when it is found and the CP is positioned at the end of the string replaced.

- |                                                                                                                                                                                          |                                                                                                                                                                                                                                        |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| * Cstring 1  string 2  | Searches for the character string specified by <b>string 1</b> , starting at the current CP position and proceeding toward the end of the edit buffer; replaces the string with the one specified by <b>string 2</b> when it is found. |
| * Cstring 1                                                                                             | Deletes the character string specified by <b>string 1</b> .                                                                                                                                                                            |

- Key in C while in the command wait state.
- Key in the string to be located followed by a delimiter.
- Key in the string which is to replace the one located.
- Press  and a search is made for the first string. Only the first occurrence of the string is replaced. The line including the string replaced is displayed and the CP is positioned at the end of that string.
- The message "Not found" is displayed if the specified string is not found and the CP is positioned to the beginning of the edit buffer.

### Q (Queue) Command

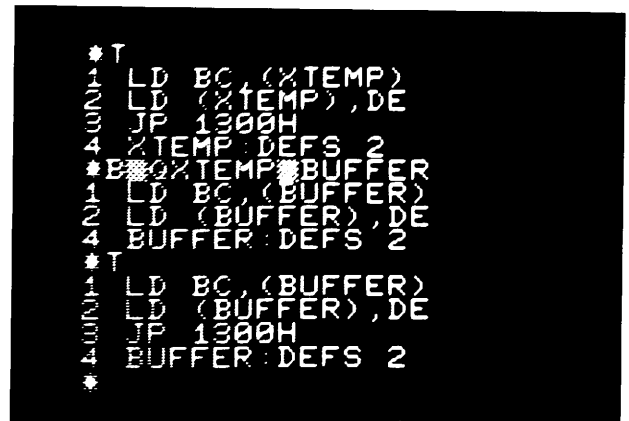
This command repeats the function of the C command each time the specified character string is found until the end of the edit buffer is reached. The CP is repositioned to the end of the string last replaced.

- |                                                                                                                                                                                              |                                                                                |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| * Qstring 1  string 2  | Causes the function of the C command to be executed repeatedly.                |
| * Qstring 1                                                                                               | Deletes all occurrences of the character string specified by <b>string 1</b> . |

- Key in Q while in the command wait state.
- The remainder of the operation is the same as for the C command.

- The photograph at right shows the result of execution of the Q command on the following text.

```
1 LD BC, (XTEMP)
2 LD (XTEMP), DE
3 JP 1300H
4 XTEMP : DEFS 2
```



## I (Insert) Command

This command inserts the specified string at the CP position. A carriage return is performed on the CRT screen if one is included in the string.

Line numbers are updated automatically when a new line is inserted. The CP is repositioned to the end of the string inserted.

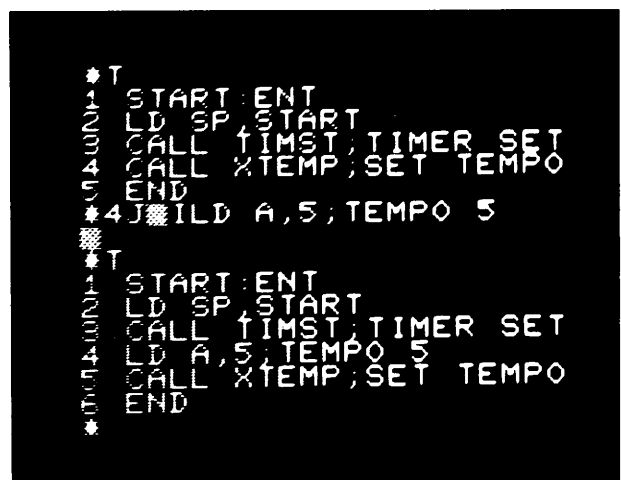
- |                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                     |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| * Istring   | Inserts the specified string at the CP position.                                                                                                                                                                                                                                                    |
| * Istring 1                                                                                  | Inserts the lines specified by string 1, string 2 and string 3 at the CP position.                                                                                                                                                                                                                  |
| string 2                                                                                     |                                                                                                                                                                                                                                                                                                     |
| string 3                                                                                     |                                                                                                                                                                                                                                                                                                     |
|             | A  is treated as a character by the I command. Therefore, a delimiter must be keyed in before  is pressed to execute the command. |

- Key in I while in the command wait state.
- Key in the string to be inserted.
- Strings keyed in are inserted at the CP position and the contents of the edit buffer following the CP are automatically shifted toward the end of the edit buffer.
- When a  is keyed in, it is inserted as a carriage return code.
- Key in a delimiter  after all the strings have been keyed in.
- Press  key to execute the I command.
- The photograph at right shows an example of using the I command.

Text:

```
1 START: ENT
2 LD SP, START
3 CALL TIMST ;TIMER SET
4 CALL XTEMP ;SET TEMPO
5 END
```

LD A, 5 ;TEMPO 5 is inserted between lines 3 and 4 of the above text.



## K (Kill) Command

This command deletes the  $n$  lines preceding or following the CP from the edit buffer.

- |                                                                              |                                                                                                                                                                                                                            |
|------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| * <b>nK</b> <span style="border: 1px solid black; padding: 0 2px;">CR</span> | Deletes the $n$ lines preceding or following the CP from the edit buffer.<br>A line is not deleted if the CP is located within it, since characters preceding or following the CP are not deleted.                         |
| * <b>K</b> <span style="border: 1px solid black; padding: 0 2px;">CR</span>  | Deletes characters preceding the CP position until a <span style="border: 1px solid black; padding: 0 2px;">CR</span> is detected.<br>The <span style="border: 1px solid black; padding: 0 2px;">CR</span> is not deleted. |

— Key in the number of lines,  $n$  and **K** while in the command wait state.

— Press CR to execute the K command.

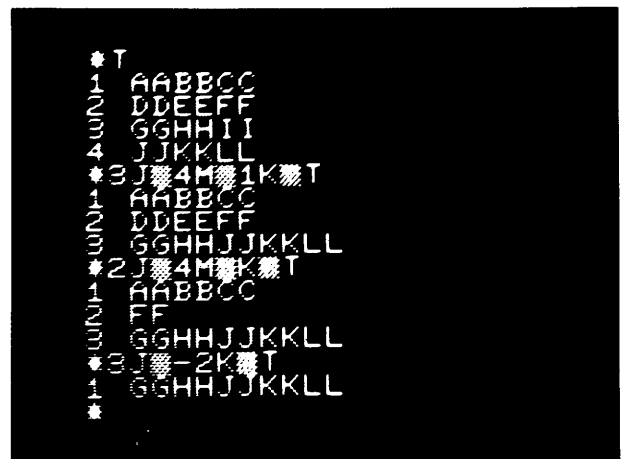
— Operation differs according to the value of  $n$  as follows.

- $n > 0$  : Deletes all characters following the CP until  $n$  CR codes are detected.  
CR codes detected are also deleted. Command execution ends after the last CR code has been deleted.
- $n < 0$  : Deletes all characters preceding the CP until  $|n| + 1$  CR codes are detected.  
The  $(|n| + 1)$ th CR code is not deleted.
- $n = 0$  or not specified : Deletes all characters preceding the CP until a CR code is detected. That is, deletes the part of the line in front of the CP. The CR code detected is not deleted.

— Line numbers are automatically updated after deletion.

— The CP position is not changed.

— The photograph at right shows an example of the result of execution of the K command with the following text. (This text is presented only for the purpose of illustration; it has no meanings in assembly language.)



```
1 AABBC
2 DDEEF
3 GGHII
4 JKKLL
```

## D (Delete) Command

This command deletes the specified number of characters from the edit buffer, starting at the CP position.

- |           |                                                                                                                                         |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------|
| * nD [CR] | Deletes the specified number of characters from the edit buffer, starting at the CP position.<br>A [CR] code is counted as a character. |
| * D [CR]  | (No operation results.)                                                                                                                 |

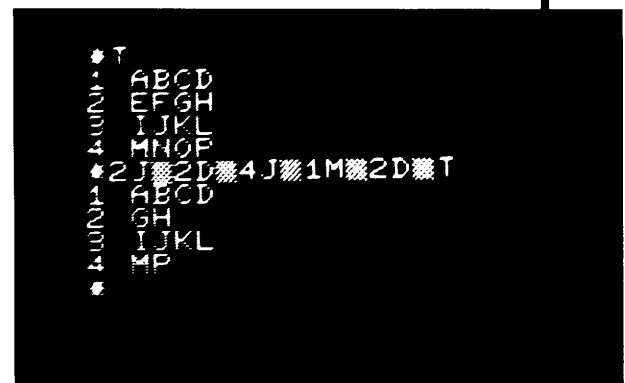
- Key in the number of character **n** and **D**.
- Press [CR] to execute the command.
- Operation differs according to the value of **n** as follows.

- n > 0 :** Deletes the **n** characters following the CP from the edit buffer.  
A [CR] code is counted as a character.
- n < 0 :** Deletes the |**n**| characters preceding the CP from the edit buffer.  
A [CR] code is counted as a character.
- n = 0 or not specified** No operation results.

- Line numbers are automatically updated if necessary.
- The CP position is not changed.

- The photograph at right shows an example of the result of execution of the D command with the following text. (This text is presented only for the purpose of this illustration; it has no meaning in assembly language.)

- 1 ABCD
- 2 EFGH
- 3 IJKL
- 4 MNOP



## —Search Command—

### S (Search) Command

This command searches for the specified character string in the contents of the edit buffer.

\* **S string** **[CR]** Searches for the specified character string, starting at the current CP position; the CP is repositioned to the end of the character string when it is found.

- Key in S.
- Key in the string to be located.
- Press **[CR]** to execute the S command.
- The search starts at the current CP position and proceeds toward the end of the buffer.
- When the specified string is found, the line which includes it is displayed and the CP is positioned to the end of the character string.
- If the specified string cannot be found, the message "Not found" is displayed and the CP is repositioned to the beginning of the edit buffer.
- The photograph at right shows the result of a search for the character string "LETNL" in the following text. The line including "LETNL" is displayed following the S command. The 2T command indicates that the CP is positioned to the end of the string.

```
1 START: ENT
2 LD SP, START
3 CALL TIMST ;TIMER SET
4 CALL LETNL ;NEW LINE
5 LD A, 04H ;TEMPO<--4
6 CALL XTEMP
7 END
```



## —Special Commands—

### \ Command

This command executes the specified built-in command. " \* " is displayed to indicate that command entry is awaited after execution.

\* \DELETE ABC  Deletes file ABC.ASC.

- Enter the \ command when " \* " appears to indicate that command entry is awaited.
- Specify the built-in FDOS command and press the  key; the command is then executed.
- The XFER and EXEC commands cannot be executed. The RUN command cannot be executed when the program is too long to be executed.
- No built-in FDOS commands can be executed in the page processing mode. If an attempt is made to execute one in the page processing mode, the message "Page opened!" is output. In this case, reset the page processing mode with the PC or PK command.

### = Command

\* =  Displays the total number of characters (including spaces and CRs) stored in the edit buffer.

- Key in " = " (equal) while in the command wait state.
- Press ; the total number of characters stored in the edit buffer is displayed.

### . Command

\* .  Displays the number of the line on which the CP is located.

- Key in . (period) while in the command wait state.
- Press ; the line number on which the CP is located is displayed.

### & Command

\* &  Clears the contents of the edit buffer.

- Key in & (ampersand) while in the command wait state.
- Press ; the contents of the edit buffer are then cleared.

## # Command

\* # **[CR]** Changes the printer list mode.

- Key in # (sharp symbol) while in the command wait state.
- Press **[CR]**; the printer list mode is then changed.
- The printer list mode is disabled when the text editor is started. It is enabled when the # command is executed once; executing it again disables it, and so on.
- The following shows a listing obtained by executing the T command when the printer list mode is enabled.

```
1 ; TYPE COMMAND
2 ;
3 .TYPE:ENT
4 LD DE,SWTBL; DE:=SWITCH TABLE
5 CALL ?GSW; CHECK GLOBAL SWITCH
6 RET C
7 CALL C&L1 ; SELECT CRT OR LPT
8 CALL ?SEP ; CHECK SEPARATER
9 RET C
10 CP 2CH ; SEPARATER="," ?
11 LD A,3 ; 3 IS ERR CODE
12 SCF
13 RET NZ ; NO, ERR RETURN
14 TYPE0:CALL ?LSW ; CHECK LOCAL SWITCH
15 RET C
16 LD A,8 ; 8 IS ERR CODE
17 SCF
18 RET NZ ; ERROR, LSW EXIST
19 LD C,128 ; LU#:=128
20 EXX
21 LD B,4; DEFAULT MODE=ASC
22 EXX
23 CALL ROPEN ; READ-OPEN
24 RET C
25 CALL &NL
26 JR C,TYPEER
27 CALL TESW ; TEST GLOBAL SWITCH
28 DEFB 88H; /P
29 CCF
30 CALL C,PPAGE; LPT PAGING
31 JR C,TYPEER
32 CALL MODECK ; FILEMODE CHECK
33 DEFB 4 ; .ASC ?
```

## ! Command

\* ! **[CR]** Returns control to FDOS.

- Key in ! (exclamation mark) while in the command wait state.
- Press **[CR]**; control is then returned to FDOS.

# ERROR MESSAGES

| Error message                                         | Explanation                                                                                                                                                                                                                                                                                                                                                                                                                          | Related commands       |
|-------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|
| <b>Full buffer</b>                                    | The edit buffer is full.                                                                                                                                                                                                                                                                                                                                                                                                             | R, A, PR, PA           |
| <b>???</b>                                            | $n < 0$ in the nT or nJ command.                                                                                                                                                                                                                                                                                                                                                                                                     | T, J                   |
| <b>Large</b>                                          | n greater than 65535 was specified.                                                                                                                                                                                                                                                                                                                                                                                                  | T, J, L, M, K, D, B, Z |
| <b>Not found</b>                                      | The string specified in the command was not found.                                                                                                                                                                                                                                                                                                                                                                                   | S, C, Q                |
| <b>Invalid</b>                                        | Other than an editor command was entered or an incorrect format was used.<br>Ex) * H <span style="border: 1px solid black; padding: 0 2px;">CR</span> : There is no H command.<br>* S <span style="border: 1px solid black; padding: 0 2px;">CR</span> : A string should be specified.                                                                                                                                               | Any case               |
| <b>Page opened ?</b>                                  | The file to be subjected to page processing is not defined (or is not opened).                                                                                                                                                                                                                                                                                                                                                       | PR, PA, PW, PC         |
| <b>Page opened !</b>                                  | An attempt to execute the ! or \command was made on a file which was subjected to page processing, but which was not closed or killed by the PC or PK command.                                                                                                                                                                                                                                                                       | !, \                   |
| <b>No file is saved after edition<br/>End of job?</b> | These messages are displayed where an attempt is made to execute a ! command after the edit buffer contents have been corrected without first executing a W or PW command. Pressing the <span style="border: 1px solid black; padding: 0 2px;">Y</span> key in this case executes the ! command. Pressing the <span style="border: 1px solid black; padding: 0 2px;">N</span> key causes the system to await entry of a new command. | !                      |

Note: Refer to the System Error Messages in the System Command manual for the system errors.

Display of the message "Already opened" during execution of a W command indicates that there is a PW command which has not been closed.

Personal Computer  
**MZ-80B**

Z-80 Assembler

**SHARP**

### NOTICE

The MZ-80 series of sophisticated personal computers is manufactured by the SHARP CORPORATION. Hardware and software specifications are subject to change without prior notice; therefore, you are requested to pay special attention to version numbers of the monitor and the system software (supplied in the form of cassette tape or mini-floppy disk files).

This manual is for reference only and the SHARP CORPORATION will not be responsible for difficulties arising out of inconsistencies caused by version changes, typographical errors or omissions in the descriptions.

This manual is based on the SB-1500 series monitor and the SB-7000 series FDOS.

— CONTENTS —

|                                                      |    |
|------------------------------------------------------|----|
| <b>INTRODUCTION</b> .....                            | 1  |
| <b>ASSEMBLY LANGUAGE RULES</b> .....                 | 3  |
| Characters .....                                     | 4  |
| Line .....                                           | 5  |
| Label Symbols .....                                  | 5  |
| Constants .....                                      | 6  |
| <b>ASSEMBLY LISTING AND ASSEMBLER MESSAGES</b> ..... | 7  |
| Definition Condition Messages .....                  | 8  |
| Error Messages .....                                 | 8  |
| <b>ASSEMBLER DIRECTIVES</b> .....                    | 10 |
| ENT (entry) .....                                    | 10 |
| EQU (equate) .....                                   | 11 |
| ORG (origin) .....                                   | 12 |
| DEFB n (define byte) .....                           | 13 |
| DEFB 'S', DEFB "S" (define byte) .....               | 13 |
| DEFW nn' (define word) .....                         | 14 |
| DEFM 'S', DEFM "S" (define message) .....            | 14 |
| DEFS nn' (define storage) .....                      | 15 |
| SKP n (skip n lines) .....                           | 16 |
| SKP H (skip home) .....                              | 16 |
| END (end) .....                                      | 16 |
| <b>MESSAGE TABLE</b> .....                           | 17 |



# INTRODUCTION

The assembler translates a source file written in assembly language to generate a relocatable binary file; the source file is one which has been generated and edited by the text editor, and the relocatable binary file is an intermediate file between the source file and object file. It is possible to link several relocatable files by the linker.

The assembly source file is coded in assembly language. It consists of labels, mnemonic operations codes, assembler directives, comments and an end directive; these are arranged according to the rules of the assembler. The source file edited by the editor is written in ASCII code. The assembler translates the source file to generate a relocatable file and outputs messages which indicate definition conditions and syntax errors. These messages are included in the assembly listing which is displayed on the CRT or printed on the printer.

The following FDOS commands activate the assembler.

- **ASM SAMPLE**

Activates the assembler. The assembler translates source file SAMPLE.ASC and generates relocatable file SAMPLE.RB.

- **ASM SAMPLE, \$LPT/L, \$CRT/E**

Activates the assembler. The assembler translates source file SAMPLE.ASC, generates relocatable file SAMPLE.RB, prints the assembly listing on the printer and display only erroneous lines of the CRT screen.

- **ASM/N SAMPLE, \$SOA/L**

Activates the assembler. The assembler translates source file SAMPLE.ASC and outputs the assembly listing to serial output port A (\$SOA), but does not generate a relocatable file since global switch/N is specified.

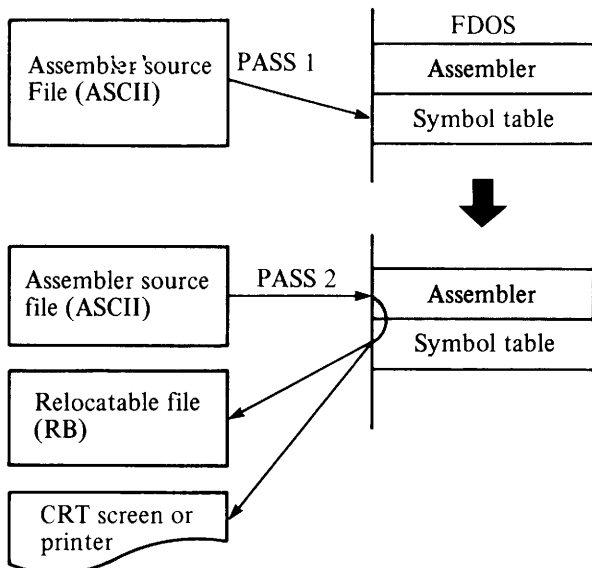
- **ASM SAMPLE, \$FD3; SAMLIST/L**

Activates the assembler. The assembler translates source file SAMPLE.ASC, generates relocatable file SAMPLE.RB and outputs the assembly listing in the same form as that printed on the printer to SAMLIST.ASC on FD3 in ASCII code.

- **ASM SAMPLE, \$LPT/L, \$4000**

Activates the assembler. The assembler translates source file SAMPLE.ASC, generates relocatable file SAMPLE.RB and prints the assembly listing on the printer with a bias of \$4000 added to the relocatable address. Relocatable file is not affected by the bias of \$4000.

The assembler basically uses a 2-pass system. A pass is the process in which the assembler reads a source file from its beginning to end. The following shows operation of the assembler with the 2-pass system.



During pass 1, the assembler stores label symbols according to the assembler rules in the symbolic label table. Label symbols help the operator to read and understand the program easily.

During pass 2, the assembler generates a relocatable file with reference to the symbol table generated during pass 1, then outputs the assembly listing (on the CRT or printer).

The relocatable file and the assembly listing do not occupy space in RAM, which is only used by the symbol table. Therefore, the size of the source file to be assembled is not limited by the amount of RAM.

The following program list will help you understand the function of the assembler. This program is only for reference and has no meaning.

```

** Z80 ASSEMBLER SB-7201 <A> PAGE 01 ??/??/?
01 0000 ;
02 0000 ; SAMPLE LIST
03 0000 ;
04 2000 ORG 2000H
05 2000 3E33 LD A, '3'
06 2002 FE43 CP 43H
07 2004 FE43 CP 'C'
08 2006 FE05 CP 'H'
09 2008 22 DEFB ' '
10 2009 27 DEFB ' '
11 200A 43 DEFB 'C'
12 200B 02 DEFB 'H'
13 200C 06050201 DEFM 'CHHLLHLLH'
14 2010 0304
15 2012 7E LD A, (HL)
16 2013 7E LD A, M ; M may be used in place of (HL).
17 2014 ;
18 2014 ;
19 2014 P XYZ: EQU 10
20 2014 C32120 JP ABC+XYZ ; Relocatable address ± EQU defined symbol value.
21 2017 C30A00 ABC: JP XYZ
22 201A C31420 JP ABC-3
23 201D C30A00 JP 10 ; Absolute address 10
24 2020 C32A20 JP +10 ; Relative address 2AH (20H+10)
25 2023 2100D0 LD HL, D000 ; Handled as a hexadecimal number.
26 2026 213930 LD HL, 12345
27 2029 212120 LD HL, ABC+XYZ
28 202C 3E0D LD A, XYZ+3 ; EQU defined label value ± numerical data
29 202E 3EFF LD A, -1 ; Negative value is converted into one's complement.
30 2030 21FFFF LD HL, -1
31 2033 21F0FF LD HL, -10H
32 2036 C33520 JP -1
33 2039
34 2039 CD4A20 CALL ZZZ
35 203C CD5420 CALL ZZZ+10
36 203F CD4B20 CALL ZZZ+XXX
37 2042 21FFFF LD HL, -XXX
38 2045 21FEFF LD HL, -XXX-XXX
39 2048 4920 DEFW ZZZ-XXX
40 204A 00 ZZZ: NOP
41 204B P XXX: EQU 1
42 204B END

** Z80 ASSEMBLER SB-7201 <A> PAGE 02 ??/??/?
ABC 2017 XXX 0001 XYZ 000A ZZZ 204A ; Indicates the contents of the symbol table.

```

# ASSEMBLY LANGUAGE RULES

The source program must be coded according to assembly language rules. This paragraph describes the structure of the source program and the assembly language rules.

The assembly source program consists of the following.

**Z80 instruction mnemonic codes**

**Label symbols**

**Comments**

**Assembler directives  
(Pseudo instructions)**

**Definition directives**

**Entry directives**

**Skip directives**

**End directive**

Comments may be used as needed by the programmer; they have no effect on execution of the program and are not included in the relocatable file.

All assembly source programs must be ended with the assembler directive END.

**Z80 instruction mnemonic codes** from the body of the assembly source program. These are explained in a separate volume.

A mnemonic code consists of an op-code of up to 4 characters, separators (space, comma, etc.) and operands.

A **label symbol** symbolically represents an address or data. A label symbol is either placed in the label column and separated from the following instruction with a colon ( : ), or placed in an operand.

The first 6 characters of a label symbol are significant and the 7th and following characters (if used) are ignored. Therefore, ABCDEFG and ABCDEFH are treated as the same label symbol.

Alphanumerics are generally used for label symbols, but any characters other than those used for separators and special symbols may be used.

**Comments** are written between the separator " ; " and a CR code; these have no influence on program execution.

**Assembler directives** will be explained later in this manual. These are written in the same column as the Z80 instruction mnemonic codes.

An **END directive** is one of the assembler directives; all assembly source programs must end with this directive.

## —Characters—

Characters which are used in an assembly source program are alphanumerics, special symbols and other characters. The special symbols have functional meanings. (Separators, `CR`, `SPACE`, etc.)

### 1) Alphabetic characters: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

These characters are used to represent symbols and instruction mnemonic codes. A ~ F are also used for representing hexadecimal values. Further, D is used to indicate decimal and H is used to indicate hexadecimal.

### 2) Numerics: 0 1 2 3 4 5 6 7 8 9

These are used to represent constants and symbols. Whether a constant is a hexadecimal number or a decimal number is determined according to the rules of constants.

### 3) Space

Spaces are treated as separators except when they are used in comments. They perform the tabulation function on the assembly listing when they are placed between op-code and operand or between operand and comment as shown below:

|          |                                                         |                    |
|----------|---------------------------------------------------------|--------------------|
| Example: | OR <code>SP</code> F0H <code>SP</code> ; A<-X0          | } Editor list      |
|          | XYZ : PUSH <code>SP</code> AF                           |                    |
|          | ADD <code>SP</code> HL, BC <code>SP</code> ; BC = COUNT |                    |
|          | ↓                                                       |                    |
|          | XYZ: OR F0H ; A<-X0                                     | } Assembly listing |
|          | PUSH AF                                                 |                    |
|          | ADD HL, BC ; BC = COUNT                                 |                    |
|          | ↑            ↑                                          |                    |
|          | Tab set    Tab set                                      |                    |

### 4) Colon " : "

A colon behaves as a separator when it is placed between a label symbol and an instruction. It performs the tabulation function on the assembly listing.

|          |                     |
|----------|---------------------|
| Example: | START: LD SP, START |
|          | MAIN: ENT           |
|          | ↑            ↑      |
|          | Tab set    Tab set  |

An address is assigned to the label symbol even if no instruction follows. (See the paragraph on symbols.)

|          |               |                                                   |
|----------|---------------|---------------------------------------------------|
| Example: | ENTRY:        | ← "ENTRY" is assigned the same address as "TOP0". |
|          | TOP0: PUSH HL |                                                   |

### 5) Semicolon " ; "

A semicolon represents the beginning of a comment. None of the characters between a semicolon and a `CR` code have any influence on execution of the program. The semicolon is placed at the top of a line or the beginning of a comment column.

|          |                  |                           |
|----------|------------------|---------------------------|
| Example: | ;                | } All lines are comments. |
|          | ; SAMPLE PROGRAM |                           |
|          | ;                |                           |
|          | CMMNT: ENT       |                           |
|          |                  | ; COMMENT                 |
|          |                  | Comment column            |

## 6) Carriage return ( CR )

A carriage return code represents the end of a line.

## 7) Other special symbols: + - ' ( ) ,

All these are special symbols used in instructions.

## 8) Other symbols

Other characters are not generally used, although they may be used as symbol labels or in the comment column.

## —Line—

Each line of a source program is formed of alphanumeric and symbols, and is ended with a carriage return. Except for comments, each line includes only one of the Z80 instructions, an assembler directive, an end statement or an empty statement for a skip.

Components on each line are arranged according to the tab settings when it is listed. (See the assembly listing on page 7.)

## —Label Symbols—

All characters other than special symbols may be used for label symbols, but generally alphanumeric are used. Each label symbol can consist of up to 6 characters; the 7th and following characters, if used, are ignored by the assembler.

|          |           |          |       |        |        |                                |
|----------|-----------|----------|-------|--------|--------|--------------------------------|
| Example: | Correct   | ABC      | START | BUFFER | 50STEP |                                |
|          | Incorrect | (ABC)    | ,HL   | IY+3   | XYZ+3  | ← Special characters are used. |
|          |           | COMPARE0 |       |        |        |                                |
|          |           | COMPARE1 |       |        |        |                                |

} The following labels are treated as the same label symbol "COMPAR".

Assembler directive EQU defines data (1 byte or 2 bytes) for a label symbol and assigns it to the label.

|          |        |     |       |
|----------|--------|-----|-------|
| Example: | ABC:   | EQU | 3     |
|          | CR:    | EQU | 0DH   |
|          | VRAM0: | EQU | D000H |

Assembler directive ENT defines a label symbol as a global symbol. A colon ( : ) placed between a label symbol and a following instruction defines the label symbol as a relocatable instruction address.

|          |        |         |
|----------|--------|---------|
| Example: | RLDR:  | ENT     |
|          | RLDR0: | PUSH HL |

When a label symbol is referenced (that is, when it is used as an operand), the assembler first searches the symbol table for the specified label symbol; if it is not found, the assembler treats it as hexadecimal data. For example, when CALL ABC is encountered, the assembler searches the symbol table for ABC; if it is not found, the assembler treats it as 0ABCH and calls address 0ABC.

A label symbol used as an operand must be defined in the assembly source program unit in which it is used, or must be defined as a global symbol in another assembly source program unit. Otherwise, it is converted into binary and left undefined.

A label symbol which has once been defined cannot be defined again.



## ASSEMBLY LISTING AND ASSEMBLER MESSAGES

The assembly listing is output to the CRT screen or printer when an FDOS system command ASM is executed with \$CRT/L or \$LPT/L specified as an argument. Examining the assembly listing is one of the most important procedures in assembly programming since this is when a check is made for errors in the source program.

The assembler translates the specified source program and outputs the assembly listing, which includes line numbers, relative addresses, relocatable binary codes, assembler messages and the source program list (including label symbols, Z80 instruction mnemonic codes and comments). The assembly listing is pages every 60 lines.

The comment column is displayed when the number of characters per line is set to 80, but is not displayed when it is set to 40.

The assembly listing format is shown below. Tabs are set at the beginnings of labels, op-codes, operands and comment columns.

| Line number | Relative address | Relocatable binary code | Assembler message | Label   | Op-code  | Operand             | Comment                                                                          |
|-------------|------------------|-------------------------|-------------------|---------|----------|---------------------|----------------------------------------------------------------------------------|
|             | **               | Z80 ASSEMBLER SB-7201   | <A>               | PAGE 01 | ??/??/?? |                     | This message is output at the top of each page.                                  |
| 01          | 0000             |                         |                   |         |          |                     | ;                                                                                |
| 02          | 0000             |                         |                   |         |          |                     | ; ASSEMBLER LIST SAMPLE                                                          |
| 03          | 0000             |                         |                   |         |          |                     | ;                                                                                |
| 04          | 0000             | P                       |                   | LETNL:  | EQU      | 0762H               |                                                                                  |
| 05          | 0000             | P                       |                   | MSG:    | EQU      | 06B3H               |                                                                                  |
| 06          | 0000             |                         |                   |         |          |                     | ;                                                                                |
| 07          | 0000             |                         |                   | START:  | ENT      |                     | ; ENTRY FROM UNIT#1                                                              |
| 08          | 0000             |                         |                   | MAIN:   | ENT      |                     | ; ENTRY FROM UNIT#2                                                              |
| 09          | 0000             | 310000                  |                   |         | LD       | SP, START           | ; INITIAL STACK POINTER                                                          |
| 10          | 0003             | 210000                  | E                 |         | LD       | HL, TEMPO           |                                                                                  |
| 11          | 0006             | DD210000                | E                 |         | LD       | IX, TEMP1           |                                                                                  |
| 12          | 000A             | DD360000                | EE                | MAIN0:  | LD       | (IX+CONST0), CONST1 |                                                                                  |
| 13          | 000E             | 00                      | Q                 |         | XOA      | A                   | ; A<--00                                                                         |
| :           | :                | :                       | :                 | :       | :        | :                   | :                                                                                |
| 47          | 005A             | 1A                      |                   | MAIN7:  | LD       | A, (DE)             |                                                                                  |
| 48          | 005B             | B7                      |                   |         | OR       | A                   |                                                                                  |
| 49          | 005C             | 2000                    | V                 |         | JR       | NZ, COMP            |                                                                                  |
| 50          | 005E             | EB                      |                   | MAIN8:  | EX       | DE, HL              | ; EXCHANGE DE, HL                                                                |
|             | **               | Z80 ASSEMBLER SB-7201   | <A>               | PAGE 02 | ??/??/?? |                     | A new page is started when the number of lines on the preceding page reaches 60. |

Errors detected during assembly and definition conditions are indicated with assembler messages.

## —Definition Condition Messages—

### E (External)

This message indicates that an external symbol reference is being made; i.e., the label symbol by the operand is not defined in the assembly source program unit assembled.

The label symbol indicated must be defined as a global symbol in another assembly program unit for linkage with the current unit by the linker. (See "Assembler Directive ENT" on page 10.)

An undefined byte of data is treated as "00"; 2 undefined bytes of data (or an address) are uncertain.

Example:

|    |      |                |                                                               |
|----|------|----------------|---------------------------------------------------------------|
| E  | LD   | B, CONST0      |                                                               |
| └─ |      |                | The byte of data "CONST0" is not defined in the program unit. |
| E  | CALL | SORT           |                                                               |
| └─ |      |                | Address SORT is not defined in the program unit.              |
| EE | BIT  | TOP, (IY+FLAG) |                                                               |
| └─ |      |                | The byte of data "FLAG" is not defined in the program unit.   |
| └─ |      |                | The byte of data "TOP" is not defined in the program unit.    |

### P (Phase)

This message indicates that the label symbol is defined by an EQU statement with a constant value assigned. A label symbol indicated by this message can be referenced from an external file. In this case, however, the program unit including the EQU statement must be loaded before the other program units which are to be linked with it.

The P message is displayed when a label symbol different from those stored in the symbol table during PASS 1 is found.

Example:

|    |        |     |       |                                                    |
|----|--------|-----|-------|----------------------------------------------------|
| P  | LETNL: | EQU | 0762H |                                                    |
| P  | DATA1: | EQU | 3     |                                                    |
| └─ |        |     |       | Indicates that LETNL and DATA1 are defined by EQU. |

The P message is displayed in the relocatable binary code column rather than in the assembler message column.

## —Error Messages—

### C (illegal Character error)

This message indicates that an illegal character has been used as an operand.

Example: C JP +1000-3

### F (Format error)

This message indicates that the instruction format is incorrect.

### N (Non label error)

This message indicates that ENT or EQU has no label symbol.

Example:

|   |              |     |       |                 |
|---|--------------|-----|-------|-----------------|
| N |              | EQU | 0012H |                 |
|   | └──────────┘ |     |       | No label symbol |

## L (erroneous Label error)

This message indicates that an illegal label symbol is used.

Example: L JR XYZ

↑ XYZ is not defined in the current source program.

No externally defined global symbol can be used as an operand of the JR or DJNZ commands.  
The L message is displayed if such a label symbol is specified.

## M (Multiple label error)

This message indicates that a label symbol is defined two or more times.

Example: M ABC: LD DE, BUFFER

}

M ABC: ENT

↑ Indicates that ABC is defined more than once.

## O (erroneous Operand)

This message indicates that an illegal operand has been specified.

## Q (Questionable mnemonic)

This message indicates that a mnemonic code is incorrect.

Example: Q CAL XYZ

CALL XYZ is correct.

Q PSH B

PUSH BC is correct.

## S (String error)

This message indicates that single quotation mark(s) are omitted from a DEFM statement.

Example: S DEFM GAME OVER

DEFM 'GAME OVER' is correct.

## V (Value over)

This message indicates that the value of the operand is out of the prescribed range.

Example: V LD A, FF8H

V SET 8, A

V JR -130

# ASSEMBLER DIRECTIVES

Assembler directives (also sometimes referred to as "pseudo instructions") control assembly, but are not converted into machine language. However, in the DEFB, DEFW and DEFM directives, their operands are sometimes converted into machine language.

## —ENT (entry)—

This assembler directive defines a label symbol as a global symbol. Label symbols which are referenced by two or more programs when multiple programs are linked must be defined by the entry directive.

Label symbols defined by the entry directive are included in the relocatable file so that the linker can identify them. The symbolic debugger can perform symbolic addressing using these label symbols.

Label symbols which are not defined by the entry directive contribute only to assembly of the current source program unit, and are not included in the relocatable file output by the assembler. However, labels defined by the EQU directive are exceptions since they are defined as global symbols and entry definition is not necessary.

The example below shows label symbols being referenced between program units GAUSS-MAIN and GAUSS-SR. The E message in the assembler message column indicates that a label symbol which is not defined in the current program unit is being referenced externally.

Program unit 1  
"GAUSS-MAIN"

|                   |        |            |                                                                            |
|-------------------|--------|------------|----------------------------------------------------------------------------|
|                   | ;      | GAUSS-MAIN |                                                                            |
|                   | ;      |            |                                                                            |
|                   | MAIN0: | ENT        | ← Entry definition of label symbol<br>MAIN0                                |
|                   |        | :          |                                                                            |
| Address undefined |        | CALL       | CMPLX                                                                      |
| CD0000 E          |        | :          |                                                                            |
| E message         |        | CALL       | CMPLX+2                                                                    |
|                   |        | :          | ← No offset can be added to a label symbol<br>which is defined externally. |
|                   |        | END        | ← END is always required at the end of a<br>program unit.                  |

Program unit 2  
"GAUSS-SR"

|                   |        |          |                                             |
|-------------------|--------|----------|---------------------------------------------|
|                   | ;      | GAUSS-SR |                                             |
|                   | ;      |          |                                             |
|                   | CMPLX: | ENT      | ← Entry definition of label symbol<br>CMPLX |
|                   |        | :        |                                             |
| Address undefined |        | RET      |                                             |
| C30000 E          |        | :        |                                             |
| E message         |        | JP       | MAIN0                                       |
|                   |        | :        |                                             |
|                   |        | END      |                                             |

## —EQU (equate)—

This assembler directive defines a label symbol with a numeric value (or address) assigned. The numeric value must be a decimal or hexadecimal constant. Any numeric value can be added to or subtracted from a label symbol once it is defined with a numeric value assigned; this allows a new label symbol to be defined.

The label symbol used as an address in the operand is generally treated as a relative address. However, when a specific address is assigned to the label symbol with an EQU directive, the address is not changed during assembly.

The EQU directive also defines a label symbol as a global symbol. A label defined by the EQU directive can be referenced by an external program unit. **However, program units including such directives must be loaded before other program units to be linked.**

The following example illustrates use of the EQU directive to define label symbols as monitor subroutine addresses and I/O port numbers for a specific device. The P messages indicate that the EQU directives define the label symbols as global symbols.

```

** Z80 ASSEMBLER SB-7201 <A> PAGE 01 ??/??/??

01 0000 ;
02 0000 ; MONITOR SUBROUTINE
03 0000 ;
04 0000 P BRKEY: EQU 0527H
05 0000 P GETKY: EQU 0610H
06 0000 P PRNTS: EQU 063AH
07 0000 P PRNT: EQU 063CH
08 0000 P MSG: EQU 06B5H
09 0000 P NL: EQU 0757H
10 0000 P LETNL: EQU 0764H
11 0000 P GETL: EQU 0BE5H
12 0000 SKP 3

16 0000 ;
17 0000 ; SET PORT#: PRINTER
18 0000 ;
19 0000 P POTFE: EQU FEH
20 0000 P POTFF: EQU POTFE+1 ; POTFF is defined with FF (hexadecimal)
21 0000 ; assigned.
22 0000 P CON1: EQU 1
23 0000 P CON2: EQU 2
24 0000 P CON3: EQU CON1+CON2 ; This results in assigned of 3 to CON 3. In this
 case, CON1 and CON2 must be defined in
 advance.
```

## —ORG (origin)—

This assembler directive determines the object program loading address. For example, when

```
ORG 2000H
```

is placed at the beginning of the program to be assembled, the assembler assembles the program with a loading address of 2000H specified.

When a relocatable binary file generated with the loading address specified with the ORG directive is linked with other programs by the linker, the loading address specified with the ORG directive is effective and that specified with the linker is not.

When relocatable files with loading addresses specified with ORG directives are linked, or when more than one ORG directives is used in a program, the loading addresses specified must not overlap and must appear in the sequential order.

When a relocatable file with a loading address specified with an ORG assembler directive is converted into a system file using the LINK / S command, the specified loading address is ignored.

```

** Z80 ASSEMBLER SB-7201 <ORG> PAGE 01 ??/??/?
01 0000 ; TYPE COMMAND
02 0000 ;
03 2000 ORG 2000H
04 2000 .TYPE: ENT
05 2000 116220 LD DE,SWTBL ; DE: = SWITCH TABLE
06 2003 CD0000 E CALL ?GSW ; CHECK GLOBAL SWITCH
07 2006 D8 RET C
08 2007 CD0000 E CALL C&L1 ; SELECT CRT OR LPT
09 200A CD0000 E CALL ?SEP ; CHECK SEPARATOR
10 200D D8 RET C
11 200E FE2C CP 2CH ; SEPARATER = " , " ?
12 2010 3E03 LD A,3 ; 3 IS ERR CODE
13 2012 37 SCF
14 2013 C0 RET NZ ; NO, ERR RETURN
15 2014 CD0000 E TYPE0: CALL ?LSW ; CHECK LOCAL SWITCH
16 2017 D8 REC C
17 2018 3E08 LD A,8 ; 8 IS ERR CODE
18 201A 37 SCF
19 201B C0 RET NZ ; ERROR, LSW EXIST
20 201C 0E80 LD C,128 ; LU#: = 128
21 201E D9 EXX
22 201F 0604 LD B,4 ; DEFAULT MODE = ASC
23 2021 D9 EXX
 : : :
55 2062 88 SWTBL: DEFB 88H ; /P
56 2063 FF DEFB FFH ; END OF SWTBL
57 2064 BUFFER: DEFS 128 ; 128 BYTE BUFFER
58 20E4 END

```

```

** Z80 ASSEMBLER SB-7021 <ORG> PAGE 02 ??/??/?
.TYPE 2000 BUFFER 2064 SWTBL 2062 TYPE0 2014 TYPE10 203C
TYPE20 2048 TYPEER 2058

```

## —DEFB n (define byte)—

This directive sets constant n (1 byte) in the address of the line on which the directive is specified. A label symbol defined with a constant (1 byte) assigned may be used in place of n.

This directive (as well as DEFW and DWFM) is used to form message data or a graphic data group for a code conversion table or other table.

The following example forms the message "ERROR" in ASCII code. Since it uses 0DH as an end mark, monitor subroutine 06B3H can be used to output the message.

```
13 1FF3 B7 OR A
14 1FF4 CA0000 E JP Z, READY
15 1FF7 110020 LD DE, MESGO
16 1FFA CDB506 CALL MSG
17 1FFD C30000 E JP MAIN2
18 2000 P MSG: EQU 06B5H
19 2000 ;
20 2000 ; MESSAGE GROUP
21 2000 ;
22 2000 MESGO: ENT ; "ERROR"
23 2000 45 DEFB 45H
24 2001 52 DEFB 52H
25 2002 52 DEFB 52H
26 2003 4F DEFB 4FH
27 2004 52 DEFB 52H
28 2005 0D DEFB 0DH
```

## —DEFB 'S', DEFB "S" (define byte)—

This directive sets the ASCII code corresponding to the character enclosed in single or double quotation marks in the address of the line on which the directive is specified.

Since this directive converts characters to ASCII code, the above example can be rewritten as follows.

```
21 2000 MESGO: ENT ; "ERROR"
22 2000 45 DEFB 'E '
23 2001 52 DEFB 'R '
24 2002 52 DEFB 'R '
25 2003 4F DEFB 'O '
26 2004 52 DEFB 'R '
27 2005 0D DEFB 0DH
28 2006 06 DEFB ' '
29 2007 03 DEFB ' '
30 2008 0D DEFB 0DH
31 2009 27 DEFB " "
32 200A 22 DEFB " "
```

## —DEFW nn' (define word)—

This directive sets n' in the address of the line on which the directive is specified and n in the following address; in other words, it sets two bytes of data. A label symbol may be used in place of nn'.

```
39 5FF1 CMDT: ENT ; COMMAND TABLE
40 5FF1 41 DEFB 41H
41 5FF2 0053 DEFW CMDA
42 5FF4 42 DEFB 42H
43 5FF5 1E53 DEFW CMDB+3
44 5FF7 53 DEFB 53H
45 5FF8 0000 E DEFW CMDS
46 5FFA 0D DEFB 0DH
47 5FFB CONST0: ENT
48 5FFB 0F01 DEFW 010FH
49 5FFD CONST1: ENT
50 5FFD 660D DEFW 0D66H
```

## —DEFM 'S', DEFM "S" (define message)—

This directive sets the character string enclosed in single or double quotation marks in ASCII code in addresses starting at that of the line on which the directive is specified. The number of characters must be within the range from 1 to 64. On the assembly listing, codes for 4 characters are output on each line.

The example on the preceding page can be written as follows with this directive.

```
21 2000 MESGO: ENT ; "ERROR"
22 2000 4552524F DEFM ' ERROR '
23 2004 52
24 2005 0D DEFB 0DH
25 2006 06034142 DEFM ' AB '
26 200A 0D DEFB 0DH
27 200B 41274247 DEFM " A' B' C' "
28 200F 4327
29 2011 0D DEFB 0DH
```

## —DEFS nn' (define storage)—

This directive reserves nn' bytes of memory area starting at the address of the line on which the directive is specified.

This directive adds nn' to the reference counter contents; the contents of addresses skipped are not defined.

The following example reserves buffer areas.

|    |      |         |      |     |            |
|----|------|---------|------|-----|------------|
| 02 | 4BB8 | TEMPO:  | ENT  |     | ; BUFFER A |
| 03 | 4BB8 |         | DEFS | 1   |            |
| 04 | 4BB9 | TEMP1:  | ENT  |     | ; BUFFER B |
| 05 | 4BB9 |         | DEFS | 2   |            |
| 06 | 4BBB | TEMP2:  | ENT  |     | ; BUFFER C |
| 07 | 4BBB |         | DEFS | 2   |            |
| 08 | 4BBD | TEMP3:  | ENT  |     | ; BUFFER D |
| 09 | 4BBD |         | DEFS | 128 |            |
| 10 | 4C3D | BFFR:   | ENT  |     | ; BUFFER E |
| 11 | 4C3D |         | DEFS | A   |            |
| 12 | 4C47 | BUFFER: | ENT  |     | ; BUFFER F |
| 13 | 4C47 |         | DEFS | 2   |            |

The addresses are increased by amounts corresponding to the values indicated by the respective DEFS statements.

## —SKP n (skip n lines)—

This directive advances the assembly listing by n lines to make the list easy to read.

```
30 COMMON: ENT ; NORMAL RETURN
31 3BB8 AF XOR A ; A ← -00
32 3BB9 32B84B LD (TEMPO), A ; CLEAR CMD BUFFER
33 3BBC 110020 LD DE, MESGO ; "READY"
34 3BBF C9 RET
35 3BC0 SKP 3
 } 3 line feeds are made.

39 3BC0 ;
40 3BC0 ; ABNORMAL RETURN
41 3BC0 ;
42 3BC0 ABNRET: ENT ; SET INVALID MODE
```

## —SKP H (skip home)—

This directive advances the page during output of the assembly listing.

## —END (end)—

This directive declares the end of the source program. All source programs must be ended with this directive. Assembly operation is not completed if this directive is omitted.

The assembly outputs

END?

when it reads a source file which doesn't include an END directive.

# MESSAGE TABLE

| Definition status message | Meaning                                                                                                                                                                   | Example                                                                                                                                                                                                                                      |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>E (External)</b>       | Indicates that a label symbol is being referenced externally; that is, the label is not defined in the current source program unit.                                       | <p>E LD B, CONST0<br/> ↑—The data byte "CONST0" is undefined.</p> <p>E CALL SORT<br/> ↑—The address "SORT" is undefined.</p> <p>EE BIT TOP, (IY+FLAG)<br/> ↑—The data byte "FLAG" is undefined.<br/> ↑—The data byte "TOP" is undefined.</p> |
| <b>P (Phase)</b>          | Defines a label symbol with a constant assigned.<br>This message is also output when a label symbol is encountered during pass 2 which was not encountered during pass 1. | <p>P LETNL : EQU 0762H</p> <p>P DATA1 : EQU 3<br/> ↑—LETNL and DATA1 are defined by EQU.<br/> The P message is displayed in the relocatable binary code column rather than in the assembler message column.</p>                              |

| Error message                      | Meaning                                                                 | Example                                                                                                                                                                                                                              |
|------------------------------------|-------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>C (illegal Character error)</b> | Indicates that an illegal character is used in the operand.             | C JP +1000-3                                                                                                                                                                                                                         |
| <b>F (Format error)</b>            | Indicates that the instruction format is incorrect.                     |                                                                                                                                                                                                                                      |
| <b>N (Non label error)</b>         | Indicates that no label symbol is specified for ENT or EQU.             | <p>N EQU 0012H<br/> ↑—No label symbol</p>                                                                                                                                                                                            |
| <b>L (erroneous Label error)</b>   | Indicates that an illegal label symbol is used.                         | <p>L JR XYZ<br/> ↑—XYZ is not defined in the current program.<br/> No externally defined global symbol can be used as the operand of a JR or DJNZ command.<br/> If such a label symbol is specified, the L message is displayed.</p> |
| <b>M (Multiple label error)</b>    | Indicates that a label symbol is defined two or more times.             | <p>M ABC : LD DE, BUFFER<br/> ?</p> <p>M ABC : ENT<br/> ↑—ABC is defined twice.</p>                                                                                                                                                  |
| <b>O (erroneous Operand)</b>       | Indicates that an illegal operand is specified.                         |                                                                                                                                                                                                                                      |
| <b>Q (Questionable mnemonic)</b>   | Indicates that the mnemonic code is incorrect.                          | <p>Q CAL XYZ<br/> CALL XYZ is correct.</p>                                                                                                                                                                                           |
| <b>S (String error)</b>            | Indicates that single or double quotation mark(s) are omitted.          | <p>S DEFM GAME OVER<br/> DEFM 'GAME OVER' is correct.</p>                                                                                                                                                                            |
| <b>V (Value over)</b>              | Indicates that the value of the operand is out of the prescribed range. | <p>V LD A, FF8H V SET 8, A<br/> V JR -130</p>                                                                                                                                                                                        |
| <b>END?</b>                        | Indicates that the END directive is missing from the source program.    |                                                                                                                                                                                                                                      |

Note: Refer to the System Error Messages in the System Command manual for other system errors.



# Personal Computer **MZ-80B**

Service Manual

**SHARP**

### NOTICE

The MZ-80 series of sophisticated personal computers is manufactured by the SHARP CORPORATION. Hardware and software specifications are subject to change without prior notice; therefore, you are requested to pay special attention to version numbers of the monitor and the system software (supplied in the form of cassette tape or mini-floppy disk files).

This manual is for reference only and the SHARP CORPORATION will not be responsible for difficulties arising out of inconsistencies caused by version changes, typographical errors or omissions in the descriptions.

This manual is based on the SB-1500 series monitor and the SB-7000 series FDOS.

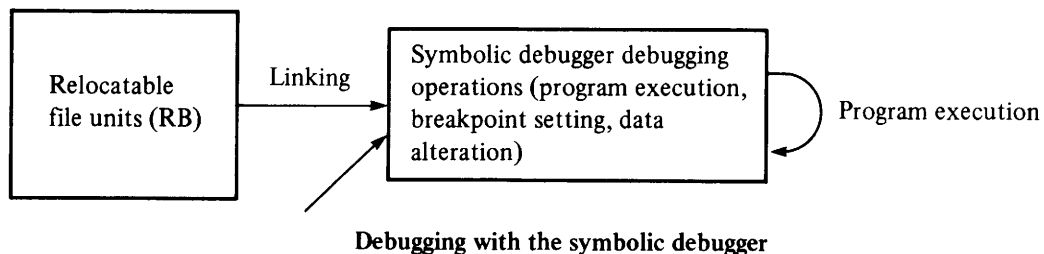
— CONTENTS —

|                                              |    |
|----------------------------------------------|----|
| <b>INTRODUCTION</b> .....                    | 1  |
| Starting the Symbolic Debugger .....         | 2  |
| <b>SYMBOLIC DEBUGGER COMMAND TABLE</b> ..... | 3  |
| <b>BREAKPOINTS</b> .....                     | 4  |
| <b>USING THE DEBUGGER COMMANDS</b> .....     | 5  |
| T (Table dump) Command .....                 | 5  |
| Link message examples .....                  | 6  |
| B (Breakpoint) Command .....                 | 7  |
| & (Clear B.P) Command .....                  | 9  |
| M (Memory dump) Command .....                | 10 |
| D (Memory list dump) Command .....           | 11 |
| W (Data wrtie) Command .....                 | 12 |
| G (Goto) Command .....                       | 13 |
| I (Indicative start) Command .....           | 14 |
| A (Accumulator) Command .....                | 15 |
| C (Complementary) Command .....              | 15 |
| P (Program counter) Command .....            | 16 |
| R (Register) Command .....                   | 16 |
| Using register commands A, C, P and R .....  | 17 |
| X (Data transfer) Command .....              | 18 |
| S (Save) Command .....                       | 19 |
| Y (Yank) Command .....                       | 20 |
| \ (FDOS) Command .....                       | 21 |
| # Command .....                              | 21 |
| ! Command .....                              | 21 |
| <b>ERROR MESSAGES</b> .....                  | 22 |



# INTRODUCTION

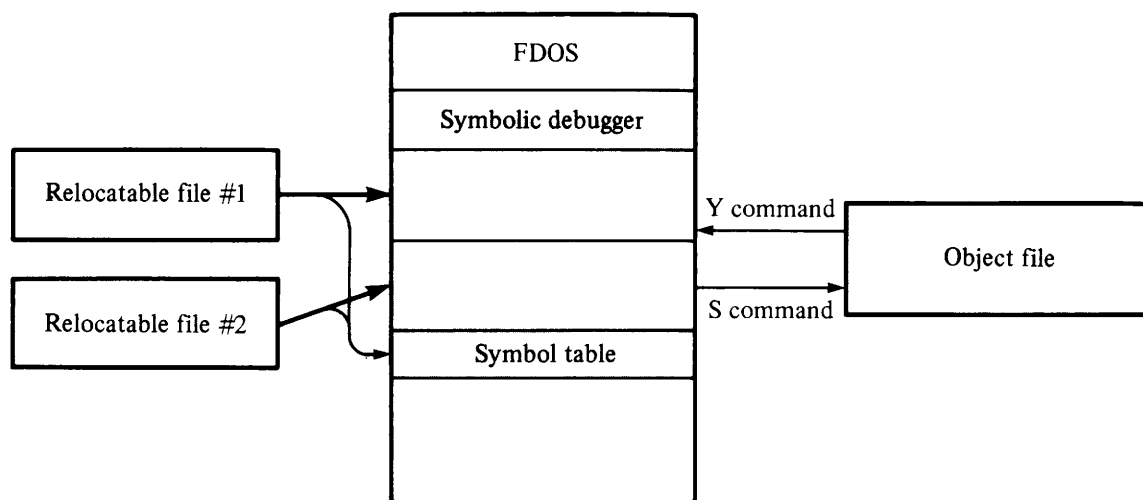
The SHARP MZ-80B symbolic debugger links and loads one or more program units from relocatable files to form an object program in memory in an immediately executable form and runs the object program for debugging. It provides the programmer with facilities for taking a memory dump of the object program in the link area, for setting a breakpoint in the program, for displaying and altering the contents of the CPU internal registers and for starting execution of the program at a given address with the CPU internal registers set to specified values (indicative start).



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 relative addresses in relocatable programs and offset values specified when they are loaded.

In normal program development process, the programmer debugs each object program unit with the symbolic debugger and, if he finds errors, he reedits its source program and reassembles it. After debugging all object program units, the programmer links and loads them with the linker to form the final object program.

Symbolic debugger commands are summarized in the table on page 3. Commands marked with a dagger permit symbolic operations. The debugger creates the symbol table in the same way as the linker.



Symbolic debugger file processing

## —Starting the Symbolic Debugger—

The symbolic debugger is started by entering one of the commands below in the FDOS command mode.

### 1. **DEBUG** CR

The debugger is invoked and the debugger command wait state entered.

### 2. **DEBUG** [filename 1, ..... filename N] CR

The debugger links and loads program units from relocatable files filename 1 through filename N and waits for entry of a debugger command.

### 3. **DEBUG/P ABC** CR

The debugger loads the program unit from file ABC.RB and prints the link information shown in Figure 1 on the printer.

### 4. **DEBUG/P/T ABC** CR

The debugger loads the program unit from file ABC.RB and prints the link and symbol table information on the printer.

### 5. **DEBUG ABC, XYZ, TBL\$20** CR

The debugger links and loads program units from relocatable files ABC.RB and XYZ.RB and waits for entry of a debugger command. It also reserves 2000 (hex) bytes (approximately 8K bytes) of space for the symbol table. Approximately 6K bytes of space are reserved when the table size is not specified.

### 6. **DEBUG ABC, \$1000, XYZ, DEF/O** CR

The debugger links and loads program units from relocatable files ABC.RB and XYZ.RB to generate an object program in object program file DEF.OBJ, then waits for entry of a debugger command. It reserves 4K bytes of free space (offset of 1000 (hex)) between program units ABC and XYZ.

**Note:** When the debugger is invoked and the command wait state entered, all files (including those specified in the **DEBUG** command) are killed.

```
Linking ABC.RB
 Top asm.bias $6100
 End asm.bias $613A

KEYIN 6138 U
Debugger area 6100-6139
```

**Figure 1.**

```
Linking ABC.RB
 Top asm.bias $6100
 End asm.bias $613A

Symbol table
CLEAR 6125 DRINO D FFEC KEYIN 6138 U MTFG D FFE6
START 6100
Debugger area 6100-6139
```

**Figure 2.**

# SYMBOLIC DEBUGGER COMMAND TABLE

| Command type         | Command name   | Function                                                                                                                                                  |
|----------------------|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Symbol table command | T              | Displays the contents of the symbol table; i.e., the label symbol name, its absolute address and the definition status for each table entry. (Table Dump) |
| Debugging commands   | B <sup>†</sup> | Displays, sets or alters a breakpoint. (Breakpoint)                                                                                                       |
|                      | &              | Clears all breakpoints set. (Clear Breakpoints)                                                                                                           |
|                      | M <sup>†</sup> | Displays the contents of the specified block in the link area in hexadecimal representation or alters them. (Memory Dump)                                 |
|                      | D <sup>†</sup> | Displays the contents of the specified block in the link area in hexadecimal representation with one instruction on a line. (Memory List Dump)            |
|                      | W <sup>†</sup> | Writes hexadecimal data, starting at the specified address in the link area. (Write)                                                                      |
|                      | G <sup>†</sup> | Executes the program at the specified address. (GOTO)                                                                                                     |
|                      | I              | Executes the program at the address designated by PC with the register buffer data set to the CPU internal registers. (Indicative Start)                  |
|                      | 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 I/O commands    | S              | Saves the object program in the link area in an output file with the specified name. (Save)                                                               |
|                      | Y              | Reads the object program from the object file with the specified file name into memory. (Yank)                                                            |
| Special commands     | \              | Executes the specified FDOS built-in command.                                                                                                             |
|                      | #              | Switches the printer list mode for listing printout.                                                                                                      |
|                      | !              | Transfers control to FDOS.                                                                                                                                |

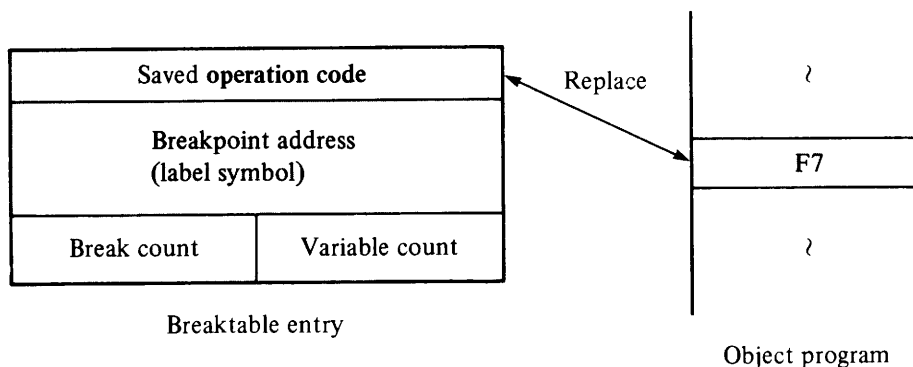
Note: Commands marked by a dagger permit symbolic operations.

# 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 F7. The debugger creates one breaktable entry for each breakpoint as shown below.



Hexadecimal code F7 is the operation code for RST 6, which initiates a break operation. When the RST 6 instruction, which is a 1-byte CALL instruction, is executed, the contents of the program counter are pushed into the stack and the program counter is loaded with new data 0030H; that is, program control jumps to address 0030H in the monitor, from which point control is immediately passed to the debugger. The debugger searches the breaktable for the pertinent breakpoint. If the breakpoint is not found, the debugger displays error message "RST6?." Thus, the RST 6 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 corresponding count and decrements the variable count (this count is initially set to the break count) by one. If the variable count reaches zero, the debugger performs break processing; otherwise, it continues program execution.

## USING THE DEBUGGER COMMANDS

### —T (Table dump) Command—

The T command displays the contents of the symbol table, that is, the label symbol name, its absolute address and its definition status.

|      |                                            |
|------|--------------------------------------------|
| * DT | Displays the contents of the symbol table. |
|------|--------------------------------------------|

- Enter a T command in response to the prompt "\*D".
- The debugger displays the label symbol name, its absolute address (in hexadecimal) and the definition status for each symbol table entry. The programmer can detect symbol definition errors by checking the definition status of the displayed label symbols.
- Messages pertaining to the symbol table definition status are identical to those issued by the linker. The definition status messages are listed below, followed by examples.
- Two symbol table entries are displayed on a line when the number of characters per line is set to 40 and four entries are displayed on a line when it is set to 80.

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

No message is attached to symbols for which an address has been defined.

U, M, X and H indicate error conditions.

## Link 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  |            |

#### "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 statement.

Note: The EQU statement 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 statement.

#### "COUNT1" D

COUNT1 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).

### 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  |            |

#### "TMDLYL" U

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

### Third program unit loaded (UNIT-#3)

|         |      |           |
|---------|------|-----------|
| INPUT:  | CALL | 001BH     |
|         | CALL | TMDLYL    |
|         | CALL | 001BH     |
|         | LD   | HL, START |
|         | CP   | 0DH       |
|         | JR   | Z, END    |
|         | LD   | (HL), A   |
|         | INC  | HL        |
|         | JR   | INPUT     |
| END:    | JP   | 0000H     |
| COUNT2: | EQU  | 12        |
|         | END  |           |

## —B (Breakpoint) Command—

The B command sets or changes a breakpoint. A breakpoint occurs after instructions immediately preceding the breakpoint are executed the number of times specified in the break counter. When a breakpoint is taken, 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. The programmer can specify the breakpoint with either an absolute hexadecimal address or a label symbol (the label symbol can be given a displacement of from -65535 to 65535 in decimal).

|              |                                                                                                                                              |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| * DB         | Sets a breakpoint.                                                                                                                           |
| addr count   |                                                                                                                                              |
| 1 7530 2     | The breakpoint is address 7530 and the break count is 2.                                                                                     |
| 2 SORT3 1    | The breakpoint is the address represented by 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 the break count must be separated by at least one blank (denoted by ) .)                                                 |

- Enter the B command in response to the prompt "\*D".
- 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 a 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, enter an address followed by a space and a break count. The break count specifies the number of allowable passes through the breakpoint before a break actually occurs. **The programmer can specify a hexadecimal value from 1 to E.**

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

- When a label symbol is entered as a breakpoint address, the debugger displays message "???" and waits for a new command if the pertinent symbol is not defined or if the symbol is a data defining symbol.
- No breakpoint can be specified for the DJNZ instruction. When a breakpoint is specified for the DJNZ instruction, the debugger displays message "DJNZ?" and waits for entry of a new command.
- No breakpoint can be specified for the CALL instruction either. Breakpoints cannot be specified for any instructions which push the program counter contents into the stack. The debugger will display the message "CALL?" if such an attempt is made.

To check a CALL instruction, set a breakpoint at the beginning of the called routine.

- **To clear a previously set breakpoint**, enter that breakpoint address with a break count of 0 (use the & command to clear all breakpoints).

The debugger displays message "???" and waits for a command when an attempt is made to clear an undefined breakpoint.

- **The programmer can specify a maximum of nine breakpoints.** When the programmer specifies nine breakpoints, the debugger displays "X" on the next line instead of the next breakpoint number. This requests the programmer to clear a breakpoint or change a break count, not to set a new breakpoint. If the programmer attempts to set a new breakpoint, the debugger will not accept it and prompts for a new command with message "Over".
- When a B command is entered after breakpoints are set, the debugger displays them; in this case, the hexadecimal address is displayed first, followed by the break count format.
- The programmer can use the DEL key while setting breakpoints. When the CR key is pressed, the debugger is returned to the command wait state.

## —& (Clear B.P) Command—

\* D&

Clears all the breakpoints which have been set.

- Enter the & command in response to the prompt "\* D".
- The debugger clears all breakpoints set and waits for entry of a new command.

— The photo at right shows an example of setting breakpoints. The breakpoints are set with a 4-digit hexadecimal number (absolute address), a global label symbol, a label symbol plus a line specification and a label symbol plus a byte displacement.

```

*DB
addr count
1 3000 1
2 3002 2
3 3004 2
4 3006 2
5 3008 1
6 300A 2
7 300C 1
8 300E 2
9 3010 1
10 3012 2
11 3014 1
12 3016 2
13 3018 1
14 301A 2
15 301C 1
16 301E 2
17 3020 1
18 3022 2
19 3024 1
20 3026 2
21 3028 1
22 302A 2
23 302C 1
24 302E 2
25 3030 1
26 3032 2
27 3034 1
28 3036 2
29 3038 1
30 303A 2
31 303C 1
32 303E 2
33 3040 1
34 3042 2
35 3044 1
36 3046 2
37 3048 1
38 304A 2
39 304C 1
40 304E 2
41 3050 1
42 3052 2
43 3054 1
44 3056 2
45 3058 1
46 305A 2
47 305C 1
48 305E 2
49 3060 1
50 3062 2
51 3064 1
52 3066 2
53 3068 1
54 306A 2
55 306C 1
56 306E 2
57 3070 1
58 3072 2
59 3074 1
60 3076 2
61 3078 1
62 307A 2
63 307C 1
64 307E 2
65 3080 1
66 3082 2
67 3084 1
68 3086 2
69 3088 1
70 308A 2
71 308C 1
72 308E 2
73 3090 1
74 3092 2
75 3094 1
76 3096 2
77 3098 1
78 309A 2
79 309C 1
80 309E 2
81 30A0 1
82 30A2 2
83 30A4 1
84 30A6 2
85 30A8 1
86 30AA 2
87 30AC 1
88 30AE 2
89 30B0 1
90 30B2 2
91 30B4 1
92 30B6 2
93 30B8 1
94 30BA 2
95 30BC 1
96 30BE 2
97 30C0 1
98 30C2 2
99 30C4 1
100 30C6 2
101 30C8 1
102 30CA 2
103 30CC 1
104 30CE 2
105 30D0 1
106 30D2 2
107 30D4 1
108 30D6 2
109 30D8 1
110 30DA 2
111 30DC 1
112 30DE 2
113 30E0 1
114 30E2 2
115 30E4 1
116 30E6 2
117 30E8 1
118 30EA 2
119 30EC 1
120 30EE 2
121 30F0 1
122 30F2 2
123 30F4 1
124 30F6 2
125 30F8 1
126 30FA 2
127 30FC 1
128 30FE 2
129 3100 1
130 3102 2
131 3104 1
132 3106 2
133 3108 1
134 310A 2
135 310C 1
136 310E 2
137 3110 1
138 3112 2
139 3114 1
140 3116 2
141 3118 1
142 311A 2
143 311C 1
144 311E 2
145 3120 1
146 3122 2
147 3124 1
148 3126 2
149 3128 1
150 312A 2
151 312C 1
152 312E 2
153 3130 1
154 3132 2
155 3134 1
156 3136 2
157 3138 1
158 313A 2
159 313C 1
160 313E 2
161 3140 1
162 3142 2
163 3144 1
164 3146 2
165 3148 1
166 314A 2
167 314C 1
168 314E 2
169 3150 1
170 3152 2
171 3154 1
172 3156 2
173 3158 1
174 315A 2
175 315C 1
176 315E 2
177 3160 1
178 3162 2
179 3164 1
180 3166 2
181 3168 1
182 316A 2
183 316C 1
184 316E 2
185 3170 1
186 3172 2
187 3174 1
188 3176 2
189 3178 1
190 317A 2
191 317C 1
192 317E 2
193 3180 1
194 3182 2
195 3184 1
196 3186 2
197 3188 1
198 318A 2
199 318C 1
200 318E 2
201 3190 1
202 3192 2
203 3194 1
204 3196 2
205 3198 1
206 319A 2
207 319C 1
208 319E 2
209 31A0 1
210 31A2 2
211 31A4 1
212 31A6 2
213 31A8 1
214 31AA 2
215 31AC 1
216 31AE 2
217 31B0 1
218 31B2 2
219 31B4 1
220 31B6 2
221 31B8 1
222 31BA 2
223 31BC 1
224 31BE 2
225 31C0 1
226 31C2 2
227 31C4 1
228 31C6 2
229 31C8 1
230 31CA 2
231 31CC 1
232 31CE 2
233 31D0 1
234 31D2 2
235 31D4 1
236 31D6 2
237 31D8 1
238 31DA 2
239 31DC 1
240 31DE 2
241 31E0 1
242 31E2 2
243 31E4 1
244 31E6 2
245 31E8 1
246 31EA 2
247 31EC 1
248 31EE 2
249 31F0 1
250 31F2 2
251 31F4 1
252 31F6 2
253 31F8 1
254 31FA 2
255 31FC 1
256 31FE 2
257 3200 1
258 3202 2
259 3204 1
260 3206 2
261 3208 1
262 320A 2
263 320C 1
264 320E 2
265 3210 1
266 3212 2
267 3214 1
268 3216 2
269 3218 1
270 321A 2
271 321C 1
272 321E 2
273 3220 1
274 3222 2
275 3224 1
276 3226 2
277 3228 1
278 322A 2
279 322C 1
280 322E 2
281 3230 1
282 3232 2
283 3234 1
284 3236 2
285 3238 1
286 323A 2
287 323C 1
288 323E 2
289 3240 1
290 3242 2
291 3244 1
292 3246 2
293 3248 1
294 324A 2
295 324C 1
296 324E 2
297 3250 1
298 3252 2
299 3254 1
300 3256 2
301 3258 1
302 325A 2
303 325C 1
304 325E 2
305 3260 1
306 3262 2
307 3264 1
308 3266 2
309 3268 1
310 326A 2
311 326C 1
312 326E 2
313 3270 1
314 3272 2
315 3274 1
316 3276 2
317 3278 1
318 327A 2
319 327C 1
320 327E 2
321 3280 1
322 3282 2
323 3284 1
324 3286 2
325 3288 1
326 328A 2
327 328C 1
328 328E 2
329 3290 1
330 3292 2
331 3294 1
332 3296 2
333 3298 1
334 329A 2
335 329C 1
336 329E 2
337 32A0 1
338 32A2 2
339 32A4 1
340 32A6 2
341 32A8 1
342 32AA 2
343 32AC 1
344 32AE 2
345 32B0 1
346 32B2 2
347 32B4 1
348 32B6 2
349 32B8 1
350 32BA 2
351 32BC 1
352 32BE 2
353 32C0 1
354 32C2 2
355 32C4 1
356 32C6 2
357 32C8 1
358 32CA 2
359 32CC 1
360 32CE 2
361 32D0 1
362 32D2 2
363 32D4 1
364 32D6 2
365 32D8 1
366 32DA 2
367 32DC 1
368 32DE 2
369 32E0 1
370 32E2 2
371 32E4 1
372 32E6 2
373 32E8 1
374 32EA 2
375 32EC 1
376 32EE 2
377 32F0 1
378 32F2 2
379 32F4 1
380 32F6 2
381 32F8 1
382 32FA 2
383 32FC 1
384 32FE 2
385 3300 1
386 3302 2
387 3304 1
388 3306 2
389 3308 1
390 330A 2
391 330C 1
392 330E 2
393 3310 1
394 3312 2
395 3314 1
396 3316 2
397 3318 1
398 331A 2
399 331C 1
400 331E 2
401 3320 1
402 3322 2
403 3324 1
404 3326 2
405 3328 1
406 332A 2
407 332C 1
408 332E 2
409 3330 1
410 3332 2
411 3334 1
412 3336 2
413 3338 1
414 333A 2
415 333C 1
416 333E 2
417 3340 1
418 3342 2
419 3344 1
420 3346 2
421 3348 1
422 334A 2
423 334C 1
424 334E 2
425 3350 1
426 3352 2
427 3354 1
428 3356 2
429 3358 1
430 335A 2
431 335C 1
432 335E 2
433 3360 1
434 3362 2
435 3364 1
436 3366 2
437 3368 1
438 336A 2
439 336C 1
440 336E 2
441 3370 1
442 3372 2
443 3374 1
444 3376 2
445 3378 1
446 337A 2
447 337C 1
448 337E 2
449 3380 1
450 3382 2
451 3384 1
452 3386 2
453 3388 1
454 338A 2
455 338C 1
456 338E 2
457 3390 1
458 3392 2
459 3394 1
460 3396 2
461 3398 1
462 339A 2
463 339C 1
464 339E 2
465 33A0 1
466 33A2 2
467 33A4 1
468 33A6 2
469 33A8 1
470 33AA 2
471 33AC 1
472 33AE 2
473 33B0 1
474 33B2 2
475 33B4 1
476 33B6 2
477 33B8 1
478 33BA 2
479 33BC 1
480 33BE 2
481 33C0 1
482 33C2 2
483 33C4 1
484 33C6 2
485 33C8 1
486 33CA 2
487 33CC 1
488 33CE 2
489 33D0 1
490 33D2 2
491 33D4 1
492 33D6 2
493 33D8 1
494 33DA 2
495 33DC 1
496 33DE 2
497 33E0 1
498 33E2 2
499 33E4 1
500 33E6 2
501 33E8 1
502 33EA 2
503 33EC 1
504 33EE 2
505 33F0 1
506 33F2 2
507 33F4 1
508 33F6 2
509 33F8 1
510 33FA 2
511 33FC 1
512 33FE 2
513 3400 1
514 3402 2
515 3404 1
516 3406 2
517 3408 1
518 340A 2
519 340C 1
520 340E 2
521 3410 1
522 3412 2
523 3414 1
524 3416 2
525 3418 1
526 341A 2
527 341C 1
528 341E 2
529 3420 1
530 3422 2
531 3424 1
532 3426 2
533 3428 1
534 342A 2
535 342C 1
536 342E 2
537 3430 1
538 3432 2
539 3434 1
540 3436 2
541 3438 1
542 343A 2
543 343C 1
544 343E 2
545 3440 1
546 3442 2
547 3444 1
548 3446 2
549 3448 1
550 344A 2
551 344C 1
552 344E 2
553 3450 1
554 3452 2
555 3454 1
556 3456 2
557 3458 1
558 345A 2
559 345C 1
560 345E 2
561 3460 1
562 3462 2
563 3464 1
564 3466 2
565 3468 1
566 346A 2
567 346C 1
568 346E 2
569 3470 1
570 3472 2
571 3474 1
572 3476 2
573 3478 1
574 347A 2
575 347C 1
576 347E 2
577 3480 1
578 3482 2
579 3484 1
580 3486 2
581 3488 1
582 348A 2
583 348C 1
584 348E 2
585 3490 1
586 3492 2
587 3494 1
588 3496 2
589 3498 1
590 349A 2
591 349C 1
592 349E 2
593 34A0 1
594 34A2 2
595 34A4 1
596 34A6 2
597 34A8 1
598 34AA 2
599 34AC 1
600 34AE 2
601 34B0 1
602 34B2 2
603 34B4 1
604 34B6 2
605 34B8 1
606 34BA 2
607 34BC 1
608 34BE 2
609 34C0 1
610 34C2 2
611 34C4 1
612 34C6 2
613 34C8 1
614 34CA 2
615 34CC 1
616 34CE 2
617 34D0 1
618 34D2 2
619 34D4 1
620 34D6 2
621 34D8 1
622 34DA 2
623 34DC 1
624 34DE 2
625 34E0 1
626 34E2 2
627 34E4 1
628 34E6 2
629 34E8 1
630 34EA 2
631 34EC 1
632 34EE 2
633 34F0 1
634 34F2 2
635 34F4 1
636 34F6 2
637 34F8 1
638 34FA 2
639 34FC 1
640 34FE 2
641 3500 1
642 3502 2
643 3504 1
644 3506 2
645 3508 1
646 350A 2
647 350C 1
648 350E 2
649 3510 1
650 3512 2
651 3514 1
652 3516 2
653 3518 1
654 351A 2
655 351C 1
656 351E 2
657 3520 1
658 3522 2
659 3524 1
660 3526 2
661 3528 1
662 352A 2
663 352C 1
664 352E 2
665 3530 1
666 3532 2
667 3534 1
668 3536 2
669 3538 1
670 353A 2
671 353C 1
672 353E 2
673 3540 1
674 3542 2
675 3544 1
676 3546 2
677 3548 1
678 354A 2
679 354C 1
680 354E 2
681 3550 1
682 3552 2
683 3554 1
684 3556 2
685 3558 1
686 355A 2
687 355C 1
688 355E 2
689 3560 1
690 3562 2
691 3564 1
692 3566 2
693 3568 1
694 356A 2
695 356C 1
696 356E 2
697 3570 1
698 3572 2
699 3574 1
700 3576 2
701 3578 1
702 357A 2
703 357C 1
704 357E 2
705 3580 1
706 3582 2
707 3584 1
708 3586 2
709 3588 1
710 358A 2
711 358C 1
712 358E 2
713 3590 1
714 3592 2
715 3594 1
716 3596 2
717 3598 1
718 359A 2
719 359C 1
720 359E 2
721 35A0 1
722 35A2 2
723 35A4 1
724 35A6 2
725 35A8 1
726 35AA 2
727 35AC 1
728 35AE 2
729 35B0 1
730 35B2 2
731 35B4 1
732 35B6 2
733 35B8 1
734 35BA 2
735 35BC 1
736 35BE 2
737 35C0 1
738 35C2 2
739 35C4 1
740 35C6 2
741 35C8 1
742 35CA 2
743 35CC 1
744 35CE 2
745 35D0 1
746 35D2 2
747 35D4 1
748 35D6 2
749 35D8 1
750 35DA 2
751 35DC 1
752 35DE 2
753 35E0 1
754 35E2 2
755 35E4 1
756 35E6 2
757 35E8 1
758 35EA 2
759 35EC 1
760 35EE 2
761 35F0 1
762 35F2 2
763 35F4 1
764 35F6 2
765 35F8 1
766 35FA 2
767 35FC 1
768 35FE 2
769 3600 1
770 3602 2
771 3604 1
772 3606 2
773 3608 1
774 360A 2
775 360C 1
776 360E 2
777 3610 1
778 3612 2
779 3614 1
780 3616 2
781 3618 1
782 361A 2
783 361C 1
784 361E 2
785 3620 1
786 3622 2
787 3624 1
788 3626 2
789 3628 1
790 362A 2
791 362C 1
792 362E 2
793 3630 1
794 3632 2
795 3634 1
796 3636 2
797 3638 1
798 363A 2
799 363C 1
800 363E 2
801 3640 1
802 3642 2
803 3644 1
804 3646 2
805 3648 1
806 364A 2
807 364C 1
808 364E 2
809 3650 1
810 3652 2
811 3654 1
812 3656 2
813 3658 1
814 365A 2
815 365C 1
816 365E 2
817 3660 1
818 3662 2
819 3664 1
820 3666 2
821 3668 1
822 366A 2
823 366C 1
824 366E 2
825 3670 1
826 3672 2
827 3674 1
828 3676 2
829 3678 1
830 367A 2
831 367C 1
832 367E 2
833 3680 1
834 3682 2
835 3684 1
836 3686 2
837 3688 1
838 368A 2
839 368C 1
840 368E 2
841 3690 1
842 3692 2
843 3694 1
844 3696 2
845 3698 1
846 369A 2
847 369C 1
848 369E 2
849 36A0 1
850 36A2 2
851 36A4 1
852 36A6 2
853 36A8 1
854 36AA 2
855 36AC 1
856 36AE 2
857 36B0 1
858 36B2 2
859 36B4 1
860 36B6 2
861 36B8 1
862 36BA 2
863 36BC 1
864 36BE 2
865 36C0 1
866 36C2 2
867 36C4 1
868 36C6 2
869 36C8 1
870 36CA 2
871 36CC 1
872 36CE 2
873 36D0 1
874 36D2 2
875 36D4 1
876 36D6 2
877 36D8 1
878 36DA 2
879 36DC 1
880 36DE 2
881 36E0 1
882 36E2 2
883 36E4 1
884 36E6 2
885 36E8 1
886 36EA 2
887 36EC 1
888 36EE 2
889 36F0 1
890 36F2 2
891 36F4 1
892 36F6 2
893 36F8 1
894 36FA 2
895 36FC 1
896 36FE 2
897 3700 1
898 3702 2
899 3704 1
900 3706 2
901 3708 1
902 370A 2
903 370C 1
904 370E 2
905 3710 1
906 3712 2
907 3714 1
908 3716 2
909 3718 1
910 371A 2
911 371C 1
912 371E 2
913 3720 1
914 3722 2
915 3724 1
916 3726 2
917 3728 1
918 372A 2
919 372C 1
920 372E 2
921 3730 1
922 3732 2
923 3734 1
924 3736 2
925 3738 1
926 373A 2
927 373C 1
928 373E 2
929 3740 1
930 3742 2
931 3744 1
932 3746 2
933 3748 1
934 374A 2
935 374C 1
936 374E 2
937 3750 1
938 3752 2
939 3754 1
940 3756 2
941 3758 1
942 375A 2
943 375C 1
944 375E 2
945 3760 1
946 3762 2
947 3764 1
948 3766 2
949 3768 1
950 376A 2
951 376C 1
952 376E 2
953 3770 1
954 3772 2
955 3774 1
956 3776 2
957 3778 1
958 377A 2
959 377C 1
960 377E 2
961 3780 1
962 3782 2
963 3784 1
964 3786 2
965 3788 1
966 378A 2
967 378C 1
968 378E 2
969 3790 1
970 3792 2
971 3794 1
972 3796 2
973 3798 1
974 379A 2
975 379C 1
976 379E 2
977 37A0 1
978 37A2 2
979 37A4 1
980 37A6 2
981 37A8 1
982 37AA 2
983 37AC 1
984 37AE 2
985 37B0 1
986 37B2 2
987 37B4 1
988 37B6 2
989 37B8 1
990 37BA 2
991 37BC 1
992 37BE 2
993 37C0 1
994 37C2 2
995 37C4 1
996 37C6 2
997 37C8 1
998 37CA 2
999 37CC 1
1000 37CE 2
1001 37D0 1
1002 37D2 2
1003 37D4 1
1004 37D6 2
1005 37D8 1
1006 37DA 2
1007 37DC 1
1008 37DE 2
1009 37E0 1
1010 37E2 2
1011 37E4 1
1012 37E6 2
1013 37E8 1
1014 37EA 2
1015 37EC 1
1016 37EE 2
1017 37F0 1
1018 37F2 2
1019 37F4 1
1020 37F6 2
1021 37F8 1
1022 37FA 2
1023 37FC 1
1024 37FE 2
1025 3800 1
1026 3802 2
1027 3804 1
1028 3806 2
1029 3808 1
1030 380A 2
1031 380C 1
1032 380E 2
1033 3810 1
1034 3812 2
1035 3814 1
1036 3816 2
1037 3818 1
1038 381A 2
1039 381C 1
1040 381E 2
1041 3820 1
1042 3822 2
1043 3824 1
1044 3826 2
1045 3828 1
1046 382A 2
1047 382C 1
1048 382E 2
1049 3830 1
1050 3832 2
1051 3834 1
1052 3836 2
1053 3838 1
1054 383A 2
1055 383C 1
1056 383E 2
1057 3840 1
1058 3842 2
1059 3844 1
1060 3846 2
1061 3848 1
1062 384A 2
1063 384C 1
1064 384E 2
1065 3850 1
1066 3852 2
1067 3854 1
1068 3856 2
1069 3858 1
1070 385A 2
1071 385C 1
1072 385E 2
1073 3860 1
1074 3862 2
1075 3864 1
1076 3866 2
1077 3868 1
1078 386A 2
1079 386C 1
1080 386E 2
1081 3870 1
1082 3872 2
1083 3874 1
1084 3876 2
1085 3878 1
1086 387A 2
1087 387C 1
1088 387E 2
1089 3880 1
1090 3882 2
1091 3884 1
1092 3886 2
1093 3888 1
1094 388A 2
1095 388C 1
1096 388E 2
1097 3890 1
1098 3892 2
1099 3894 1
1100 3896 2
1101 3898 1
1102 389A 2
1103 389C 1
1104 389E 2
1105 38A0 1
1106 38A2 2
1107 38A4 1
1108 38A6 2
1109
```

## —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 label symbols. The M command permits the programmer to alter data with the cursor.

|                             |                                                                                                                                                                                                     |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| * DM 7800_7850 [CR]         | Displays the contents of the memory block from 7800 to 7850.                                                                                                                                        |
| * DM MAIN7_MAIN9 [CR]       | Displays the contents of the memory block from the address identified by "MAIN7" to the address identified by "MAIN9".                                                                              |
| * DM STEP0-9_STEP3+15L [CR] | Displays the contents of the memory block from the address 9 bytes before the address identified by label symbol "STEP0" to the address of the instruction 15 lines away from label symbol "STEP3". |

- Enter the M command in response to the prompt "\* D".
- The 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 debugger displays a dump of memory contents on the screen with 8 bytes on a line in the 40 characters per line mode and with 16 bytes on a line in the 80 characters per line mode.
- If the printer is placed in the enable mode, the debugger prints the memory dump on the printer with 16 bytes on a line.
- The cursor appears on the screen when the memory block dump is completed. The programmer can then alter byte data in the memory dump 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 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 the [SPACE] key to suspend display of the memory dump. To resume display, press the [SPACE] key again.
- Press the [BREAK] key to force the debugger into the command mode.

## —D (Memory list dump) Command—

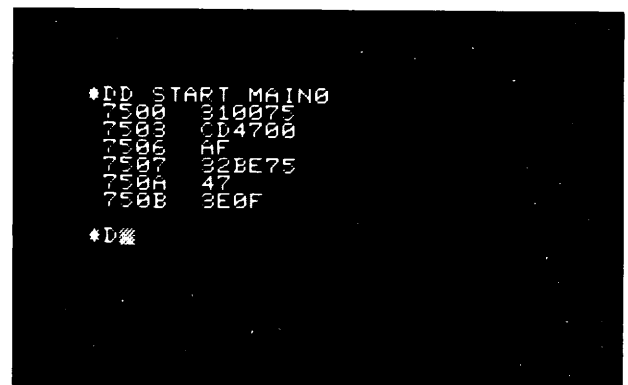
The D command displays the contents of the specified memory block in hexadecimal representation with one instruction on a line. The memory block may be specified with either absolute hexadecimal addresses or label symbols. The programmer cannot alter memory contents through cursor manipulation.

|                          |                                                                                                                                                                           |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| * DD 7800_7850 [CR]      | Displays the contents of the memory block from addresses 7800 to 7850 with one instruction on a line.                                                                     |
| * DD START_MAIN0 [CR]    | Displays the contents of the memory block from the addresses identified by "START" to the address identified by "MAIN0" with one instruction on a line.                   |
| * DD 7500_START+12L [CR] | Displays the contents of the memory block from address 7500 to the address of the instruction 12 lines away from the label symbol "START" with one instruction on a line. |

- Enter the D command in response to the prompt "\*D".
- The debugger displays the cursor with a space between it and the letter D, then 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 either with 4-digit hexadecimal numbers or global symbols. As with the M command, the starting address must be smaller than or equal to the ending address.
- Press the [CR] key after specifying the required memory block; the debugger then displays an address and machine language code on each line.

Consider the source program shown below, which contains the label symbols "START" and "MAIN0". Assume that the corresponding object code is loaded in memory starting at address 7500. When a D command is entered, the debugger displays a dump listing on the screen as shown in the photo at right.

```
START: ENT
 LD SP, START
 CALL MSTP
 XOR A
 LD (? TABP), A
 LD B, A
MAIN0: ENT
 LD A, 0FH
```



- It must be noted that the memory block starting address specified in the D command must contain an operation code. If the starting address contains a data byte, subsequent lines dumped will display meaningless instructions which read that data byte as an operation code. The same note applies to the data areas (defined by DEFB and DEFW, etc.) in the memory block.

- Display of the memory dump listing can be suspended and resumed with the **SPACE** key.
- The D command does not allow memory alteration; after the memory dump is completed, the debugger is returned to the command wait state.
- Press the **BREAK** key to terminate this command in the middle of a dump.

## —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 label symbol.

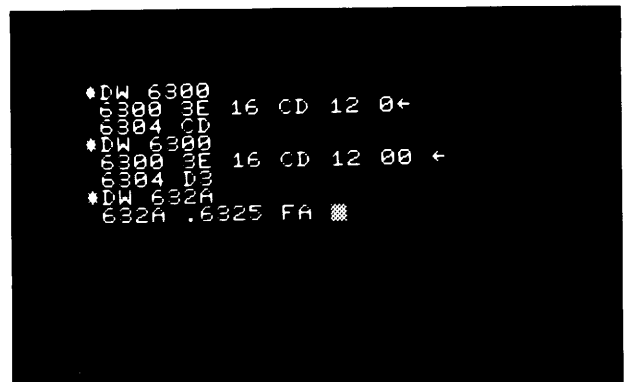
- \* **DW 8000** **CR**      Writes machine language data, starting at address 8000.
- \* **DW DATA1** **CR**      Writes machine language data, starting at the address identified by label symbol "DATA1".

- Enter the W command in response to the prompt "\*D".
- The debugger displays the cursor with a space between it and the letter W, then waits for the programmer to enter the starting address of the memory area to be written.  
The programmer may specify the memory block starting address with a 4-digit hexadecimal number or a global symbol.
- The memory area to be written must be inside the link area.

```
*DW 1111 }
 1111 } Address 1111 is not in the link area.
 ???
```

- When the programmer presses the **CR** key after specifying an address, the debugger displays that address on the next line to prompt the programmer to enter 2-digit hexadecimal data.  
The debugger enters 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 the **←** key to return the cursor to the byte of data just entered and correct it. The photo on the right shows an example. As the photo shows, when the **←** key is pressed, the cursor is placed on the next line and the address of the byte of data to which the cursor is moved is displayed.



- To specify a displacement for a JR, DJNZ or other Z80 relative jump instruction, enter a period; the 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 debugger computes the displacement and stores the 1-byte result in the current byte position. The seventh and eighth lines in the photo above show an example of specifying a displacement.
- After the necessary data has been written, press **CR**; the debugger then 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.

- \* DG 7700 CR      Executes the program at address 7700.
- \* DG START CR      Executes the program at the address identified by label symbol "START".
- \* DG CR      Restarts the program at the breakpoint. The restart address and CPU register data are stored in the register buffer.

- Enter a G command in response to the prompt "\*D".
- The 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 defined with the ENT assembler directive.

When using a label symbol, the programmer can specify the execution address on a line or byte basis.

- \* DG MAIN0      Executes the program at address "MAIN0".
- \* DG MAIN0+3L      Executes the program at the address 3 lines after "MAIN0".
- \* DG 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 debugger returns to the command wait state without executing the program.

The contents of the CPU registers to be 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.

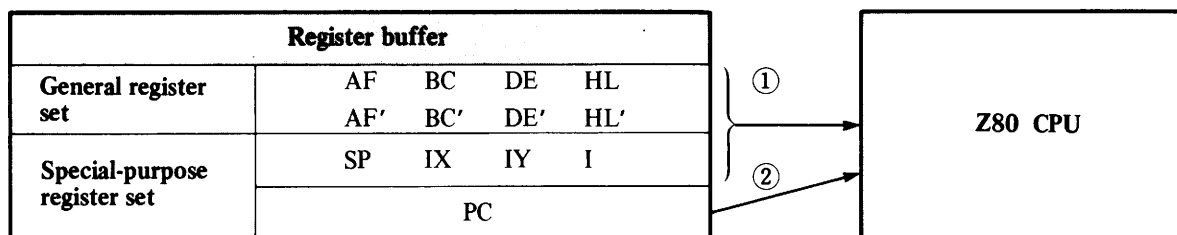
- Press the BREAK key to terminate entry of a G command.

## —I (Indicative start) Command—

The I command executes the program with the CPU registers loaded with the register buffer contents. The execution address is designated by the program counter. The contents of the CPU registers can be specified by the programmer through use of the A, C and P commands.

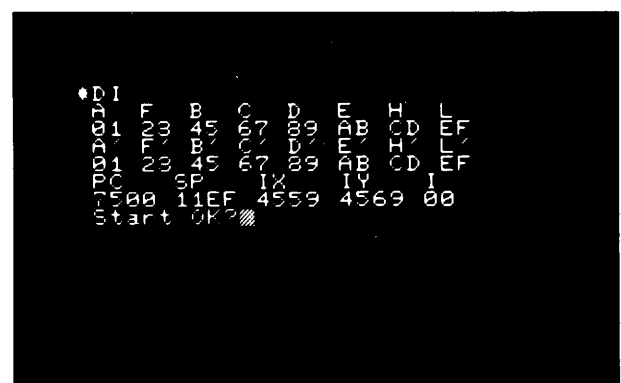
|                                               |    |      |    |      |    |      |    |                                                                                                                                               |
|-----------------------------------------------|----|------|----|------|----|------|----|-----------------------------------------------------------------------------------------------------------------------------------------------|
| * DI                                          |    |      |    |      |    |      |    | Executes the program at the address<br>designated by the program counter<br>with the data shown on the screen<br>loaded in the CPU registers. |
| A                                             | F  | B    | C  | D    | E  | H    | L  |                                                                                                                                               |
| 01                                            | 23 | 45   | 67 | 89   | AB | CD   | ED |                                                                                                                                               |
| A'                                            | F' | B'   | C' | D'   | E' | H'   | L' |                                                                                                                                               |
| 01                                            | 23 | 45   | 67 | 89   | AB | CD   | EF |                                                                                                                                               |
| PC                                            |    | SP   |    | IX   |    | IY   | I  |                                                                                                                                               |
| 78AB                                          |    | 1FEA |    | 5F70 |    | 4F50 | 00 |                                                                                                                                               |
| Start OK? <input checked="" type="checkbox"/> |    |      |    |      |    |      |    |                                                                                                                                               |

- Enter the I command in response to the prompt " \* D".
- The debugger displays the 2- and/or 4-digit hexadecimal values to be loaded into the CPU registers. These values are stored in the register buffer. They can be displayed with the R command.
- The debugger then displays message "Start OK?". To start the program in this environment, press **CR**. The debugger then executes the program, starting at the address designated in the program counter. To change register values or terminate the I command, press the **BREAK** key; the debugger then returns to the command wait state.
- The figure below shows how the CPU registers are set with the I command.



The CPU general registers and special-purpose registers SP, IX, IY and I are loaded first; the program counter is then loaded with the execution address and the program is executed.

- The photo at right shows how the debugger responds to the I command and executes the program (at address 7500 in this example.)
- The status of the CPU registers is displayed on a line in the 80 characters per line mode.



## —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 cursor manipulation.

\* DA

| 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 the A command in response to the prompt "\*D".
- 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 stored in the register buffer for use in subsequent restart operations (see the G command description) at the breakpoint.
- The debugger displays the cursor on the line following the one last displayed. 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 cursor will move to the beginning of the next line).

The register values displayed or altered with the A command are those values which will be restored to the CPU internal registers on a restart at a breakpoint or on an indicative start with the I command.

- Press **CR** with the cursor on the new line; the debugger then 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 manipulation.

\* DC

| 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 the C command in response to the prompt "\*D".
- 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. The contents of the registers and the meanings of the register contents and data altered through cursor manipulation are the same as with the A command. They are used for restart at a breakpoint or with the I command.
- Press the **CR** key with the cursor on the new line; the debugger 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 manipulation.

|      |                                                                          |      |      |    |  |
|------|--------------------------------------------------------------------------|------|------|----|--|
| * DP | Displays the contents of special-purpose registers PC, SP, IX, IY and I. |      |      |    |  |
| PC   | SP                                                                       | IX   | IY   | I  |  |
| 78AB | 1FEA                                                                     | 5F70 | 5F50 | 00 |  |
| ■    |                                                                          |      |      |    |  |

- Enter the P command in response to the prompt "\*D".
- The debugger displays the contents of special-purpose registers PC, SP, IX, IY and I with 2- and/or 4-digit hexadecimal numbers. The meanings of the register contents and the data altered through cursor manipulation are the same as with the A and C commands.

The register values displayed or altered through cursor manipulation are restored into the pertinent registers upon restart at a breakpoint or upon indicative start with the I command. The program does not have to restart at the breakpoint; the programmer can specify another restart address by altering the PC value.

- Press **CR** with the cursor on the new line; the debugger then returns to the command wait state.

## —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.

|      |                                             |      |      |    |    |    |    |  |
|------|---------------------------------------------|------|------|----|----|----|----|--|
| * DR | Displays the contents of all CPU registers. |      |      |    |    |    |    |  |
| 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 |  |
| PC   | SP                                          | IX   | IY   | I  |    |    |    |  |
| 78AB | 1FEA                                        | 5F70 | 5F50 | 00 |    |    |    |  |

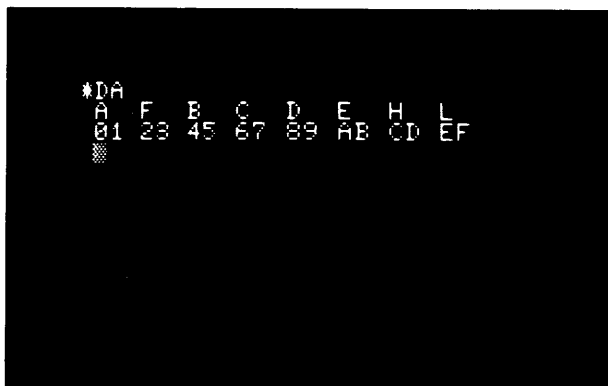
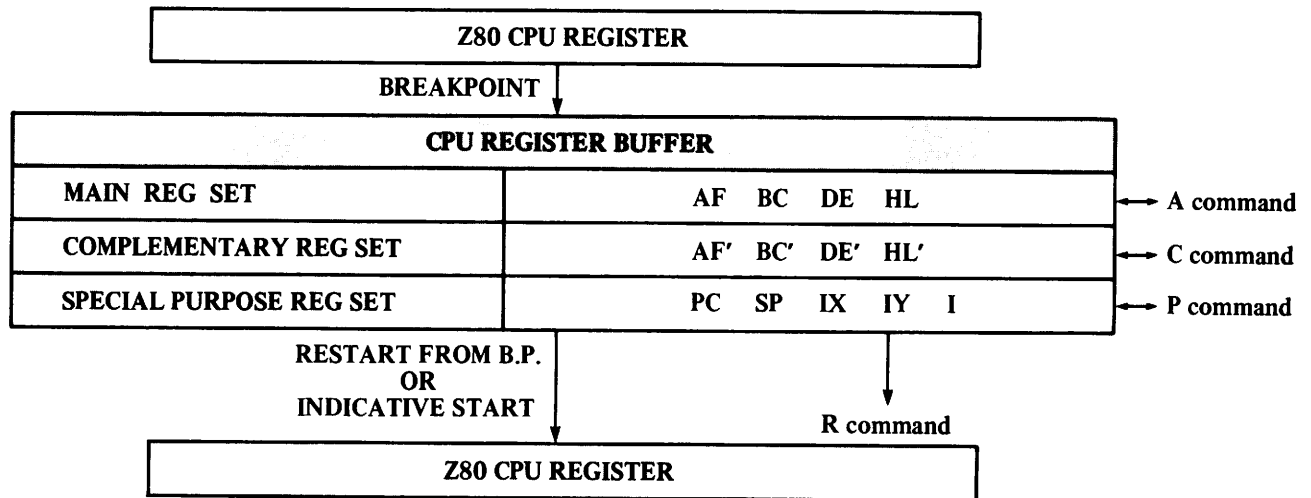
- Enter the R command in response to the prompt "\*D".
- The debugger displays the contents of all CPU registers with 2- and/or 4-digit hexadecimal numbers. The cursor does not appear in the screen, so the programmer cannot alter their values.

The same data is automatically displayed when a break occurs or when an indicative start is initiated with the I command.
- The debugger enters the command wait state after displaying all the register contents.
- The above display is on 1 line in the 80 characters per line mode.

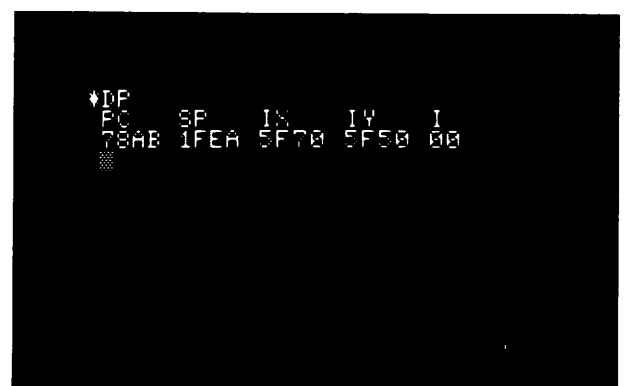
## Using register commands A, C, P and R

Values displayed with register commands (A, C, P and R) are the actual contents of the register buffer in the debugger. The register buffer in the debugger contains values loaded when breaks occur or when changes are made through cursor manipulation with the A, C or P command. The values are restored to the CPU registers when a restart is made from a breakpoint or when an indicative start is made.

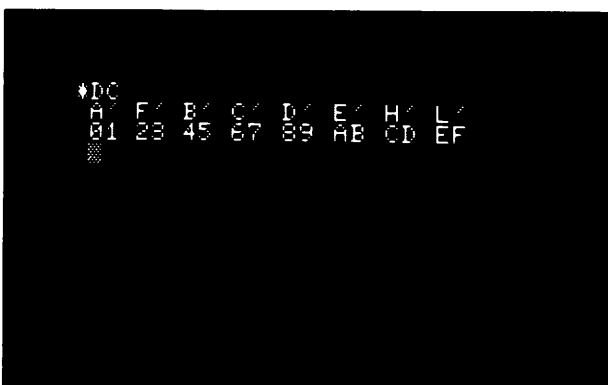
The figure below shows the relationship between the CPU registers and the register commands; the photos show examples of use of the register commands.



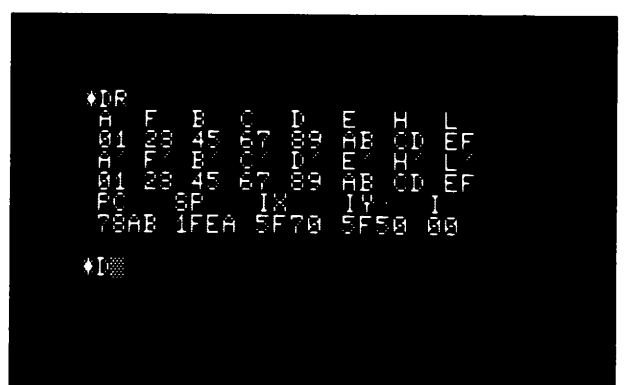
A command



P command



C command



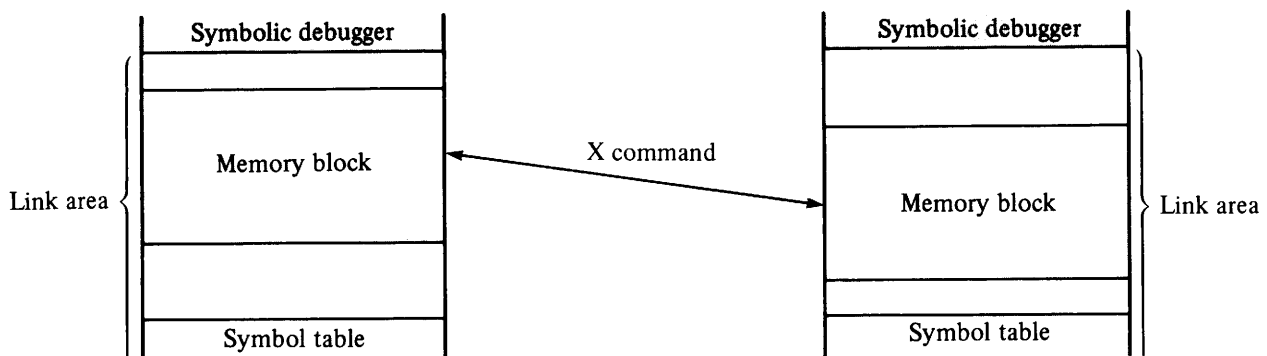
R command

## —X (Data transfer) Command—

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

|                                               |                                                                                                                            |
|-----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| <p>* DX<br/>From? 7500 To? 811F Top? 9000</p> | <p>Transfers the contents of the memory block from addresses 7500 to 811F to the memory area starting at address 9000.</p> |
|-----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|

- Enter the X command in response to the prompt "\*D".
- The 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 the programmer to enter the ending address of the source memory block with a 4-digit hexadecimal number. When the ending address is entered, the debugger displays the message "Top?" to prompt the programmer to enter the starting address of the destination memory area with a 4-digit hexadecimal number (symbolic addresses are disallowed).
- When the last address is entered, the debugger starts transferring the memory block. After completing the transfer, it returns to the command wait state.
- 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 photo at right shows how the debugger transfers the memory block starting at address 7500 and ending at address 750F to the memory area starting at address 7508.

Compare the memory contents displayed with the two M commands.

The contents of 8 memory bytes are displayed on each line in the 40 characters per line mode and the contents of 16 memory bytes are displayed on each line in the 80 characters per line mode.

```

*DM 7500 751F
7500 54 4E 4C 32 07 68 00 4C TNL2.h.L
7501 00 4D 4E 4E 42 00 00 00 MONLY...
7502 10 4D 4F 41 4D 45 00 00 LOAMES...
7503 00 4C 4E 47 00 00 00 04 .LONG...

*DX
From? 7500 To? 750F Top? 7508
*DM 7500 751F
7500 54 4E 4C 32 07 68 00 4C TNL2.h.L
7501 00 4D 4E 4E 42 00 00 00 MONLY...
7502 10 4D 4F 41 4D 45 00 00 LOAMES...
7503 00 4C 4E 47 00 00 00 04 .LONG...

*DM

```

## —S (Save) Command—

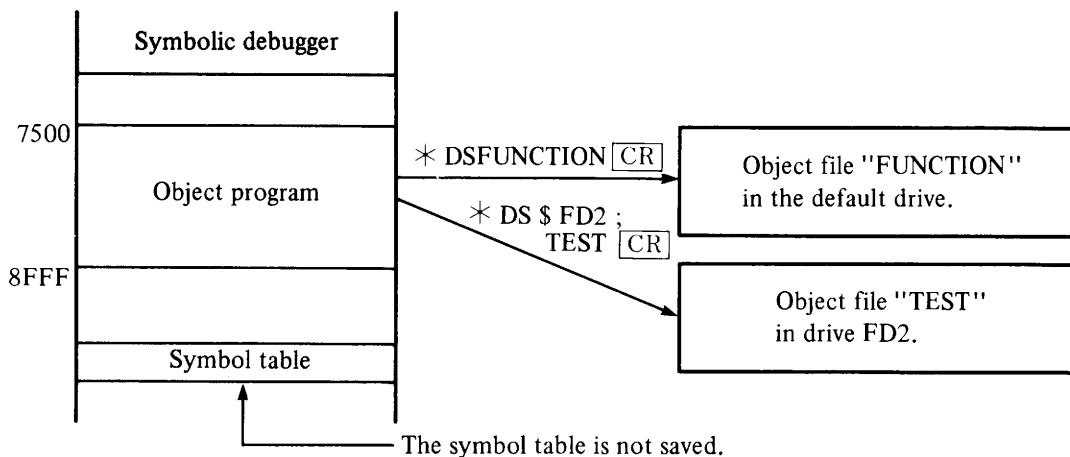
The S command saves a specified block of the object program in the symbolic debugger link area into a named output file in immediately executable form. The contents of this file can be restored to the link area with the Y command.

\* DSfilename

TBE  7500  8FFF  7500

Saves the immediately executable object program from addresses 7500 to 8FFF in the link area to an object file with a file name of filename. OBJ.

- Enter the S command followed by a file name in response to the prompt "\* D".
- Press  after entering a file name. The debugger displays a TBE (Top-Bottom-Execute) message after verifying that the specified file does not exist on the specified diskette.
- Enter the starting and ending addresses of the block to be saved and the execution address with 4-digit hexadecimal numbers or symbolic label names. When the execution address is omitted, the debugger assumes the block starting address as the execution address.
- The figure below shows how the object program block from addresses 7500 to 8FFF is saved to an output file with the file name "FUNCTION".



—Y (Yank) Command—

The Y command reads the object file identified by filename into the link area.

\* DY filename

CR

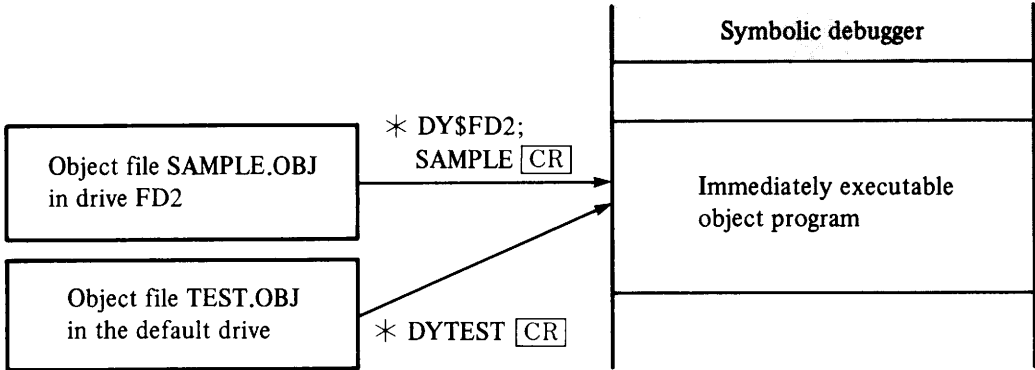
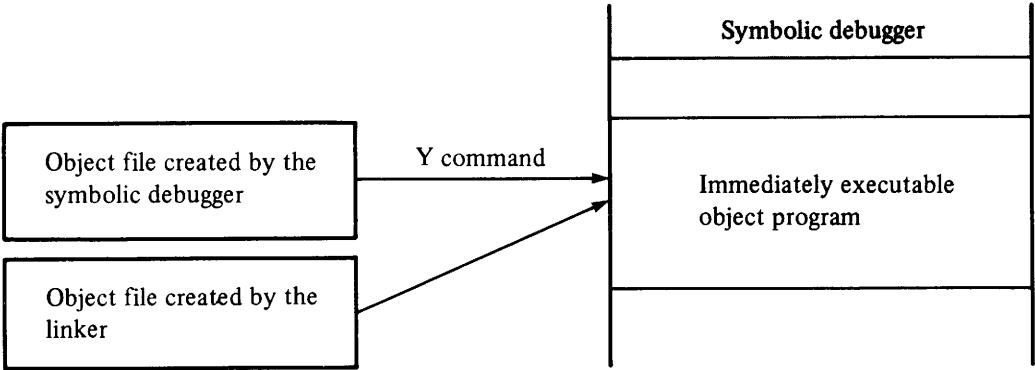
Reads the object file named filename into the link area under loading

Loading address \$7500–8FFF conditions established when the file was saved.

Execute address \$7500

- Enter the Y command followed by a file name in response to the prompt "\*D".
- Press CR after entering the file name. The debugger then searches for the file named filename. OBJ and reads it.
- The program in the filename.OBJ file is loaded into the link area block between the starting and ending addresses specified when the file was saved with the S command.

Note: Files opened before the Y command is issued are all killed.



## — \ (FDOS) Command—

The \ command executes a built-in FDOS command. "\*D" is displayed to prompt for the next command.

|                                                                                                                                           |
|-------------------------------------------------------------------------------------------------------------------------------------------|
| <b>* D \FREE \$FDn</b> <b>[CR]</b> Outputs the number of used and unused sectors on the floppy disk in the disk drive indicated by \$FDn. |
|-------------------------------------------------------------------------------------------------------------------------------------------|

- Enter the \ command followed by the desired built-in FDOS command in response to the prompt "\*D".
- Press the **[CR]** key; the debugger then executes the specified FDOS command and displays "\*D" to prompt for the next command.
- The XFER and EXEC commands cannot be executed. The RUN command cannot be executed when the program to be executed by the RUN command is too long.

## — # Command—

|                                                                 |
|-----------------------------------------------------------------|
| <b>* D#</b> Switches the list mode for printout on the printer. |
|-----------------------------------------------------------------|

- Enter the # command in response to the prompt "\*D".
- The debugger then switches the list mode. When the debugger 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).

## — ! Command—

|                                      |
|--------------------------------------|
| <b>* D!</b> Returns control to FDOS. |
|--------------------------------------|

- Enter the ! command in response to the prompt "\*D".
- Control is then transferred to FDOS.

# ERROR MESSAGES

| Error message | Description                                                                                                                                                                                                                                                                                                                                       | Related commands                   |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------|
| ???           | <ul style="list-style-type: none"> <li>○ The command operand fields does not match the 4-digit hexadecimal format.</li> <li>○ A symbolic label is missing.</li> <li>○ A data defining symbol is used as a label.</li> </ul>                                                                                                                       | M, D, W, B, G                      |
| Error         | <ul style="list-style-type: none"> <li>○ An invalid number of digits was entered when altering register or memory contents, or a key other than 0 through 9 or A through F was pressed.</li> </ul>                                                                                                                                                | A, C, P, M                         |
| DJNZ?         | A breakpoint was set for a DJNZ instruction.                                                                                                                                                                                                                                                                                                      | B                                  |
| CALL?         | A breakpoint was set for a CALL instruction.                                                                                                                                                                                                                                                                                                      | B                                  |
| RST 6?        | A breakpoint was set for a RST 6 instruction.                                                                                                                                                                                                                                                                                                     | B                                  |
| Over          | An attempt was made to set more than 9 breakpoints.                                                                                                                                                                                                                                                                                               | B                                  |
| ?             | <ul style="list-style-type: none"> <li>○ An attempt was made to access outside the link area.</li> <li>○ The starting address is greater than the ending address.</li> <li>○ An attempt was made to clear an undefined breakpoint.</li> <li>○ The breakpoint counter was set to F (the maximum permissible value is E in hexadecimal).</li> </ul> | M, D, W, B, G, X<br>M, D<br>B<br>B |

Note: Refer to the System Error Messages in the System Command manual for other system error messages.

Personal Computer  
**MZ-80B**

1-800-4-A-SHARP

**SHARP**

### NOTICE

The MZ-80 series of sophisticated personal computers is manufactured by the SHARP CORPORATION. Hardware and software specifications are subject to change without prior notice; therefore, you are requested to pay special attention to version numbers of the monitor and the system software (supplied in the form of cassette tape or mini-floppy disk files).

This manual is for reference only and the SHARP CORPORATION will not be responsible for difficulties arising out of inconsistencies caused by version changes, typographical errors or omissions in the descriptions.

This manual is based on the SB-1500 series monitor and the SB-7000 series FDOS.

— CONTENTS —

|                                                                         |    |
|-------------------------------------------------------------------------|----|
| INTRODUCTION .....                                                      | 1  |
| LOADING ADDRESS .....                                                   | 2  |
| RELATIONSHIP BETWEEN THE EXECUTION ADDRESS<br>AND LOADING ADDRESS ..... | 3  |
| OFFSET .....                                                            | 6  |
| LOADING ADDRESS AND EXECUTION ADDRESS<br>(ORG STATEMENT) .....          | 7  |
| SYMBOL TABLE .....                                                      | 8  |
| LINK/T COMMAND .....                                                    | 9  |
| LINK MESSAGE EXAMPLES .....                                             | 11 |
| ERROR MESSAGES .....                                                    | 12 |



# INTRODUCTION

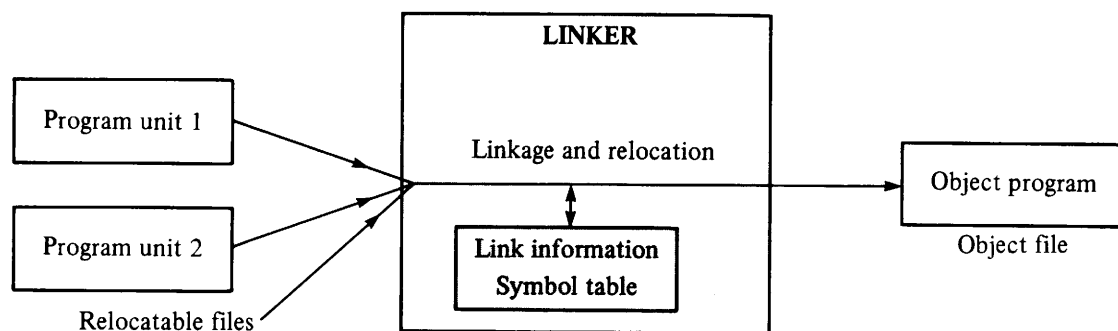
The linker for the SHARP MZ-80B inputs relocatable files output by the assembler or the BASIC compiler and outputs object files.

Relocatable files are not programs which are directly executable by the CPU, but are files which contain information used to keep programs relocatable. They also contain global symbols in ASCII code which are declared to link two or more program units.

The linker fetches relocation information and loads object programs into the link area in main memory while adding the programmer-specified loading address to the relocatable addresses. When two or more relocatable program units are loaded, units are appended to the first program unit (file), if the loading address is specified for the first unit.

The linkage operation itself is described in detail in Section 2.3, "Linker" of the System Command manual. However, the programmer does not need to be aware of details of the linkage operation details.

When outputting the object program (object file), it is necessary to specify the loading address and the execution address.



# LOADING ADDRESS

The loading address specifies the address at which the object program is to be loaded. When this address is not specified, FDOS assumes the starting address which can be managed by FDOS as loading address.

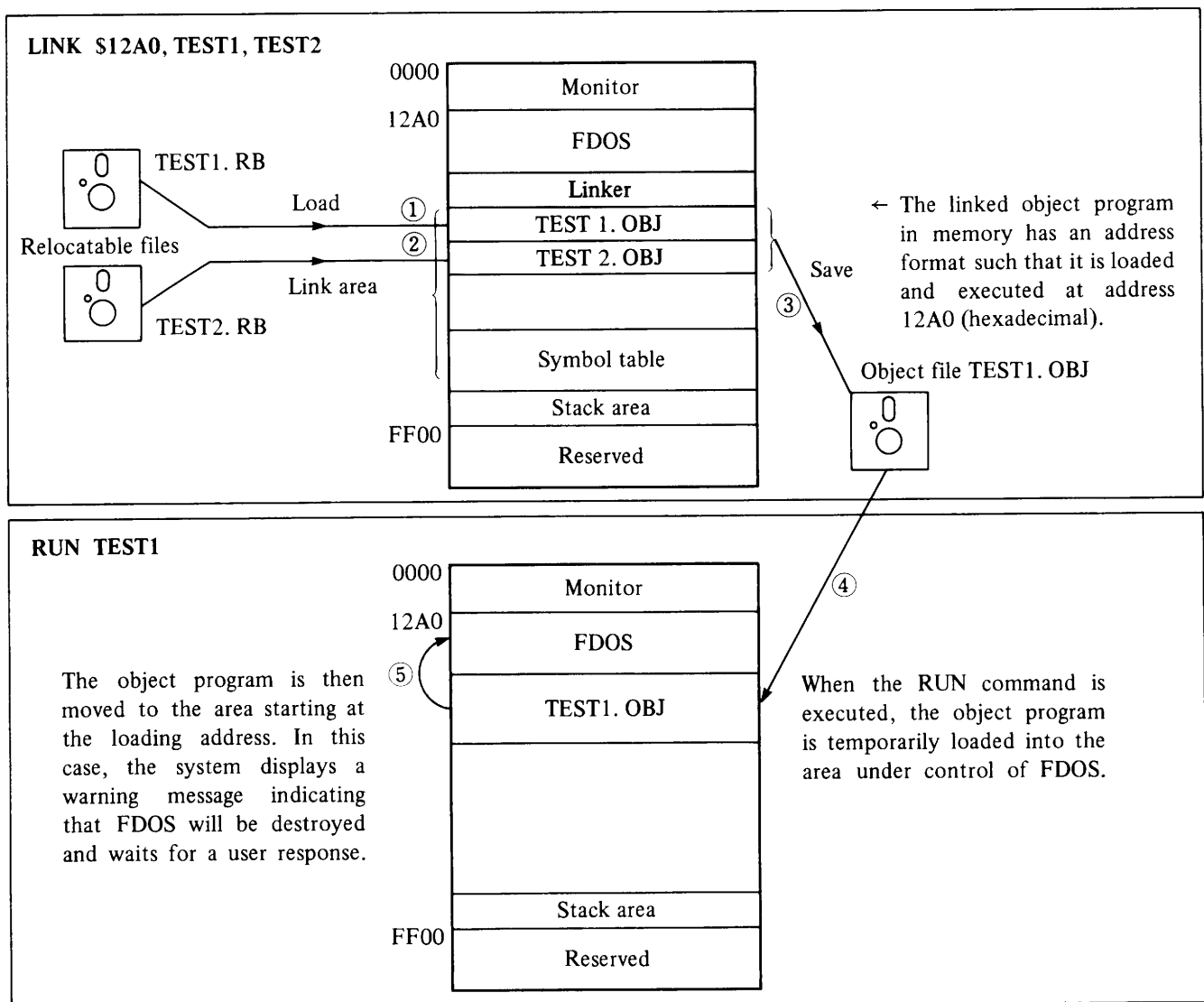
**LINK TEST1, TEST2**

Links TEST1 and TEST2 and assigns the loading address to the beginning of the area managed by FDOS.

**LINK \$12A0, TEST1, TEST2**

Links TEST1 and TEST2 and assigns the loading address to 12A0H.

The figure below shows the flow of files from the time they are linked by the linker until they are executed with the RUN command. Numbers ① through ⑤ in the figure denote the processing sequence.



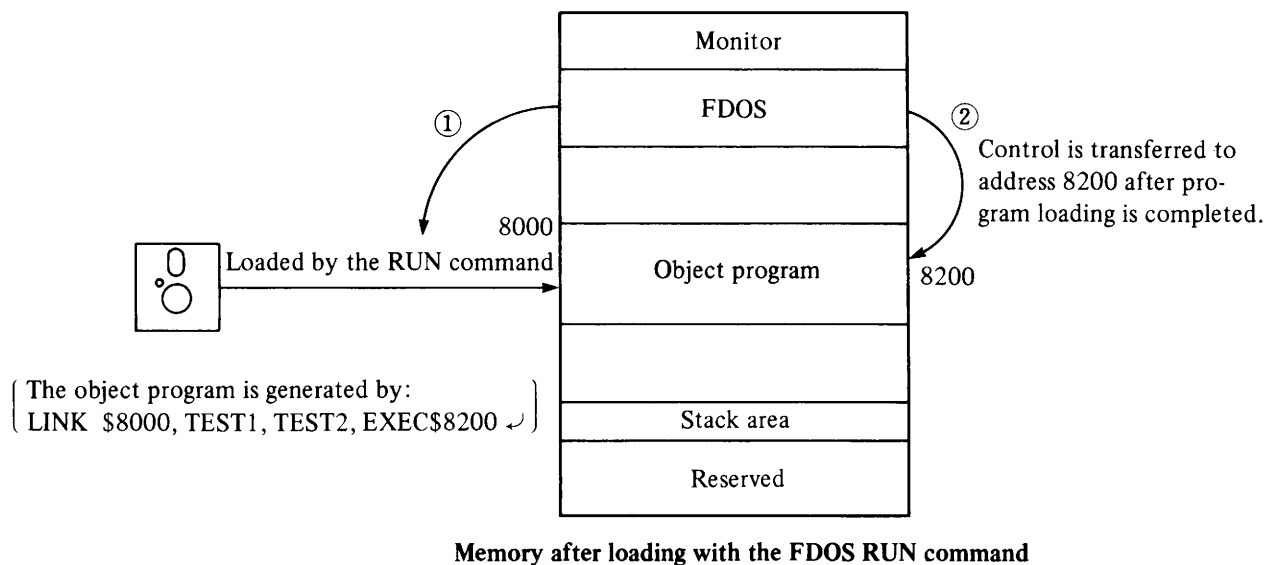
## RELATIONSHIP BETWEEN THE EXECUTION ADDRESS AND LOADING ADDRESS

The programmer may specify the execution address as well as the loading address when outputting an object file through the linker.

```
LINK $8000, TEST1, TEST2, EXEC$8200
```

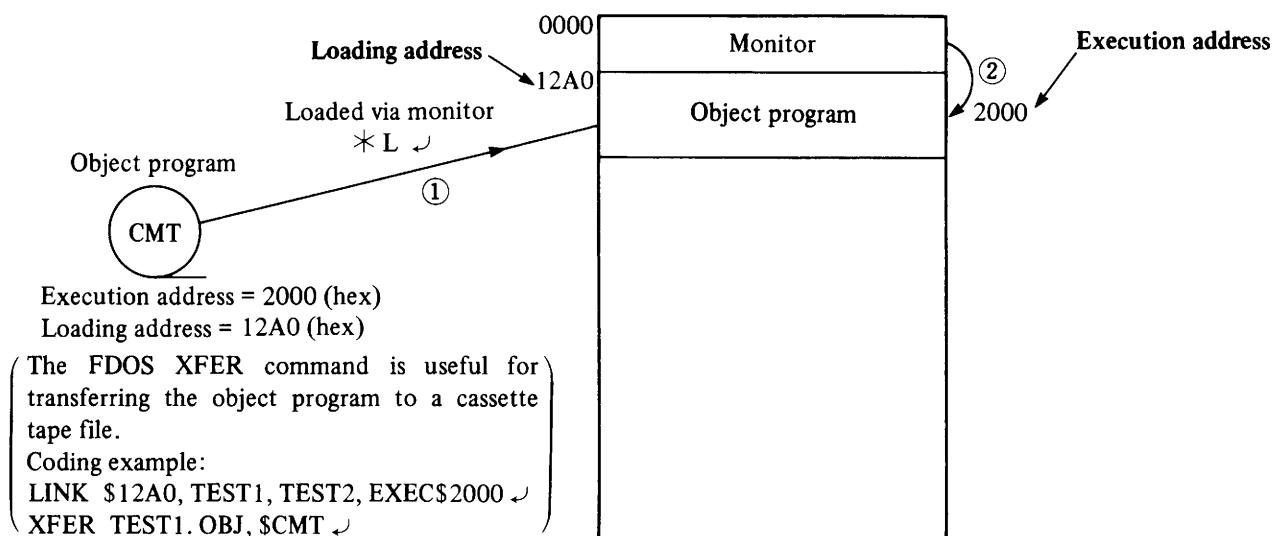
The above command links and loads relocatable program unit files TEST1 and TEST2 into memory, specifying a loading address of 8000 (hex) and an execution address of 8200 (hex).

Examples of linkage and loading are given below (numbers in circles in the figures denote the processing steps). The first example uses a simple RUN command.

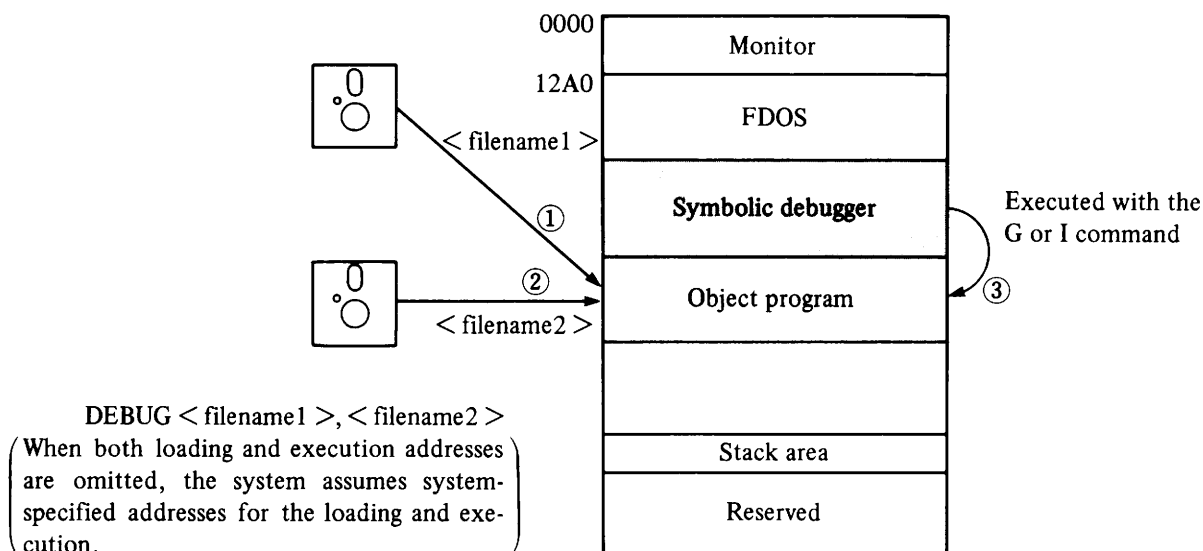


Note: Any loading address or execution address is invalid for LINK/S even if specified.

When the monitor is used to load the object program, its starting address in memory is designated by the loading address. The program counter is set to the address designated by the execution address after the object program is loaded. The figure below shows how an object program with a loading address of 12A0 and an execution address of 2000 is loaded and how control is transferred.



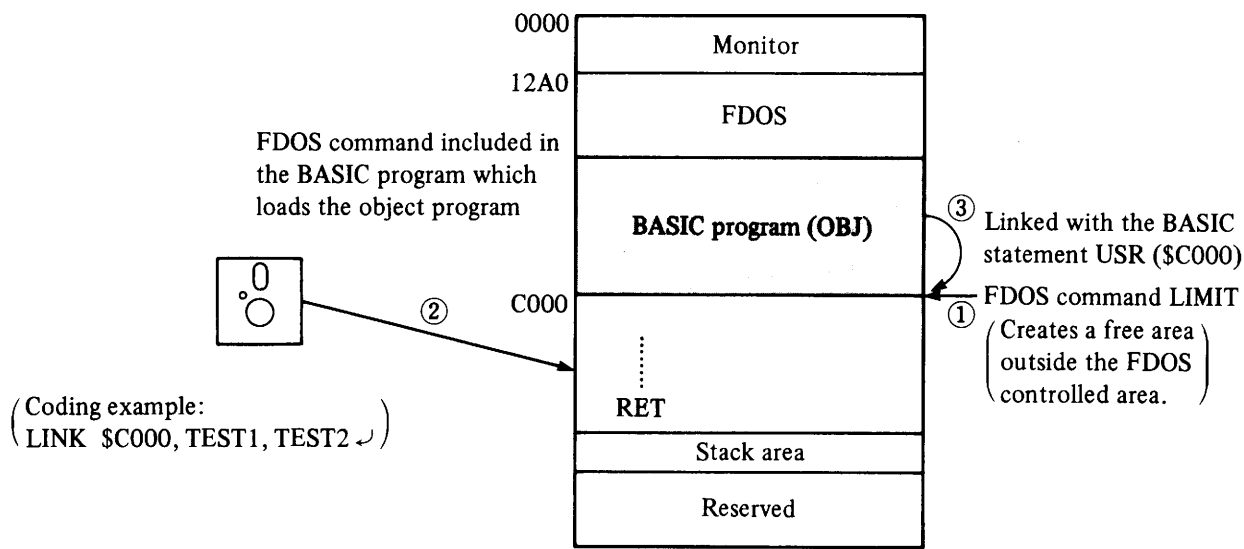
Memory after loading with the monitor program



Memory after loading with the symbolic debugger

Subroutine programs created with the assembler and BASIC programs created with the BASIC compiler may be linked using a library (see the "Programming Utility" manual) or the BASIC USR statement. Here, an example is given of linking an object program with a BASIC program using the USR statement.

The figure below shows how an object program is loaded and linked with a BASIC program. The area in memory which is managed by FDOS is reduced with the FDOS LIMIT command to create a free area. The object program is loaded into this free area with the FDOS or BASIC LOAD statement. The BASIC program can then call the object program as a subroutine using the USR( ) statement.



Memory after loading with an FDOS command in a BASIC program

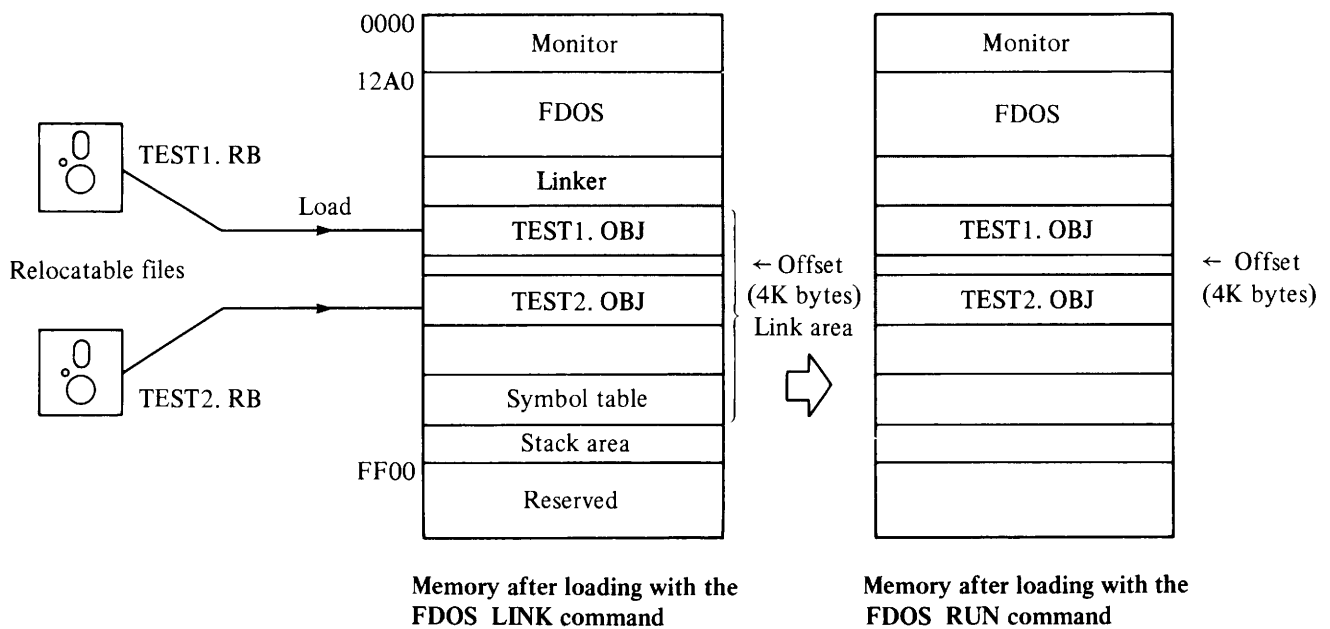
# OFFSET

The programmer can specify an offset to reserve a free area between two object program units.

**LINK TEST1, \$1000, TEST2**

Links TEST1 and TEST2 so that the object program is loaded at the area equivalent to 1000 (hex) addresses reserved between them.

Execution of the above command is illustrated below.



Note that the loading address and offset are carefully distinguished in the following command:

A 4-digit hexadecimal number preceded by a \$ symbol in the first argument position is always interpreted as the loading address.

LINK \$8000, TEST1, \$1000, TEST2, TBL\$20, EXEC\$8200

↑                      ↑                      ↑                      ↑

Loading address      Offset (4K bytes)      Symbol table size (approx. 8K bytes)      Execution address

Note: Any loading address or execution address is invalid for LINK/S even if specified.

## LOADING ADDRESS AND EXECUTION ADDRESS (ORG STATEMENT)

Although a loading address can be specified with the linker, it can also be specified with the ORG assembler directive during assembly. Assume that there are two relocatable files.

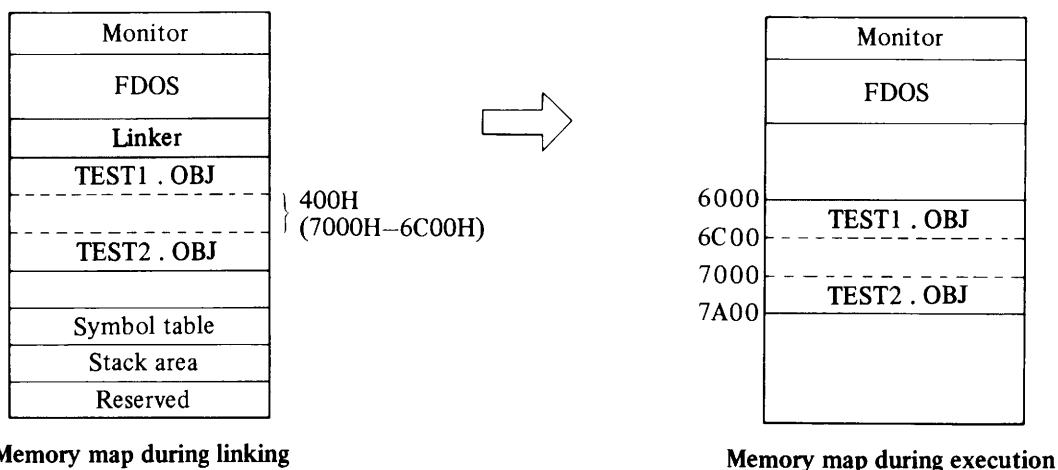
TEST1: Assembled with loading address 6000H specified. The object file will be loaded in the area from 6000H through 6C00H.

TEST2: Assembled with loading address 7000H specified. The object file will be loaded in the area from 7000H through 7A00H.

These are linked as follows.

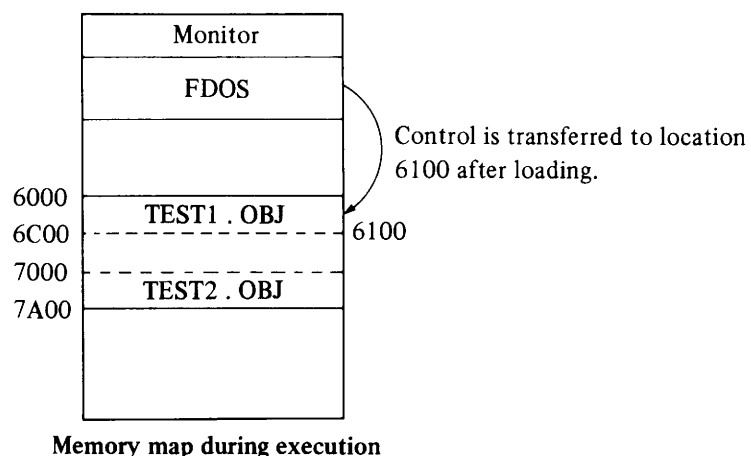
LINK TEST1, TEST2

Then, the object files are loaded as shown in the memory map below and the execution address of TEST1 . OBJ is automatically set to 6000H.



The loading addresses specified during assembly are valid even if the loading addresses and offsets are specified in the LINK command. However, when no loading address is specified for TEST2 during assembly, the offset specified in the LINK command is valid. The execution address specified in the LINK command is valid.

LINK \$5000, TEST1, \$3000, TEST2, EXEC\$6100



Loading addresses specified during assembly are invalid when the LINK/S command is used to generate a system file.

# SYMBOL TABLE

Information referred to as symbols in the linker and symbolic debugger indicates globally declared labels (that is, label symbols defined by the ENT or EQU assembler directive) in the source program. This information is stored in the relocatable file by the assembler for use in linking with other relocatable programs.

The linker loads label symbols into the symbol table while inputting program units in the relocatable files. The symbol table is placed at the end of the link area; its size is set to approximately 6K bytes by the linker unless otherwise specified by the programmer. The programmer can specify an area of more than 6K bytes for the symbol table area using the LINK command as follows:

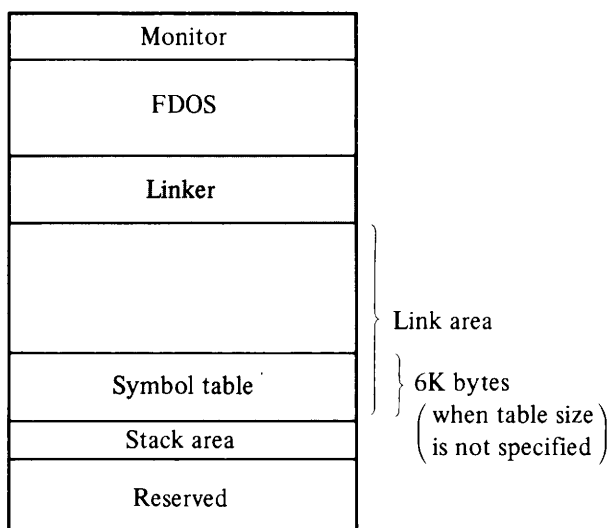
**LINK TEST1, TEST2, TBL\$20**

This command links TEST1 and TEST2 and specifies a symbol table size of 2000H (approximately 8K bytes).

TBL\$20 in the above command specifies that a symbol table of approximately 8K bytes is to be created. In other words, the programmer can reserve a symbol table area in 256-byte units. As shown in the memory map, the symbol table is constructed at the end of the link area.

Each symbol table entry is 9 bytes long. The format of the symbol table entry is shown at right. Section 2.3, "Linker" in the System Command manual describes how the linker uses this 9-byte information to link relocatable program units.

|             |   |   |   |   |                   |   |         |   |
|-------------|---|---|---|---|-------------------|---|---------|---|
| 1           | 2 | 3 | 4 | 5 | 6                 | 7 | 8       | 9 |
| Symbol name |   |   |   |   | Definition status |   | Address |   |



Linker memory map

# LINK/T COMMAND

The LINK/T command is used to display the contents of the symbol table after program linking is completed. It displays a symbol name, its absolute address (in hexadecimal representation) and the definition status for each symbol table entry. The user can detect symbol definition errors by checking the definition status.

The LINK/T command has two basic formats:

**LINK/T TEST1, TEST2**

Links TEST1 and TEST2 and displays the symbol table on the CRT screen.

**LINK/T/P TEST1, TEST2**

Links TEST1 and TEST2 and prints the symbol table on the printer.

— The photo at right shows link and symbol table information displayed on the CRT screen with the LINK/T command for the three program units shown on Page 11. Undefined symbols are labeled "U".

— Symbol definition messages are listed below.

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

No message is attached to symbols for which an address has been defined. U, M, X and H indicate error conditions.

— If global switch /T is not specified, only error symbols (whose definition messages are U, M, X or H) are displayed or printed.

```

2>LINK/T UNIT-#1,UNIT-#2,UNIT-#3
Linking UNIT-#1.RB
Top asm.bias $4A00
End asm.bias $4A1A
Linking UNIT-#2.RB
Top asm.bias $4A1A
End asm.bias $4A2B
Linking UNIT-#3.RB
Top asm.bias $4A2B
End asm.bias $4A42
Save UNIT-#1.OBJ
Loading address $4A00
Execute address $4A00
Bytesize $0042

Symbol table
COUNT 4A00 X COUNT0 D 0050
COUNT1 D 0000 COUNT2 D 0000 H
END 4A10 M START 5E55 X
IMCLVL 5E80 U
2>

```

The listing below shows a printout of link and symbol table information. The symbol table entries have been sorted as may be seen from this listing.

```

Linking M-LANG#1.RB
 Top asm.bias $4A00
 End asm.bias $5678
Linking M-LANG#2.RB
 Top asm.bias $5678
 End asm.bias $5B14
Linking MONEQU.LIB
 Top asm.bias $5B14
 End asm.bias $5B14
Save M-LANG.OBJ
 Loading address $4A00
 Execute address $4A00
 Bytesize $1114

```

```

Symbol table
&MSG 57C2 &NL 57B4 &PRNT 57A1 &PRNTS 5796
1HEX0 574D 1SET 57F7 2HEX0 5760 2SET 5823
4HEX0 5776 4SET 5832 8880T 5888 ??KEY 0D77
?FEED 52D6 ?TABP 5374 @ERR1 D 0003 @ERR2 D 0004
@ERR3 D 0005 @ERR4 D 0006 ACCUM 4BF7 ARST7 D 0038
BDRIVE 4CF9 BELL 0A80 BKTBL 5A19 BPDUM 5A58
BPFLG 5A57 BPSIM 5A40 BREAD 4F9B BREAK 4C7C
BRKEY 0527 BRST8 D 0039 BUFR 5A74 BUSY 4F3A
BWRIT 50A4 CLBF0 4CED CLBF1 4CEE CLBF2 4CEF
CLBF3 4CF0 CLBP 5678 CLEAR 4A82 CMD 5A3F
COMMON 5511 COMPL 4BFC COMPR 5688 CONT 5565
CONT9 5635 CR D 00D8 CTBL 58E6 CURSOL 5375
DI 0E0E DM D 00DC DR D 00DB EFREE D FFF0
ERCODE 4CF2 ERJMP 4CF5 ERJPAD 4CF6 ERSECT 4CF4
ERTRK 4CF3 ESCPRT 52DA FCMD 4CEB FDERR 4AB5
GET1 56DF GET1K 5741 GET2 5716 GET4 5731
GET44 572C GETKY 0610 GETL 0BE5 GOTO 4CB0
HS D 00DD IBUF 1180 JRTBL 58B2 LFLG 5A5A
LIST 4A9F LISTA 529C LISTM 5263 LISTN 5256
LISTS 529A LOAD 4ABD LOOK0 586B LOOK1 5878
LPNT 5A3D MAIN1 4A3A MAIN2 4A48 MELDY 0AA3
MEMRY 4BA1 MES0 5914 MES1 5933 MES10 598A
MES11 5990 MES12 59A8 MES13 59C1 MES14 59D8
MES15 59E3 MES16 59E4 MES17 59E7 MES18 59ED
MES19 59F3 MES2 593D MES20 59F7 MES21 59F9
MES22 59FE MES23 5A05 MES24 5A0B MES25 5A13
MES3 5944 MES4 594E MES5 5959 MES6 5966
MES7 5972 MES8 597E MES9 5985 MSG 06B5
MTFG 4CEC MTOFF 4D71 MTON 4D44 NL 0757
NLMSG 57BF PMSGX 5279 PNL 5251 POTFE D 00FE
POTFF D 00FF PRNT 063C PRNTS 063A PROG 4C01
PROTOC 57CF PRTAB 52AE PTAB0 58BD PTAB1 58C3
PUSHR 0DF1 PWORK 58C6 RCLB 4F13 READY 4CFC
REGST 4C07 RETRY 4CF1 RSTRT 552B SACC 569B
SAVE 4ABA SCOMP 56B9 SCR D 00DA SEARCH 583D
SEEK 4D5A SFREE D 0000 SKPBL 57E7 SOUND 568E
SPROG 56C1 STAFG 5222 START 4A00 TR D 00D9
TYPE0 577F TYPE1 5782 TYPE2 578C TYPE3 578F
VERIFY 5182 VRFONT 4CF8 WARN 5A48 WRITE 4C13
WRK0 5A4E WRK1 5A4F WRK2 5A50 WRK3 5A51
WRK4 5A52 WRK5 5A53 WRK6 5A54 X1HEX 5752
X2HEX 5765 XFER 4B30 XGET1 56E2 XGET2 5719
XGET22 571D XGET4 5734 XTEMP 09BE Z80TB 589D
ZAF 5A5B ZAF0 5A63 ZBC 5A5D ZBCC 5A65
ZDE 5A5F ZDEC 5A67 ZHL 5A61 ZHLC 5A69
ZIR 5A73 ZIX 5A6F ZIY 5A71 ZPC 5A6B
ZSP 5A6D

```

(Note: This listing is not related to the programs on page 11.)

# LINK 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

```

Refer to photo on page 9.

### "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 statement.

#### Note:

The EQU statement 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 statement.

### "COUNT1" D

COUNT1 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).

## Second program unit loaded (UNIT-#2)

```

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

```

## Third program unit loaded (UNIT-#3)

```

INPUT : CALL 001BH
 CALL TMDLYL
 CALL 001BH
 LD HL, START
 CP 0DH
 JR Z, END
 LD (HL), A
 INC HL
 JR INPUT
END : JP 0000H
COUNT2 : EQU 12
 END

```

### "TMDLYL" U

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

# ERROR MESSAGES

The error messages issued by the linker are described in the System Command manual. Here, only error messages which require particular attention are described.

## **no memory space**

Indicates that the symbol table is full; that is, that there are too many symbols to be cataloged. The symbol table size is set to approximately 6K bytes by the linker unless specified by the programmer. It is necessary to specify the TBL\$ argument in the LINK command to increase or decrease the symbol table size.

## **memory protection**

Indicates that the link area is inadequate, that is, that the linked data has reached the symbol table area located at the end of the link area. In this case, MLINK command is available.

## **il data**

Indicates that the data read from the specified relocatable file has an illegal link format. This condition may be caused by a hardware read error in the floppy disk drive or by an assembly error in the source program.

Personal Computer  
**MZ-80B**

**SHARP**

### NOTICE

The MZ-80 series of sophisticated personal computers is manufactured by the SHARP CORPORATION. Hardware and software specifications are subject to change without prior notice; therefore, you are requested to pay special attention to version numbers of the monitor and the system software (supplied in the form of cassette tape or mini-floppy disk files).

This manual is for reference only and the SHARP CORPORATION will not be responsible for difficulties arising out of inconsistencies caused by version changes, typographical errors of omissions in the descriptions.

This manual is based on the SB-1500 series monitor and the SB-7000 series FDOS.

# — CONTENTS —

|                                                               | U/PROM    |
|---------------------------------------------------------------|-----------|
| <b>PROM FORMATTER</b> .....                                   | 1         |
| Activation of the PROM Formatter .....                        | 1         |
| <b>PROM WRITER FORMATS</b> .....                              | 3         |
| BNPF .....                                                    | 3         |
| B10F .....                                                    | 4         |
| HEXADECIMAL .....                                             | 5         |
| BINARY .....                                                  | 6         |
| Performance Boards of Various Companies .....                 | 7         |
| <b>PROM FORMATTER COMMANDS</b> .....                          | 9         |
| File Input/Output Commands .....                              | 9         |
| Y (Yank file) Command .....                                   | 9         |
| S (Save file) Command .....                                   | 9         |
| CY (Yank disk) Command .....                                  | 10        |
| CS (Save disk) Command .....                                  | 10        |
| Formatting Commands .....                                     | 11        |
| P (Punch) Command .....                                       | 11        |
| R (Read) Command .....                                        | 11        |
| Other Commands .....                                          | 12        |
| M (Memory dump/modify) Command .....                          | 12        |
| V (Verify) Command .....                                      | 13        |
| \ (FDOS) Command .....                                        | 13        |
| # (Change printer mode) Command .....                         | 13        |
| & (Clear) Command .....                                       | 13        |
| ? (Disp free area) Command .....                              | 14        |
| ! (Return) Command .....                                      | 14        |
| Format Commands .....                                         | 14        |
| <b>PROM FORMATTER (SB-7501) COMMANDS &amp; MESSAGES</b> ..... | 16        |
|                                                               | U/PLOTTER |
| <b>EXAMPLE OF PLOTTER CONTROL APPLICATION</b> .....           | 1         |
| Interface Card .....                                          | 1         |
| Plotter Control Program .....                                 | 1         |
| 1. Conditions for linkage with a BASIC program .....          | 1         |
| 2. Linkage conditions when an error occurs .....              | 2         |
| 3. Use of external subroutines .....                          | 2         |
| 4. Plotter control codes .....                                | 2         |
| 5. Program outline .....                                      | 3         |
| Command Table .....                                           | 6         |
| Outline of the BASIC Program .....                            | 7         |
| Sample Program (Plotter Control Routines) .....               | 10        |
| BASIC Main Program .....                                      | 19        |

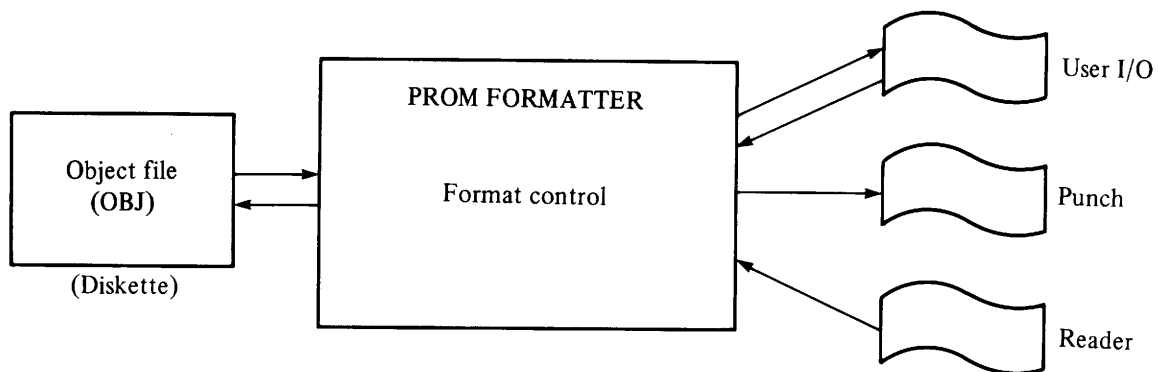


# PROM FORMATTER

The rapid advances in LSI technology have allowed the functions of a computer's CPU to be concentrated onto a single semiconductor chip. These microprocessors are becoming ever more sophisticated, while at the same time they are becoming less expensive. As a result, the range of fields in which microprocessors are being utilized is growing rapidly. One subject of great importance to the development of new device applications is that of developing efficient application programs; it is not too much to say that the quality of the application program determines how well a newly developed device performs.

On the other hand, developments in LSI technology have also stimulated efforts to develop low cost, large capacity memory elements (RAM and ROM). The increased availability of PROMs which are erasable with ultraviolet rays has had a particularly strong influence on the development of devices which incorporate microprocessors.

The procedure which is most suitable for efficiently developing application programs is to create an object file from a source file created through assembly programming using an assembly language, then to reassemble it after debugging. The function of the PROM formatter is to load one or more object programs created with the assembler and linker, then to output it to a paper tape punch after converting it to PROM writer format.



Functions of the PROM formatter

It also allows programs which are written in different formats to be input from a reader for storage on diskette and enables conversion of programs to the required format for output on paper tape or the like.

## —Activation of the PROM Formatter—

Entering PROM  while in the FDOS command entry mode activates the PROM formatter. Commands may be entered as soon as activation is completed.

The following formats are provided for in the PROM formatter:

### 1. BNPF

- Britronics
- Intel
- Takeda Riken

### 2. B10F

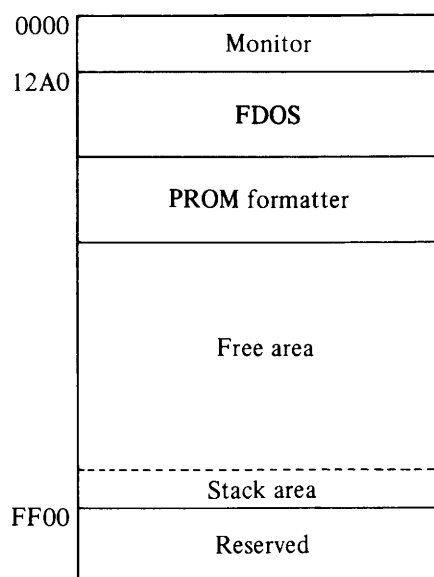
- Takeda Riken

### 3. HEXADECIMAL

- Britronics
- Takeda Riken
- Minato Electronics

### 4. BINARY

- Britronics



PROM formatter memory map

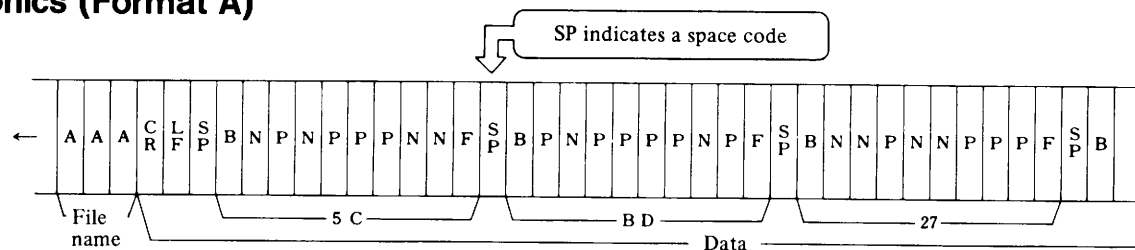
# PROM WRITER FORMATS

PROM writers are provided in many formats by different companies. This section discusses forms which are converted by the PROM formatter; refer to the individual PROM writer manuals for details.

The examples in the figures include the file name "AAA", the address "0000", and the data "5C", "BD" and "27". The leader section for the start of punched output and the trailer section for the end of punched output are created automatically.

## —BNPF—

### Britronics (Format A)

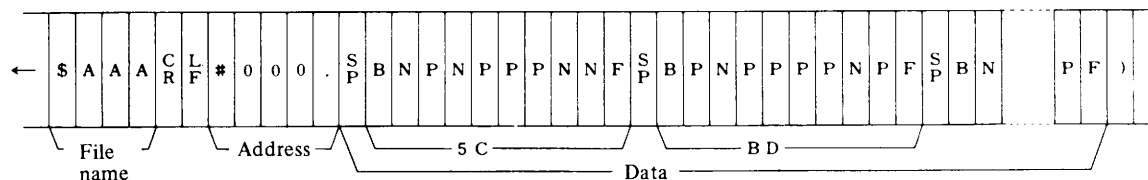


- The file name is punched in ASCII code (if one is specified). (Using the character "B" as a file name will result in incorrect identification of the beginning of data.)
- **CR** and **LF** are punched in ASCII code.
- The space code (20H) and the byte of data at the address specified for "RAM from?" are punched in BNPF format. The address is incremented successively.
- **CR** and **LF** are punched after each 6 items of data are punched in the BNPF format.
- Punching is performed in BNPF format up to the address specified for "to?".

### Intel (Format D)

- This is the same as the Britronics format. The BNPF format is one which has a relatively high degree of standardization; thus, the PROM formatter can also be used with devices other than those which are discussed in this manual.

## Takeda Riken (Format E)

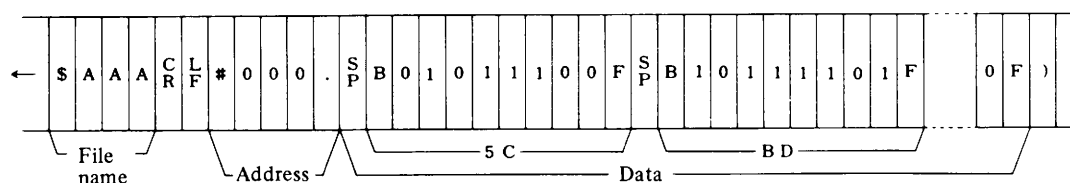


- The "\$" mark, which denotes the file name, is punched in ASCII code.
- The file name is punched in ASCII code (if one is specified).
- CR and LF are punched. The "\$" mark is regarded as denoting the beginning of a comment statement; the end of a comment statement is denoted with an LF code.
- The "#" mark (which indicates the beginning of an address) is punched, followed by the first three digits of the address specified for "PROM address?". The separator between the address and the data is punched as ".".
- The data item at the address specified for "RAM from?" is punched in BNPF format. The address is incremented successively.
- CR and LF are punched after each 6 items of data are punched in the BNPF format.
- A tape leader stop mark ")" is punched after the data has been punched up to the address specified for "to?".

Note: Care must be taken to ensure that characters which act as control characters (B, :, \$, #, etc.) are not used when a file name is specified. (Otherwise, incorrect operation will result.)

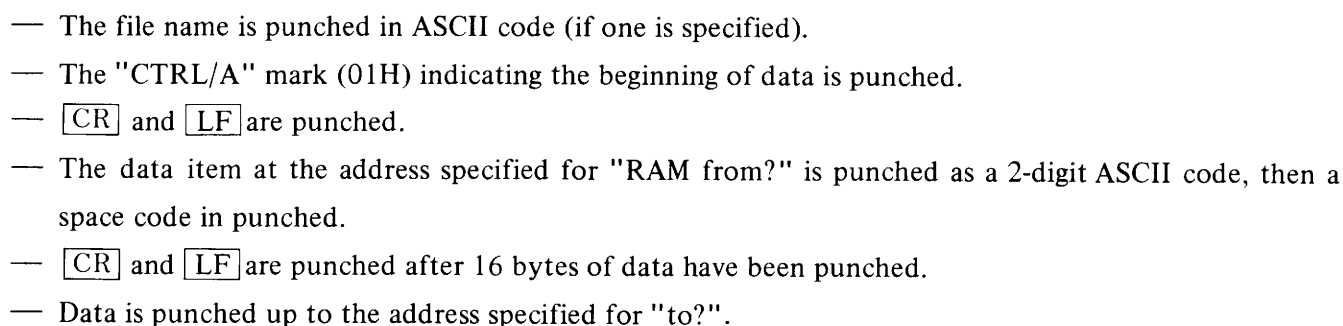
## —B10F—

### Takeda Riken (Format F)



- Except for the NP section, this is the same as Takeda Riken's BNPF format.
- The B10F format corresponds to the BNPF format in that 1 = P and 0 = N.

### Britronics (Format B)

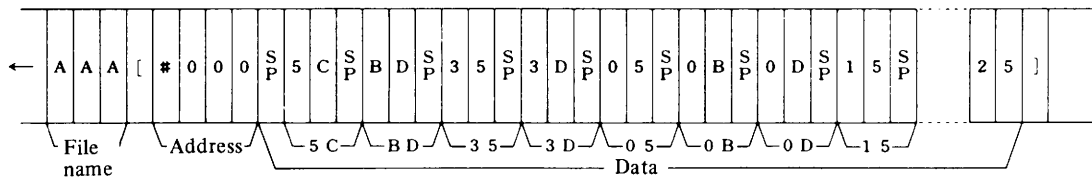


← \$ A A A C R # 0 0 0 · S P 5 C · S P B D · S P 3 5 · S P 3 D · S P 0 5 · S P 0 B D · )

File name Address 5 C B D 3 5 3 D 0 5 Data

- U/PROM-5**

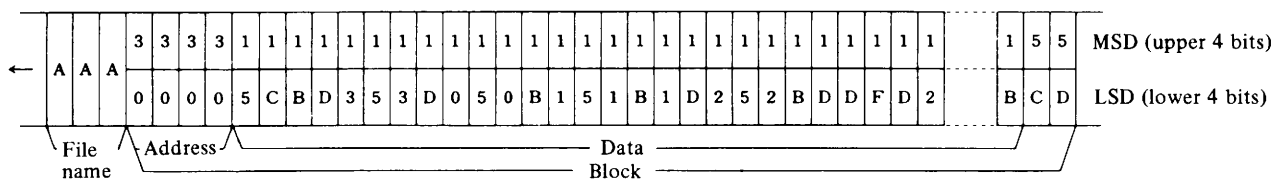
## Minato Electronics (Format H)



- The file name is punched in ASCII code when file name is specified.
- The start-of-data mark " [ " is punched.
- The address designation mark " # " is punched, followed by a 3-digit ASCII code for the address specified for "PROM address?".
- A space code is punched, then the data at the address specified for "RAM from?" is converted to a 2-digit ASCII code and punched.
- 16 combinations of space codes and data items are punched, then CR and LF are punched.
- The end of data mark " ] " is punched after data has been punched up to the address specified for "to?".

**—BINARY—**

### Britronics (Format C)



- In the binary format, the 4-bit mark section and the 4-bit data section are expressed together as one character (8 bits). The mark section is punched as the upper 4 bits of the paper tape, while the data section is punched as the lower 4 bits.
- The file name is punched in ASCII code (if one is specified). Specifications which result in a "3" in the upper 4 bits of the ASCII code file name are not permitted. Such specifications will result in incorrect operation, since incorrect determination that the lower 4 bits of the file name are an address will result.
- Three binary digits for the address specified for "PROM address?" are punched in the lower 4 bits. The address designation mark ("3") is punched in the upper 4 bits.
- A data mark ("1") is punched in the upper 4 bits and data at the address specified for "RAM from?" is punched in the lower 4 bits.
- Data is punched 4 bits at a time (with the upper and lower 4 bits punched in alternation) up to the address specified for "to?".
- Check sum marks ("5") are punched in the upper 4 bits, followed in alternation by check sum data in the lower 4 bits.

**—Performance Boards of Various Companies—** (Note: Consult the various manufacturers for details.)

**a) Intel**

2716  
 2732  
 8748/8741  
 3621, 3602, 3622, 3602A, 3622A, 3604, 3624, 3604A, 3624A, 3605, 3625, 3605A, 3625A, 3628, 3608,  
 3604AL-6, 3604AL  
 8702A/1702A  
 8708/8704/2708/2704  
 8755A

**b) Britronics**

| Company                | Element                                                                                                                                                                                                                                                                                                                                                               |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Intel                  | 3602 A / 22 A, 3604 A / 24 A, 3604 A L / 24 L, 3605 / 25, 3608 / 28                                                                                                                                                                                                                                                                                                   |
| Intersil               | 5600 / 10, 5603 A / 23, 5604 / 24, 5605 / 25                                                                                                                                                                                                                                                                                                                          |
| Fujitsu                | 7055, 7051, 7052, 7058, 7053, 7059, 7054, 7057                                                                                                                                                                                                                                                                                                                        |
| Monolithic Memory      | 5330 / 6330, 5331 / 6331, 5300 / 6300, 5335 / 6335, 5336 / 6336, 5308 / 6308, 5309 / 6309, 53134 / 63134, 53135 / 63135, 5305 / 6305, 5306 / 6306, 53137 / 63137, 53141 / 63141, 5340 / 6340, 5341 / 6341, 5348 / 6348, 5349 / 6349, 5350 / 6350, 5351 / 6351, 5352 / 6352, 5353 / 6353, 5380 / 6380, 5381 / 6381, 5384 / 6384, 5385 / 6385, 5386 / 6386, 5387 / 6387 |
| Harris                 | 7602 / 03, 7610 A / 11 A, 7620 A / 21 A, 7640 A / 41 A, 7640 A R / 41 A R, 7642 / 43, 7644, 7646 R / 47 R, 7648 / 49, 7608, 7680 / 81, 7680 R / 81 R, 7680 P / 81 P, 7680 R P / 81 R P, 7683, 7684 / 85, 7684 P / 85 P, 7686 / 87, 7686 R / 87 R, 7686 P / 87 P, 7686 R P / 87 R P                                                                                    |
| Fairchild              | 93417 / 27, 93436 / 36, 93438 / 48, 93452 / 52                                                                                                                                                                                                                                                                                                                        |
| National Semiconductor | 54 / 74 S 387, 54 / 74 S 287, 54 / 74 S 470, 54 / 74 S 471, 54 / 74 S 570, 54 / 74 S 571, 77 / 87 S 295, 77 / 87 S 296, 54 / 74 S 473, 54 / 74 S 472, 54 / 74 S 572, 54 / 74 S 573                                                                                                                                                                                    |
| NEC                    | 403 D, 406 D                                                                                                                                                                                                                                                                                                                                                          |
| Raytheon               | 29660 / 61, 29600 / 01, 29612 / 13                                                                                                                                                                                                                                                                                                                                    |
| Signetics              | 82 S 114 / 115, 82 S 126 / 127, 82 S 130 / 131, 82 S 140 / 141, 82 S 136 / 137, 82 S 180 / 181, 82 S 2708, 82 S 184 / 185, 82 S 190 / 191                                                                                                                                                                                                                             |
| Texas Instruments      | 54 / 7488A, 54 / 74 S / 88, 54 / 74 S 288, 54 / 74 S 470, 54 / 74 S 71, 54 / 74 S 73, 54 / 74 S 72, 54 / 74 S 75                                                                                                                                                                                                                                                      |

**c) Minato Electronics**

Adaptable to all PROMs.

# d) Takeda Riken

## MOS Type

| Element      | Bit configuration, capacity<br>(words x bits = capacity) | Maker name |                      |                  |            |            |          |                       |                                             |           |          |
|--------------|----------------------------------------------------------|------------|----------------------|------------------|------------|------------|----------|-----------------------|---------------------------------------------|-----------|----------|
|              |                                                          | Okai Denki | Toshiba              | Fujitsu          | Hitachi    | Mitsubishi | Intersil | Intel                 | Texas Instruments                           | Fairchild | AMD      |
| MOS          | 256 x 8 = 2048                                           |            |                      | MB8503<br>MB8513 | HN351702A⑦ | M5L1702⑦   |          | 1602A<br>1702A⑦       |                                             |           | AM1702A⑦ |
|              | 512 x 4 = 2048                                           |            | TMM121C<br>TMM121C-1 |                  |            |            |          |                       |                                             |           |          |
|              | 512 x 8 = 4096                                           |            |                      |                  |            |            |          | 2704<br>2708<br>2758⑧ |                                             |           |          |
|              | 1024 x 8 = 8192                                          |            |                      |                  |            |            |          | 2716<br>2732          | TMS2708⑧<br>TMS2716⑧<br>TMS2516⑧<br>TMS2532 | F2708 ⑧   | AM2708 ⑧ |
|              | 2048 x 8 = 16384                                         |            |                      |                  |            |            |          |                       |                                             |           |          |
| CMOS         | 4096 x 8 = 32768                                         |            |                      |                  |            |            | 1M6654④  |                       |                                             |           |          |
|              | 512 x 8 = 4096                                           |            |                      |                  |            |            | 1M6653④  |                       |                                             |           |          |
|              | 1024 x 4 = 4096                                          |            |                      |                  |            |            |          |                       |                                             |           |          |
|              | 1024 x 8 = 8192                                          |            |                      |                  |            | M5B460 S   |          | 8755                  |                                             |           |          |
| Compound MOS |                                                          |            |                      |                  |            |            |          |                       |                                             |           |          |

## Bipolar Type

| Element | Bit configuration, capacity<br>(words x bits = capacity) | Maker name              |         |                    |            |                   |                |                                    |                                                                                                                                |                |                                                                     |                               |                             |                  |
|---------|----------------------------------------------------------|-------------------------|---------|--------------------|------------|-------------------|----------------|------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|----------------|---------------------------------------------------------------------|-------------------------------|-----------------------------|------------------|
|         |                                                          | Nihon Denki             | Hitachi | Fujitsu            | Mitsubishi | Intersil          | Intel          | Texas Instruments                  | Harris                                                                                                                         | Fairchild      | Raytheon                                                            | MMI                           | AMD                         | SIGNE            |
| Bipolar | 32 x 8 = 256                                             |                         |         | MB7051<br>MB7056   | M54730     | 1M5600<br>1M5610  |                | SN74188A<br>SN74S188<br>SN74S288   | HM7602<br>HM7603                                                                                                               | HM761S03       | 6330-1<br>6331-1                                                    | AM27S18<br>AM27S19            | 82S23<br>82S123             |                  |
|         | 256 x 4 = 1024                                           | μPB403D②                |         | MB7052<br>MB7057②  | M54700     | 1M5603A<br>1M5623 | 3601<br>3621   | SN74S287<br>SN74S387               | HM7610A<br>HM7611A<br>HM7611                                                                                                   | 93417<br>93427 | 29662<br>29663<br>29660<br>29661                                    | AM27S20<br>AM27S21            | 82S126<br>82S129            |                  |
|         | 256 x 8 = 2048                                           |                         |         |                    |            |                   |                | SN74S470 ③<br>SN74S471             | HM7625R<br>HM7629                                                                                                              |                | 6308-1<br>6309-1<br>6335-1<br>6336-1<br>63135-1                     |                               | 82S114                      |                  |
|         | 512 x 4 = 2048                                           |                         |         | MB7053②<br>MB7058  |            | 1M5604<br>1M5624  | 3602<br>3622   |                                    | HM7620<br>HM7621<br>HM7621A                                                                                                    | 93436<br>93446 | 29612<br>29613                                                      | AM27S12<br>AM27S13            | 82S130<br>82S131            |                  |
|         | 512 x 8 = 4096                                           | μPB405D<br>μPB425D      |         |                    |            | 1M5605<br>1M5625  | 3604A<br>3624A | SN74S473<br>SN74S474 ③<br>SN74S475 | HM7640<br>HM7641<br>HM7640A R<br>HM7641A R<br>HM7647R<br>HM7648<br>HM7649                                                      | 93438<br>93448 | 6341-1<br>6340-1<br>6348-1<br>6349-1                                | AM27S15<br>AM27S26<br>AM27S27 | 82S115<br>82S140<br>82S141  |                  |
|         | 1024 x 4 = 4096                                          | μPB406D ⑤<br>μPB426D ⑤  |         | MB7054<br>MB7059 ⑤ |            | 1M5606<br>1M5626  | 3605A<br>3625A |                                    | HM7642<br>HM7643<br>HM7644<br>HM7642P<br>HM7643P                                                                               | 93452<br>93453 | 6350-1<br>6351-1<br>6352-1<br>6353-1                                | AM27S32<br>AM27S33            | 82S136<br>82S137            |                  |
|         | 1024 x 8 = 8192                                          | μPB417 D ⑥<br>μPB427D ⑥ |         | MB7055<br>MB7060 ⑥ |            |                   | 3608A<br>3628A |                                    | HM7680<br>HM7681<br>HM7680P<br>HM7681R<br>HM7680RP<br>HM7681RP<br>HM7608<br>HM7683                                             | 93450<br>93451 | 6380-1<br>6381-1<br>6384-1<br>6385-1<br>6386-1<br>6387-1<br>63133-1 |                               | 82S180<br>82S181<br>82S2708 |                  |
|         | 2048 x 4 = 8192                                          |                         |         |                    |            |                   |                |                                    | HM7684<br>HM7685<br>HM7684P<br>HM7685P<br>HM7686<br>HM7687<br>HM7686P<br>HM7687P<br>HM7686R<br>HM7687R<br>HM7686RP<br>HM7687RP |                |                                                                     |                               | 82S184<br>82S185            |                  |
|         | 2048 x 8 = 16384                                         |                         |         |                    |            |                   | 3636           |                                    | HM7616<br>HM76160<br>HM76161                                                                                                   |                |                                                                     |                               |                             | 82S190<br>82S191 |

Elements annotated with the same figures in circles can be used with the same performance boards.

# PROM FORMATTER COMMANDS

## —File Input/Output Commands—

### Y (Yank file) Command

Reads the object (OBJ) file specified by the file name into the free area.

\* YCHARAGEN   
RAM from? 8000 to 87FF

Reads in CHARAGEN.OBJ  
Specifies read-in from address 8000  
(read up to address 87FF)

- File name can be specified by entering a Y (Yank file command) when "\*" appears to indicate that command entry is awaited.
- Specify the starting address of the file to be read in as a 4-digit hexadecimal number. (Reading will start at the address specified regardless of the actual data address of the object file.)
- The last address read is displayed when file read-in is completed.

**Caution:** The address specified for read-in must be in the free area.

### S (Save file) Command

Writes the specified program (or data) in the free area onto a diskette.

\* STEST# 2   
RAM from? 8400 to? 87E7  
exec? 1300 data? 1300

Output program (data) to TEST# 2.OBJ  
Output program (data) from address 8400 to 87E7  
Execute address 1300, data address 1300

- Another file name can be specified by entering an S (Save file) command when "\*" appears to indicate that command entry is awaited.
- The addresses of the memory block in the free area which is to be output are specified with 4-digit hexadecimal numbers.
- The execute address and data address of the object (OBJ) file created are specified with 4-digit hexadecimal numbers. The data address is the address to which the program (data) is to be reloaded into memory by a later RUN or LOAD command. The execute address is the address from which a program reloaded into memory is to be executed. Specify 0000 when either of these addresses is not necessary.

## CY (Yank disk) Command

This command loads data in units of 256 bytes from the specified sector(s) of the specified track on the floppy disk in the specified drive into RAM.

```
* CY
drive? 1
track, sect? 0301
byte size? 0100 adrs? 7000
```

- Enter the CY command when "\*" appears to indicate that command entry is awaited.
- Specify the number of disk drive containing the floppy disk storing the data to be loaded.
- Enter the track number and the first of the sector numbers in which the data to be loaded is stored as 4 continuous hexadecimal digits.
- Enter the 4 hexadecimal digits which indicate the number of 256 byte units of data, then enter the starting address (4 hexadecimal digits) of the RAM area into which the data is to be loaded.
- The 4-digit hexadecimal number which indicates the amount of data is constructed as shown below.

XX XX

— No numbers other than 0 should be entered in these two places (otherwise, the number of data units specified in first two places will be affected).

— Indicate the number of 256-byte units of data to be loaded.

Ex.) 512 bytes of data are loaded when 0200 is entered (0101 is also interpreted as 0200).

## CS (Save disk) Command

This command saves data in 256-byte units from RAM memory in the specified sector(s) of the specified track of the floppy disk in the specified disk drive.

```
* CS
drive? 2
track, sect? 3002
byte size? 0100 adrs? 7000
```

- Enter the CS command when "\*" appears to indicate that command entry is awaited.
- Specify the number of the disk drive containing the floppy disk on which the data is to be saved.
- Enter the track number and the first of the sector numbers in which the data is to be saved as a 4-digit hexadecimal number.
- Enter the 4-digit hexadecimal number which indicates the number of 256 byte units of data, then enter the starting address (4 hexadecimal digits) of the RAM area from which the data is to be saved.
- The 4-digit hexadecimal number which indicates the amount of data is constructed in the same manner as for the CY command.

For example, 0101 is interpreted as 0200.

- The track number must be 3 or greater. Great care must be taken not to destroy existing data on the floppy disk.

## —Formatting Commands—

### P (Punch) Command

Punches data in the free area in the specified format.

|                                         |                                                     |
|-----------------------------------------|-----------------------------------------------------|
| * P <input type="text"/>                | Punch command                                       |
| filename? CHARAGEN <input type="text"/> | File name assigned to the paper tape to be punched. |
| format? C <input type="text"/>          | Format C                                            |
| RAM from? 8000 to? 87FF                 | Addresses 8000 to 87FF in the free area             |
| PROM address? 0000                      | PROM write address 0000                             |

- Enter the P (Punch) command when "\*" appears to indicate that command entry is awaited.
- Next, the file name is specified. This is not the file name which is included on the diskette, but the name which is to be punched at the beginning of the tape. Refer to the explanations of the various formats for details. When no file name is needed, enter only .
- Next, specify the conversion format (A~H) and enter .
- Specify the starting and ending addresses of the memory block in the free area which is to be output with 4-digit hexadecimal numbers.
- Finally, specify the PROM write address. (This step may not be required, depending on the format.)

The P command described above outputs formatted data to a PTP device. (More precisely, \$PTP/LF is used as the output device.)

The PROM FORMATTER can also output converted format data to devices other than PTP (including user I/O and diskette).

|         |                                    |                                                                                                        |
|---------|------------------------------------|--------------------------------------------------------------------------------------------------------|
| (Ex. 1) | *PSUSR1 <input type="text"/>       | Outputs to user I/O                                                                                    |
| (Ex. 2) | *PXYZ <input type="text"/>         | Outputs file name XYZ.ASC to the diskette.                                                             |
| (Ex. 3) | *P\$PTP/PE/LF <input type="text"/> | Adds even parity to PTP, affixes <input type="text"/> after <input type="text"/> and outputs the file. |

As an application, data may be sent directly to the PROM writer by creating hardware and user routines for its online interface.

### R (Read) Command

This command reads in data formatted in the BNPF, HEXADECIMAL or other format from a paper tape reader.

|                                |                                                           |
|--------------------------------|-----------------------------------------------------------|
| * R <input type="text"/>       | Read command                                              |
| format? C <input type="text"/> | Format C                                                  |
| RAM from? 8000 to 83FF         | Addresses in the free area into which data is to be read. |
| filename PROM#2                |                                                           |

- Enter the **R** (Read) command when "\*" appears to indicate that command entry is awaited.
- Next, specify the format of the data to be read.
- Finally, specify the starting address of the free area into which the data is to be read with a 4-digit hexadecimal number.
- The last data address and the file name are displayed after the read is completed and entry of the next command is awaited.
- With this PROM writer format, it may not be possible to read tapes punched using other programs because of the need to maintain a certain minimum degree of redundancy.

The R command described above reads in formatted data from a PTR device, (More precisely, \$PTR is used as the input device.)

The PROM FORMATTER can also read in converted format data from devices other than PTR (including user I/O and diskette).

|         |            |                                 |                                                |
|---------|------------|---------------------------------|------------------------------------------------|
| (Ex. 1) | *R\$USR2   | <input type="text" value="CR"/> | Inputs from user I/O                           |
| (Ex. 2) | *RXYZ      | <input type="text" value="CR"/> | Inputs from XYZ.ASC on a diskette              |
| (Ex. 3) | *R\$PTR/PE | <input type="text" value="CR"/> | Inputs data with even parity affixed from PTR. |

## —Other Commands—

### M (Memory dump/modify) Command

This command is used to display and modify the contents of the free area.

|                         |    |           |    |    |    |    |    |    |                      |   |   |   |   |   |   |   |  |  |
|-------------------------|----|-----------|----|----|----|----|----|----|----------------------|---|---|---|---|---|---|---|--|--|
| * M                     | CR | M command |    |    |    |    |    |    |                      |   |   |   |   |   |   |   |  |  |
| RAM from? 7000 to? 7014 |    |           |    |    |    |    |    |    | Area to be displayed |   |   |   |   |   |   |   |  |  |
| 7000                    | ED | 73        | C8 | 5F | 01 | C6 | 5F | 09 | m                    | S | H | □ | ↓ | F | □ | . |  |  |
| 7008                    | 60 | 22        | D8 | CD | 3E | 28 | CA | 27 | \                    | " | X | M | > | ( | J | ' |  |  |
| 7010                    | 28 | 22        | DE | 26 | CD |    |    |    | (                    | " | ^ | & | M |   |   |   |  |  |

- Enter the **M** (Memory dump/modify) command when "\*" appears to indicate that command entry is awaited.
- Next, specify the starting and ending addresses in the free area of the data to be displayed with 4-digit hexadecimal numbers.
- The PROM formatter, in the 40 (80) characters per line mode, divides data in the specified addresses into 8-byte segments and displays the address (4 hexadecimal digits), the 8 (16) bytes of data (as groups of 2 hexadecimal digits) and the 8 (16) corresponding ASCII characters in that order. However, when the corresponding ASCII character cannot be displayed, a "." is displayed in its place. Further, data is printed in 16 byte segments when the printer is used with the "#" command.
- Execution of the M command can be suspended or resumed by pressing . A switch can be made to the command entry mode by pressing .
- If no change is required in data displayed using the M command, just press . When a change is required, move the cursor to the position where the change is to be made and press  after entering the 2 new hexadecimal digits. (The change is made when  is pressed.) After data modification is completed, move the cursor to an empty line and press  to return to the command wait state.

- Data can also be changed using the cursor when display is halted with `SPACE`. In this case, display is resumed when the cursor is moved to an empty line and `CR` is pressed.

**Caution:** Data is only printed when the printer is used with the " # " command; modification of data is not possible in this case.

## V (Verify) Command

Reads data formatted in BNPF, HEXADECIMAL and so forth from the paper tape reader and compares it with the contents of the RAM free area.

|                           |                                                    |
|---------------------------|----------------------------------------------------|
| * V <code>CR</code>       | Verify command                                     |
| format? C <code>CR</code> | Format C                                           |
| RAM from? 8000 to 8615    | The start and end addresses of the area containing |
| filename ABC              | the data to be compared.                           |
| Verify OK.                |                                                    |

- Enter the V (Verify) command when " \* " appears to indicate that command entry is awaited.
- Specify the format of data to be read.
- Specify the start address (4-digit hexadecimal) of the area containing the data to be compared.
- The end address of the data read is displayed and "Verify OK." is displayed when the data read matches the data compared. If not, the end address is not displayed and "Verify error." is displayed.

## \ (FDOS) Command

This command invokes the specified built-in FDOS command. Command entry is awaited after execution of the FDOS command.

\* \ DIR `CR`

- Enter the \ command when " \* " appears to indicate that command entry is awaited.
- Next, specify the built-in FDOS command and press the `CR` key.
- The PROM formatter executes the built-in FDOS command, then awaits entry of the next PROM formatter command.
- The XFER and EXEC commands cannot be executed. The RUN command cannot be executed if the program executed by the RUN command is too long.

## # (Change printer mode) Command

This command starts and stops output to the printer. Printer output is OFF when the PROM formatter is activated, and is changed from ON to OFF to ON each time the " # " command is executed.

When printer output is ON, data is printed almost as it appears on the display screen.

## & (Clear) Command

Buries the entire free area in hexadecimal code FFH.

## **? (Disp free area) Command**

Displays the free area.

## **! (Return) Command**

Terminates the PROM formatter and returns to FDOS.

## **—Format Commands—**

Format commands are commands entered when "format?" is displayed during execution of the P, R or V commands. Selecting one of these commands during execution of the P command determines whether data is to be punched in BNPF, HEXADECIMAL or other format. Failure to specify the correct format command during execution of the R command will result in failure to correctly read the program into the free area.

### **A Command**

- Used to specify the Britronics BNPF format. The control character "B" may not be used when the file name is specified.

### **B Command**

- Used to specify the Britronics HEXADECIMAL format.

### **C Command**

- Used to specify the Britronics BINARY format. Numerals and the codes (; < = > ?) may not be used when the file name is specified.
- The message "PROM address?" is displayed during execution of the P command to request specification of the PROM loading address; specify it as a 4-digit number.
- Check sums are written following the data (with the P command).
- Data from the address specification to the check sums constitutes one block; if data is to be loaded into an address which has been skipped, the operation must be divided into two or more parts. This also applies when two or more blocks are read in with the R command.

### **D Command**

- This command is used to specify the Intel BNPF format. The character "B" cannot be used in the file name.

### **E Command**

- This command is used to specify the Takeda Riken BNPF format. The character "B" may be used in the file name.
- A file is a block which begins with " \$ " and ends with " ) ".
- The message "PROM address?" is displayed during execution of the P command to request specification of the PROM loading address; specify it as a 4-digit number.
- If two or more blocks are to be read out or written in, the operation must be divided into two or more parts.

## F Command

- This command is used to specify the Takeda Riken B10F format. The character "B" may be used in the file name.
- A file is a block which begins with "\$" and ends with ")".
- The message "PROM address?" is displayed during execution of the P command to request specification of the PROM loading address; specify it as a 4-digit number.
- If two or more blocks are to be read out or written in, the operation must be divided into two or more parts.

## G Command

- This command is used to specify the Takeda Riken HEXADECIMAL format.
- A file is block which begins with "\$" and ends with ")".
- The message "PROM address?" is displayed during execution of the P command to request specification of the PROM loading address; specify it as a 4-digit number.
- If two or more blocks are to be read out or written in, the operation must be divided into two or more parts.

## H Command

- This command is used to specify the Minato Electronics HEXADECIMAL format.
- The start-of-data symbol "[" may not be used in the file name.
- The message "PROM address?" is displayed during execution of the R command to request specification of the PROM loading address; specify it as a 4-digit number.
- Denote the end of data with the symbol "]".

# PROM FORMATTER (SB-7501) COMMANDS & MESSAGES

| COMMAND                        |                                                                               | OPERATION                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------------------------|-------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| File Input/<br>Output commands | Y (Yank)                                                                      | Loads a program (data) from the diskette into the free area.                                                                                                                                                                                                                                                                                                                                                    |
|                                | S (Save)                                                                      | Saves the program (data) in the free area on diskette.                                                                                                                                                                                                                                                                                                                                                          |
|                                | CY (Yank disk)                                                                | Loads data in 256-byte units from the specified sector(s) of the specified track on the diskette into RAM.                                                                                                                                                                                                                                                                                                      |
|                                | CS (Save disk)                                                                | Saves data in 256-byte units from RAM memory in the specified sector(s) of the specified track of the diskette.                                                                                                                                                                                                                                                                                                 |
| Format commands                | P (Punch)<br>R (Read)                                                         | Punches the specified contents of the free area in the specified format.<br>Reads in a paper tape punched in the format specified.                                                                                                                                                                                                                                                                              |
| Other commands                 | M (Memory)<br>V (Verify)<br><br>\ (FDOS)<br>#<br>& (Clear)<br>?<br>! (Return) | Displays and modifies data in the free area.<br>Reads data from the paper tape reader and compares it with the contents of the RAM free area.<br><br>Executes the specified built-in FDOS command.<br>Switches the list mode for listing on a printer.<br>Buries all data in the free area in hexadecimal code FFH.<br>Displays the starting and ending addresses of the free area.<br>Returns control to FDOS. |

| Error message      | Error content                                        | Related command  |
|--------------------|------------------------------------------------------|------------------|
| memory protection  | An address outside of the free area was specified.   | Y, S, P, R, M, V |
| il command         | The command was not entered correctly.               |                  |
| il data            | The format specified does not match the format read. | R, V             |
| check sum          | Check sum error.                                     | R, V             |
| \$ LPT : not ready | The printer is not ready.                            | #                |
| \$ PTP : not ready | The paper tape punch is not ready.                   | P                |
| \$ PTR : not ready | The paper tape reader is not ready.                  | R, V             |

See the "System Error Messages" in System Command for other error messages.

## Caution:

Entry of characters other than S, Y, CS, CY, P, R, M, V, \, &, #, ? or ! will cause a return to the command wait state after the command table is displayed.

If a character other than A~H is input while "format?" is displayed and format entry awaited, the format table will be displayed, after which the format entry wait state will be reentered. A return can be made to the command wait state at this time by pressing **BREAK**.

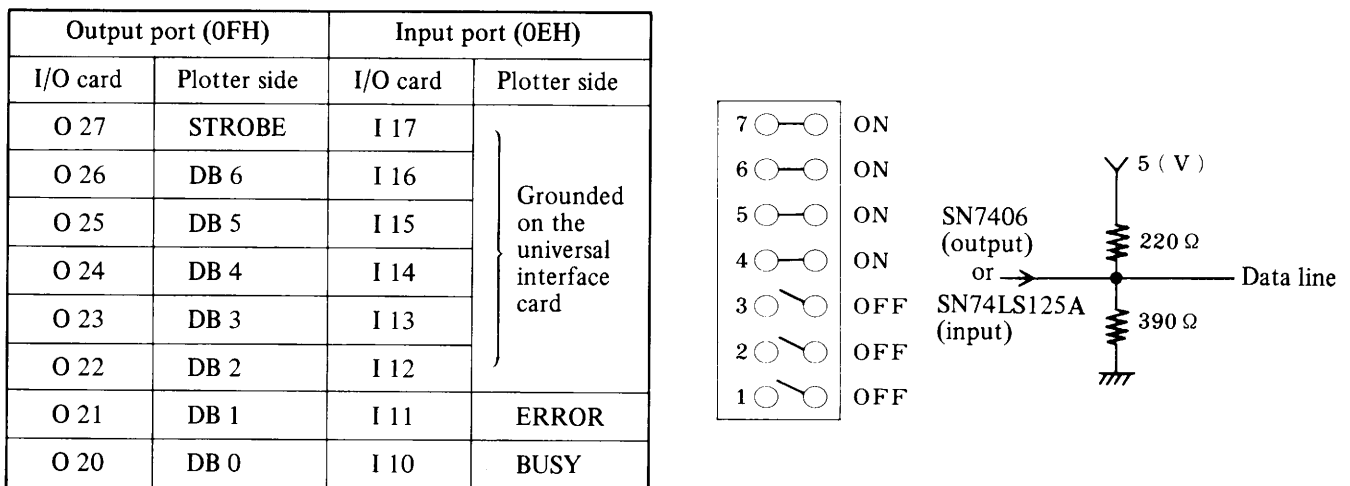
# EXAMPLE OF PLOTTER CONTROL APPLICATION

Use of the FDOS editor, assembler and BASIC compiler allows the familiar BASIC language to be used to write main programs without loss in processing speed, as long as control programs are created for control of the various devices. Another benefit is that commands developed by the user can be used as BASIC commands.

## —Interface Card—

Universal interface card MZ-80IO2 is used for the interface between the MZ-80B computer and the MILOT WX4671 plotter.

The connection conditions are as shown in Figure 1.



a) Connection conditions for all input/output terminals    b) Address switch settings    c) Data line termination conditions

Fig. 1 Connection conditions

SN7404 is included as the data driver for the universal interface card, but ICs 16 and 17 only are changed to SN7406. All data line and status input terminations are made as shown in Figure 1-c). A 1.5 m cable can be used for this purpose.

See the universal interface card instructions for details.

## —Plotter Control Program—

This section may skip if you are not interested in assembly subroutines.

### 1. Conditions for linkage with a BASIC program

Conditions for linkage with a BASIC program

- Command names must be externally declared with the ENT statement .....SIZE : ENT
- The number of parameters must be specified ..... DEFB 1 (1 parameter)
- The parameter type must be specified. .... DEFB 0 (real number)
- Buffers must be specified for parameters. .... SE : DEFS 2 (2 bytes reserved)
- The RET instruction must be included at the end of all control routines.

The above are the linkage conditions; the processing program is written between items (d) and (e).

## 2. Linkage conditions when an error occurs

- (a) A subroutine is used from FDOS library RELO.LIB ..... CALL BEERR
- (b) The error number is written. .... DEFB 80
- (c) The error message is written. .... DEFM 'PLOTTER ERROR'
- (d) The terminator is written. .... DEFB 0DH

This causes \* ER 80:PLOTTER ERROR to be output on the display screen when a plotter error occurs.

## 3. Use of external subroutines

This control program uses 4 routines out of the subroutines included in FDOS library RELO.LIB. One of these is BEERR, which was shown above; the remaining three are as follows:

- (a) . . INTO

16 bit binary data is set in the HL register with a sign attached when an address with a parameter is loaded in the HL register and called. All registers except the AF register are protected in the event of an overflow if the carry flag is set.

- (b) CASC'

The unsigned 16-bit binary data from the HL register is converted to ASCII code and stored in the address indicated in the DE register, then 0DH is set.

- (c) . MOVE'

When the parameter contains a type 1 string and an address with data is loaded in the HL register and called, a type 2 string is set in the address indicated by the DE register and 0DH is set.

See the "LIBRARY/PACKAGE" instructions for details on all subroutines.

## 4. Plotter control codes

All data for the MILOT WX4671 currently used is in 7-bit ASCII code. Input statuses include the BUSY signal and the ERROR signal. Data output is possible when the BUSY signal goes low, and the data is taken in on the plotter side when the STROBE signal is output.

", " (that is, 2CH) is used as the data delimiter and 03H~0DH are valid as data terminators. However, only 0AH will be accepted when an error occurs; an error condition is not cleared by any data terminator other than 0AH.

At the port on the MZ-80 side, 0EH is used for the status (that is, as the input for the BUSY and ERROR signals) and 0FH is used for the data and STROBE signal output.

## 5. Program outline

Linkage conditions for a BASIC program have already been indicated; however, string type becomes applicable in the parameter type specification with the 80H. Moreover, parameter types and parameter buffers must be added depending on the number of parameters; the steps described in subparagraphs (c) and (d) of that section are not required if the number of parameters is zero. See routines CTYPE, SIZE, PLOT, HOME and so forth of the assembly listing for this.

This illustrated using the PLOT routine as a representative example. The flowchart is as shown in Figure 2 (pages 4 and 5).

### (a) Data output

Although subroutines COUT, PLOT1 and DOUT are used, DOUT is the one which actually outputs the data. With DOUT, the data set in the accumulator is output to the plotter with the STROBE signal if the BUSY signal of the plotter is LOW. If BUSY is HIGH the routine repeats a loop.

With COUT and PLOT1, the data at the address indicated in the HL register is loaded in the accumulator and then DOUT is executed; this is repeated until 0DH is output. After 0DH is output, a check is made for plotter errors and a jump is made to the error routine if any are found. Note that continuation of program execution is possible if ON ERROR processing is provided on the BASIC side.

### (b) Data conversion routine

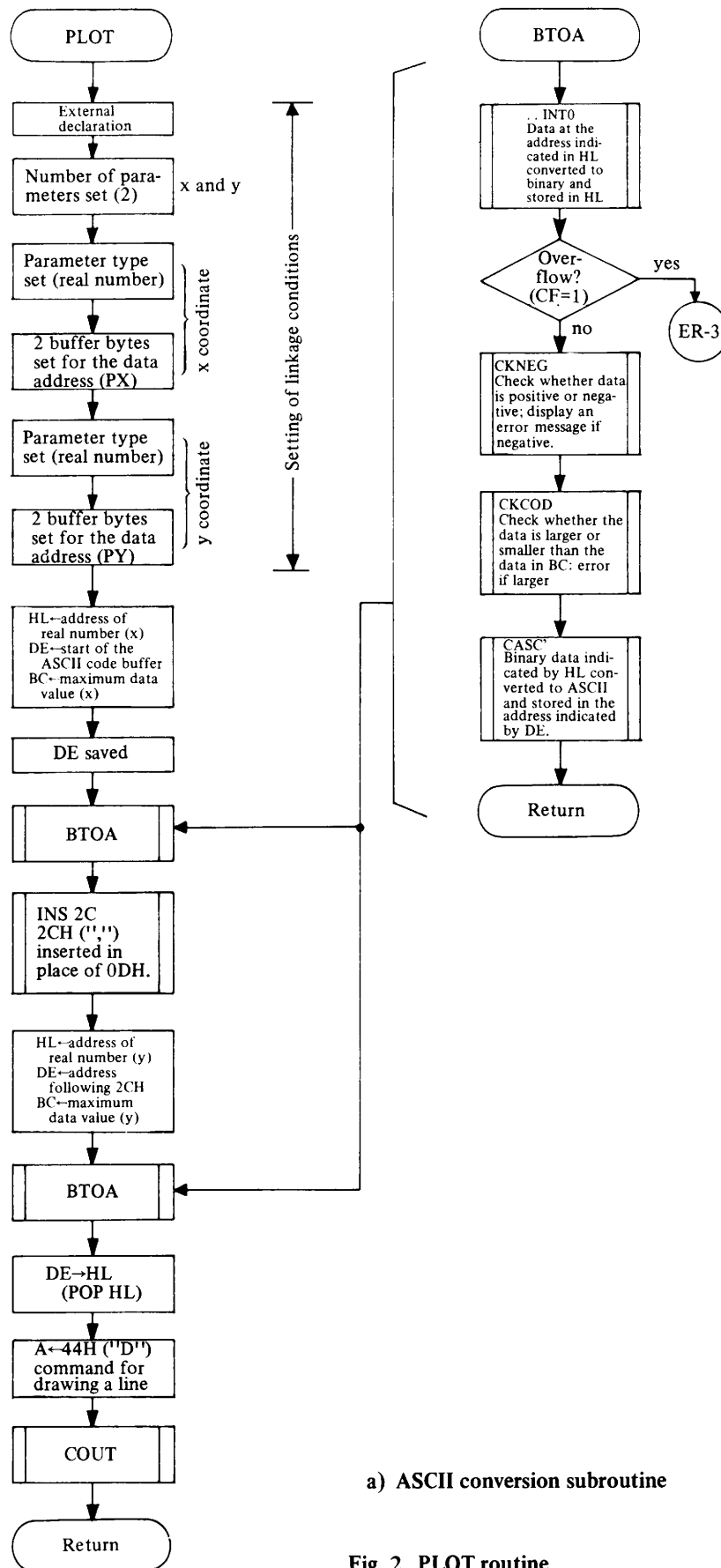
As was indicated previously, all data must be converted into ASCII code since the plotter will not accept data in any other form.

Subroutine BTOA uses the RELO.LIB routine in FDOS to convert data to ASCII code. This is as shown in the flowchart in Figure 2-a) and paragraph 3, "Use of external subroutines"; however, checks are also made for positive, negative and overflow data.

### (c) PLOT x, y routine

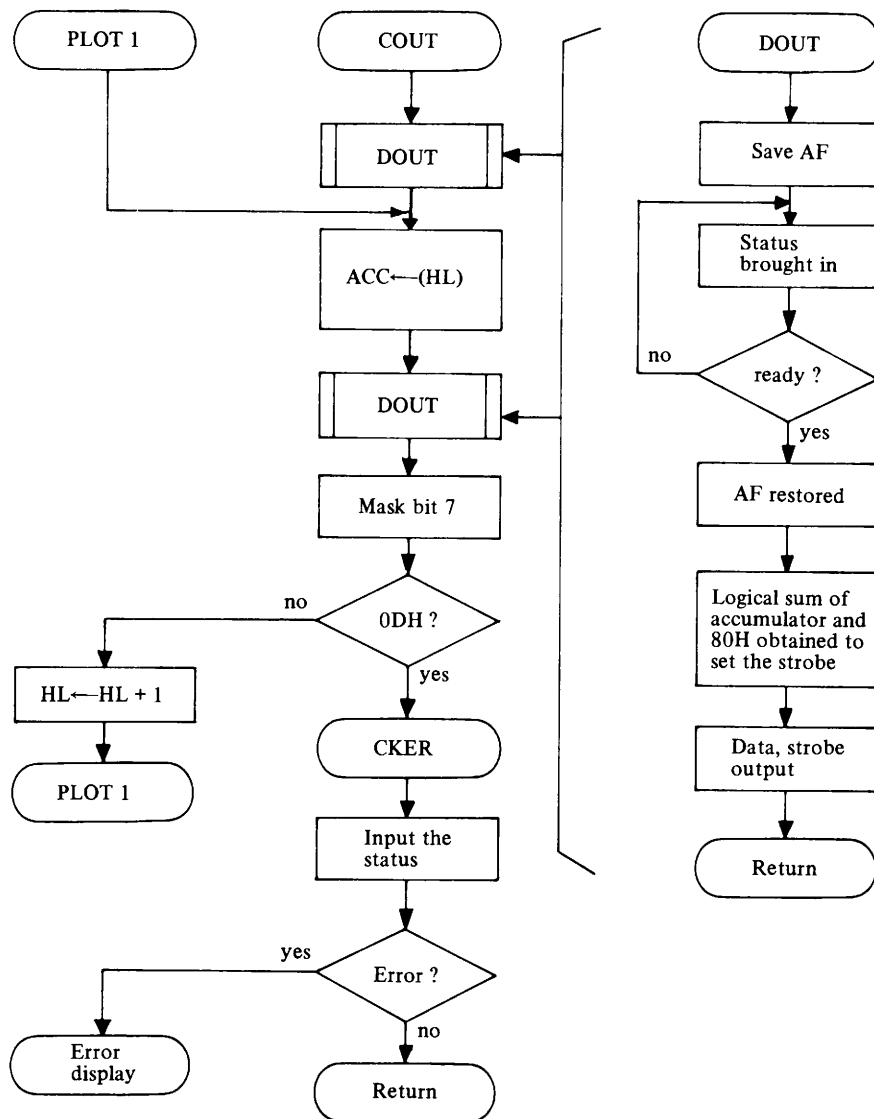
As is shown in the flowchart, the two parameters x and y are specified following the external declaration. The parameter type is real number and two bytes are set in the parameter buffer for both x and y.

The pre-conversion data address, the starting address of the buffer for storage of the data after conversion and the maximum value of x are loaded in registers HL, DE and BC, respectively, then BTOA is called and the data is checked and converted to ASCII code. Afterwards, the data delimiter for the plotter " , " is loaded and the process is repeated for y. The starting address of the data converted into ASCII code is loaded into the HL register, the 44H ("D") command (which draws a line on the plotter) is loaded and COUT is called. With COUT, the contents of the address indicated in the HL register are output to the plotter following the command.



a) ASCII conversion subroutine

Fig. 2 PLOT routine



b) Data output subroutine

Fig. 2 PLOT routine

## —Command Table—

Table 1. Plotter control commands

| Command name                       | Function                                                                                                                                                                         |
|------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>PLOT</b> x, y                   | Draws a straight line from the current position of the pen to the coordinates x, y;<br>x = 0 ~ 3600, y = 0 ~ 2540 are the possible range (other values will result in an error). |
| <b>IPLLOT</b> $\Delta x, \Delta y$ | Draws a straight line from the current position of the pen for the values of $\pm \Delta x, \pm \Delta y$ only.                                                                  |
| <b>MOVE</b> x, y                   | Lifts the pen and moves it to the position indicated by coordinates x and y.<br>x = 0 ~ 3600, y = 0 ~ 2540 are the possible range (other values will result in an error).        |
| <b>IMOVE</b> $\Delta x, \Delta y$  | Lifts the pen and moves it by an amount indicated by $\pm \Delta x, \pm \Delta y$ .                                                                                              |
| <b>XAXIS</b> x, n                  | Draws a scale with n divisions on the X axis at intervals indicated by x.                                                                                                        |
| <b>YAXIS</b> x, n                  | Draws a scale with n divisions on the Y axis at intervals indicated by y.                                                                                                        |
| <b>CTYPE</b> A\$                   | Prints the character string indicated by A\$.                                                                                                                                    |
| <b>SIZE</b> n                      | Specifies that characters are to be written in the size indicated by n; n = 0 ~ 15                                                                                               |
| <b>CSIZE</b>                       | Specifies that characters are to be written in the size initially set for the plotter.<br>(n = 4 is output automatically.)                                                       |
| <b>DOT</b> x                       | Draws a dotted line over the interval specified by x. ( $x \leq 127$ )                                                                                                           |
| <b>DOTM</b>                        | Draws a line of dots spaced at intervals of 3 mm.                                                                                                                                |
| <b>DOT0</b>                        | Clears DOT x or DOTM and returns to a straight line.                                                                                                                             |
| <b>PUP</b>                         | Pen up (same function as IMOVE0, 0)                                                                                                                                              |
| <b>PDOWN</b>                       | Pen down (same function as IPLLOT0,0)                                                                                                                                            |
| <b>ANGL</b> n                      | Rotates characters by the angle (in the counterclockwise direction) specified by n. n = 0 ~ 3<br>n = 0 ... 0°      n = 1 ... 90°      n = 2 ... 180°      n = 3 ... 270°         |
| <b>MARK</b> n                      | Writes the mark specified by n. n = 1 ~ 6<br>n = 1 ... ·      n = 2 ... ◊      n = 3 ... ⊥      n = 4 ... ▲      n = 5 ... ✕      n = 6 ... ⊕                                    |
| <b>HOME</b>                        | Moves the pen to the x, y coordinates it was at when the power was turned on, clears the error lamp and clears plotter error.                                                    |
| <b>ERCLR</b>                       | Clears plotter errors when they occur; the error lamp is not cleared, however.                                                                                                   |
| <b>(Note)</b>                      | Values indicated by x and y are specified as integral multiples of 0.1 mm; for example, PLOT 100, 2000 results in a line being drawn to x = 10 mm and y = 200 mm.                |

## —Outline of the BASIC Program—

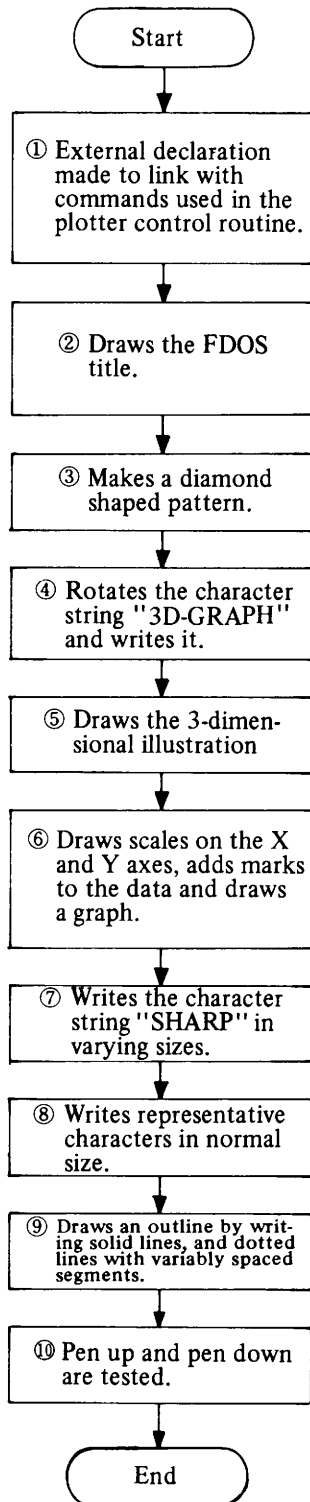


Fig. 3 Flowchart of the BASIC program

As is indicated in the flowchart at left, the program consists of 10 subroutines.

At ①, the EXTERNAL statement is used for linking the program with the plotter control program commands. Although it is not necessary to use line numbers with statements other than GOTO and GOSUB, they are used in other locations to make the program easier to understand.

At ②, the title of FDOS is drawn from line number 110 to 910. This routine uses the MOVE, IMOVE, PLOT and IPLOT commands.

At ③, a diamond pattern is drawn from line number 920 to 1050. MOVE X, Y is used to return the pen to the starting point, while IPLOT A, B is used to draw the pattern.

At ④, "3D-GRAPH" specified by A\$ is written at 90° angles with ANGL 1 and CTYPE A\$ from line number 1090 to 1140.

At ⑤, repetitions of attenuated SIN(X) and a 3-dimensional graph are drawn from line number 1190 to 1910. The 3-dimensional graph is drawn in dots using the PUP and PDOWN commands.

Try changing the program from line number 1500 to 1510 as shown below to draw the graph with solid lines.

```

1500 IF (A - 2) > T THEN 1520
1502 T = A
1504 IF S = 0 THEN MOVE DX, DY : GOTO 1508
1506 PLOT DX, DY
1508 PDOWN : S = 1
1510 RETURN
1520 MOVE DX, DY : T = A : S = 0 : GOTO 1510

```

With this program, S indicates the pen status; S = 0 raises the pen while S = 1 lowers it.

At ⑥, scales are drawn on the X and Y axes with the XAXIS and YAXIS commands and marks are drawn on the graph with the MARK command; this processing is performed from line 1990 to line 2190. The dotted line is drawn with the DOTM command, and the DOT0 command is used to return to a solid line. The curve is a drawing of  $\sin(X)/X$ . Refer to the command table for these commands.

At ⑦, the SIZE command is used from line number 2390 to 2460 to write the word SHARP in characters of varying sizes. Afterwards, the CSIZE command is used to return to standard size.

At ⑧, representative characters of standard size are drawn from line number 2790 to 2820.

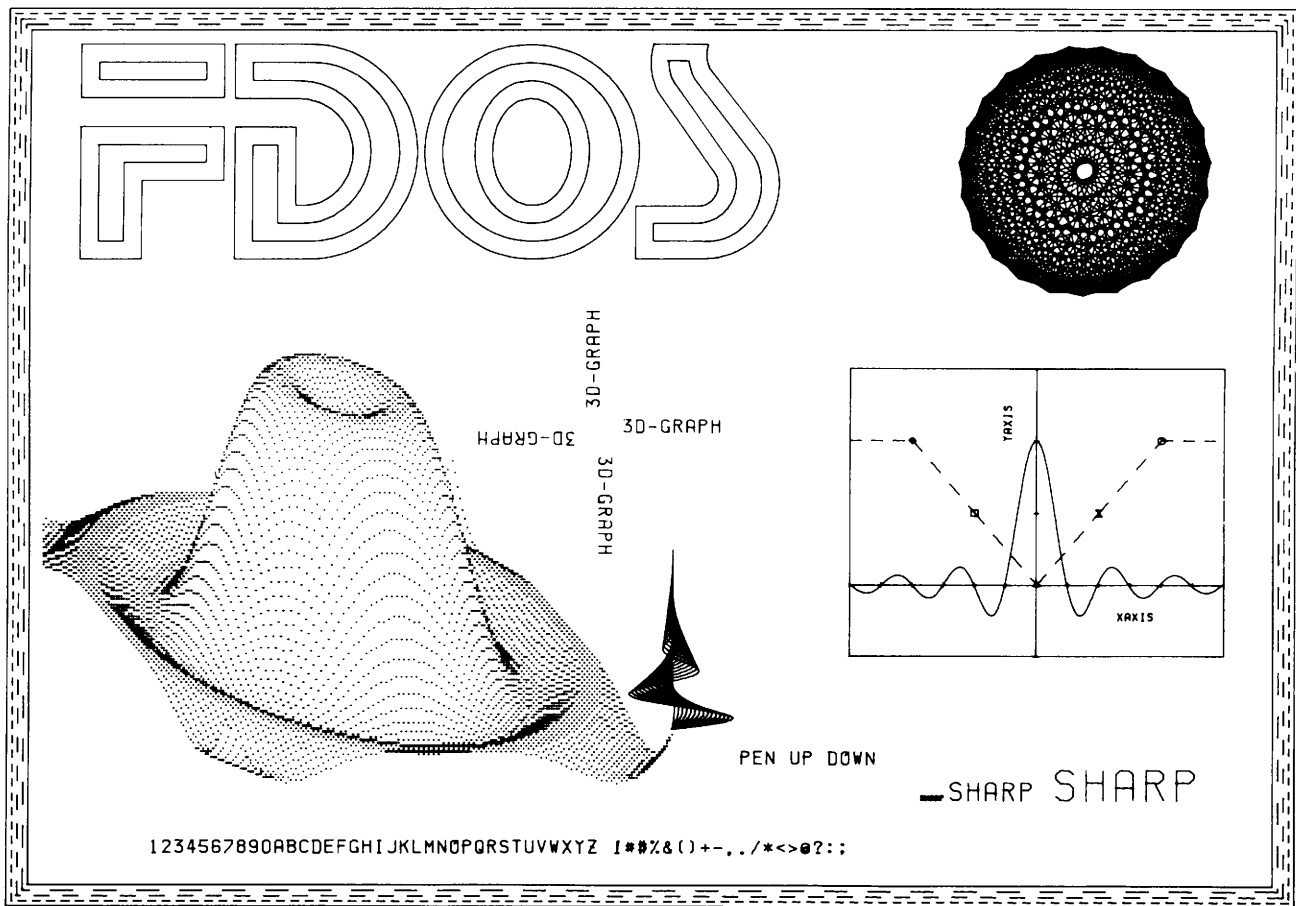
At ⑨, the HOME, DOT and DOT0 commands are used from line number 3000 to 5010 to draw the the surrounding outline by changing between solid lines and dotted lines of varying pitch.

At ⑩, pen up and pen down are tested from line number 5100 to END.

If the program is written so that the HOME command can be executed if an error occurs with the plotter, the error can be cleared, the error lamp extinguished and the pen returned to the starting position. If the ERCLR command is used (it is not in this program), the error can be cleared without moving the pen; however, in this case the error lamp is not extinguished.

## —Conclusion—

As has been shown above, user-written programs can be easily linked with BASIC. By observing the conditions for linkage, it is possible to connect many devices other than the plotter. When a main program is created using the assembler, it may be necessary to output a picture on the display screen. The BASIC compiler and FDOS are very useful for this purpose. Further, the processing is the same as with machine language. The photograph on page 9 shows the result of execution of this program.



## —References—

1. Watanabe Manufacturing Co., MILOT WX4671 Instruction Manual
2. Sharp Corp., Universal Interface Card Instruction Manual

# —Sample Program (Plotter Control Routines)—

\*\* Z80 ASSEMBLER SB-7201 <PLOTTER> PAGE 01

07/09/81

```

01 0000 ;
02 0000 ;
03 0000 ; MZ-80B Plotter control package.
04 0000 ; Plotter WX-4671 & universal I/O.
05 0000 ;
06 0000 ; Copyright 1981 by SHARP Corp.
07 0000 ;
08 0000 ;
09 0000 ; BASIC SUBROUTIN LINK
10 0000 ;
11 0000 ;
12 0000 ; CALL ..INT0
13 0000 ; WITHOUT AF KEEP
14 0000 ; 16BIT BINARY TO HL
15 0000 ;
16 0000 ; CALL CASC'
17 0000 ; WITHOUT AF KEEP
18 0000 ; HL(16BIT BINARY)TO DE ADDRESS
19 0000 ; (DE)=ASCII END=0DH
20 0000 ;
21 0000 ; CALL BEERR
22 0000 ; ERROR MSG DSP-OUT
23 0000 ;
24 0000 ; CALL .MOVE'
25 0000 ; CHR TYPE PARAMETER TO ASCII CODE(HL TO DE)
26 0000 ;
27 0000 ;
28 0000 ; PORT
29 0000 ;
30 0000 P PLTS: EQU 0EH ;PLOTTER STATUS READ
31 0000 P PLTD: EQU 0FH ;PLOTTER DATA OUT
32 0000 ;
33 0000 ; MOVE X,Y(PEN UP)
34 0000 ;
35 0000 MBUFF: DEFS +16
36 0010 ;
37 0010 MOVE: ENT
38 0010 02 DEFB 2 ;NO OF PARAMETER
39 0011 00 DEFB 0 ;REAL NO.
40 0012 MX: DEFS +2
41 0014 00 DEFB 0 ;REAL NO.
42 0015 MY: DEFS +2
43 0017 ;
44 0017 2A1200 LD HL,(MX)
45 001A 110000 LD DE,MBUFF
46 001D D5 PUSH DE
47 001E 01110E LD BC,0E11H ;3600
48 0021 CD5202 CALL BTOA
49 0024 CD8302 CALL INS2C
50 0027 2A1500 LD HL,(MY)
51 002A 01ED09 LD BC,09EDH ;2540
52 002D CD5202 CALL BTOA
53 0030 E1 POP HL
54 0031 3E4D LD A,4DH ;'M' MOVE CMD
55 0033 C34201 JP COUT
56 0036 ;
57 0036 ; INCREASE MOVE(PEN UP)DX,DY
58 0036 ;
59 0036 IMBUFF: DEFS +16
60 0046 ;

```

```

01 0046 IMOVE: ENT
02 0046 02 DEFB 2 ;NO OF PARAMETER
03 0047 00 DEFB 0 ;REAL
04 0048 IMX: DEFS +2
05 004A 00 DEFB 0 ;REAL
06 004B IMY: DEFS +2
07 004D ;
08 004D 2A4800 LD HL,(IMX)
09 0050 113600 LD DE,IMBUFF
10 0053 D5 PUSH DE
11 0054 01110E LD BC,0E11H ;3600
12 0057 CD6202 CALL NSBTOA
13 005A CD8302 CALL INS2C
14 005D 2A4B00 LD HL,(IMY)
15 0060 01ED09 LD BC,09EDH ;2540
16 0063 CD6202 CALL NSBTOA
17 0066 E1 POP HL
18 0067 3E52 LD A,52H ;'R' RELATIVE MOVE CMD
19 0069 C34201 JP COUT
20 006C ;
21 006C ; X-AXIS
22 006C ;
23 006C QRXBUF: DEFS +16
24 007C ;
25 007C XAXIS: ENT
26 007C 02 DEFB 2 ;NO OF PARAMETER
27 007D 00 DEFB 0 ;REAL
28 007E QX: DEFS +2
29 0080 00 DEFB 0 ;REAL
30 0081 RX: DEFS +2
31 0083 ;
32 0083 2A7E00 LD HL,(QX)
33 0086 116C00 LD DE,QRXBUF
34 0089 D5 PUSH DE
35 008A 01110E LD BC,0E11H ;3600
36 008D CD5202 CALL BTOA
37 0090 CD8302 CALL INS2C
38 0093 2A8100 LD HL,(RX)
39 0096 01110E LD BC,0E11H ;3600
40 0099 CD5202 CALL BTOA
41 009C 3E31 LD A,31H ;'1'
42 009E 1832 JR AXOUT
43 00A0 ;
44 00A0 ; Y-AXIS
45 00A0 ;
46 00A0 QRYBUF: DEFS +16
47 00B0 ;
48 00B0 YAXIS: ENT
49 00B0 02 DEFB 2 ;NO OF PARAMETER
50 00B1 00 DEFB 0 ;REAL
51 00B2 QY: DEFS +2
52 00B4 00 DEFB 0 ;REAL
53 00B5 RY: DEFS +2
54 00B7 ;
55 00B7 2AB200 LD HL,(QY)
56 00BA 11A000 LD DE,QRYBUF
57 00BD D5 PUSH DE
58 00BE 01ED09 LD BC,09EDH ;2540
59 00C1 CD5202 CALL BTOA
60 00C4 CD8302 CALL INS2C

```

```

01 00C7 2AB500 LD HL,(RY)
02 00CA 01ED09 LD BC,09EDH ;2540
03 00CD CD5202 CALL BTOA
04 00D0 3E30 LD A,30H ;'0'
05 00D2 CD2302 AXOUT: CALL AXIS
06 00D5 E1 POP HL
07 00D6 186D JR PLOT1
08 00D8 ;
09 00D8 ; INCREASE PLOT X+DX,Y+DY
10 00D8 ;
11 00D8 IPBUFF: DEFS +16
12 00E8 ;
13 00E8 IPLOT: ENT
14 00E8 02 DEFB 2 ;NO OF PARAMETER
15 00E9 00 DEFB 0 ;REAL
16 00EA IPX: DEFS +2
17 00EC 00 DEFB 0 ;REAL
18 00ED IPY: DEFS +2
19 00EF ;
20 00EF 2AEA00 LD HL,(IPX)
21 00F2 11D800 LD DE,IPBUFF
22 00F5 D5 PUSH DE
23 00F6 01110E LD BC,0E11H ;3600
24 00F9 CD6202 CALL NSBTOA
25 00FC CD8302 CALL INS2C
26 00FF 2AED00 LD HL,(IPY)
27 0102 01ED09 LD BC,09EDH ;2540
28 0105 CD6202 CALL NSBTOA
29 0108 E1 POP HL
30 0109 3E49 LD A,49H ;'I' INCREASE CMD
31 010B 1835 JR COUT
32 010D ;
33 010D ; PLOT X,Y
34 010D ;
35 010D PTBUFF: DEFS +16
36 011D ;
37 011D ;
38 011D PLOT: ENT
39 011D 02 DEFB 2 ;NO OF PARAMETER
40 011E 00 DEFB 0 ;REAL
41 011F PX: DEFS +2
42 0121 00 DEFB 0 ;REAL
43 0122 PY: DEFS +2
44 0124 ;
45 0124 2A1F01 LD HL,(PX)
46 0127 11D001 LD DE,PTBUFF
47 012A D5 PUSH DE
48 012B 01110E LD BC,0E11H ;3600
49 012E CD5202 CALL BTOA
50 0131 CD8302 CALL INS2C
51 0134 2A2201 LD HL,(PY)
52 0137 01ED09 LD BC,09EDH ;2540
53 013A CD5202 CALL BTOA
54 013D E1 POP HL
55 013E 3E44 LD A,44H ;'D' DRAW CMD
56 0140 1800 JR COUT
57 0142 ;
58 0142 ; ASCII CODE OUT SUB
59 0142 ;
60 0142 CD2F02 COUT: CALL DOUT

```

```

01 0145 7E PLOT1: LD A,(HL)
02 0146 CD2F02 CALL DOUT
03 0149 E67F AND 7FH
04 014B FE0D CP 0DH
05 014D 2803 JR Z,CKER
06 014F 23 INC HL
07 0150 18F3 JR PLOT1
08 0152 ;
09 0152 ; ERROR CHECK
10 0152 ;
11 0152 CDCE02 CKER: CALL DLY
12 0155 DB0E IN A,(PLTS)
13 0157 E602 AND 02H ;CKER(BIT 1=0?)
14 0159 C0 RET NZ
15 015A CD0000 E CKER1: CALL BEERR
16 015D 50 DEFB 50H ;80
17 015E 504C4F54 DEFM 'PLOTTER ERROR'
18 0162 54455220
19 0166 4552524F
20 016A 52
21 016B 0D DEFB 0DH
22 016C ;
23 016C ; PRINT ASCII
24 016C ;
25 016C CTYPE: ENT
26 016C 01 DEFB 1 ;NO OF PARAMETER
27 016D 80 DEFB 80H ;CHR
28 016E TP: DEFS +2
29 0170 ;
30 0170 2A6E01 LD HL,(TP)
31 0173 CD0000 E CALL ,MOVE'
32 0176 EB EX DE,HL
33 0177 3E50 LD A,50H ;'P' PRINT CMD
34 0179 18C7 JR COUT
35 017B ;
36 017B ; SIZE ALPHA
37 017B ;
38 017B SBUF: DEFS +5
39 0180 ;
40 0180 SIZE: ENT
41 0180 01 DEFB 1 ;NO OF PARAMETER
42 0181 00 DEFB 0 ;REAL
43 0182 SE: DEFS +2
44 0184 ;
45 0184 2A8201 LD HL,(SE)
46 0187 117B01 LD DE,SBUF
47 018A 011000 LD BC,0010H ;15
48 018D CD5202 CALL BTOA
49 0190 EB EX DE,HL
50 0191 3E53 LD A,53H ;'S' SCALE CMD
51 0193 18AD JR COUT
52 0195 ;
53 0195 ; DOT LINE
54 0195 ;
55 0195 DBUF: DEFS +5
56 019A ;
57 019A DOT: ENT
58 019A 01 DEFB 1 ;NO OF PARAMETER
59 019B 00 DEFB 0 ;REAL
60 019C DT: DEFS +2

```

```

01 019E ;
02 019E 2A9C01 LD HL,(DT)
03 01A1 119501 LD DE,DBUF
04 01A4 018000 LD BC,0080H
05 01A7 CD5202 CALL BTOA
06 01AA EB EX DE,HL
07 01AB 3E42 DOTWR: LD A,42H ;'B' LINE GAGE CMD
08 01AD CD4201 CALL COUT
09 01B0 21A202 LD HL,DOTMSG
10 01B3 188D JR COUT
11 01B5 ;
12 01B5 DOTM: ENT
13 01B5 00 DEFB 0 ;NO PARAMETER
14 01B6 219F02 LD HL,DOIMSG
15 01B9 18F0 JR DOTWR
16 01BB DOT0: ENT
17 01BB 00 DEFB 0 ;NO PARAMETER
18 01BC 219C02 LD HL,LTMSG
19 01BF 180A JR CNCT
20 01C1 CSIZE: ENT
21 01C1 00 DEFB 0 ;NO PARAMETER
22 01C2 21A502 LD HL,ISMSG
23 01C5 1804 JR CNCT
24 01C7 ;
25 01C7 ; PEN UP
26 01C7 ;
27 01C7 PUP: ENT
28 01C7 00 DEFB 0 ;NO PARAMETER
29 01C8 21A802 LD HL,PUMSG
30 01CB C34501 CNCT: JP PLOT1
31 01CE ;
32 01CE ; PEN DOWN
33 01CE ;
34 01CE PDOWN: ENT
35 01CE 00 DEFB 0 ;NO PARAMETER
36 01CF 21AD02 LD HL,PDMSG
37 01D2 18F7 JR CNCT
38 01D4 ;
39 01D4 ; ERROR CLEAR
40 01D4 ;
41 01D4 ERCLR: ENT
42 01D4 00 DEFB 0 ;NO PARAMETER
43 01D5 1806 JR LFOUT
44 01D7 ;
45 01D7 ; HOME
46 01D7 ;
47 01D7 HOME: ENT
48 01D7 00 DEFB 0 ;NO PARAMETER
49 01D8 3E48 LD A,48H ;'H' HOME
50 01DA CD2F02 CALL DOUT
51 01DD 3E0A LFOUT: LD A,0AH ;ER CLEAR CMD
52 01DF 184E JR DOUT
53 01E1 ;
54 01E1 ; ROTATE ALPHA
55 01E1 ;
56 01E1 PBUF: DEFS +5
57 01E6 ;
58 01E6 ANGL: ENT
59 01E6 01 DEFB 1 ;NO OF PARAMETER
60 01E7 00 DEFB 0 ;REAL

```

```

01 01E8 AL: DEFS 2
02 01EA ;
03 01EA 2AE801 LD HL,(AL)
04 01ED 11E101 LD DE,PBUF
05 01F0 010400 LD BC,0004H
06 01F3 CD5202 CALL BTOA
07 01F6 EB EX DE,HL
08 01F7 3E51 LD A,51H ;'Q' ROTATE CMD
09 01F9 C34201 JP COUT
10 01FC ;
11 01FC ; MARK
12 01FC ;
13 01FC MBUF: DEFS +5
14 0201 ;
15 0201 MARK: ENT
16 0201 01 DEFB 1 ;NO OF PARAMETER
17 0202 00 DEFB 0 ;REAL
18 0203 MK: DEFS +2
19 0205 ;
20 0205 2A0302 LD HL,(MK)
21 0208 CD0000 CALL ..INT0
22 020B DA0000 JP C,ER3
23 020E CDB202 CALL CKNEG
24 0211 010700 LD BC,0007H
25 0214 CD8D02 CALL CKC0D0
26 0217 11FC01 LD DE,MBUF
27 021A CD0000 CALL CASC'
28 021D EB EX DE,HL
29 021E 3E4E LD A,4EH ;'N' MARK CND
30 0220 C34201 JP COUT
31 0223 ;
32 0223 ; AXIS SUB
33 0223 ;
34 0223 F5 AXIS: PUSH AF
35 0224 3E58 LD A,58H ;'X' AXIS CMD
36 0226 CD2F02 CALL DOUT
37 0229 F1 POP AF
38 022A CD2F02 CALL DOUT
39 022D 3E2C AXIS1: LD A,2CH ;',' DELIMITER
40 022F ;
41 022F ;
42 022F ;;;;;;;;;;
43 022F ;
44 022F ; PLOTTER DATA CONTROL
45 022F ; A=DATA
46 022F ;
47 022F ;;;;;;;;;;
48 022F ;
49 022F F5 DOUT: PUSH AF
50 0230 DB0E DOUTP: IN A,(PLTS)
51 0232 CB47 BIT 0,A
52 0234 20FA JR NZ,DOUTP
53 0236 F1 POP AF
54 0237 D30E OUT (PLTS),A
55 0239 F680 OR 80H
56 023B CDCE02 CALL DLY
57 023E D30E OUT (PLTS),A
58 0240 F5 PUSH AF
59 0241 DB0E DOUTQ: IN A,(PLTS)
60 0243 CB47 BIT 0,A

```

```

01 0245 28FA JR Z,DOUTQ
02 0247 3EFF LD A,FFH
03 0249 3D DEC A
04 024A 20FD JR NZ,-1
05 024C F1 POP AF
06 024D E67F AND 7FH
07 024F D30E OUT (PLTS),A
08 0251 C9 RET
09 0252 ;
10 0252 ; BINARY TO ASCII
11 0252 ;
12 0252 CD0000 E BTOA: CALL ..INT0
13 0255 DA0000 E JP C,ER3
14 0258 CDB202 CALL CKNEG
15 025B CD9202 CALL CKCOD
16 025E CD0000 E CALL CASC'
17 0261 C9 RET
18 0262 ;
19 0262 ;
20 0262 CD0000 E NSBTOA: CALL ..INT0
21 0265 DA0000 E JP C,ER3
22 0268 CD7202 CALL INS2D
23 026B CD9202 CALL CKCOD
24 026E CD0000 E CALL CASC'
25 0271 C9 RET
26 0272 ;
27 0272 ; IF NEG... INSERT '-'
28 0272 ;
29 0272 7C INS2D: LD A,H
30 0273 CB7F BIT 7,A
31 0275 C8 RET Z
32 0276 3E2D LD A,2DH
33 0278 12 LD (DE),A
34 0279 13 INC DE
35 027A 7D LD A,L
36 027B 2F CPL
37 027C 6F LD L,A
38 027D 7C LD A,H
39 027E 2F CPL
40 027F 67 LD H,A
41 0280 23 INC HL
42 0281 C9 RET
43 0282 ;
44 0282 ; SEEK 0D,INSERT 2C
45 0282 ;
46 0282 13 INC DE
47 0283 1A INS2C: LD A,(DE)
48 0284 FE0D CP 0DH
49 0286 20FA JR NZ,INS2C-1
50 0288 3E2C LD A,2CH
51 028A 12 LD (DE),A
52 028B 13 INC DE
53 028C C9 RET
54 028D ;
55 028D ; CHECK DATA VALUE
56 028D ;
57 028D 7C CKCOD0: LD A,H
58 028E B5 OR L
59 028F CA0000 E JP Z,ER3
60 0292 E5 CKCOD: PUSH HL

```

```

01 0293 B7 OR A
02 0294 ED42 CKMAX: SBC HL,BC
03 0296 F20000 E JP P,ER3
04 0299 E1 POP HL
05 029A B7 OR A
06 029B C9 RET
07 029C ;
08 029C ; LINE MSG
09 029C ;
10 029C 4C LTMSG: DEFB 4CH ;'L' LINE CMD
11 029D 30 DEFB 30H ;'0'
12 029E 0D DEFB 0DH
13 029F ;
14 029F ; BETWEEN DOT
15 029F ;
16 029F 33 DOIMSG: DEFB 33H ;'3'
17 02A0 30 DEFB 30H ;'0'
18 02A1 0D DEFB 0DH
19 02A2 ;
20 02A2 ; DOT MSG
21 02A2 ;
22 02A2 4C DOTMSG: DEFB 4CH ;'L' LINE CMD
23 02A3 31 DEFB 31H ;'1'
24 02A4 0D DEFB 0DH
25 02A5 ;
26 02A5 ; INITIAL SIZE SET
27 02A5 ;
28 02A5 53 ISMSG: DEFB 53H ;'S'
29 02A6 34 DEFB 34H ;'4'
30 02A7 0D DEFB 0DH
31 02A8 ;
32 02A8 ; PEN UP MSG
33 02A8 ;
34 02A8 52 PUMSG: DEFB 52H ;'R' RELATIVE CMD
35 02A9 30 DEFB 30H ;'0'
36 02AA 2C DEFB 2CH ;','
37 02AB 30 DEFB 30H ;'0'
38 02AC 0D DEFB 0DH
39 02AD ;
40 02AD ; PEN DOWN MSG
41 02AD ;
42 02AD 49 PDMSG: DEFB 49H ;'I' INCREASE CMD
43 02AE 30 DEFB 30H ;'0'
44 02AF 2C DEFB 2CH ;','
45 02B0 30 DEFB 30H ;'0'
46 02B1 0D DEFB 0DH
47 02B2 ;
48 02B2 ; CHECK DATA SIGN
49 02B2 ;
50 02B2 7C CKNEG: LD A,H
51 02B3 CB7F BIT 7,A
52 02B5 C8 RET Z
53 02B6 CD0000 E CKER2: CALL BEERR
54 02B9 51 DEFB 51H ;81
55 02BA 554E464F DEFB 'UNFORMAT DATA ERROR'
56 02BE 524D4154
57 02C2 20444154
58 02C6 41204552
59 02CA 524F52
60 02CD 0D DEFB 0DH

```

```

01 02CE ;
02 02CE C5 DLY: PUSH BC
03 02CF F5 PUSH AF
04 02D0 060A LD B,10
05 02D2 AF DLY2: XOR A
06 02D3 3D DEC A
07 02D4 20FD JR NZ,-1
08 02D6 10FA DJNZ DLY2
09 02D8 F1 POP AF
10 02D9 C1 POP BC
11 02DA C9 RET
12 02DB END

```

```

AL 01E8 ANGL 01E6 AXIS 0223 AXIS1 022D AXOUT 00D2
BTOA 0252 CKCOD 0292 CKCOD0 028D CKER 0152 CKER1 015A
CKER2 02B6 CKMAX 0294 CKNEG 02B2 CNCT 01CB COUT 0142
CSIZE 01C1 CTYPE 016C DBUF 0195 DLY 02CE DLY2 02D2
DOIMSG 029F DOM0 01BB DOT 019A DOTM 01B5 DOTMSG 02A2
DOTWR 01AB DOUT 022F DOUTP 0230 DOUT0 0241 DT 019C
ERCLR 01D4 HOME 01D7 IMBUFF 0036 IMOVE 0046 IMX 0048
IMY 004B INS2C 0283 INS2D 0272 IPBUFF 00D8 IPLOT 00E8
IPX 00EA IPY 00ED ISMSG 02A5 LFOUT 01DD LTMSG 029C
MARK 0201 MBUF 01FC MBUFF 0000 MK 0203 MOVE 0010
MX 0012 MY 0015 NSBTOA 0262 PBUF 01E1 PDMSG 02AD
PDOWN 01CE PLOT 011D PLOT1 0145 PLTD 000F PLTS 000E
PTBUFF 010D PUMSG 02A8 PUP 01C7 PX 011F PY 0122
QRXBUF 006C QRYBUF 00A0 QX 007E QY 00B2 RX 0081
RY 00B5 SBUF 017B SE 0182 SIZE 0180 TP 016E
XAXIS 007C YAXIS 00B0

```

# —BASIC Main Program—

BASIC compiler SB-7701 <X-YDEMO> page 1

07.09.81

```
000F EXTERNAL IPLOT,PLOT,IMOVE,MOVE,PUP,PDOWN,XAXIS,YAXIS,CTYPE
000F EXTERNAL CSIZE,SIZE,HOME,ANGL,MARK,ERCLR,DOTM,DOT0,DOT
000F DIM D(1,255),E(50)
0042 DIM A(23),B(23)
005F HOME : CSIZE : ANGL 0: DOT0
00A8 REM *FDOS NO SAKUGA*
00A8 REM * F *
00A8 MOVE 200,2400
00DD IPLOT 400,0: IPLOT 0,-150: IPLOT -400,0: IPLOT 0,150
018C IMOVE 50,-50: IPLOT 300,0: IPLOT 0,-50: IPLOT -300,0: IPLOT 0,50
0256 MOVE 200,2170: IPLOT 400,0: IPLOT 0,-150: IPLOT -230,0
0300 IPLOT 0,-220: IPLOT -170,0: IPLOT 0,370
038A IMOVE 50,-50: IPLOT 300,0: IPLOT 0,-50
03FF IPLOT -230,0: IPLOT 0,-220: IPLOT -70,0
047B IPLOT 0,270
04A9 REM * D *
04A9 MOVE 630,2400: IPLOT 210,0
0505 A=840: B=300: C=2100: GOSUB 1100
0540 IPLOT -210,0: IPLOT 0,370: IPLOT 170,0: IPLOT 0,-220
05DC A=810: B=150: C=2100: GOSUB 1000
060B IPLOT -180,0: IPLOT 0,150: IMOVE 50,-50: IPLOT 160,0
06B5 A=840: B=250: C=2100: GOSUB 1100
06E4 IPLOT -160,0: IPLOT 0,270: IPLOT 70,0: IPLOT 0,-220: IPLOT 60,0
07AE A=810: B=200: C=2100: GOSUB 1000
07D6 IPLOT -130,0: IPLOT 0,50: GOTO 1200
0836 1000 FOR I=- $\pi/2$ TO $\pi/2$ STEP $\pi/60$
0890 X=INT(A+B*COS(I)): Y=INT(C+B*SIN(I))
090A PLOT X,Y
091F NEXT I
092A RETURN
0930 1100 FOR I= $\pi/2$ TO $-\pi/2$ STEP $-\pi/60$
0978 X=INT(A+B*COS(I)): Y=INT(C+B*SIN(I))
09DE PLOT X,Y
09F3 NEXT I
09F9 RETURN
09FF 1200 REM * O *
0A05 X=0: Y=0: A=0: B=0: C=0: D=0: MOVE 1760,2100
0A69 A=1460: B=300: C=2100: D=300: GOSUB 1300
0AA1 IMOVE -50,0
0AC8 A=1460: B=250: C=2100: D=250: GOSUB 1300
0AF9 IMOVE -90,0
0B27 A=1460: B=160: C=2100: D=200: GOSUB 1300
0B58 IMOVE -50,0
0B7F A=1460: B=110: C=2100: D=150: GOSUB 1300
0BB7 MOVE 1840,2350: GOTO 1400
0BF2 1300 FOR I=0* π TO 2* π STEP $\pi/60$
0C28 X=INT(A+B*COS(I)): Y=INT(C+D*SIN(I))
0C8E PLOT X,Y
0CA3 NEXT I: A=0: B=0: C=0: D=0: X=0: Y=0: RETURN
0CE5 1400 REM * S *
0CEB FOR I= π TO $\pi*7/6$ STEP $\pi/60$
0D32 X=INT(1960+120*COS(I)): Y=INT(2350+120*SIN(I))
0DA6 PLOT X,Y: NEXT I: X=0: Y=0
0DD3 A=1920: B=110: C=2010: GOSUB 1500
0E09 IPLOT -120,0: IPLOT 0,-50: IPLOT 140,0
```

```

0E85 A=1940: B=160: C=2010: GOSUB 1600
0EB4 FOR I=π*7/6 TO π*186/180 STEP -π/60
0F0C X=INT(2030+135*COS(I)): Y=INT(2370+135*SIN(I))
0F87 PLOT X,Y
0F9C NEXT I: IPLOT -60,0: X=0: Y=0: IMOVE -38,45
1010 IPLOT 150,0
1037 FOR I=π*8/9 TO π*7/6 STEP π/60
108D X=INT(2030+85*COS(I)): Y=INT(2370+85*SIN(I))
10FA PLOT X,Y
110F NEXT I: X=0: Y=0
1127 A=1940: B=210: C=2010: GOSUB 1500
114F IPLOT -190,0: IPLOT 0,150: IPLOT 170,0
11CB A=1920: B=60: C=2010: GOSUB 1600
11F3 FOR I=π*7/6 TO π*8/9 STEP -π/60
1244 X=INT(1960+170*COS(I)): Y=INT(2350+170*SIN(I))
12AA PLOT X,Y
12BF NEXT I: X=0: Y=0: GOTO 1700
12DD 1500 REM * SUB A *
12E3 FOR I=π/6 TO -π/2 STEP -π/60
1325 X=INT(A+B*COS(I)): Y=INT(C+B*SIN(I))
138B PLOT X,Y
13A0 NEXT I: X=0: Y=0: RETURN
13BE 1600 REM * SUB B *
13C4 FOR I=-π/2 TO π/6 STEP π/60
13FD X=INT(A+B*COS(I)): Y=INT(C+B*SIN(I))
1463 PLOT X,Y
1478 NEXT I: X=0: Y=0: RETURN
1496 REM * DIAMOND *
1496 MOVE 3000,2050
14CB 1700 FOR Z=1 TO 23
14F1 A(Z)=INT(350*COS(2*π*(Z-1)/23+π/2)+3000)
158D B(Z)=INT(-350*SIN(2*π*(Z-1)/23+π/2)+2050)
1621 NEXT Z
1627 FOR S=2 TO 11
1649 L=2-S
1655 X=A(23): Y=B(23)
1682 MOVE X,Y
1697 FOR I=1 TO 24
16B9 J=L+S
16C5 IF J<24 THEN 1800
16E7 J=J-23
16F3 1800 X1=A(J): L=J: Y1=B(J)
172A A=X1-X: B=Y1-Y
1742 IPLOT A,B
1757 X=X1: Y=Y1
1769 NEXT I: NEXT S: X=0: Y=0: X1=0: Y1=0: A=0: B=0
17AB MOVE 1650,1320
17E0 REM *PRINT " 3D-GGRAPH"*
17E0 A$=" 3D-GGRAPH"
1802 FOR I=3 TO 0 STEP -1
1830 MOVE 1650,1320
1857 ANGL I
1866 CTYPE A$
187D NEXT I: A$=""
1891 REM * 3D-GRAPHIC *
```

```

1891 T=0: S=0
18A3 FOR L=0 TO 255
18BE D(0,L)=-1: D(1,L)=-1: NEXT L
18F2 MOVE 100,250
1920 FOR Y=-180 TO 180 STEP 4
194E FOR X=-180 TO 180 STEP 4
1975 IF (Y=-180)*(X=180) THEN 2600
19B3 1900 R=PI/180*SQR(X*X+Y*Y)
19FE Z=100*COS(R)-30*COS(3*R)
1AA7 A=INT((110+X/2+(16-Y/2)/2)
1AA8 B=INT((116-Y/2-Z)/2): B=192-B
1B04 IF (A<0)+(A>255) THEN 2000
1B39 IF D(0,A)=-1 THEN 2100
1B68 IF B<=D(0,A) THEN 2300
1B93 IF B>=D(1,A) THEN 2400
1BBE 2000 NEXT X
1BCA IMOVE 0,0: T=0: S=0
1C03 NEXT Y
1C09 GOTO 2700
1C0F 2100 IF A=0 THEN 2200
1C2D IF D(0,A-1)=-1 THEN 2200
1C68 IF D(0,A+1)=-1 THEN 2200
1CA3 D(0,A)=INT((D(0,A-1)+D(0,A+1))/2)
1D0F D(1,A)=INT((D(1,A-1)+D(1,A+1))/2)
1D7B GOSUB 2500: GOTO 2000
1D8E 2200 D(0,A)=B: D(1,A)=B: GOSUB 2500: GOTO 2000
1DD5 2300 GOSUB 2500: D(0,A)=B: IF D(1,A)=-1 THEN D(1,A)=B
1E42 GOTO 2000
1E48 2400 GOSUB 2500: D(1,A)=B: IF D(0,A)=-1 THEN D(1,A)=B
1EB5 GOTO 2000
1EBB 2500 REM *SUB DATA OUT *
1EC1 DX=100+7*A: DY=8*B
1EE5 MOVE DX,DY: PDOWN: PUP
1F0C RETURN
1F12 2600 REM *SUB GENSUI *
1F18 FOR K=1 TO 50: E(K)=0: NEXT K
1F4D FOR I=1 TO 50
1F68 E(I)=SIN(I*PI/12.5)
1FA4 NEXT I
1FAA N=15
1FBA FOR I=1 TO N
1FD5 D=5*I/N+5: G=(N-I+0.5)/N*4: O=DX: P=DY
2049 MOVE O,P
205E FOR J=1 TO 50
2079 O=DX+E(J)*G*(50-J): P=P+D
20C0 PLOT O,P
20D5 NEXT J: NEXT I: MOVE DX,DY: GOTO 1900
20FC 2700 REM * SIN(X)/X *
2102 MOVE 2350,900
2130 XAXIS 87,12
2165 MOVE 2870,700
219A YAXIS 200,4: MOVE 2350,900
21E8 FOR I=-6 TO 6 STEP 0.08
2216 IF I=0 THEN Y=1: X=0: GOTO 2800
2243 X=I*PI

```

```
224F Y=SIN(X)/X
2264 2800 A=2870+INT(27.5*X+0.5)
229E B=900+INT(400*Y)
22BF PLOT A,B
22D4 NEXT I
22DA MOVE 2350,1300: DOTM : MARK 1
2329 IPLOT 174,0: MARK 2: IPLOT 174,-200: MARK 3: IPLOT 174,-200: MARK 4
23ED IPLOT 174,200: MARK 5: IPLOT 174,200: MARK 6: IPLOT 174,0
2492 DOT0
249B MOVE 2350,700: IPLOT 0,800: IPLOT 1044,0: IPLOT 0,-800: IPLOT -1044,0
256C MOVE 3100,800: SIZE 2: A$="XAXIS": CTYPE A$
25D7 A$="YAXIS": MOVE 2800,1300: ANGL 1: CTYPE A$: ANGL 0: A$=""
2668 REM * SHARP SIZE *
2668 MOVE 2550,300
2696 B$="SHARP "
26AA FOR I=1 TO 15 STEP 5
26D1 SIZE I
26E0 CTYPE B$
26F2 NEXT I: B$=""
2706 CSIZE
270F REM * CHARACTER SET *
270F MOVE 400,150
2736 C$="1234567890ABCDEFGHIJKLMN0PQRSTUVWXYZ !#$%&()+-.,/*<>@?;:"
277C CTYPE C$
278E REM * DOT0 TYPE *
278E HOME : PLOT 0,2500: PLOT 3590,2500: PLOT 3590,0: PLOT 0,0: IMOVE 15,15
2868 H=3560: V=2470: R=15
2891 FOR I=20 TO 60 STEP 20
28BF DOT I
28CE IPLOT 0,V: IPLOT H,0: IPLOT 0,-V: IPLOT -H,0: IMOVE R,R
296D V=V-2*R: H=H-2*R
299D NEXT I
29A3 HOME : IMOVE 60,60
29D3 DOT0 : IPLOT 0,V: IPLOT H,0: IPLOT 0,-V: IPLOT -H,0
2A66 D$="PEN UP DOWN"
2A7F MOVE 2050,400
2AA6 CTYPE D$
2AB8 IMOVE 50,0
2ADF FOR I=1 TO 5: PUP : PDOWN : NEXT I
2B12 CSIZE : DOT0 : ANGL 0: HOME
2B45 END
```

\*\* Compiler found no errors.

Personal Computer  
**117-80B**

Package

**SHARP**

### NOTICE

The MZ-80 series of sophisticated personal computers is manufactured by the SHARP CORPORATION. Hardware and software specifications are subject to change without prior notice; therefore, you are requested to pay special attention to version numbers of the monitor and the system software (supplied in the form of cassette tape or mini-floppy disk files).

This manual is for reference only and the SHARP CORPORATION will not be responsible for difficulties arising out of inconsistencies caused by version changes, typographical errors of omissions in the descriptions.

This manual is based on the SB-1500 series monitor and the SB-7000 series FDOS.

— CONTENTS —

|                                                      |    |
|------------------------------------------------------|----|
| <b>USING LIBRARY ROUTINES</b> .....                  | 1  |
| <b>MONITOR SUBROUTINES (MONEQU . LIB)</b> .....      | 2  |
| <b>FDOS SUBROUTINES (FDOSEQU . LIB)</b> .....        | 7  |
| Outline .....                                        | 7  |
| CLI (Command Line Interpreter) Subroutines .....     | 8  |
| IOCS (Input Output Control System) Subroutines ..... | 11 |
| Utility Subroutines .....                            | 17 |
| FDOS Common Variables .....                          | 24 |
| CLI Intermediate Code Table .....                    | 26 |
| <b>BASIC RELOCATABLE LIBRARY (RELO . LIB)</b> .....  | 27 |
| Type 1 and Type 2 Character String Formats .....     | 31 |
| <b>INDEX OF LIBRARY NAMES</b> .....                  | 32 |



# USING LIBRARY ROUTINES

The FDOS master diskette contains three libraries (MONEQU.LIB, FDOSEQU.LIB and RELO.LIB).

MONEQU.LIB is a file of monitor subroutine addresses defined with the EQU statement. That is, it contains a program such as that shown at right which has been assembled and converted to the library (.LIB) mode.

Monitor subroutines are often used when creating programs with the assembler; in such cases, they are used as described below.

First, the subroutine names are written as is (without addresses defined) as external names when the program is written. These are then assembled with the assembler. When the assembly listing is reviewed at this time, the symbol "E" is affixed to indicate that the names are external names. Next, the program is linked; MONEQU.LIB is linked at this time also. For example,

|        |     |       |
|--------|-----|-------|
| WRINF: | EQU | 021DH |
| WRDAT: | EQU | 024EH |
| RDINF: | EQU | 025FH |
| RDDAT: | EQU | 027DH |
| VERFY: | EQU | 0286H |
| BRKEY: | EQU | 0527H |
| PRTHL: | EQU | 0568H |
| PRTHX: | EQU | 056DH |
| ASCI:  | EQU | 0583H |

⋮

Part of the contents of MONEQU . ASC

```
2 > LINK GAMEPRG1, $FD1 ; MONEQU.LIB ↵
```

↖ Program created

MONEQU.LIB must be written last at this time.

FDOSEQU.LIB is a file of subroutine addresses in FDOS which are defined with the EQU statement; it is used in the same manner as MONEQU.LIB. Since MONEQU.LIB is contained in FDOSEQU.LIB, it is only necessary to link FDOSEQU.LIB when both monitor and FDOS subroutines are to be used at the same time.

RELO.LIB is a library of subroutines for programs created with the BASIC compiler. It contains subroutines for the four basic arithmetic operations, functional calculations, character string processing, error message display and many others. In other words, whereas MONEQU.LIB and FDOSEQU.LIB are simple collections of EQU statements, RELO.LIB contains actual subroutines.

When the linker is used for linkage with RELO.LIB, it is possible to select only the routines required from the many available for linkage.

RELO.LIB is used in the same manner as MONEQU.LIB and FDOSEQU.LIB. Further, the contents of MONEQU.LIB and FDOSEQU.LIB are included in RELO.LIB.

Source programs MONEQU.ASC and FDOSEQU.ASC are also included on the master diskette along with MONEQU.LIB and FDOSEQU.LIB. It is possible to modify and add to the libraries by regenerating the source programs to recreate the libraries as necessary.

## Note:

Detailed procedures for using FDOSEQU.LIB are contained in "LINKING ASSEMBLY PROGRAM WITH FDOS" in Appendix; see "EXAMPLE OF PLOTTER CONTROL APPLICATION" in Programming Utility for details on RELO.LIB.

# MONITOR SUBROUTINES (MONEQU.LIB)

| Subroutine name<br>(hexadecimal address) | Function                                                                                                                                                                                                                                                                                                                                                                                                                            | Registers preserved            |
|------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------|
| <b>CALL LETNL</b><br>(0764H)             | To change the line and set the cursor to the beginning of the next line.                                                                                                                                                                                                                                                                                                                                                            | All registers except AF        |
| <b>CALL NL</b><br>(0757H)                | Changes the line and sets the cursor to its beginning if the cursor is not already located at the beginning of a line.                                                                                                                                                                                                                                                                                                              | All registers except AF        |
| <b>CALL PRNTS</b><br>(063AH)             | Displays one space only at the cursor position on the display screen.                                                                                                                                                                                                                                                                                                                                                               | All registers except AF        |
| <b>CALL PRNT</b><br>(063CH)              | Handles data in A register (accumulator) as ASCII code and displays it on the screen, starting at the cursor position. However, a carriage return is performed for 0DH and the various cursor control operations are performed for 01H–06H when these are included.                                                                                                                                                                 | All registers except AF        |
| <b>CALL MSG</b><br>(06B5H)               | Displays a message, starting at the cursor position on the screen. The starting address of the message must be specified in the register pair DE in advance. The message is written in ASCII code and must end in 0DH. A carriage return is not executed, but cursor control operations (control codes: 01H to 06H) are performed.                                                                                                  | All registers except AF        |
| <b>CALL MSGX</b><br>(06AFH)              | Almost the same as MSG, except that cursor control codes are for reverse character display.                                                                                                                                                                                                                                                                                                                                         | All registers except AF        |
| <b>CALL BELL</b><br>(0A80H)              | Sounds a momentary tone (approximately 880 Hz)                                                                                                                                                                                                                                                                                                                                                                                      | All registers except AF        |
| <b>CALL MELDY</b><br>(0AA3H)             | Plays musical data. The starting address of the musical data must be specified in advance in the register pair DE. As with BASIC, the musical interval and the duration of notes of the musical data are expressed in that order in ASCII code. The end mark must be either 0DH or 2AH (for the character "*"). The melody is over if C flag is 0 when a return is made; if C flag is 1 it indicates that <b>BREAK</b> was pressed. | All registers except AF        |
| <b>CALL XTEMP</b><br>(09BEH)             | Sets the musical tempo. The tempo data (1 to 7) is set in and called from A register.<br>A ← 01H      Slowest<br>A ← 04H      Medium speed<br>A ← 07H      Fastest<br>Care must be taken here to ensure that the tempo data is entered in A register in binary code, and not in the ASCII code corresponding to the numbers "1" to "7" (31H to 37H).                                                                                | All registers                  |
| <b>CALL TIMST</b><br>(09CAH)             | Sets the built-in clock. (The clock is activated by this call.) The call conditions are.<br>A ← 0 (AM), A ← 1 (PM)<br>DE ← the time in seconds (2 bytes)                                                                                                                                                                                                                                                                            | All registers except AF        |
| <b>CALL TIMRD</b><br>(0A16H)             | Reads the value of the built-in clock. The conditions upon return are:<br>A ← 0 (AM), A ← 1 (PM)<br>DE ← the time in seconds (2 bytes)                                                                                                                                                                                                                                                                                              | All registers except AF and DE |

| Subroutine name<br>(hexadecimal address) | Function                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Registers preserved                  |
|------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------|
| <b>CALL BRKEY</b><br>(0527H)             | Checks whether <b>BREAK</b> was pressed. Z flag is set if it was pressed, and Z flag is reset if it was not.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | All registers except AF              |
| <b>CALL GETL</b><br>(0BE5H)              | Inputs one line entered from the keyboard. The starting address in which the data input is to be stored and the number of characters which can be input must be specified in advance in the register pair DE and memory location KNUMBS (0BE3H), respectively. Key input is terminated by pressing the <b>CR</b> (or <b>ENT</b> ) key, at which time end mark 0DH is stored following the data entered. The maximum number of characters which can be input (including the end mark) is 160. The data input is displayed on the screen. Cursor control, insertion and deletion are accepted. Pressing the <b>BREAK</b> key during key input sets break code 0BH at the beginning of the address specified in the register pair DE and returns control to the caller. This subroutine is also called by the monitor program with the register pair DE loaded with memory location BUFR (1100H) and location KNUMBS loaded with 39 (27H). | All registers                        |
| <b>CALL GETKY</b><br>(0610H)             | Takes one character only into the A register from the keyboard. For example, when this subroutine is called with the <b>B</b> key held down, ASCII code 42H, corresponding to the character "B", is loaded into the A register and control is returned. If no key is held down, control is returned with the A register loaded with 00H. Key input is not displayed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | All registers except AF              |
| <b>CALL CHR40</b><br>(098FH)             | Sets the number of characters per line on the CRT screen to 40.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | All registers except AF,BC,DE and HL |
| <b>CALL CHR80</b><br>(0958H)             | Sets the number of characters per line on the CRT screen to 80.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | All registers except AF,BC,DE and HL |
| <b>CALL GETCRT</b><br>(0C7CH)            | Takes the line on which the cursor is located from the display data. The starting address where the data taken is to be stored and the number of characters which can be taken must be specified in advance in the register pair DE and memory location KNUMBS, respectively. End mark 0DH is stored automatically following the data. The maximum number of characters which can be taken (including the end mark) is 160.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | All registers except AF              |
| <b>CALL PRTHL</b><br>(0568H)             | Displays the contents of the register pair HL on the display screen as a 4-digit hexadecimal number.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | All registers except AF              |
| <b>CALL PRTHX</b><br>(056DH)             | Displays the contents of the A register on the display screen as a 2-digit hexadecimal number.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | All registers except AF              |
| <b>CALL ASCI</b><br>(0583H)              | Converts the contents of the lower 4 bits of A register from hexadecimal to ASCII code and returns after setting the converted data in A register.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | All registers except AF              |
| <b>CALL HEX</b><br>(058DH)               | Converts the 8 bits of A register from ASCII code to hexadecimal and returns after setting the converted data in the lower 4 bits of A register.<br>When C flag = "0" upon return    A ← hexadecimal<br>When C flag = "1" upon return    A is not assured.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | All registers except AF              |

| Subroutine name<br>(hexadecimal address) | Function                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Registers preserved            |                  |           |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |      |     |                  |     |     |                  |     |     |                  |      |     |                  |      |     |                  |          |     |            |  |     |                  |     |     |                  |    |     |                                                   |  |     |                                    |  |                         |
|------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------|------------------|-----------|-----|------------------|---|-----|------------------|---|-----|------------------|---|-----|------------------|---|-----|------------------|------|-----|------------------|-----|-----|------------------|-----|-----|------------------|------|-----|------------------|------|-----|------------------|----------|-----|------------|--|-----|------------------|-----|-----|------------------|----|-----|---------------------------------------------------|--|-----|------------------------------------|--|-------------------------|
| <b>CALL HLHEX<br/>(05A2H)</b>            | <p>Handles a consecutive string of 4 characters in ASCII code as hexadecimal string data and returns after setting the data in the register pair HL. The call and return conditions are as follows.</p> <p>∴ DE ← starting address of the ASCII string (e.g., "3" "1" "A" "5")</p> <p style="text-align: center;">↑<br/>DE</p> <p>CALL HLHEX<br/>C flag = 0 HL ← hexadecimal number (e.g., HL=31A5H)<br/>C flag = 1 HL is not assured.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | All registers except AF and HL |                  |           |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |      |     |                  |     |     |                  |     |     |                  |      |     |                  |      |     |                  |          |     |            |  |     |                  |     |     |                  |    |     |                                                   |  |     |                                    |  |                         |
| <b>CALL 2HEX<br/>(05B1H)</b>             | <p>Handles 2 consecutive ASCII strings as hexadecimal strings and returns after setting the data in A register. The call and return conditions are as follows.</p> <p>DE ← starting address of the ASCII string (e.g., "3" "A")</p> <p style="text-align: center;">↑<br/>DE</p> <p>CALL 2HEX<br/>C flag = 0 A ← hexadecimal number (e.g., A = 3AH)<br/>C flag = 1 A is not assured.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | All registers except AF and DE |                  |           |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |      |     |                  |     |     |                  |     |     |                  |      |     |                  |      |     |                  |          |     |            |  |     |                  |     |     |                  |    |     |                                                   |  |     |                                    |  |                         |
| <b>CALL ??KEY<br/>(0D77H)</b>            | <p>Awaits key input while causing the cursor to flash. When a key entry is made it is converted to display code and set in A register, then a return is made.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | All registers except AF        |                  |           |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |      |     |                  |     |     |                  |     |     |                  |      |     |                  |      |     |                  |          |     |            |  |     |                  |     |     |                  |    |     |                                                   |  |     |                                    |  |                         |
| <b>CALL ?DPCT<br/>(0714H)</b>            | <p>Controls the display on the display screen. The relationship between A register at the time of the call and control is as follows.</p> <p>A register</p> <table><tr><td>00H</td><td>Same function as</td><td>SHIFT + 0</td></tr><tr><td>01H</td><td>Same function as</td><td>↓</td></tr><tr><td>02H</td><td>Same function as</td><td>↑</td></tr><tr><td>03H</td><td>Same function as</td><td>→</td></tr><tr><td>04H</td><td>Same function as</td><td>←</td></tr><tr><td>05H</td><td>Same function as</td><td>HOME</td></tr><tr><td>06H</td><td>Same function as</td><td>CLR</td></tr><tr><td>07H</td><td>Same function as</td><td>DEL</td></tr><tr><td>08H</td><td>Same function as</td><td>INST</td></tr><tr><td>09H</td><td>Same function as</td><td>GRPH</td></tr><tr><td>0AH</td><td>Same function as</td><td>SFT LOCK</td></tr><tr><td>0BH</td><td>No control</td><td></td></tr><tr><td>0CH</td><td>Same function as</td><td>RVS</td></tr><tr><td>0DH</td><td>Same function as</td><td>CR</td></tr><tr><td>0EH</td><td>Cancels the GRAPHIC and SHIFT LOCK key input mode</td><td></td></tr><tr><td>0FH</td><td>Cancels the REVERSE key input mode</td><td></td></tr></table> | 00H                            | Same function as | SHIFT + 0 | 01H | Same function as | ↓ | 02H | Same function as | ↑ | 03H | Same function as | → | 04H | Same function as | ← | 05H | Same function as | HOME | 06H | Same function as | CLR | 07H | Same function as | DEL | 08H | Same function as | INST | 09H | Same function as | GRPH | 0AH | Same function as | SFT LOCK | 0BH | No control |  | 0CH | Same function as | RVS | 0DH | Same function as | CR | 0EH | Cancels the GRAPHIC and SHIFT LOCK key input mode |  | 0FH | Cancels the REVERSE key input mode |  | All registers except AF |
| 00H                                      | Same function as                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | SHIFT + 0                      |                  |           |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |      |     |                  |     |     |                  |     |     |                  |      |     |                  |      |     |                  |          |     |            |  |     |                  |     |     |                  |    |     |                                                   |  |     |                                    |  |                         |
| 01H                                      | Same function as                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | ↓                              |                  |           |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |      |     |                  |     |     |                  |     |     |                  |      |     |                  |      |     |                  |          |     |            |  |     |                  |     |     |                  |    |     |                                                   |  |     |                                    |  |                         |
| 02H                                      | Same function as                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | ↑                              |                  |           |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |      |     |                  |     |     |                  |     |     |                  |      |     |                  |      |     |                  |          |     |            |  |     |                  |     |     |                  |    |     |                                                   |  |     |                                    |  |                         |
| 03H                                      | Same function as                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | →                              |                  |           |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |      |     |                  |     |     |                  |     |     |                  |      |     |                  |      |     |                  |          |     |            |  |     |                  |     |     |                  |    |     |                                                   |  |     |                                    |  |                         |
| 04H                                      | Same function as                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | ←                              |                  |           |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |      |     |                  |     |     |                  |     |     |                  |      |     |                  |      |     |                  |          |     |            |  |     |                  |     |     |                  |    |     |                                                   |  |     |                                    |  |                         |
| 05H                                      | Same function as                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | HOME                           |                  |           |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |      |     |                  |     |     |                  |     |     |                  |      |     |                  |      |     |                  |          |     |            |  |     |                  |     |     |                  |    |     |                                                   |  |     |                                    |  |                         |
| 06H                                      | Same function as                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | CLR                            |                  |           |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |      |     |                  |     |     |                  |     |     |                  |      |     |                  |      |     |                  |          |     |            |  |     |                  |     |     |                  |    |     |                                                   |  |     |                                    |  |                         |
| 07H                                      | Same function as                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | DEL                            |                  |           |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |      |     |                  |     |     |                  |     |     |                  |      |     |                  |      |     |                  |          |     |            |  |     |                  |     |     |                  |    |     |                                                   |  |     |                                    |  |                         |
| 08H                                      | Same function as                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | INST                           |                  |           |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |      |     |                  |     |     |                  |     |     |                  |      |     |                  |      |     |                  |          |     |            |  |     |                  |     |     |                  |    |     |                                                   |  |     |                                    |  |                         |
| 09H                                      | Same function as                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | GRPH                           |                  |           |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |      |     |                  |     |     |                  |     |     |                  |      |     |                  |      |     |                  |          |     |            |  |     |                  |     |     |                  |    |     |                                                   |  |     |                                    |  |                         |
| 0AH                                      | Same function as                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | SFT LOCK                       |                  |           |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |      |     |                  |     |     |                  |     |     |                  |      |     |                  |      |     |                  |          |     |            |  |     |                  |     |     |                  |    |     |                                                   |  |     |                                    |  |                         |
| 0BH                                      | No control                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                                |                  |           |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |      |     |                  |     |     |                  |     |     |                  |      |     |                  |      |     |                  |          |     |            |  |     |                  |     |     |                  |    |     |                                                   |  |     |                                    |  |                         |
| 0CH                                      | Same function as                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | RVS                            |                  |           |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |      |     |                  |     |     |                  |     |     |                  |      |     |                  |      |     |                  |          |     |            |  |     |                  |     |     |                  |    |     |                                                   |  |     |                                    |  |                         |
| 0DH                                      | Same function as                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | CR                             |                  |           |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |      |     |                  |     |     |                  |     |     |                  |      |     |                  |      |     |                  |          |     |            |  |     |                  |     |     |                  |    |     |                                                   |  |     |                                    |  |                         |
| 0EH                                      | Cancels the GRAPHIC and SHIFT LOCK key input mode                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                                |                  |           |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |      |     |                  |     |     |                  |     |     |                  |      |     |                  |      |     |                  |          |     |            |  |     |                  |     |     |                  |    |     |                                                   |  |     |                                    |  |                         |
| 0FH                                      | Cancels the REVERSE key input mode                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                |                  |           |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |      |     |                  |     |     |                  |     |     |                  |      |     |                  |      |     |                  |          |     |            |  |     |                  |     |     |                  |    |     |                                                   |  |     |                                    |  |                         |

| Subroutine name<br>(hexadecimal address) | Function                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | Registers preserved            |
|------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------|
| <b>CALL ?PONT</b><br>(0904H)             | <p>Sets the current position of the cursor on the display screen in register pair HL. The return conditions are as follows.</p> <p>CALL ?PONT</p> <p>HL ← cursor position on the display screen (V-RAM address)</p> <p>(Note) The X–Y coordinates of the cursor are contained in DSPXY (10D1H). The current position of the cursor is loaded as follows.</p> <p>LD HL, (DSPXY) ; H ← Y coordinate on the screen.<br/>L ← X coordinate on the screen.</p> <p>The cursor position is set as follows.</p> <p>LD (DSPXY), HL</p> | All registers except AF and HL |
| <b>CALL WRINF</b><br>(021DH)             | <p>Writes the current contents of a certain part of the header buffer (described later) onto the tape, starting at the current tape position.</p> <p>Return conditions</p> <p>C flag = 0 No error occurred.<br/>C flag = 1 The <b>BREAK</b> key was pressed.</p>                                                                                                                                                                                                                                                             | All registers except AF        |
| <b>CALL WRDAT</b><br>(024EH)             | <p>Writes the contents of the specified memory area onto the tape as a CMT data block in accordance with the contents of a certain part of the header buffer.</p> <p>Return conditions</p> <p>C flag = 0 No error occurred.<br/>C flag = 1 The <b>BREAK</b> key was pressed.</p>                                                                                                                                                                                                                                             | All registers except AF        |
| <b>CALL RDINF</b><br>(025FH)             | <p>Reads the first CMT header found starting at the current tape position into a certain part of the header buffer.</p> <p>Return conditions</p> <p>C flag = 0 No error occurred.<br/>C flag = 1, A = FFH A check sum error occurred.<br/>C flag = 1, A ≠ FFH The <b>BREAK</b> key was pressed.</p>                                                                                                                                                                                                                          | All registers except AF        |
| <b>CALL RDDAT</b><br>(027DH)             | <p>Reads in the CMT data block according to the current contents of a certain part of the header buffer.</p> <p>Return conditions</p> <p>C flag = 0 No error occurred.<br/>C flag = 1, A = FFH A check sum error occurred.<br/>C flag = 1, A ≠ FFH The <b>BREAK</b> key was pressed.</p>                                                                                                                                                                                                                                     | All registers except AF        |
| <b>CALL VERFY</b><br>(0286H)             | <p>Compares the first CMT header found starting at the current tape position with the contents of the memory area indicated by the header.</p> <p>Return conditions</p> <p>C flag = 0 No error occurred.<br/>C flag = 1, A = FFH A match was not obtained.<br/>C flag = 1, A ≠ FFH The <b>BREAK</b> key was pressed.</p>                                                                                                                                                                                                     | All registers except AF        |
| <b>CALL PUSHR</b><br>(0DF1H)             | <p>Pushes registers IX, HL, DE and BC. The RET instruction at the end of this subroutine then automatically POPs these registers.</p> <p>SUBR : CALL PUSHR</p> <p>RET Z ; POP and RET<br/>if Z flag = 1</p> <p>RET ; POP and RET</p>                                                                                                                                                                                                                                                                                         | All registers except IX        |
| <b>CALL PUSHR2</b><br>(0DFDH)            | <p>Pushes registers IX, HL and BC. The RET instruction at the end of this subroutine then automatically POPs these registers.</p> <p>SUBR2 : CALL PUSHR2</p> <p>RET Z ; POP and RET<br/>if Z flag = 1</p> <p>RET ; POP and RET</p>                                                                                                                                                                                                                                                                                           | All registers except IX        |

(Note) The contents of the header buffer at the specific addresses are as follows. The buffer starts at address 1180H and consists of 128 bytes.

| Address                  | Contents                                                                                                                                                                                                                                                                                                                        |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>IBUFE<br/>(1180H)</b> | This byte indicates one of the following file modes.<br>01H    Object file (machine language program)<br>02H    BASIC text file<br>03H    BASIC data file<br>04H    Source file (ASCII file)<br>05H    Relocatable file (relocatable binary file)<br>A0H    PASCAL interpreter text file<br>03H    PASCAL interpreter data file |
| <b>IBU1<br/>(1181H)</b>  | These 17 bytes indicate the file name. However, since 0DH is used as the end mark, in actuality the file name is limited to 16 bytes.<br>Example: <b>S A M P L E 0D</b>                                                                                                                                                         |
| <b>IBU18<br/>(1192H)</b> | These two bytes indicate the byte size of the data block which is to follow.                                                                                                                                                                                                                                                    |
| <b>IBU20<br/>(1194H)</b> | These two bytes indicate the data address of the data block which is to follow. The loading address of the data block which is to follow is indicated by "CALL RDDAT". The starting address of the memory area which is to be output as the data block is indicated by "CALL WRDAT".                                            |
| <b>IBU22<br/>(1196H)</b> | These two bytes indicate the execution address of the data block which is to follow.                                                                                                                                                                                                                                            |
| <b>IBU24<br/>(1198H)</b> | These bytes are used for supplemental information, such as comments.                                                                                                                                                                                                                                                            |

#### Example

| Address | Content    |                                                       |
|---------|------------|-------------------------------------------------------|
| 1180    | 01         | ; indicates an object file (machine language program) |
| 1181    | 'S'        | ; the file name is 'SAMPLE'.                          |
| 1182    | 'A'        |                                                       |
| 1183    | 'M'        |                                                       |
| 1184    | 'P'        |                                                       |
| 1185    | 'L'        |                                                       |
| 1186    | 'E'        |                                                       |
| 1187    | 0D         |                                                       |
| 1188    | } Variable |                                                       |
| 1191    |            |                                                       |
| 1192    |            |                                                       |
| 1192    | 00         | ; the size of the file is 2000H bytes.                |
| 1193    | 20         |                                                       |
| 1194    | 00         | ; the data address of the file is 1300H.              |
| 1195    | 13         |                                                       |
| 1196    | 60         | ; the execution address of the file is 1360H.         |
| 1197    | 13         |                                                       |

# FDOS SUBROUTINES (FDOSEQU.LIB)

## —Outline—

FDOS subroutines can be broadly divided into three groups. That is,

1. CLI (Command Line Interpreter) subroutines
2. IOCS (Input Output Control System) subroutines
3. Utility subroutines

CLI subroutines are used to translate command lines appearing within user programs. That is, when programs are called in which switches and arguments appear in appended format (such as RUN PROG/P FILE1, FILE2 CR), these subroutines translate those switches and arguments.

IOCS subroutines are used to open and close files and devices. Utility subroutines are other general purpose subroutines.

Command lines are strings of characters (which have been converted to intermediate code) which are input from the keyboard as FDOS commands or other character strings in the same format. In the explanation below, except where otherwise indicated, command lines appear in intermediate code. See the table on page 26 for the intermediate code.

## —CLI (Command Line Interpreter) Subroutines—

### TRS10

Function: Converts FDOS command lines written in ASCII code into intermediate code.

Input registers: The HL register contains the starting address of the command line written in ASCII code. The DE register contains the starting address of the area storing the command line converted to intermediate code.

Calling procedure: CALL TRS10

Output register: CF = 0 ..... Normal  
CF = 1 ..... Error (A ← error code)

Note: See the "System Error Messages" in System Command for details. The same applies below.

Registers preserved: All registers except AF.

### . CLI (Command Line Interpreter)

Function: Translates and executes FDOS command lines.

Input registers: The HL register contains the command line pointer.

Calling procedure: LD DE, (RJOB)  
PUSH DE  
CALL .CLI  
POP HL  
LD (RJOB), HL

Output registers: CF = 0 ..... Normal  
CF = 1 ..... Error (A ← FFH)

Example of use (DATE/P)

```
LD HL, DATE
LD DE, (RJOB)
PUSH DE
CALL .CLI
POP HL
LD (RJOB), HL
JP C, ERROR
:
DATE: DEFB B1H } Intermediate
 DEFB 88H } code for DATE/P
 DEFB 0DH }
```

Registers preserved: None

Caution: The LIMIT, RUN, EXEC and DEBUG commands cannot be executed.

See page 25 for the RJOB command.

### ? HEX (Check Hexadecimal)

Function: Converts a 4-digit hexadecimal data item starting with "\$" into sixteen bit, binary notation.

Input registers: HL contains the pointer; it should specify "\$".

Calling procedure: CALL ?HEX

Output registers: CF = 1 ..... Not a hexadecimal number. (A ← 3, and HL are preserved)  
CF = 0 ..... a hexadecimal number. (DE ← data, HL indicates the address following the hexadecimal number)

Registers preserved: All registers except AF, DE and HL.

## ? SEP (Check Separator)

Function: Checks whether the contents of the address indicated by the HL register are a separator (one of the following: `CR` `␣` , : /).

Input registers: Register HL is the pointer.

Calling procedure: `CALL ?SEP`

Output registers: `CF = 1` ..... Not a separator. `A ← 3` (error code) and the HL register are preserved.  
`CF = 0` ..... A separator.

`A = 2CH` ... The separator is a space or a comma " `␣` ", " , ", " (the HL register then points to the address following the separator)

`A = 0DH` ... The separator is `CR` or slant " / " (the HL register points to the separator)

`A = 3AH` ... The separator is a colon " : " (the HL register points to the separator)

Registers preserved: All registers except AF and HL.

## ? GSW (Check Global Switch)

Function: Determines whether the global switch on the command line is correct and, if so, stores it in the area within FDOS.

Input registers: The DE register contains the starting address of the switch table. The HL register contains the command line pointer which points to the global switch.

Calling procedure: `LD DE, SWTBL`  
`CALL ?GSW`

`⋮`  
`⋮`  
`⋮`

`SWTBL : DEFB SW1`  
`DEFB SW2`  
`⋮`  
`DEFB SWn`  
`DEFB FFH`

} List of items which may be used as global switches  
 (these are written in intermediate code, from 0 to a maximum of 5.  
 See page 26)

End of table

Output registers: `CF = 1` ..... Error (`A ←` error code)

`CF = 0` ..... Normal. The HL register points to the address following the global switch.

Registers preserved: All registers except AF, DE and HL.

## TESW (Test Global Switch)

Function: Determines the presence or absence of the specified global switch. Subroutine "?GSW" must be called before this subroutine is used.

Input registers: None

Calling procedure: CALL TESW  
DEFB global switch

Output registers: CF = 0 .....  
The specified global switch is present.  
CF = 1 .....  
The specified global switch is not present.

Registers preserved: All registers except AF.

Example: This routine outputs whether or not global switch /P is present to the line printer or the CRT.

```
CALL TESW
DEFB 88H ; intermediate code for /P
PUSH AF
CALL C, MSG ; displayed on the CRT if the switch is not present.

POP AF
CCF ; CF ← CF̄
CALL C, PMSG ; Printed on the line printer if the switch is present.

JP C, ERROR ; indicates a line printer error.
```

## ? LSW (Check Local Switch)

Function: Used to determine the local switch which is attached to the file name on the command line.

Input registers: The HL register is the command line pointer which indicates the start of the file name.

Calling procedure: CALL ?LSW

Output registers: CF = 1 ..... Error (A ← error code)  
CF = 0 ..... Normal  
ZF = 1 ..... No local switch. (A ← 0)  
ZF = 0 ..... Local switch is present. (A ← intermediate code for the local switch)

Registers preserved: All registers except AF.

Example: Read-opens (ROPEN) a file with logical number 2 if a local switch is not present; if local switch /0 is present the file is write-opened (WOPEN) with logical number 3; otherwise, an error occurs.

```
EXX
LD B, 4 ; default file mode .ASC
EXX
CALL ?LSW
JP C, ERROR
JR NZ, L2
LD C, 2 ; logical number 2
CALL ROPEN
JR L3
L2: CP 89H ; intermediate code for /O
LD A, 8 ; error code (if local switch)
JP NZ, ERROR
LD C, 3 ; logical number 3
CALL WOPEN
L3: JP C, ERROR
```

## —IOCS (Input Output Control System) Subroutines—

### ROPEN (Read Open)

Function: Read-opens a file (including the input/output device).

Input registers: HL: Pointer which indicates the start of the file name.  
C : Logical number (see note 3)  
B' : Default file mode (see note 1)

Calling procedure: CALL ROPEN

Output registers: CF = 1 ..... Error (A ← error code)  
CF = 0 ..... Normal  
HL: Pointer (indicates the next separator)  
B' : File mode (see note 1)  
C' : File attribute (see note 2)  
L' : Device number  
IY: Starting address of the device table (see note 4)

Registers preserved: Only registers BC, DE and IX.

Example (when \$FD1 ; ABC)

```
LD HL, FL
LD C, 2 (logical number)
EXX
LD B, 4 (. ASC)
EXX
CALL ROPEN
CALL C, ERR (see page 23)
RET C
:
:
FL : DEFB 90H ($ FD1)
DEFB ' ; ABC '
DEFB 0DH
```

### WOPEN (Write Open)

Function: Write-opens a file (including an input/output device).

Input registers: HL: Pointer which indicates the start of the file name.  
C : Logical number (see note 3)  
B' : Default file mode (see note 1)

Calling procedure: CALL WOPEN

Output registers: CF = 1 ..... Error (A ← error code)  
CF = 0 ..... Normal  
HL: Pointer (indicates the next separator)  
B' : File mode (see note 1)  
C' : File attribute (only for "0")  
L' : Device number  
IY: Starting address of the device table (see note 4)

Registers preserved: Only registers BC, DE and IX.

Example (\$PTP/PE/LF)

```
LD HL, PTP
LD C, 3 (logical number)
EXX
LD B, 4 (. ASC)
EXX
CALL WOPEN
JP C, ERROR
:
:
PTP : DEFB A1H ($ PTP)
DEFB 8FH (/PE)
DEFB 8CH (LF)
DEFB 0DH
```

## MODECK (Filemode Check)

**Function:** Checks whether the file mode indicated in register B' for the file opened is correct or not.

**Input registers:** Register B' contains the file mode of the opened file.

**Calling procedure:** CALL MODECK  
DEFB file mode number (see page 26 concerning file modes)

**Output registers:** CF = 0 ..... The file mode is correct.  
CF = 1 ..... The file mode is not correct. A ← error code.

**Registers preserved:** All registers except AF.

(Note 1) The default file mode is the mode which is assumed when no mode is specified in the command line. The numbers enclosed in parentheses indicate the file mode number. (see page 26.)

### Example:

| Command line | Default file mode | Actual file mode |
|--------------|-------------------|------------------|
| ABC .ASC     | .ASC (4)          | .ASC (4)         |
| ABC .LIB     | .RB (5)           | .LIB (7)         |
| ABC          | .OBJ (1)          | .OBJ (1)         |
| ABC          | .ASC (4)          | .ASC (4)         |

(Note 2) The file attribute indicates the type of file access, and is expressed as one of the following ASCII codes.

"0" a file with no attribute.

"R" a file for which reading is inhibited. (Read protected file)

"W" a file for which writing is inhibited. (Write protected file)

"P" a file for which both reading and writing are inhibited. (Permanent file)

However, files with the attribute "P" can be read and written if the file mode is .OBJ. The EXEC command can be executed if the file mode is .ASC.

Normally, the programmer does not need to be aware of file attributes since they are managed by FDOS.

(Note 3) Logical file numbers are numbers within FDOS which have a one-to-one correspondence with physical files opened (including input/output devices). Numbers from 1 to 249 may be used as logical numbers; however, since programs within FDOS use all of the numbers from 128 on, user programs should use only the numbers from 1 to 127 to avoid conflict.

(Note 4) An explanation of the device table is contained in "USER CODED I/O ROUTINES" in Appendix; however, except for special I/O operations, the programmer normally does not need to be aware of the contents of the device table.

## GET1L (Get 1 Line)

**Function:** Reads in one line from the file whose logical number is specified in the C register. The line read is one which is terminated with 0DH. The data read is stored in the area indicated by the address in the DE register. The length of the line, including 0DH, must be no more than 128 bytes.

**Input registers:** The C register contains the logical number. The DE register contains the address of the area in which the data is stored.

**Calling procedure:** CALL GET1L

**Output registers:** CF = 0 ..... Normal  
CF = 1, A = 0 ..... File end  
CF = 1, A ≠ 0 ..... Error (A ← error code)  
IY : Starting address of the device table (see note 4 on page 12)

**Registers preserved:** Only registers BC, DE, HL and IX.

## GET1C (Get 1 Character)

**Function:** Reads one byte from the file whose logical number is specified in the C register.

**Input registers:** The C register contains the logical number.

**Calling procedure:** CALL GET1C

**Output registers:** CF = 0 ..... Normal (A ← data read)  
CF = 1, A = 0 ..... File end  
CF = 1, A ≠ 0 ..... Error (A ← error code)  
IY : Starting address of the device table (see note 4 on page 12)

**Registers preserved:** Only registers BC, DE, HL and IX.

## GETBL (Get Block)

**Functions:** Read data into the address indicated in the DE register from the file whose logical number is specified in the C register; only the number of bytes of data indicated in the HL register are read in.

**Input registers:** The C register contains the logical number. The DE register contains the address in which the data is to be stored. The HL register contains the number of bytes of data to be read.

**Calling procedure:** CALL GETBL

**Output registers:** CF = 0 ..... Normal  
CF = 1, A = 0 ..... File end  
CF = 1, A ≠ 0 ..... Error (A ← error code)  
IY: Starting address of the device table (see note 4 on page 12)

**Registers preserved:** Only registers BC and IX.

## ? EOF (Check End-of-file)

Function: Checks for the end of a read-opened file. Z flag becomes "1" when an attempt is made to read beyond the end of data.

Input registers: The C register contains the logical number.

Calling procedure: `CALL ?EOF`

Output registers: `CF = 1` ..... Error ( $A \leftarrow$  error code)  
`CF = 0, ZF = 1` ..... Not file end  
`CF = 0, ZF = 0`..... File end  
IY : Starting address of the device table (see note 4 on page 12)

Registers preserved: Only registers BC, DE, HL and IX.

## PUT1C (Put 1 Character)

Function: Outputs one byte of data to the file whose logical number is specified in the C register.

Input registers: The C register contains the logical number. The A register contains the data to be output.

Calling procedure: `CALL PUT1C`

Output registers: `CF = 0` ..... Normal  
`CF = 1` ..... Error ( $A \leftarrow$  error code)  
IY : Starting address of the device table (see note 4 on page 12)

Registers preserved: Only registers BC, DE, HL and IX.

## PUT1L (Put 1 Line)

Function: Outputs the line starting at the address specified in the DE register to the file whose logical number is specified in the C register. Outputs the ending carriage return.

Input registers: The C register contains the logical number. The DE register contains the starting address of the data to be output.

Calling procedure: `CALL PUT1L`

Output registers: `CF = 0` ..... Normal  
`CF = 1` ..... Error ( $A \leftarrow$  error code)  
IY : Starting address of the device table (see note 4 on page 12)

Registers preserved: Only registers BC, DE, HL and IX.

## PUTBL (Put Block)

- Function:** Outputs the number of bytes of data indicated in the HL register to the file whose logical number is specified in the C register, starting at the address indicated in the DE register.
- Input registers:** The C register contains the logical number. The DE register contains the starting address of the data to be output. The HL register contains the number of bytes of data to be output.
- Calling procedure:** CALL PUTBL
- Output registers:** CF = 0 ..... Normal (DE ← address following the end of the block output)  
CF = 1 ..... Error (A ← error code)  
IY : Starting address of the device table (see note 4 on page 12)
- Registers preserved:** Only registers BC and IX. (Register HL is also preserved if C flag = 0)

## PUTCR (Put Carriage Return)

- Function:** Outputs a carriage return to the file whose logical number is specified in the C register.
- Input registers:** The C register contains the logical number.
- Calling procedure:** CALL PUTCR
- Output registers:** CF = 0 ..... Normal  
CF = 1 ..... Error (A ← error code)  
IY : Starting address of the device table (see note 4 on page 12)
- Registers preserved:** Only registers BC, DE, HL and IX.

## PUTM (Put Message)

### PUTMX

- Function:** Outputs the line starting at the address indicated in register DE to the file whose logical number is specified in the C register. PUTM and PUTMX operate in the same manner except for their handling of \$CRT and \$LPT. Cursor control operations (▢, ▣, etc.) are executed only when PUTM is used; when PUTMX is used, they are only displayed or printed as reverse characters. The end code (ODH) is not output.
- Input registers:** The C register contains the logical number. The DE register contains the starting address of the data to be output.
- Calling procedure:** CALL PUTM or CALL PUTMX
- Output registers:** CF = 0 ..... Normal  
CF = 1 ..... Error (A ← error code)  
IY : Starting address of the device table (see note 4 on page 12)
- Registers preserved:** Only registers BC, DE, HL and IX.

## **CLOSE (Close File)**

## **KILL (Kill File)**

Function: Closes or kills the file whose logical number is specified in the C register. If this subroutine is called when the C register contains 0, all currently opened files will be closed or killed. (This excludes files which were opened by FDOS itself.)

Input registers: The C register contains 0 or a logical number.

Calling procedure: CALL CLOSE or CALL KILL

Output registers: CF = 0 ..... Normal  
CF = 1 ..... Error (A ← error code)  
IY : Starting address of the device table (see note 4 on page 12)

Registers preserved: Only registers BC, DE, HL and IX.

## **LUCHK (LU Number Check)**

Function: Checks whether a logical number (contained in the C register) has been defined.

Input registers: The C register contains the logical number.

Calling procedure: CALL LUCHK

Output registers: CF = 1 ..... The logical number has not been defined.  
CF = 0 ..... The logical number has been defined.  
L' ← device number (see page 26 concerning device numbers)  
IY ← starting address of the device table. (see note 4 on page 12)

Registers preserved: All registers except AF, HL, IY, D' and L'.

Example: LD C, 5 ; logical number  
CALL LUCHK  
JP C, NOTUSE  
EXX  
LD A, L ; device number  
EXX  
CP 4  
JP C, FD  
.  
.  
.

## —Utility Subroutines—

### MTOFF (Motor Off)

Function: Stops the motor of the floppy disk drive. (The drive motor is activated automatically when necessary.)

Calling procedure: `CALL MTOFF`

Registers preserved: All registers except AF.

### BREAK (Check Break Key)

Function: Checks whether `BREAK` has been pressed.

Input registers: None

Calling procedure: `CALL BREAK`

Output registers: CF = 0 ..... Not pressed.  
CF = 1 ..... Pressed. (In this event,  $A \leftarrow 37$ . 37 is the error code.)

Registers preserved: All registers except AF.

### HALT (Halt Action with Break Action)

Function: Checks the keyboard and, if the `SPACE` key is pressed, stops execution until the `SPACE` key is pressed again. If `BREAK` is pressed,  $A \leftarrow 37$  and  $CF \leftarrow 1$ . (37 is the error code.)

Input registers: None

Calling procedure: `CALL HALT`

Output registers: CF = 0 ..... Normal  
CF = 1 ..... `BREAK` was pressed. (In this event,  $A \leftarrow 37$ .)

Registers preserved: All registers except AF.

## SGETL (Screen Get Line)

**Function:** Inputs one line from the keyboard. The keyboard is provided with the automatic repeat feature. The line which is actually input is the line in which the cursor is located when **CR** is pressed; the maximum number of characters which can be input is 160.

**Input registers:** The DE register contains the starting address of the area (80 bytes required) in which the data is to be stored.

**Calling procedure:** CALL SGETL

**Output registers:** CF = 0 ..... Normal  
CF = 1 ..... **BREAK** was pressed. A ← 0 (not 37)

**Registers preserved:** All registers except AF.

## LTPNL (Let Printer New Line)

## PMSGX (Printer Message X)

## PMSG (Printer Message)

## PPRNT (Printer Print)

## PPAGE (Printer Page)

**Function:** These are printer control routines. Each routine performs the same function for the printer as does the corresponding monitor subroutine shown below for the CRT.

| <b>Printer</b>    | <b>CRT</b> |
|-------------------|------------|
| LTPNL             | LETNL      |
| (carriage return) |            |
| PMSGX             | MSGX       |
| PMSG              | MSG        |
| PPRNT             | PRNT       |
| PPAGE             | _____      |

**Output registers:** CF = 0 ..... Normal  
CF = 1 ..... Error (A ← error code)

**Registers preserved:** All registers except AF and IY.

**C&L1**

**&NL**

**&PRNT**

**&NMSG**

**&MSG**

**&1L**

Function: Each subroutine directs output to the printer or CRT depending on the presence or absence of the global switch (/P). &NL, &NMSG and &1L include the HALT function (see page 17 for the HALT function).

C&L1 ..... Prepares either the printer or the CRT. This routine must be called before any other routines are used. Further, "?GSW" must be called before this routine is called.

&HL ..... Performs the same function as LETNL.

&PRNT ..... Performs the same function as PRNT.

&MSG ..... Performs the same function as MSG.

&NMSG ..... Executes &NL, then executes &MSG.

&1L ..... Executes &MSG, then executes &NL.

Output registers: CF = 0 ..... Normal  
CF = 1 ..... Error (A ← error code)

Registers preserved: All registers except AF and IY.

See "LINKING ASSEMBLY PROGRAM WITH FDOS" in Appendix for an example of use.

## CHKACC (Check Acc)

Function: Checks whether the contents of A register (accumulator) match any of several different given data items.

Input registers: A contains the data items to be checked.

Calling procedure: `CALL CHKACC`  
`DEFB n` ; number of data items (1-255)  
`DEFB data 1`  
`DEFB data 2` } n items of data to be compared  
`DEFB data 3` } `DEFM '.....'` may be used with ASCII.  
`DEFB data n`

Output registers: `ZF = 1` ..... One of the data items matches the contents of A.  
`ZF = 0` ..... No match was found.

Registers preserved: All registers except the flags.

## MULT (Multiply)

Function: Multiplies the contents of the DE register and the HL register (handling them as 16-bit unsigned integers) and places the result in the DE register.

Input registers: DE, HL

Calling procedure: `CALL MULT`

Output registers: `CF = 1` ..... Overflow (result cannot be expressed in 16 bits)  
`CF = 0` ..... Normal. The DE register contains the result of the calculation.

Registers preserved: All registers except AF, DE and HL.

## SOUND (Warning Sound)

Function: Produces the sound "A0+ARA+AR" to indicate that an error has occurred.

Calling procedure: `CALL SOUND`

Registers preserved: All registers.

## BINARY (Convert ASCII to Binary)

Function: Converts an ASCII numeric string into a 16-bit unsigned integer.

Input registers: The HL register contains the starting address of the ASCII numeric string.

Calling procedure: CALL BINARY

Output registers: CF = 1 ..... Overflow (cannot be expressed within 16 bits)  
CF = 0 ..... Normal. The DE register contains the converted data. The HL register contains the address following the end of the numeric string.  
If the ASCII characters indicated by HL are not a numeric string,  
CF ← 0 and DE ← 0.

Registers preserved: All registers except AF, DE and HL.

Example:

```
LD HL, BUFFER
CALL BINARY
JP C, ERROR ; if CF = 0, DE becomes 400H.
 : HL points to 0DH.
BUFFER: DEFM ' 1024 '
 DEFB 0DH ; must be an ASCII code for other than ' 0 ' - ' 9 '.
```

## CASCII (Convert Binary to ASCII)

Function: Converts a 16-bit unsigned integer into an ASCII numeric string.

Input registers: The HL register contains the 16-bit unsigned integer. The DE register contains the address of the area in which the ASCII numeric string is to be stored.

Calling procedure: CALL CASCII

Output registers: The DE register contains the ending address of the ASCII numeric string obtained.

Registers preserved: All registers except AF and DE.

Example:

```
LD HL, 1024
LD DE, BUFFER
CALL CASCII
 :
BUFFER: DEFS 10 ; after conversion the ASCII numeric string ' 1024 '
 is stored.
```

## CLEAR (Clear Area)

Function: Loads a continuous area in the memory with zeros. (The memory area must be 255 bytes or less.)

Input registers: None

Calling procedure: CALL CLEAR

DEFB length ; number of bytes to be cleared.

DEFW address ; the memory is cleared starting at this address.

Output registers: None

Registers preserved: All registers.

## CHLDE (Compare HL, DE)

Function: Compares the contents of the HL register with the contents of the DE register.

Input registers: HL and DE

Calling procedure: CALL CHLDE

Output registers: FLAG  $\leftarrow$  HL – DE; that is

CF = 0, ZF = 0 ..... HL > DE

CF = 1, ZF = 0 ..... HL < DE

CF = 0, ZF = 1 ..... HL = DE

Registers preserved: All registers except AF.

## LCHK (Limit Check)

Function: Compares the last usable memory area (the address indicated by the stack pointer minus 256) with the contents of the HL register.

Input registers: HL

Calling procedure: CALL LCHK

Output registers: CF = 0 ..... HL  $\leq$  SP–256

CF = 1 ..... HL > SP–256

At this time, A  $\leftarrow$  21. 21 is an error code. (memory protect error)

Registers preserved: All registers except AF.

## ERR (Display Error Message)

**Function:** Displays an error message (see the System Error Messages in System Command for details). The contents of the C register and the IY register must be preserved from the time the error occurs until this routine is called. Further, the CLOSE or KILL routine must not be called during that time (otherwise, the contents of the error message may be incorrect).

**Input registers:** The A register contains the error code (no error message is output if the error code is FFH).

The C register contains the logical number.

The IY register contains the starting address of the device table (see note 4 on page 12) | These may not be necessary depending on the type of error.

**Calling procedure:** CALL ERR

**Output registers:** A ← FFH

CF ← 1

**Registers preserved:** All registers except AF.

Example:

```
CALL SGETL (Page 18)
CALL NC, & 1L (Page 19)
JR NC, - 6
CALL C, ERR
RET C
```

## ERRX

**Function:** This function displays a colon (":"), followed by the contents of the area from the address following a specified 0DH until the next 0DH; the specified 0DH is the one which is the (ACC-1)th from the address indicated in the DE register.

**Input registers:** The DE register contains the starting address of the message block.

The A register contains a number (1-255).

**Calling procedure:** CALL ERX

**Output registers:** A ← FFH

CF ← 1

**Registers preserved:** All registers except AF.

Example:

```
ERMSG: DEFM 'SYNTAX'
 DEFB 0DH
 DEFM 'OVERFLOW'
 DEFB 0DH
 DEFM 'IL DATA'
 DEFB 0DH
 :
 LD A, 2
 LD DE, ERMSG
 CALL ERX
 This displays ' : OVERFLOW '.
```

## ERWAIT

### (Display Error Message and Wait Space Key)

**Function:**

1. Calls subroutine ERR if A ≠ 0.
2. Displays the contents of the area starting with the address indicated in the DE register until 0DH.
3. Displays ", ↵ space key" if A = 0.
4. Waits until SPACE or BREAK is pressed.

**Input registers:** A and DE

**Calling procedure:** CALL ERWAIT

**Output registers:** CF = 0 ..... SPACE was pressed.

CF = 1 ..... BREAK was pressed. (A ← 37)

**Registers preserved:** All registers except AF.

## —FDOS Common Variables—

### LIMIT (Limit of Memory)

Number of bytes: 2

Meaning: Contains the last address plus one of RAM mounted.

### ISTACK (Initial Stack Pointer)

Number of bytes: 2

Meaning: Contains the last address plus one of the memory area which is available to FDOS. This data is used by FDOS for initialization of the stack pointer. The contents of ISTACK may be changed by the FDOS LIMIT instruction. The contents of ISTACK must not be changed by any other means.

### ZMAX

Number of bytes: 2

Meaning: Contains the last address of the area being used by FDOS (excluding the stack). The contents of ZMAX may be changed depending on the next subroutine called. (ROPEN, WOPEN, CLOSE, KILL, . CLI)

Caution: The area which may be used within the user program as free area is as follows.

1. [Lowest address] = [value contained in ZMAX when the user program was entered]  
+ [number of files which are simultaneously opened  
(ROPEN or WOPEN)] × 350  
+ [number of files which are simultaneously write-opened]  
× 72  
+ [number of floppy disk units used] × 128

[Maximum address] = [stack pointer (SP)] -  $\alpha$ ,  $\alpha$  is approximately 256.

2. From ISTACK to LIMIT-1.

3. Area reserved by the DEFS statement within the assembly program.

### **.DNAME (Default File Name)**

Number of bytes: 17

Meaning: The file name and succeeding 0DH contained in this area will be used as the default file name when the file name is omitted. For example, when this area contains "ABCD CR", "\$FD3" appearing on the command line will be interpreted as "\$FD3;ABCD".

### **MDRIVE (Master Boot Drive)**

Number of bytes: 1

Meaning: Contains the drive number minus one (0–3) of the drive containing the master diskette.

### **BDRIVE (Boot Drive)**

Number of bytes: 1

Meaning: Contains the default drive number minus 1 (0–3). The default drive number is the number which appears to the left of the prompt ">" when FDOS is in the command wait state.

### **MAXDVR (Maximum Drive)**

Number of bytes: 1

Meaning: Contains the number of floppy disk drives connected (1–4).

### **TODAY**

Number of bytes: 7

Meaning: Contains the month, day and year followed by 0DH; each element of the date is indicated with a two-digit ASCII code.

### **RJOB (Running Job Pointer)**

Number of bytes: 2

Meaning: Area which indicates how far command line interpretation has proceeded. When command lines are interpreted in a user program, the address following that of the last command line interpreted must be placed in RJOB.

## —CLI Intermediate Code Table—

| Switch |                   | Device name |               |                   |        |               |                   |
|--------|-------------------|-------------|---------------|-------------------|--------|---------------|-------------------|
| ASCII  | Intermediate code | ASCII       | Device number | Intermediate code | ASCII  | Device number | Intermediate code |
|        | 80 H              | \$FD1       | 0             | 90 H              | \$LPT  | 16            | A0H               |
| /D     | 81 H              | \$FD2       | 1             | 91 H              | \$PTP  | 17            | A1H               |
| /C     | 82 H              | \$FD3       | 2             | 92 H              | \$CRT  | 18            | A2H               |
| /E     | 83 H              | \$FD4       | 3             | 93 H              |        | 19            | A3H               |
| /G     | 84 H              | \$CMT       | 4             | 94 H              | \$SOA  | 20            | A4H               |
| /L     | 85 H              | \$MEM       | 5             | 95 H              | \$SOB  | 21            | A5H               |
| /N     | 86 H              |             | 6             | 96 H              |        | 22            | A6H               |
| /S     | 87 H              |             | 7             | 97 H              |        | 23            | A7H               |
| /P     | 88 H              |             | 8             | 98 H              | \$USR1 | 24            | A8H               |
| /O     | 89 H              |             | 9             | 99 H              | \$USR2 | 25            | A9H               |
| /T     | 8AH               | \$PTR       | 10            | 9AH               | \$USR3 | 26            | AAH               |
|        | 8BH               |             | 11            | 9BH               | \$USR4 | 27            | ABH               |
| /LF    | 8CH               | \$KB        | 12            | 9CH               |        | 28            | ACH               |
| /PN    | 8DH               | \$SIA       | 13            | 9DH               |        | 29            | ADH               |
| /PO    | 8EH               | \$SIB       | 14            | 9EH               |        | 30            | AEH               |
| /PE    | 8FH               |             | 15            | 9FH               |        | 31            | AFH               |

| Built-in commands |                   |       |                   |  | File mode |                  |                   |
|-------------------|-------------------|-------|-------------------|--|-----------|------------------|-------------------|
| ASCII             | Intermediate code | ASCII | Intermediate code |  | ASCII     | File mode number | Intermediate code |
| RUN               | B0H               | KEY   | C2H               |  | . *       | 255              | F0H               |
| DATE              | B1H               | KLIST | C3H               |  |           |                  | F1H               |
| XFER              | B2H               | BOOT  | C4H               |  | . OBJ     | 1                | F2H               |
| DIR               | B3H               | FAST  | C5H               |  | . BTX     | 2                | F3H               |
|                   | B4H               | REW   | C6H               |  |           | 3                | F4H               |
| RENAME            | B5H               |       |                   |  | . ASC     | 4                | F5H               |
| DELETE            | B6H               |       |                   |  | . RB      | 5                | F6H               |
| TYPE              | B7H               |       |                   |  |           | 6                | F7H               |
| CHATR             | B8H               |       |                   |  | . LIB     | 7                | F8H               |
| FREE              | B9H               |       |                   |  |           | 8                | F9H               |
| MON               | BAH               |       |                   |  |           | 9                | FAH               |
| TIME              | BBH               |       |                   |  | . SYS     | 10               | FBH               |
| EXEC              | BCH               |       |                   |  |           | 11               | FCH               |
|                   | BDH               |       |                   |  |           | 12               | FDH               |
|                   | BEH               |       |                   |  |           | 13               | FEH               |
| POKE              | BFH               |       |                   |  |           | 14               | FFH               |
|                   | COH               |       |                   |  |           |                  |                   |
| CONSOLE           | C1H               |       |                   |  |           |                  |                   |

### Other

Codes other than those shown in this table are expressed as is in ASCII code. However, this applies only to 01H-7FH. The codes for some small characters and graphic characters are the same as CLI intermediate codes; therefore, they cannot be used.

# BASIC RELOCATABLE LIBRARY (RELO.LIB)

The basic relocatable library contains a collection of subroutines which are required by programs created using the basic compiler. These routines are useful when basic program subroutines (external functions, external commands, etc.) are created using the assembler.

Routines contained in RELO . LIB can only be used as basic subroutines; they cannot be executed as independent assembly programs.

## **. .INT0**

## **. .INT1**

## **. .INT2 (Convert Floating to Fixed)**

Function: Converts a real number expressed in 5 bytes into a 16-bit integer. The absolute value or any decimal fraction is discarded. (Examples:  $1.5 \rightarrow 1$      $-2.7 \rightarrow -2$ )

. . INT0 ..... The input range is from  $-32768 \sim 32767$

. . INT1 ..... The input range is from  $0 \sim 255$

. . INT2 ..... The input range is from  $-32768 \sim 65535$

Input registers: The HL register contains the starting address of the 5 byte real number.

Calling procedure: `CALL . .INT0`    `CALL . .INT1`    `CALL . .INT2`

Output registers: HL  $\leftarrow$  integer

Error processing: . . INT0 ..... Upon overflow, CF  $\leftarrow$  1.

. . INT1 ..... Upon overflow, JP ER3.

. . INT2 ..... Upon overflow, CF  $\leftarrow$  1.

Registers preserved: All registers except AF and HL.

Note: The . .FLT0 and CONST subroutines (described below) are used to create the 5-byte real number.

## **. .FLT0 (Convert Fixed to Floating)**

Function: Converts a 16-bit signed integer into a 5-byte real number.

Input registers: The HL register contains the 16-bit signed integer. The DE register contains the starting address of the area in which the real number is stored.

Calling procedure: `CALL . .FLT0`

Registers preserved: All registers except AF, DE and HL.

## **CASC' (Change ASCII)**

Function: Converts a 16-bit unsigned integer into an ASCII character string and appends 0DH to the end of it.

Input registers: The HL register contains the 16-bit unsigned integer. The DE register contains the starting address of the area in which the ASCII character string is stored.

Calling procedure: CALL CASC'

Registers preserved: All registers except AF.

## **MSGs (High Speed Message)**

Function: Performs the same function as the monitor subroutine MSG, but at high speed.

Registers preserved: All registers except AF.

## **.MOVE' (Move String)**

Function: Converts a character string from type 1 to type 2. The converted character string is stored in an area called .WORD. (The type 1 and type 2 character string formats are explained on page 31.)

Input registers: The HL register contains the starting address of the character string (type 1).

Calling procedure: CALL .MOVE'

Output registers: The DE register contains the starting address of the converted character string. (The address of .WORD)

Registers preserved: All registers except AF, BC, DE and HL.

## **FASCX (Convert Floating to ASCII)**

Function: Converts a 5-byte real number into an ASCII character string and appends 0DH to the end of it.

Input registers: The HL register contains the starting address of the real number. The DE register contains the starting address of the area in which the ASCII character string is stored.

Calling procedure: CALL FASCX

Registers preserved: None

## CONST (Convert ASCII to Const)

Function: Converts a constant expressed in ASCII code into a 5-byte real number.

Input registers: The HL register contains the starting address of the constant expressed in ASCII code. The DE register contains the starting address of the area in which the result is stored.

Calling procedure: CALL CONST

Output registers: The HL register contains the first address following the constant converted.

Registers preserved: None

Error processing: JP ER3

## CHCOND (Character Condition)

Function: Compares the two character strings (type 1.)

Input registers: The HL and DE registers contain the starting addresses of each of the two character strings being compared.

Calling procedure: CALL CHCOND

Output registers:  $\text{FLAG} \leftarrow (\text{DE}) - (\text{HL})$   
that is,  
 $\text{CF} = 0, \text{ZF} = 0 \dots\dots\dots (\text{DE}) > (\text{HL})$   
 $\text{CF} = 1, \text{ZF} = 0 \dots\dots\dots (\text{DE}) < (\text{HL})$   
 $\text{CF} = 0, \text{ZF} = 1 \dots\dots\dots (\text{DE}) = (\text{HL})$

Registers preserved: All registers except AF, BC, DE and HL.

|            |             |
|------------|-------------|
| <b>ER1</b> | <b>ER13</b> |
| <b>ER2</b> | <b>ER14</b> |
| <b>ER3</b> | <b>ER21</b> |
| <b>ER4</b> | <b>ER24</b> |
| <b>ER5</b> | <b>ER37</b> |
| <b>ER6</b> | <b>ER64</b> |

Function: Error message display routine used during BASIC program execution. See the Error Message table in the BASIC compiler instruction manual (available separately).

Calling procedure: JP ER1 (SYNTAX ERROR), etc.

## BEERR (Basic Executing Error)

Function: Error message display routine used during BASIC program execution.

Calling procedure: `CALL BEERR`  
`DEFB error code (error number in BASIC)`  
`DEFM ' ERROR MESSAGE '`  
`DEFB 0DH`  
→ No return made.

## BABORT (Basic Abort)

Function: When a system error occurs during BASIC program execution, this routine displays the applicable error message and interrupts execution.

Input registers: The A register contains the error code (system error number).  
The C register is the logical number.  
The IY register contains the starting address of the device table (see note 4 on page 12).  
} May not be required depending upon the type of error.

Calling procedure: `JP BABORT`

Example: `LD C, 2`  
`CALL GET1C`  
`JP C, BABORT`

Caution: BEERR is a routine which displays \*ER nn: message in linenumber (where nn is the error number in BASIC compiler) when an error occurs in a BASIC program; BABORT is a routine which displays -ERR message in linenumber when an error occurs at the FDOS level. ON ERROR processing will be performed in both cases, if specified.

## **..STOP**

Function: Interrupts BASIC program execution. (Corresponds to the STOP instruction of the BASIC compiler.)

Calling procedure: JP ..STOP

## **...END**

Function: Terminates BASIC program execution. (This corresponds to the END instruction of the BASIC compiler.)

Calling procedure: JP ...END

## **.WORD**

Function: 257-byte general purpose area.

## **—Type 1 and Type 2 Character String Formats—**

There are two types of character strings which are handled by BASIC; these should be used as appropriate.

Type 1

DEFB length (character string length: 0 ~ 255)

DEFM '.....'

Type 2

DEFM '.....'

DEFB 0DH

# INDEX OF LIBRARY NAMES

| Name   | Type | Page |
|--------|------|------|
| &1L    | UTYL | 19   |
| &MSG   | "    | 19   |
| &NMSG  | "    | 19   |
| &NL    | "    | 19   |
| &PRNT  | "    | 19   |
| ...END | RELO | 31   |
| ..FLT0 | "    | 27   |
| ..INT0 | "    | 27   |
| ..INT1 | "    | 27   |
| ..INT2 | "    | 27   |
| ..STOP | "    | 31   |
| .CLI   | CLI  | 8    |
| .DNAME | VAR  | 25   |
| .MOVE' | RELO | 28   |
| .WORD  | "    | 31   |
| 2HEX   | MON  | 4    |
| ??KEY  | "    | 4    |
| ?DPCT  | "    | 4    |
| ?EOF   | IOCS | 14   |
| ?GSW   | CLI  | 9    |
| ?LSW   | "    | 10   |
| ?HEX   | "    | 8    |
| ?PONT  | MON  | 5    |
| ?SEP   | CLI  | 9    |
| ASCI   | MON  | 3    |
| BABORT | RELO | 30   |
| BEERR  | "    | 30   |
| BELL   | MON  | 2    |
| BDRIVE | VAR  | 25   |
| BINARY | UTYL | 21   |
| BREAK  | "    | 17   |
| BRKEY  | MON  | 3    |
| C & L1 | UTYL | 19   |
| CASC'  | RELO | 28   |
| CASCII | UTYL | 21   |
| CHCOND | RELO | 29   |
| CHKACC | UTYL | 20   |
| CHLDE  | "    | 22   |
| CHR80  | MON  | 3    |
| CHR40  | "    | 3    |
| CLEAR  | UTYL | 22   |
| CLOSE  | IOCS | 16   |

| Name   | Type | Page |
|--------|------|------|
| CONST  | RELO | 29   |
| ER1    | "    | 29   |
| ER2    | "    | 29   |
| ER3    | "    | 29   |
| ER4    | "    | 29   |
| ER5    | "    | 29   |
| ER6    | "    | 29   |
| ER13   | "    | 29   |
| ER14   | "    | 29   |
| ER21   | "    | 29   |
| ER24   | "    | 29   |
| ER37   | "    | 29   |
| ER64   | "    | 29   |
| ERR    | UTYL | 23   |
| ERRX   | "    | 23   |
| ERWAIT | "    | 23   |
| FASCX  | RELO | 28   |
| GET1C  | IOCS | 13   |
| GET1L  | "    | 13   |
| GETBL  | "    | 13   |
| GETCRT | MON  | 3    |
| GETL   | "    | 3    |
| GETKY  | "    | 3    |
| HALT   | UTYL | 17   |
| HEX    | MON  | 3    |
| HLHEX  | "    | 4    |
| IBU1   | "    | 6    |
| IBU18  | "    | 6    |
| IBU20  | "    | 6    |
| IBU22  | "    | 6    |
| IBU24  | "    | 6    |
| IBUFE  | "    | 6    |
| ISTACK | VAR  | 24   |
| KILL   | IOCS | 16   |
| LCHK   | UTYL | 22   |
| LETNL  | MON  | 2    |
| LIMIT  | VAR  | 24   |
| LTPNL  | UTYL | 18   |
| LUCHK  | IOCS | 16   |
| MAXDVR | VAR  | 25   |
| MDRIVE | "    | 25   |
| MELDY  | MON  | 2    |

| Name   | Type | Page |
|--------|------|------|
| MODECK | IOCS | 12   |
| MSG    | MON  | 2    |
| MSGS   | RELO | 28   |
| MSGX   | MON  | 2    |
| MTOFF  | UTYL | 17   |
| MULT   | "    | 20   |
| NL     | MON  | 2    |
| PMSG   | UTYL | 18   |
| PMSGX  | "    | 18   |
| PPAGE  | "    | 18   |
| PPRNT  | "    | 18   |
| PRNT   | MON  | 2    |
| PRNTS  | "    | 2    |
| PRTHL  | "    | 3    |
| PRTHX  | "    | 3    |
| PUSHR  | "    | 5    |
| PUSHR2 | "    | 5    |
| PUT1C  | IOCS | 14   |
| PUT1L  | "    | 14   |
| PUTBL  | "    | 15   |
| PUTCR  | "    | 15   |
| PUTM   | "    | 15   |
| PUTMX  | "    | 15   |
| RDDAT  | MON  | 5    |
| RDINF  | "    | 5    |
| RJOB   | VAR  | 25   |
| ROPEN  | IOCS | 11   |
| SGETL  | UTYL | 18   |
| SOUND  | "    | 20   |
| TESW   | CLI  | 10   |
| TIMRD  | MON  | 2    |
| TIMST  | "    | 2    |
| TODAY  | VAR  | 25   |
| TRS10  | CLI  | 8    |
| VERFY  | MON  | 5    |
| WOPEN  | IOCS | 11   |
| WRDAT  | MON  | 5    |
| WRINF  | "    | 5    |
| XTEMP  | "    | 2    |
| ZMAX   | VAR  | 24   |

Type: MON ..... Monitor subroutine  
 CLI ..... CLI subroutine  
 IOCS ..... IOCS subroutine  
 UTYL ..... Utility subroutine  
 VAR ..... FDOS common variable subroutine  
 RELO ..... BASIC relocatable library

} FDOS subroutines

# Personal Computer **mz-80B**

Appendix

**SHARP**

### NOTICE

The MZ-80 series of sophisticated personal computers is manufactured by the SHARP CORPORATION. Hardware and software specifications are subject to change without prior notice; therefore, you are requested to pay special attention to version numbers of the monitor and the system software (supplied in the form of cassette tape or mini-floppy disk files).

This manual is for reference only and the SHARP CORPORATION will not be responsible for difficulties arising out of inconsistencies caused by version changes, typographical errors or omissions in the descriptions.

This manual is based on the SB-1500 series monitor and the SB-7000 series FDOS.

— CONTENTS —

|                                                           |    |
|-----------------------------------------------------------|----|
| <b>LINKING ASSEMBLY PROGRAM WITH FDOS</b> .....           | 1  |
| Sample Program (Command) .....                            | 2  |
| <b>USER CODED I/O ROUTINES</b> .....                      | 4  |
| User I/O Routine .....                                    | 4  |
| Relocating User I/O Routines .....                        | 6  |
| Linking User I/O Routines with FDOS .....                 | 6  |
| Sample Program (I/O Driver) .....                         | 8  |
| <b>CONVERSION OF CASSETTE BASED SYSTEM PROGRAMS</b> ..... | 11 |
| <b>MEMORY EXPANSION</b> .....                             | 12 |
| <b>I/O MAP</b> .....                                      | 12 |
| <b>PAPER TAPE PUNCH AND READER INTERFACE</b> .....        | 13 |
| Single Name .....                                         | 13 |
| I/O Ports .....                                           | 13 |
| Timing Chart .....                                        | 14 |
| Preparing a Paper Tape Punch/Reader I/O Card .....        | 15 |
| <b>COMMAND TABLES &amp; ERROR MESSAGES</b> .....          | 17 |
| FDOS Built-in Commands .....                              | 17 |
| FDOS Transient Commands .....                             | 20 |
| System Error Messages .....                               | 23 |
| Editor Commands .....                                     | 25 |
| Editor Error Messages .....                               | 26 |
| Assembler Messages .....                                  | 27 |
| Symbolic Debugger Commands .....                          | 28 |
| Symbolic Debugger Error Messages .....                    | 29 |
| PROM Formatter Commands .....                             | 30 |
| File Mode .....                                           | 31 |
| I/O Devices Handled by FDOS .....                         | 31 |
| File Attributes .....                                     | 32 |
| <b>ASCII CODE TABLE</b> .....                             | 33 |
| <b>MZ-80B CIRCUIT DIAGRAMS</b> .....                      | 34 |
| <b>UNIVERSAL I/O CARD CIRCUIT DIAGRAM</b> .....           | 45 |



# LINKING ASSEMBLY PROGRAM WITH FDOS

An object program generated with the FDOS editor, assembler and linker can be executed with the RUN command.

Example 1: 1 > RUN GALAXY [CR]

This command loads GALAXY.OBJ into memory from the floppy disk and executes it. Execution of a RET statement in the object program returns control to FDOS. The contents of the stack pointer must be restored to the value contained when the object program was called before the RET statement is executed. CF must be reset before control is returned because an error message will be output on the assumption that the ACC contains a system error code if CF is set.

Global switches and/or arguments can be assigned after the file name in the RUN command as shown below.

Example 2: 2 > RUN ASMZ8/P CONTROL-A [CR]

                    ↑                    ↑  
                    Global switch  Argument

In this case, FDOS converts the entire command line into intermediate codes (refer to the LIBRARY/PACKAGE Manual), and loads ASMZ8.OBJ into memory from the floppy disk, then executes it. At this time, the HL register points to the intermediate code corresponding to /P (88H). The RJOB area in FDOS has the same value as the HL register.

Switches and arguments following the file name (ASMZ8) must be decoded by user object program. They can be decoded using FDOS subroutines. When the last character (" : " or 0DH) is decoded, the HL register contents must be stored in RJOB. To return control to FDOS, execute a RET statement in the object program.

The sample program listed on the following pages illustrates command line decoding. It outputs an ASCII file to the CRT display or printer. This program operates in a manner similar to the FDOS TYPE command. The file name of this program is TYPE'. Thus, executing

Example 3: 2 > RUN TYPE' ABC [CR]

outputs file ABC.ASC to the CRT display and executing

Example 4: 2 > RUN TYPE' /P ABC [CR]

outputs ABC.ASC to the printer.

All external labels (indicated by the E message) in this program list are defined in FDOSEQU. LIB.

See page SYS- 51 is System Command for the RUN command.

## —Sample Program (Command)—

\*\* Z80 ASSEMBLER SB-7201 <TYPE> PAGE 01

??/??/??

```

01 0000 ; TYPE COMMAND
02 0000 ;
03 0000 .TYPE: ENT
04 0000 116200 LD DE,SWTBL ; DE:=SWITCH TABLE
05 0003 CD0000 E CALL ?GSW ; CHECK GLOBAL SWITCH
06 0006 D8 RET C
07 0007 CD0000 E CALL C&L1 ; SELECT CRT OR LPT
08 000A CD0000 E CALL ?SEP ; CHECK SEPARATER
09 000D D8 RET C
10 000E FE2C CP 2CH ; SEPARATER="," ?
11 0010 3E03 LD A,3 ; 3 IS ERR CODE
12 0012 37 SCF
13 0013 C0 RET NZ ; NO, ERR RETURN
14 0014 CD0000 E TYPE0: CALL ?LSW ; CHECK LOCAL SWITCH
15 0017 D8 RET C
16 0018 3E08 LD A,8 ; 8 IS ERR CODE
17 001A 37 SCF
18 001B C0 RET NZ ; ERROR, LSW EXIST
19 001C 0E80 LD C,128 ; LH#:=128
20 001E D9 EXX
21 001F 0604 LD B,4 ; DEFAULT MODE=ASC
22 0021 D9 EXX
23 0022 CD0000 E CALL ROPEN ; READ-OPEN
24 0025 D8 RET C
25 0026 CD0000 E CALL &NL
26 0029 3820 JR C,TYPEER
27 002B CD0000 E CALL TESW ; TEST GLOBAL SWITCH
28 002E 88 DEFB 88H ; /P
29 002F 3F CCF
30 0030 CD0000 E CALL C,PPAGE ; LPT PAGING
31 0033 3823 JR C,TYPEER
32 0035 CD0000 E CALL MODECK ; FILEMODE CHECK
33 0038 04 DEFB 4 ; .ASC ?
34 0039 116400 LD DE,BUFFER
35 003C D40000 E TYPE10: CALL NC,GET1L ; GET 1 LINE
36 003F D40000 E CALL NC,&1L ; DISP OR PRINT 1 LINE
37 0042 30FB JR NC,TYPE10 ; NO ERROR
38 0044 A7 AND A
39 0045 C25800 JP NZ,TYPEER ; ERROR
40 0048 ; ; END-OF-FILE
41 0048 CD0000 E TYPE20: CALL CLOSE ; CLOSE FILE
42 004B CD0000 E CALL ?SEP ; CHECK SEPARATER
43 004E D8 RET C
44 004F FE2C CP 2CH ; SEPARATER="," ?
45 0051 28C1 JR Z,TYPE0 ; YES, TYPE NEXT FILE
46 0053 220000 E LD (RJOB),HL ; SAVE CLI POINTER
47 0056 AF XOR A
48 0057 C9 RET
49 0058 CD0000 E TYPEER: CALL ERR ; ERROR OCCUR
50 005B CD0000 E CALL KILL ; KILL FILE (C=128)
51 005E 37 SCF ; SET ERROR FLAG
52 005F 3EFF LD A,FFH ; ALREADY DISP ERR MSG
53 0061 C9 RET
54 0062 88 SWTBL: DEFB 88H ; /P
55 0063 FF DEFB FFH ; END OF SWTBL
56 0064 BUFFER: DEFS 128 ; 128 BYTE BUFFER
57 00E4 END

```

\*\* Z80 ASSEMBLER SB-7201 <TYPE> PAGE 02

??/??/??

.TYPE 0000 BUFFER 0064 SWTBL 0062 TYPE0 0014 TYPE10 003C  
TYPE20 0048 TYPEER 0058

# USER CODED I/O ROUTINES

FDOS supports control programs not only for the floppy disk drive but for the printer (\$LPT) and the paper tape reader (\$PTR), etc. Other I/O devices can be operated under the control of FDOS by means of user coded control programs.

## —User I/O Routine—

A user I/O routine consists of the following sections.

- A. Device table (57 bytes)
- B. ROPEN or WOPEN procedure
- C. Data transfer program
- D. CLOSE and KILL procedure

These sections are explained below using the FDOS paper tape reader control program (\$PTR) as an example.

### A. Device Table (line 7 through 20, bytes 0 through 56)

- \* FDOS uses bytes 0 through 2 (FLAG 0 ~ FLAG 2).

This area must be written exactly as it is shown.

- \* Byte 3 (FLAG 3) represents the attribute of the I/O device.

Bit 7 : 0

Bit 6 : 1 indicates that tabulation is possible. (This bit is set to 1 for the printer. See Note 1 on page 7.)

Bit 5 : 1 indicates that parity specification (\$PTR/PE, etc.) can be made.

Bit 4 : 1 indicates that only .ASC mode files can be transferred.

Bit 3 : 0

Bit 2 : 0

Bit 1 : 1 indicates that WOPEN is possible. (See note 2 on page 7.)

Bit 0 : 1 indicates that ROPEN is possible. (See note 2 on page 7.)

- \* Byte 4 indicates the data transfer format. (described later)
- \* Bytes 5 and 6 are the starting address of the subroutine to be called during ROPEN execution.
- \* Bytes 7 and 8 are the starting address of the subroutine to be called during WOPEN execution. (WOPEN is not executed for \$PTR so DEFW 0 is specified in this program.)
- \* Bytes 9 and 10 are loaded with data by the FDOS STATUS command. (Not used for \$PTR.)
- \* Bytes 11 through 14 are the starting addresses of the subroutines for CLOSE and KILL processing.
- \* Bytes 15 through 22 (Procedures 1 through 4) are loaded with data transfer routine addresses. The data transfer procedure differs according to the transfer format.

### ROPEN

| Transfer format | 1                                 | 4                                                                |
|-----------------|-----------------------------------|------------------------------------------------------------------|
| Procedure 1     | Input 1 character<br>(ACC ← data) | Input 1 line<br>(From the address indicated by DE to a CR code.) |
| Procedure 2 ~ 4 | Unused                            | Unused                                                           |

### WOPEN

| Transfer format | 1                                               | 5                                                        |
|-----------------|-------------------------------------------------|----------------------------------------------------------|
| Procedure 1     | Unused                                          | Carriage return                                          |
| Procedure 2     | Output 1 character<br>(ACC : data) <sup>†</sup> | Output 1 character<br>(ACC : data) <sup>†</sup>          |
| Procedure 3     | Unused                                          | Output 1 line<br>(Corresponds to monitor subroutine MSG) |
| Procedure 4     | Unused                                          | Output line<br>(Corresponds to monitor subroutine MSGX)  |

<sup>†</sup> On .ASC mode, 0DH means carriage-return, and 0CH means form-feed.

- \* Bytes 23 and 24 are used by FDOS.
- \* Byte 25 is used only when bit 6 of FLAG 3 is 1, in which case it must be loaded with the number of characters of the line which have been output by I/O routine.
- \* Byte 26 is loaded with the file mode by FDOS.
- \* Bytes 27 through 56 are the device name (up to 16 characters); the rest area must be reserved with DEFS.
- \* When the transfer format is 4, a buffer area for 1 line is reserved after the byte 56 with DEFS.

#### B. ROPEN or WOPEN procedure (lines 39 through 50)

Only ROPEN is needed for the paper tape reader (\$PTR). The tape feeder is skipped by this procedure. WOPEN is also used to start a new page during output of an assembly listing.

#### C. Data transfer program (lines 51 through 83)

Program which performs actual transfer of data.

#### D. CLOSE and KILL procedures (lines 49 through 50)

No function with \$PTR.

To return control to FDOS from the ROPEN, WOPEN, Procedure 1 ~ 4, CLOSE or KILL routines, set registers as follows before executing the RET statement.

Normal : CR ← 0

Error : CF ← 1, ACC ← error code (refer to the System Error Messages in the System Command Manual.)

File end : CF ← 1, ACC ← 0

The contents of the IY, BC', DE' and HL' registers must be saved in any case.

## —Relocating User I/O Routines—

First, assemble the program coded (the program name DVM is used below).

Example 1: 2 > ASM DVM, \$LPT/L CR

Next, relocate the file to generate the object program. A higher loading address must be specified at this time because of factors related to the LIMIT command described later. Take care to ensure that addresses do not overlap when two or more user I/O programs are used. If necessary, link MONEQU.LIB or FDOSEQU.LIB with the user I/O programs.

Example 2: 2 > LINK \$C000, DVM CR

Example 2': 2 > LINK \$C400, CDISP, \$FD1 ; FDOSEQU.LIB CR

## —Linking User I/O Routines with FDOS—

User I/O routines must be linked with FDOS I/O controller every time FDOS is activated.

First, use the LIMIT command to reserve an area in memory for loading the object program (DVM.OBJ).

Example 3: 2 > LIMIT \$F000 CR

Next, load the object program.

Example 4: 2 > LOAD DVM CR

Finally, link the routine to the FDOS I/O controller. \$USR1 through 4 are provided in FDOS as device names for user I/O routines; assign the user I/O control routine to one of these device names.

Example 5: 2 > ASSIGN \$USR1, \$F000 CR

Now the user program is linked with FDOS and can be called by specifying \$USR1 (~4). It is convenient to prepare EXEC files which include LIMIT, LOAD and ASSIGN commands such as those shown above. (Refer to the System Command Manual).

User I/O programs are called as shown below.

Example of use by FDOS commands

2 > TYPE \$USR1 CR

2 > XFER DATA4, \$USR2 CR

Example of use by BASIC compiler

```
10 ROPEN #2, "$USR1"
20 INPUT #2, A$
30 IF EOF (#2) THEN 100
40 PRINT A$
.....
999 CLOSE #2
```

Note 1: Bit 6 determines the functions of BASIC statements PRINT # and INPUT #.

When bit 6 = 1, data is treated in the same manner as with the PRINT and INPUT statements. When bit 6 = 0, separators (" , " and " ; ") in the PRINT # statement are replaced with CR and commas included in the input character string for the INPUT # statement are treated as data; only CR is regarded as a data separator. (This is the same as with the PRINT # statement supported by SB-6510 and the PRINT/T statement supported by SB-5510.)

Note 2: Both ROPEN and WOPEN are possible, when both bit 1 and bit 0 are set, but they cannot be executed simultaneously.

# —Sample Program (I/O Driver)—

\*\* Z80 ASSEMBLER SB-7201 <PTRP> PAGE 01

07/06/81

```

01 0000 ;
02 0000 ; PTR/PTP DRIVER for MZ-80B FDOS.
03 0000 ; Copyright 1981 by SHARP Corp.
04 0000 ;
05 0000 ;
06 0000 $PTR: ENT
07 0000 0000 DEFW 0
08 0002 00 DEFB 0
09 0003 21 DEFB 21H ;PAR,READ-ENABLE
10 0004 01 DEFB 1
11 0005 7900 DEFW $PTR0 ;ROPEN
12 0007 0000 DEFW 0
13 0009 0000 DEFW 0 ;STATUS
14 000B 8600 DEFW CLC ;CLOSE
15 000D 8600 DEFW CLC ;KILL
16 000F 8800 DEFW $PTR1
17 0011 DEFS 10
18 001B 24505452 $PTRNM: DEFM '$PTR'
19 001F 0D DEFB 0DH
20 0020 DEFS 25
21 0039 ;
22 0039 $PTP: ENT
23 0039 0000 DEFW 0H
24 003B 00 DEFB 0H
25 003C 22 DEFB 22H ;PAR,WOPEN-ENABLE
26 003D 01 DEFB 1
27 003E DEFS 2
28 0040 2901 DEFW $PTPFD ;WOPEN
29 0042 0000 DEFW 0H ;STATUS
30 0044 2901 DEFW $PTPFD ;CLOSE
31 0046 8600 DEFW CLC
32 0048 DEFS 2
33 004A BE00 DEFW $PTP1C
34 004C DEFS 8
35 0054 24505450 $PTPNM: DEFM '$PTP'
36 0058 0D DEFB 0DH
37 0059 DEFS 25
38 0072 ;
39 0072 111B00 $PTRNR: LD DE,$PTRNM
40 0075 CD0000 CALL IOWAIT
41 0078 D8 RET C
42 0079 CD9500 $PTR0: CALL $PTRIN ;ROPEN
43 007C 38F4 JR C,$PTRNR
44 007E 78 LD A,B
45 007F A7 AND A
46 0080 28F7 JR Z,$PTR0
47 0082 78 LD A,B
48 0083 32BD00 LD ($PTRD),A
49 0086 AF CLC: XOR A
50 0087 C9 RET
51 0088 CD9500 $PTR1: CALL $PTRIN ;GET1C
52 008B D8 RET C
53 008C 21BD00 LD HL,$PTRD
54 008F 7E LD A,M
55 0090 70 LD M,B
56 0091 A7 AND A
57 0092 C0 RET NZ
58 0093 37 SCF ;EOF
59 0094 C9 RET
60 0095 ;

```

```

01 0095 3EEF $PTRIN: LD A,FEH
02 0097 D3FD OUT (FDH),A
03 0099 CDB000 $PTR2: CALL $PTRCK
04 009C CB67 BIT 4,A
05 009E 28F9 JR Z,$PTR2
06 00A0 CDB000 $PTR3: CALL $PTRCK
07 00A3 CB67 BIT 4,A
08 00A5 28F9 JR NZ,$PTR3
09 00A7 DBFC IN A,(FCH)
10 00A9 2F CPL
11 00AA 47 LD B,A
12 00AB 3EFF $PTR5: LD A,FFH
13 00AD D3FD OUT (FDH),A
14 00AF C9 RET
15 00B0 DBFD $PTRCK: IN A,(FDH)
16 00B2 CB6F BIT 5,A
17 00B4 C8 RET Z
18 00B5 F1 POP AF
19 00B6 CDAB00 CALL $PTR5
20 00B9 37 SCF
21 00BA 3E3C LD A,60 ;NOT READY
22 00BC C9 RET
23 00BD $PTRD: DEFS 1
24 00BE ;
25 00BE ;
26 00BE F5 $PTP10: PUSH AF
27 00BF DBFD IN A,(FDH)
28 00C1 E601 AND 1
29 00C3 2818 JR Z,$PTP20
30 00C5 3EFE LD A,FEH
31 00C7 D3FD OUT (FDH),A
32 00C9 210000 LD HL,0H
33 00CC 2B $PTP10: DEC HL
34 00CD DBFD IN A,(FDH)
35 00CF E601 AND 1
36 00D1 280A JR Z,$PTP20
37 00D3 7C LD A,H
38 00D4 B5 OR L
39 00D5 00 NOP
40 00D6 00 NOP
41 00D7 3E3C LD A,60 ;NOT READY
42 00D9 2837 JR Z,$PTP60
43 00DB 18EF JR $PTP10
44 00DD ;
45 00DD 3EFE $PTP20: LD A,FEH
46 00DF D3FD OUT (FDH),A
47 00E1 DBFD $PTP30: IN A,(FDH)
48 00E3 CB47 BIT 0,A
49 00E5 20FA JR NZ,$PTP30
50 00E7 F1 POP AF
51 00E8 F5 PUSH AF
52 00E9 2F CPL
53 00EA D3FC OUT (FCH),A
54 00EC 3EFC LD A,FCH
55 00EE D3FD OUT (FDH),A
56 00F0 210000 LD HL,0H
57 00F3 2B $PTP40: DEC HL
58 00F4 7C LD A,H
59 00F5 B5 OR L
60 00F6 3E4E LD A,78 ;TIME OUT

```

```

01 00F8 2818 JR Z,$PTP60
02 00FA DBFD IN A,(FDH)
03 00FC CB4F BIT 1,A
04 00FE 20F3 JR NZ,$PTP40
05 0100 DBFD IN A,(FDH)
06 0102 CB4F BIT 1,A
07 0104 20ED JR NZ,$PTP40
08 0106 DBFD $PTP50: IN A,(FDH)
09 0108 CB4F BIT 1,A
10 010A 28FA JR Z,$PTP50
11 010C DBFD IN A,(FDH)
12 010E CB4F BIT 1,A
13 0110 28F4 JR Z,$PTP50
14 0112 E1 $PTP60: POP HL
15 0113 F5 PUSH AF
16 0114 3EFE LD A,FEH
17 0116 D3FD OUT (FDH),A
18 0118 CD3601 CALL DLY80U
19 011B 3D DEC A ;A<--FFH
20 011C D3FC OUT (FCH),A
21 011E F1 POP AF
22 011F C0 RET NZ
23 0120 37 SCF
24 0121 C9 RET
25 0122 ;
26 0122 115400 $PTPNR: LD DE,$PTPNM
27 0125 CD0000 E CALL IOWAIT
28 0128 D8 RET C
29 0129 0696 $PTPFD: LD B,150 ;FEEDER
30 012B C5 PUSH BC
31 012C AF XOR A
32 012D CDBE00 CALL $PTP1C
33 0130 C1 POP BC
34 0131 38EF JR C,$PTPNR
35 0133 10F6 DJNZ $PTPFD+2
36 0135 C9 RET
37 0136 ;
38 0136 111000 DLY80U: LD DE,16
39 0139 1B DEC DE
40 013A 7A LD A,D
41 013B B3 OR E
42 013C 20FB JR NZ,-3
43 013E C9 RET
44 013F END

```

```

$PTP 0039 $PTP10 00CC $PTP1C 00BE $PTP20 00DD $PTP30 00E1
$PTP40 00F3 $PTP50 0106 $PTP60 0112 $PTPFD 0129 $PTPNM 0054
$PTPNR 0122 $PTR 0000 $PTR1 0088 $PTR2 0099 $PTR3 00A0
$PTR5 00AB $PTRCK 00B0 $PTRD 00BD $PTRIN 0095 $PTRNM 001B
$PTRNR 0072 $PTR0 0079 CLC 0086 DLY80U 0136

```

# CONVERSION OF CASSETTE BASED SYSTEM PROGRAMS

The following cassette based system programs (for MZ-80K) have thus far been released.

- **MACHINE LANGUAGE** SP-2001
- **EDITOR-ASSEMBLER** SP-2201, SP-2102
- **RELOCATABLE LOADER** SP-2301
- **SYMBOLIC DEBUGGER** SP-2401

These system programs (and the MZ-80K FDOS) generate source files (with file mode .ASC), relocatable files (with file mode .RB), object files (with file mode .OBJ) and debug mode save files (i.e., object files with symbol tables). Of these, **source files and object files can be transferred to FDOS diskettes.**

The procedure for transferring a cassette file to an FDOS file is as follows.

When the file name consists of characters which are usable with FDOS:

**XFER \$CMT1, \$FDn (n = 1 – 4)**

When the file name includes characters which are not allowed by FDOS, a new file name must be assigned as follows:

**XFER \$CMT1, \$FDn ; filename (n = 1 – 4)**

When an assembly source file is to be transferred, use the following procedures to determine whether or not pseudo instruction REL is used: load the file with the FDOS text editor and search for REL with the S command. **Delete all REL instructions;** this is because FDOS system programs do not require REL.

Next, assemble the file from which REL instructions have been deleted to generate a relocatable file with the FDOS assembler. The object file is obtained by relocating it.

When object files generated by cassette based system programs are transferred to an FDOS file, they can be executed with the following command.

**RUN \$FDn ; filename**

The following message is displayed on the CRT screen when the specified object file has a loading address which results in destruction of the FDOS area.

**destroy FDOS?**

Pressing the **[Y]** key at this time performs the transfer operation, destroying the FDOS area; pressing the **[N]** key stops the operation and returns the system to the FDOS command wait state.

## MEMORY EXPANSION

### FDOS requires 64K bytes of RAM

The standard memory size of the MZ-80B is 32K bytes; this is expandable to 64K bytes.

The optional 32K byte expansion RAM card, MZ-80RM, is inserted into the 20-pin connector which is located on the right rear side of the CPU board (as viewed from the rear). The standard 32K byte RAM card is located next to the expansion RAM connector. The connector pins on the bottom of the expansion RAM card are inserted into the 20 pin connector on the CPU board.

Visually check orientation of the expansion RAM card before inserting it.

## I/O MAP

I/O ports with addresses equal to or higher than B0 are reserved by the manufacturer for control of external devices; those used by FDOS are assigned device names such as \$LPT.

|    |                                               |
|----|-----------------------------------------------|
| 00 | User ports                                    |
| B0 | (RS-232C)                                     |
| C0 |                                               |
| D0 |                                               |
| D8 | Floppy disk<br>(\$FD1 ~ \$FD4)                |
| E0 | 8255, 8253, PIO                               |
| EC |                                               |
| EE |                                               |
| F0 | Graphic display                               |
| F8 | EX-ROM                                        |
| FA |                                               |
| FC | Paper tape punch and<br>reader (\$PTP, \$PTR) |
| FE | Printer<br>(\$LPT)                            |

# PAPER TAPE PUNCH AND READER INTERFACE

FDOS has built-in paper tape punch and reader control programs. These are assigned the device names \$PTP and \$PTR, respectively. In actuality, however an interface circuit must be established with a universal interface I/O card to connect the paper tape punch and reader with the MZ-80 series micro-computer. The circuit diagram is shown on page 45.

The method for controlling the paper tape punch and reader is not standardized. A paper tape punch and reader which can be controlled by FDOS must have the following signal timing system. The signal names and timing charts shown below are based on the RP-600 paper tape punch and reader manufactured by Nada Electronics Laboratory. (For details, refer to the manual included with the paper tape punch and reader.)

## —Signal Name—

### < Puncher >

$\overline{DT}_1 \sim \overline{DT}_8$  : Data (PTP  $\leftarrow$  CPU)

$\overline{MI}^*$  : Motor ON/OFF control signal (PTP  $\leftarrow$  CPU)

$\overline{ST}$  : START/STOP control signal (PTP  $\leftarrow$  CPU)

$\overline{TO}$  : Timing signal (PTP  $\rightarrow$  CPU)

$(\overline{RDY})^{**}$  : Ready state signal (PTP  $\rightarrow$  CPU)

(This signal is not output from the RP-600 since it can be used in remote operation. Ground it when the RP-600 is used.)

### < Reader >

$\overline{RD}_1 \sim \overline{RD}_8$  : Data (PTR  $\rightarrow$  CPU)

$\overline{STA}$  : START/STOP control signal (PTR  $\leftarrow$  CPU)

$\overline{SPR}$  : Sprocket signal (PTR  $\rightarrow$  CPU)

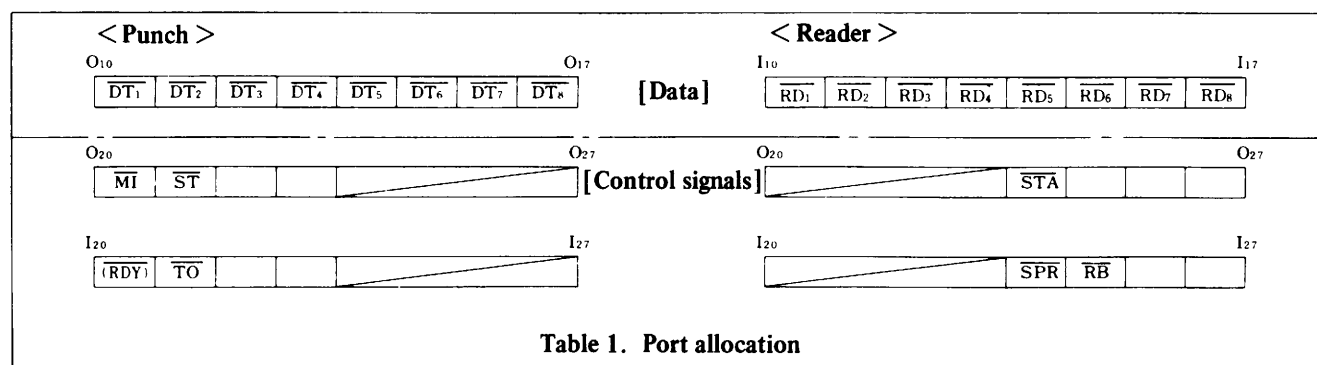
$\overline{RB}$  : Tape end signal (abnormal stop signal) (PTR  $\rightarrow$  CPU)

\* Do not connect when the motor is not remotely controlled.

\*\* The DPT26A manufactured by the Anritsu Electric Co. outputs this signal, but the RP-600 does not.

## —I/O Ports—

Port FC<sub>H</sub> is used for data by both the punch and the reader. Port FD<sub>H</sub> is used for control signals. See Table 1.



## —Timing Chart—

### Punch

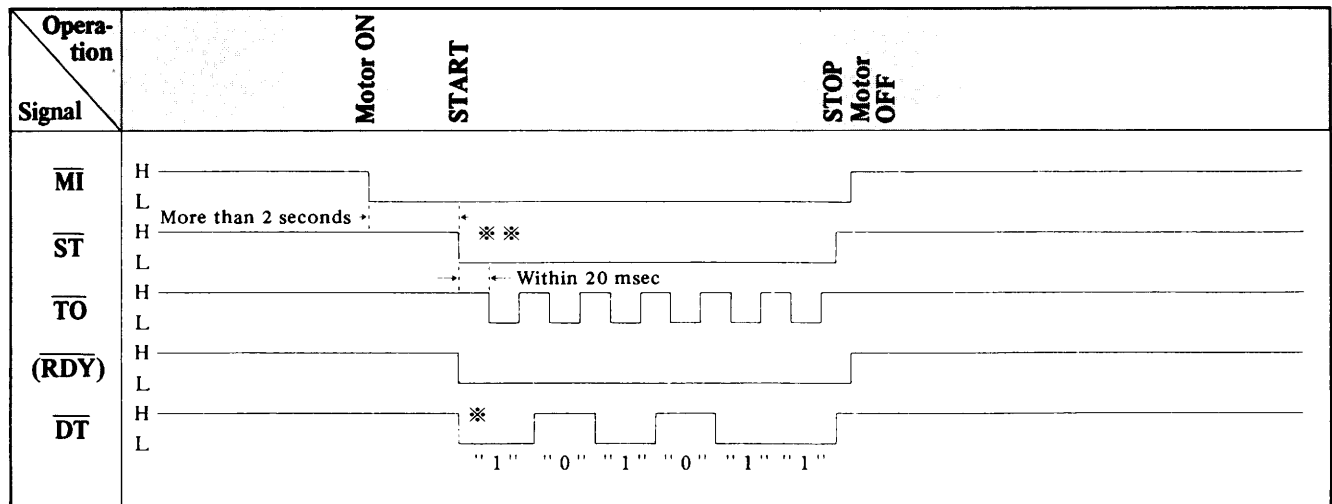


Figure 1. Punch timing chart

- \* The next data to be punched is readied while  $\overline{TO}$  is H and maintained while  $\overline{TO}$  is L.
- \*\*  $\overline{ST}$  is set to L 2 or more seconds after the motor has been started, and is set to H after  $\overline{TO}$  has risen from L to H for the last data.

### Reader

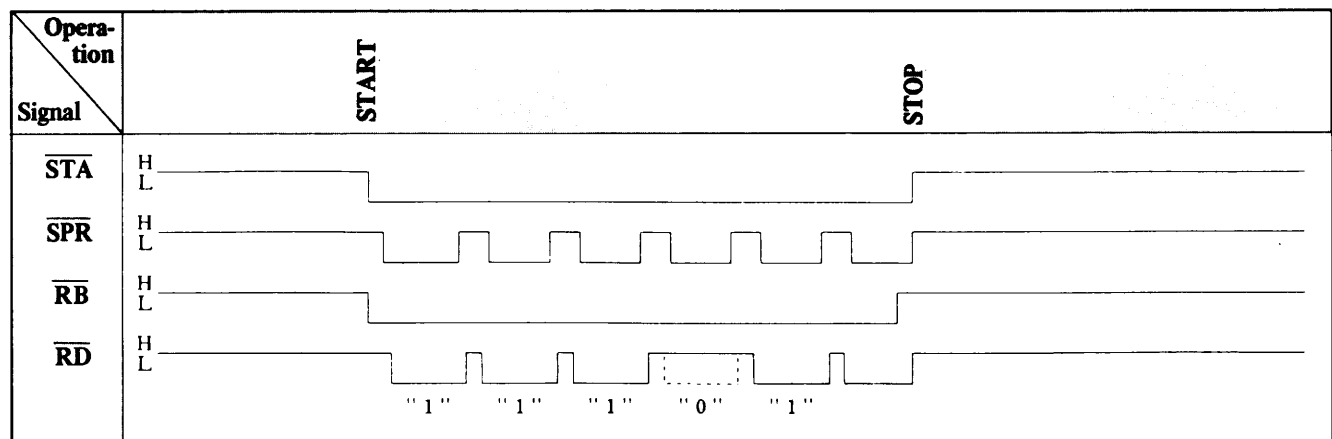


Figure 2. Reader timing chart

## —Preparing a Paper Tape Punch/Reader I/O Card—

It is convenient to use a universal I/O card (MZ-80I02) for preparing a paper tape punch and reader I/O interface circuit. Markings such as O<sub>10</sub> or O<sub>17</sub> in the port allocation table on page 13 match those on the universal I/O card. See page 16 for setting the universal I/O card switches to select port addresses FC and FD.

The RP-600 internal interface circuit and input and output pin connections are shown below for reference. (For details, refer to the manual included with the RP-600).

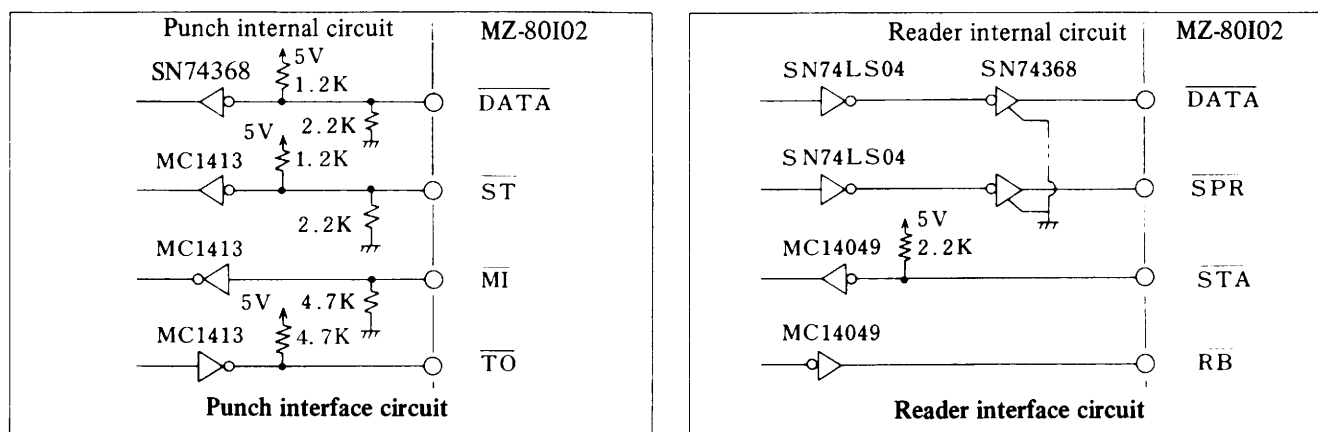


Figure 3. Interface circuit (RP-600)

Punch I/O connector

| Pin | Signal                        | Pin | Signal                              |
|-----|-------------------------------|-----|-------------------------------------|
| 1   | $\overline{DT}_1$             | 14  |                                     |
| 2   | $\overline{DT}_2$             | 15  |                                     |
| 3   | $\overline{DT}_3$             | 16  |                                     |
| 4   | $\overline{DT}_4$             | 17  |                                     |
| 5   | $\overline{DT}_5$             | 18  |                                     |
| 6   | $\overline{DT}_6$             | 19  |                                     |
| 7   | $\overline{DT}_7$             | 20  |                                     |
| 8   | $\overline{DT}_8$             | 20  |                                     |
| 9   |                               | 21  |                                     |
| 10  |                               | 22  | $\overline{MI}$ Motor ON/OFF signal |
| 11  | $\overline{TO}$ Timing signal | 23  | $\overline{ST}$ START/STOP signal   |
| 12  | GND                           | 24  | FG Frame ground                     |
| 13  |                               |     |                                     |

Reader I/O connector

| Pin | Signal                           | Pin | Signal                             |
|-----|----------------------------------|-----|------------------------------------|
| 1   | $\overline{RD}_1$                | 20  |                                    |
| 2   | $\overline{RD}_2$                | 21  |                                    |
| 3   | $\overline{RD}_3$                | 22  |                                    |
| 4   | $\overline{RD}_4$                | 23  |                                    |
| 5   | $\overline{RD}_5$                | 24  |                                    |
| 6   | $\overline{RD}_6$                | 25  |                                    |
| 7   | $\overline{RD}_7$                | 26  |                                    |
| 8   | $\overline{SPR}$ Sprocket signal | 27  |                                    |
| 9   | $\overline{RD}_8$ Data           | 28  | $\overline{STA}$ START/STOP signal |
| 10  |                                  | 29  | $\overline{RB}$ Operating state    |
| 11  |                                  | 30  | FG Frame ground                    |
| 12  | GND                              | 31  |                                    |
| 13  |                                  | 32  |                                    |
| 14  |                                  | 33  |                                    |
| 15  |                                  | 34  |                                    |
| 16  |                                  | 34  |                                    |
| 17  |                                  | 35  |                                    |
| 18  |                                  | 36  |                                    |
| 19  |                                  |     |                                    |

Table 2. Connector pin connections

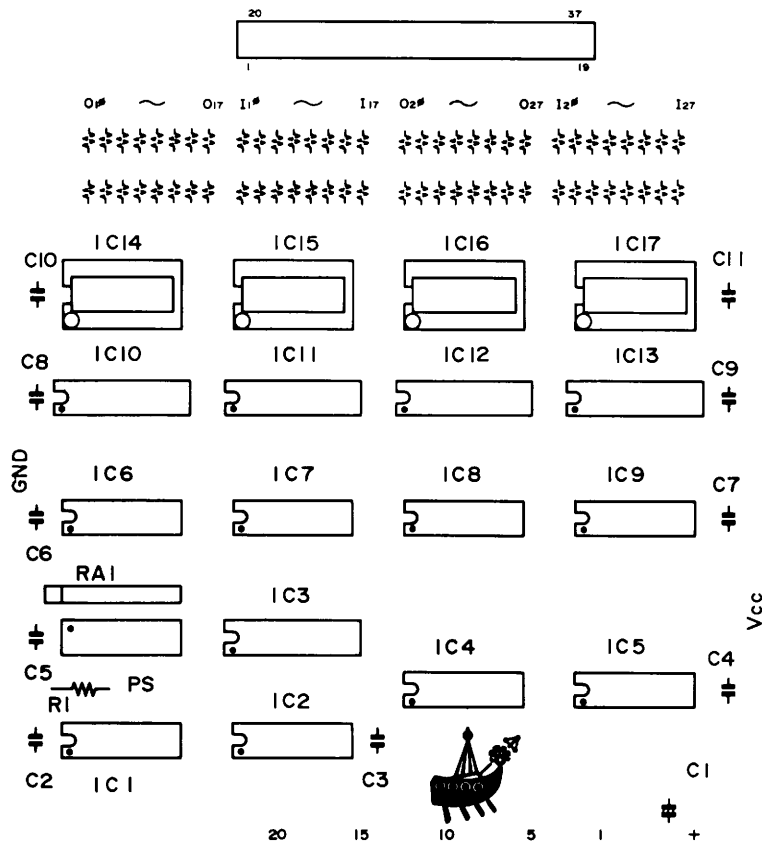


Figure 4. Universal I/O card component location (parts)

#### Universal I/O card port address setting

##### (1) Number of ports

Input : 2 ports      Output : 2 ports

##### (2) Port address

All port addresses can be set. (However, FDOS uses B0<sub>H</sub> and higher locations.)

The input port for I<sub>10</sub> ~ I<sub>17</sub> is set to an even address.

The input port for I<sub>20</sub> ~ I<sub>27</sub> is set to an odd address.

The output port for O<sub>10</sub> ~ O<sub>17</sub> is set to an even address.

The output port for O<sub>20</sub> ~ O<sub>27</sub> is set to an odd address.

##### (3) Port address setting switches (PS)

Numbers marked on the PS switches correspond to the address bus lines shown below. Turning a PS switch OFF sets the corresponding address bit to logical "1" and turning it ON to logical "0".

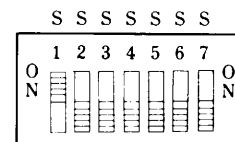
| Switch No.  | 7              | 6              | 5              | 4              | 3              | 2              | 1              |
|-------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| Address bit | A <sub>7</sub> | A <sub>6</sub> | A <sub>5</sub> | A <sub>4</sub> | A <sub>3</sub> | A <sub>2</sub> | A <sub>1</sub> |

Example: Setting the PS switches as shown below sets the port address to FC<sub>H</sub>.

1 1 1 1 1 1 0 0  
 ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑  
 S<sub>7</sub> S<sub>6</sub> S<sub>5</sub> S<sub>4</sub> S<sub>3</sub> S<sub>2</sub> S<sub>1</sub>

When the PS switches are set as shown above, ports FC<sub>H</sub> and FD<sub>H</sub> are used for this card.

S<sub>7</sub> ..... OFF  
 S<sub>6</sub> ..... OFF  
 S<sub>5</sub> ..... OFF  
 S<sub>4</sub> ..... OFF  
 S<sub>3</sub> ..... OFF  
 S<sub>2</sub> ..... OFF  
 S<sub>1</sub> ..... ON



Caution: Installing two or more interface cards which have the same port address settings will result in destruction of ICs.

# COMMAND TABLES & ERROR MESSAGES

## —FDOS Built-in Commands—

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |               |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| <b>BOOT</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |               |
| Terminates the FDOS and activates sytem IPL.<br>Example: <b>BOOT ↵</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |               |
| <b>CHATR sign, filename1, attribute [ , ...filenameN, attribute ]</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |               |
| Matches the password's sign and changes the file attribute(s) of the matching file(s) identified by filename to attribute(s).<br>P: Permanent file                      R: Read inhibit<br>0: No protection                        W: Write inhibit<br>Examples: <b>CHATR KEY, ABC, 0, XYZ, P ↵</b> : Deletes the file attribute of file ABC and changes the file attribute of file XYZ to PERMANENT if matches occur with the password KEY.<br><b>CHATR KEY, \$FD2 ; UVW, R ↵</b> : Changes the file attribute of file UVW in FD2 to READ INHIBIT if a match occurs with the password KEY.<br><b>CHATR ↵</b> : This allows the programmer to interactively specify the password, file name and attribute.                         |               |
| <b>CONSOLE Sscrolling-start-line, end-line [ , Ccharacter-number, R, N]</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |               |
| Sets the scrolling area on the CRT screen, sets the character display mode and/or reverses the picture on the screen.<br>Example: <b>CONSOLE C80 ↵</b> : Sets the number of characters per line to 80.<br><b>CONSOLE R ↵</b> : Reverses the picture on the screen.                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |               |
| <b>DATE [MM/DD/YY]</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |               |
| Displays the current date or sets the specified date in month, date, year format. The set information is used as file information when new files are created.<br>Global switch /P : Specifies that the date is to be printed on the LPT.<br>Examples: <b>DATE /P ↵</b> : Lists the current date on the LPT.<br><b>DATE 12/25/80 ↵</b> : Sets the current date to December 25, 1980.                                                                                                                                                                                                                                                                                                                                                |               |
| <b>DELETE filename1 [ , ..., filenameN]</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | <b>(? ,*)</b> |
| Deletes the file(s) specified by filename(s)<br>Global switch /C : Specifies that each file name is to be displayed on the screen for verification. The programmer must enter Y to delete it or N to suppress deletion.<br>Examples: <b>DELETE ABC . * ↵</b> : Deletes all files identified by ABC . * .<br><b>DELETE /C A * . * ↵</b> : Displays files identified by A * . * on the screen for verification before deletion.<br>filename : ABC.ASC deleted ← Indicates that the file is deleted since "Y" is entered.<br>filename : ABC.RB ← Indicates that the file is not deleted "N" is entered.<br>filename : AXY.OBJ permanent ← Indicates that the file is not deleted because it is assigned the PERMANENT file attribute. |               |

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |        |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|
| <b>DIR [\$FDn] or [filename]</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | (? ,*) |
| <p>Displays file information in the directory specified by \$FDn or of the file specified by filename on the screen.</p> <p>Global switch /P : Specifies that the file information is to be output to LPT. The file information is displayed on the screen when this switch is not specified.</p> <p>Examples: DIR ↵ : Displays all file information in the current directory on the screen.<br/> DIR/P \$FD2 ↵ : Outputs all FD2 file names to LPT and switches the currently logged disk to FD2.<br/> DIR \$FD2 ; ABC. * ↵ : Displays the file information of files in FD2 identified by ABC. * .</p> |        |
| <b>EXEC filename</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |        |
| <p>Executes the contents of the file identified by filename as FDOS commands.</p> <p>Example: EXEC ABC . ASC ↵ : Sequentially executes the FDOS commands in file ABC.</p>                                                                                                                                                                                                                                                                                                                                                                                                                               |        |
| <b>FAST</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |        |
| <p>Fast forwards the cassette tape.</p> <p>Example: FAST ↵</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |        |
| <b>FREE [\$FDn]</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |        |
| <p>Lists statistical information about the disk identified by \$FDn on the screen or on the LPT.</p> <p>Example: FREE \$FD2 ↵<br/> \$FD2 master left : XXXX used : YYYY<br/> Indicates that the diskette on FD2 is a master diskette, that the number of unused sectors is XXXX and that the number of used sectors is YYYY.</p>                                                                                                                                                                                                                                                                        |        |
| <b>KEY keynumber = "S"</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |        |
| <p>Assigns a function to the definable function key indicated by a keynumber from 1 through 20. The function is specified by writing a string or command name enclosed in double quotation marks.</p> <p>Example: KEY 1 = "RUN↵" ↵ : Assigns the function of the RUN command to key 1.</p>                                                                                                                                                                                                                                                                                                              |        |
| <b>KLIST</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |        |
| <p>Lists the definition status of the definable function keys on the screen.</p> <p>Example: KLIST ↵</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |        |
| <b>MON</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |        |
| <p>Terminates FDOS processing and returns control to the monitor.</p> <p>Example: MON ↵</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |        |
| <b>POKE \$nnnn, data [, ..., \$uuuu, dataN]</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |        |
| <p>Stores data in the specified addresses in memory.</p> <p>Example: POKE \$000D, 2010, \$000F, 40 ↵</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |        |
| <b>RENAME oldname1, newname1 [, ..., oldnameN, newnameN]</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |        |
| <p>Renames the file specified by oldname to newname.</p> <p>Examples: RENAME ABC, XYZ ↵ : Renames file ABC to XYZ.<br/> RENAME ABC, DEF, UVW, XYZ ↵ : Renames file ABC to DEF and UVW to XYZ.</p>                                                                                                                                                                                                                                                                                                                                                                                                       |        |

|                                                                                                                                                                                                                                                                                                                                                                                                                                   |                          |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|
| <b>REW</b>                                                                                                                                                                                                                                                                                                                                                                                                                        |                          |
| Rewinds the cassette tape.<br>Example: REW ↵                                                                                                                                                                                                                                                                                                                                                                                      |                          |
| <b>RUN filename</b>                                                                                                                                                                                                                                                                                                                                                                                                               |                          |
| Executes the program in the object file identified by filename.<br>Example: RUN ABC ↵ : Executes the program in file ABC, assuming it ot be ABC.OBJ.                                                                                                                                                                                                                                                                              |                          |
| <b>TIME [HH : MM : SS]</b>                                                                                                                                                                                                                                                                                                                                                                                                        |                          |
| Displays the current time or sets specified time in hour, minute, second format. This information is used as file information when new files are created. The current time is set to 00 : 00 : 00 upon system start.<br>Global switch /P : Specifies that the current time is to be listed on the LPT.<br>Examples: TIME /P ↵ : Lists the current time on the LPT.<br>TIME 16 : 30 : 30 ↵ : Sets the current time to 16 : 30 : 30 |                          |
| <b>TYPE filename1 [ , ..., filenameN]</b>                                                                                                                                                                                                                                                                                                                                                                                         | (? , * )                 |
| Lists the contents of the file(s) identified by filename(s) on the screen or on LPT.<br>Global switch /P : Lists the file contents on LPT.<br>Examples: TYPE ABC, DEF ↵ : Displays the contents of files ABC and DEF on the screen.<br>TYPE /P \$FD3 ; XYZ ↵ : Lists the contents of file XYZ in FD3 on LPT.<br>TYPE \$PTR ↵ : Reads paper tape data from PTR and displays it on the screen.                                      |                          |
| <b>XFER sourcefile1, destinationfile2 [ , ..., sourcefileN, destinationfileN]</b>                                                                                                                                                                                                                                                                                                                                                 | (sourcefile only ? , * ) |
| Transfers the source file(s) to the destination file(s).<br>Examples: XFER ABC, XYZ ↵ : Copies file ABC to XYZ.<br>XFER \$PTR, DEF ↵ : Transfers the file at the PTR to file DEF.<br>XFER XYZ, \$PTP / PE ↵ : Transfers file XYZ to the PTP with even partiy in ASCII code.                                                                                                                                                       |                          |

## —FDOS Transient Commands—

|                                                                                                                                                                                                                                                                                            |                                                                                                                           |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| <b>ASM filename</b>                                                                                                                                                                                                                                                                        |                                                                                                                           |
| Assembles the source file identified by filename and produces a relocatable file and an assembly listing.                                                                                                                                                                                  |                                                                                                                           |
| Global switch (none)                                                                                                                                                                                                                                                                       | : Specifies that the relocatable file is to be output.                                                                    |
| Global switch/N                                                                                                                                                                                                                                                                            | : Suppresses generation of the relocatable file.                                                                          |
| Local switch/O                                                                                                                                                                                                                                                                             | : Specifies that the relocatable file is to be output with the specified file name.                                       |
| Local switch/E                                                                                                                                                                                                                                                                             | : Specifies that error statements are to be output to the specified file.                                                 |
| Local switch/L                                                                                                                                                                                                                                                                             | : Specifies that the listing is to be directed to the specified file.                                                     |
| Examples: ASM ABC ↵                                                                                                                                                                                                                                                                        | : Assembles source file ABC and generates relocatable file ABC.RB.                                                        |
| ASM/N ABC, \$CRT/E ↵                                                                                                                                                                                                                                                                       | : Assembles source file ABC and displays error statements on the screen (no relocatable file is created).                 |
| ASM ABC, XYZ/O, \$LPT/L ↵                                                                                                                                                                                                                                                                  | : Assembles source file ABC and generates relocatable file XYZ.RB and an assembly listing on the LPT.                     |
| ASM ABC, \$FD2 ; XYZ/L, \$LPT/E ↵                                                                                                                                                                                                                                                          | : Assembles source file ABC outputs the assembly listing to file XYZ.ASC in FD2 and outputs error statements on the LPT.  |
| <b>ASSIGN devicename, address</b>                                                                                                                                                                                                                                                          |                                                                                                                           |
| Sets the address of a user device drive routine.                                                                                                                                                                                                                                           |                                                                                                                           |
| Examples: ASSIGN \$USR1, \$B000 ↵                                                                                                                                                                                                                                                          | : Sets the drive routine address of user device \$USR1 to B000 (hexadecimal).                                             |
| <b>BASIC filename</b>                                                                                                                                                                                                                                                                      |                                                                                                                           |
| Invokes the BASIC compiler to compile the source program identified by filename.                                                                                                                                                                                                           |                                                                                                                           |
| Example: BASIC XYZ ↵                                                                                                                                                                                                                                                                       | : Invokes the BASIC compiler, compiles source file XYZ.ASC and generates relocatable file XYZ.RB.                         |
| <b>CONVERT</b>                                                                                                                                                                                                                                                                             |                                                                                                                           |
| Converts a file generated with the SB-5000 series BASIC interpreter or the D-BASIC SB-6000 series into a file which can be used under FDOS, or converts a file generated with FDOS into a file which can be used under the SB-5000 series BASIC interpreter or the D-BASIC SB-6000 series. |                                                                                                                           |
| Example: CONVERT ↵                                                                                                                                                                                                                                                                         |                                                                                                                           |
| <b>COPY</b>                                                                                                                                                                                                                                                                                |                                                                                                                           |
| Copies the files on the diskette in drive 1 to the diskette in drive 2. The system matches the passwords in these diskettes before carrying out a copy operation.                                                                                                                          |                                                                                                                           |
| Example: COPY ↵                                                                                                                                                                                                                                                                            |                                                                                                                           |
| <b>DEBUG filename [, ..., filenameN]</b>                                                                                                                                                                                                                                                   |                                                                                                                           |
| Invokes the symbolic debugger and links and loads relocatable file(s).                                                                                                                                                                                                                     |                                                                                                                           |
| Global switch /T                                                                                                                                                                                                                                                                           | : Specifies that the symbol table information is to be output.                                                            |
| Global switch /P                                                                                                                                                                                                                                                                           | : Specifies that the listing is to be directed to the LPT (the listing is displayed on the screen if omitted).            |
| Local switch /O                                                                                                                                                                                                                                                                            | : Specifies that the object file is to be generated with the specified file name.                                         |
| Example: DEBUG ABC, DEF ↵                                                                                                                                                                                                                                                                  | : Invokes the symbolic debugger, links and loads relocatable files ABC and DEF and waits for a symbolic debugger command. |
| <b>EDIT [filename]</b>                                                                                                                                                                                                                                                                     |                                                                                                                           |
| Loads the text editor and reads in the file (if specified). The file must be an ASC mode file.                                                                                                                                                                                             |                                                                                                                           |
| Examples: EDIT ↵                                                                                                                                                                                                                                                                           | : Loads the text editor and waits for an editor command.                                                                  |
| EDIT \$FD2 ; ABC ↵                                                                                                                                                                                                                                                                         | : Loads the text editor and reads in file ABC from FD2.                                                                   |

|                                                                                                                                                                                                                                                                  |                                                                                                                                                                                                                                          |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>FORMAT [\$FDn]</b>                                                                                                                                                                                                                                            |                                                                                                                                                                                                                                          |
| Initializes the diskette in \$FDn in the system format. The password set by the SIGN command is checked before execution.                                                                                                                                        |                                                                                                                                                                                                                                          |
| Examples:                                                                                                                                                                                                                                                        | FORMAT ↵ : Initializes the currently logged-on diskette.<br>FORMAT \$FD2 ↵ : Initializes the diskette in FD2.                                                                                                                            |
| <b>HCOPY n</b>                                                                                                                                                                                                                                                   |                                                                                                                                                                                                                                          |
| Copies a data frame from the CRT screen to the LPT.                                                                                                                                                                                                              |                                                                                                                                                                                                                                          |
| Examples:                                                                                                                                                                                                                                                        | HCOPY 4 ↵ : Copies a data frame from the CRT where the contents of graphic areas 1 and 2 are displayed simultaneously.                                                                                                                   |
| <b>LIBRARY filename1 [, ..., filenameN]</b>                                                                                                                                                                                                                      |                                                                                                                                                                                                                                          |
| Links specified file(s) into a library file.                                                                                                                                                                                                                     |                                                                                                                                                                                                                                          |
| Global switch (none) : Specifies that the link information is to be displayed on the screen.                                                                                                                                                                     |                                                                                                                                                                                                                                          |
| Global switch /P : Specifies that the link information is to be printed on the LPT.                                                                                                                                                                              |                                                                                                                                                                                                                                          |
| Examples:                                                                                                                                                                                                                                                        | LIBRARY ABC, DEF, ↵ : Links relocatable files ABC and DEF and stores their contents into library file ABC.LIB<br>LIBRARY ABC, DEF, XYZ /O ↵ : Links relocatable files ABC and DEF and stores their contents into library file XYZ.LIB.   |
| <b>LIMIT address</b>                                                                                                                                                                                                                                             |                                                                                                                                                                                                                                          |
| Sets or changes the end address of the memory area managed by FDOS.                                                                                                                                                                                              |                                                                                                                                                                                                                                          |
| Examples:                                                                                                                                                                                                                                                        | LIMIT \$B000 ↵ : Sets the FDOS area to B000 (hexadecimal).<br>LIMIT MAX ↵ : Sets the FDOS area to the maximum available address.                                                                                                         |
| <b>LINK filename1 [, ..., filenameN]</b>                                                                                                                                                                                                                         |                                                                                                                                                                                                                                          |
| Links relocatable files identified by filename1 through filenameN and outputs an object file with a link table listing.                                                                                                                                          |                                                                                                                                                                                                                                          |
| Global switch /T : Specifies that the symbol table information is to be listed.                                                                                                                                                                                  |                                                                                                                                                                                                                                          |
| Global switch /P : Specifies that the listing is to be directed to the LPT (the listing is displayed on the screen if the switch is omitted).                                                                                                                    |                                                                                                                                                                                                                                          |
| Global switch /S : Specifies that a system file is to be generated.                                                                                                                                                                                              |                                                                                                                                                                                                                                          |
| Examples:                                                                                                                                                                                                                                                        | LINK ABC, DEF ↵ : Links relocatable files ABC and DEF and outputs object file ABC.OBJ<br>LINK /T /P ABC, DEF, XYZ /O ↵ : Links relocatable files ABC and DEF and outputs object file XYZ.OBJ with the link table information on the LPT. |
| <b>LOAD filename</b>                                                                                                                                                                                                                                             |                                                                                                                                                                                                                                          |
| Loads the object file identified by filename into the area immediately following the area established by the LIMIT command.                                                                                                                                      |                                                                                                                                                                                                                                          |
| Example:                                                                                                                                                                                                                                                         | LOAD ABC.OBJ ↵ : Loads object file ABC.OBJ into memory.                                                                                                                                                                                  |
| <b>MLINK filename1 [, ..., filenameN]</b>                                                                                                                                                                                                                        |                                                                                                                                                                                                                                          |
| Links relocatable files identified by filename1 through filenameN and outputs an object file with a link table listing. This command can link files to generate an object file of up to 46K bytes, although the LINK command can only deal with up to 36K bytes. |                                                                                                                                                                                                                                          |
| Global switch /T : Specifies that the symbol table information is to be listed.                                                                                                                                                                                  |                                                                                                                                                                                                                                          |
| Global switch /P : Specifies that the listing is to be output on the LPT (the listing is displayed on the screen if this switch is omitted).                                                                                                                     |                                                                                                                                                                                                                                          |
| Example:                                                                                                                                                                                                                                                         | MLINK ABC, DEF ↵ : Links relocatable files ABC and DEF and outputs object file ABC.OBJ.                                                                                                                                                  |

|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>PAGE [output-device] or nn</b>                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <p>Performs a form feed operation on the output device identified by output-device, or sets the number of lines per page on the LPT.</p> <p>Examples:   PAGE ↵ : Moves the print position to the home position of the printer form.<br/>                     PAGE 22 ↵ : Sets the number of lines per page on the LPT to 22. The print form is fed to the top of the next page when a page feed code is issued or the TOP OF FORM button is pressed.</p> |
| <b>PROM</b>                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <p>Generates formatted code on the paper tape punch from an object file. Applicable PROM writers are those which are supplied by Britronics, Intel, Takeda and Minato Electronics.</p> <p>Example:     PROM ↵</p>                                                                                                                                                                                                                                        |
| <b>SIGN [\$FDn]</b>                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <p>Changes the password of the diskette in \$FDn.</p> <p>During a diskette copy or formatting operation, the system checks the programmer-specified password with that stored in the diskette directory for a match and carries out the specified operation only when a match occurs.</p> <p>Example:     SIGN ↵ : Changes the password of the diskette currently logged on.</p>                                                                         |
| <b>STATUS devicename, status</b>                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <p>Sets the status of the I/O device identified by devicename to status.</p> <p>Example:     STATUS \$SIA, \$1234 ↵ : Sets the control status of serial input port A to 1234 (hexadecimal).</p>                                                                                                                                                                                                                                                          |
| <b>VERIFY filename1, filename2 [, ..., filenameN-1, filenameN]                      (? , * only for filename1, ..., filenameN-1)]</b>                                                                                                                                                                                                                                                                                                                    |
| <p>Compares the contents of files filename1 through filenameN.</p> <p>Global switch /P : Specifies that the results of the comparison are to be listed on the LPT.</p> <p>Example:     VERIFY \$CMT, \$FD2 ; ABC ↵ : Compares the first file on the cassette tape with source file ABC in FD2.</p>                                                                                                                                                       |

## —System Error Messages—

There are four system error message formats.

— ERR: error message

Pertains mainly to coding errors. Issued when invalid commands are detected.

— ERR filename (device name) : error message

Indicates errors pertaining to file or device specifications.

— ERR logical number: error message

Indicates errors pertaining to logical number specifications.

— ERR logical number file name (device name): error message

Indicates errors pertaining to logical number specifications and file (or device) specifications.

The system error messages are listed below. The error numbers are not output.

|        |                      |                                                                              |
|--------|----------------------|------------------------------------------------------------------------------|
| ERR- 1 | syntax               |                                                                              |
| 2      | il command           |                                                                              |
| 3      | il argument          |                                                                              |
| 4      | il global switch     |                                                                              |
| 5      | il data              |                                                                              |
| 6      | il attribute         | ; Illegal file attribute found                                               |
| 7      | different file mode  |                                                                              |
| 8      | il local switch      |                                                                              |
| 9      | il device switch     |                                                                              |
| 10     |                      |                                                                              |
| 11     | no usable device     | ; Device unavailable                                                         |
| 12     | double device        |                                                                              |
| 13     | directory in use     |                                                                              |
| 14     |                      |                                                                              |
| 15     |                      |                                                                              |
| 16     | not enough arguments |                                                                              |
| 17     | too many argument    |                                                                              |
| 18     |                      |                                                                              |
| 19     |                      |                                                                              |
| 20     | no memory space      |                                                                              |
| 21     | memory protection    |                                                                              |
| 22     | END ?                |                                                                              |
| 37     | Break                |                                                                              |
| 38     | system id            | ; Diskette not conforming to FDOS format.                                    |
| 39     | System error         | ; System malfunction, user program error, diskette replaced improperly, etc. |

|    |                      |                                                       |
|----|----------------------|-------------------------------------------------------|
| 50 | not found            |                                                       |
| 51 | too long file        | ; File size exceeds 65535 bytes                       |
| 52 | already exist        |                                                       |
| 53 | already opened       | ; The file has been already opened or                 |
| 54 | not opened           | the logical number is already used.                   |
| 55 | read protected       |                                                       |
| 56 | write protected      |                                                       |
| 57 | permanent            |                                                       |
| 58 | end of file          |                                                       |
| 59 | no byte file         |                                                       |
| 60 | not ready            |                                                       |
| 61 | too many files       | ; Number of files exceeds 96                          |
| 62 | disk volume          | ; Diskette replaced improperly                        |
| 63 | no file space        |                                                       |
| 64 | unformat             | ; Diskette unformatted                                |
| 65 | FD hard error        | ; Hardware related disk error                         |
| 66 | il data              |                                                       |
| 67 | no usable diskette   |                                                       |
| 68 | (sub)master diskette |                                                       |
| 69 | mismatch sign        |                                                       |
| 70 | il file name         | ; Invalid file name                                   |
| 71 | il file attribute    | ; Invalid file attribute                              |
| 72 | il file type         | ; Invalid file type                                   |
| 73 | il file mode         | ; Invalid file mode                                   |
| 74 | il lu#               | ; Invalid logical number                              |
| 75 | not ready            | } ; Printer error                                     |
| 76 | alarm                |                                                       |
| 77 | paper empty          | } ; Paper tape reader or punch error                  |
| 78 | time out             |                                                       |
| 79 | parity               |                                                       |
| 80 | check sum            | } ; Serial I/O errors (to be implemented later)       |
| 81 | flaming              |                                                       |
| 82 | over run             |                                                       |
| 83 | interconnect         |                                                       |
| 84 | full buffer          | } ; IEEE-488 related errors (to be implemented later) |
| 85 | uncontrollable       |                                                       |
| 86 | interface            |                                                       |
| 87 | less data            | } ; Attempt made to open too many files               |
| 88 | much data            |                                                       |
| 89 | lu table overflow    |                                                       |
| 90 | source ?             |                                                       |
| 91 | destination ?        |                                                       |
| 92 | can't xopen          |                                                       |
| 93 | too long line        | ; Line exceeding 128 bytes                            |
| 94 | end of record        |                                                       |
| 95 | diff record length   |                                                       |

## —Editor Commands—

| Command type            | Command name         | Function                                                                                                                                                                                                                                                |
|-------------------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Input command           | <b>R</b>             | Clears the edit buffer and loads it with the input file indicated by the filename. The CP is positioned at the beginning of the edit buffer after execution of this command.                                                                            |
|                         | <b>A</b>             | Appends the input file indicated by the filename to the contents of the edit buffer. The CP position is not changed.                                                                                                                                    |
| Output command          | <b>W</b>             | Writes the edit buffer contents to the output file specified by the filename in ASCII code.                                                                                                                                                             |
| Page processing command | <b>PR</b>            | Same as the R command, except that the maximum amount of data read is 1 page.                                                                                                                                                                           |
|                         | <b>PA</b>            | Same as the A command, except that the maximum amount of data read is the unused area of the edit buffer.                                                                                                                                               |
|                         | <b>PW</b>            | Same as the W command, except that the maximum amount of data output is 1 page.                                                                                                                                                                         |
|                         | <b>PC</b>            | Terminates execution of the processing command. This command is required whenever a PR, PA or PW command is executed.                                                                                                                                   |
|                         | <b>PK</b>            | Kills the file opened by a page processing command.                                                                                                                                                                                                     |
| Type command            | <b>T</b>             | Displays the entire contents of the edit buffer. The CP position is not changed.                                                                                                                                                                        |
|                         | <b>nT</b>            | Displays n lines starting at the CP position.                                                                                                                                                                                                           |
| CP positioning command  | <b>B</b>             | Positions the CP at the beginning of the edit buffer.                                                                                                                                                                                                   |
|                         | <b>nJ</b>            | Positions the CP at the beginning of the line indicated by n.                                                                                                                                                                                           |
|                         | <b>nL</b>            | Moves the CP to the beginning of the line n lines after the current CP position.                                                                                                                                                                        |
|                         | <b>L</b>             | Moves the CP to the beginning of the current line. This is the same as when n = 0 in the nL command.                                                                                                                                                    |
|                         | <b>nM</b>            | Changes the CP position by n characters.                                                                                                                                                                                                                |
|                         | <b>M</b>             | Does not move the CP. This is the same as when n = 0 in the nM command.                                                                                                                                                                                 |
|                         | <b>Z</b>             | Moves the CP to the end of the text in the edit buffer.                                                                                                                                                                                                 |
| Correction command      | <b>C</b>             | Searches for the specified character string and replaces it with another character string; the search starts at the current CP position and proceeds to the end of the edit buffer. The CP is repositioned to the end of the character string replaced. |
|                         | <b>Q</b>             | Repeats the C command each time the specified character string is found until the end of the edit buffer is reached. The CP is repositioned to the end of the character string last replaced.                                                           |
|                         | <b>I</b>             | Inserts the specified character string at the position of the CP. The CP is repositioned to the end of the character string inserted. Line numbers are updated when a line is inserted with this command.                                               |
|                         | <b>nK</b>            | Deletes the n lines following the CP. The CP position is not changed.                                                                                                                                                                                   |
|                         | <b>K</b>             | Deletes all characters preceding the CP position until a <b>[CR]</b> code is detected. The <b>[CR]</b> code is not deleted.                                                                                                                             |
|                         | <b>nD</b>            | Deletes the n characters following the CP.                                                                                                                                                                                                              |
|                         | <b>D</b>             | No operation                                                                                                                                                                                                                                            |
| Search command          | <b>S</b>             | Searches for the specified character string, starting at the CP position and proceeding to the end of the buffer. The CP is repositioned to the end of the character string when it is found.                                                           |
| Special command         | <b>\</b>             | Executes the specified built-in command.                                                                                                                                                                                                                |
|                         | <b>=</b>             | Displays the number of characters stored in the edit buffer (including spaces and CRs).                                                                                                                                                                 |
|                         | <b>.</b>             | Displays the number of the line at which the CP is located.                                                                                                                                                                                             |
|                         | <b>&amp;</b>         | Deletes the entire contents of the edit buffer.                                                                                                                                                                                                         |
|                         | <b>#</b><br><b>!</b> | Changes the list mode for listing to the printer.<br>Transfers control to the FDOS.                                                                                                                                                                     |

Most of the above commands are compatible with those used in the NOVA editor program manufactured by the Data General Corporation.

## —Editor Error Messages—

| Error message                                         | Explanation                                                                                                                                                                                                                                                                                                                                                                                                                          | Related commands       |
|-------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|
| <b>Full buffer</b>                                    | The edit buffer is full.                                                                                                                                                                                                                                                                                                                                                                                                             | R, A, PR, PA           |
| <b>???</b>                                            | $n < 0$ in the nT or nJ command.                                                                                                                                                                                                                                                                                                                                                                                                     | T, J                   |
| <b>Large</b>                                          | n greater than 65535 was specified.                                                                                                                                                                                                                                                                                                                                                                                                  | T, J, L, M, K, D, B, Z |
| <b>Not found</b>                                      | The string specified in the command was not found.                                                                                                                                                                                                                                                                                                                                                                                   | S, C, Q                |
| <b>Invalid</b>                                        | Other than an editor command was entered or an incorrect format was used.<br>Ex) * H <span style="border: 1px solid black; padding: 0 2px;">CR</span> : There is no H command.<br>* S <span style="border: 1px solid black; padding: 0 2px;">CR</span> : A string should be specified.                                                                                                                                               | Any case               |
| <b>Page opened ?</b>                                  | The file to be subjected to page processing is not defined (or is not opened).                                                                                                                                                                                                                                                                                                                                                       | PR, PA, PW, PC         |
| <b>Page opened !</b>                                  | An attempt to execute the ! or \command was made on a file which was subjected to page processing, but which was not closed or killed by the PC or PK command.                                                                                                                                                                                                                                                                       | !, \                   |
| <b>No file is saved after edition<br/>End of job?</b> | These messages are displayed where an attempt is made to execute a ! command after the edit buffer contents have been corrected without first executing a W or PW command. Pressing the <span style="border: 1px solid black; padding: 0 2px;">Y</span> key in this case executes the ! command. Pressing the <span style="border: 1px solid black; padding: 0 2px;">N</span> key causes the system to await entry of a new command. | !                      |

Note: Refer to the System Error Messages in the System Command manual for the system errors.

Display of the message "Already opened" during execution of a W command indicates that there is a PW command which has not been closed.

## —Assembler Messages—

| Definition status message | Meaning                                                                                                                                                                   | Example                                                                                                                                                                                                                                |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>E (External)</b>       | Indicates that a label symbol is being referenced externally; that is, the label is not defined in the current source program unit.                                       | <pre> E   LD   B, CONST0 ↑   —The data byte "CONST0" is undefined. E   CALL SORT ↑   —The address "SORT" is undefined. EE  BIT   TOP, (IY+FLAG) ↑↑   —The data byte "FLAG" is undefined.     —The data byte "TOP" is undefined. </pre> |
| <b>P (Phase)</b>          | Defines a label symbol with a constant assigned.<br>This message is also output when a label symbol is encountered during pass 2 which was not encountered during pass 1. | <pre> P   LETNL : EQU 0762H P   DATA1 : EQU 3 ↑   —LETNL and DATA1 are defined by EQU. </pre> <p>The P message is displayed in the relocatable binary code column rather than in the assembler message column.</p>                     |

| Error message                      | Meaning                                                                 | Example                                                                                                                                                                                                                                   |
|------------------------------------|-------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>C (illegal Character error)</b> | Indicates that an illegal character is used in the operand.             | <pre> C   JP   +1000-3 </pre>                                                                                                                                                                                                             |
| <b>F (Format error)</b>            | Indicates that the instruction format is incorrect.                     |                                                                                                                                                                                                                                           |
| <b>N (Non label error)</b>         | Indicates that no label symbol is specified for ENT or EQU.             | <pre> N           EQU 0012H ↑   —No label symbol </pre>                                                                                                                                                                                   |
| <b>L (erroneous Label error)</b>   | Indicates that an illegal label symbol is used.                         | <pre> L   JR   XYZ ↑   —XYZ is not defined in the current program. </pre> <p>No externally defined global symbol can be used as the operand of a JR or DJNZ command. If such a label symbol is specified, the L message is displayed.</p> |
| <b>M (Multiple label error)</b>    | Indicates that a label symbol is defined two or more times.             | <pre> M   ABC : LD DE, BUFFER       ? M   ABC : ENT ↑   —ABC is defined twice. </pre>                                                                                                                                                     |
| <b>O (erroneous Operand)</b>       | Indicates that an illegal operand is specified.                         |                                                                                                                                                                                                                                           |
| <b>Q (Questionable mnemonic)</b>   | Indicates that the mnemonic code is incorrect.                          | <pre> Q   CAL   XYZ     CALL XYZ is correct. </pre>                                                                                                                                                                                       |
| <b>S (String error)</b>            | Indicates that single or double quotation mark(s) are omitted.          | <pre> S   DEFM  GAME OVER     DEFM 'GAME OVER' is correct. </pre>                                                                                                                                                                         |
| <b>V (Value over)</b>              | Indicates that the value of the operand is out of the prescribed range. | <pre> V   LD A, FF8H   V   SET 8, A V   JR  -130 </pre>                                                                                                                                                                                   |
| <b>END?</b>                        | Indicates that the END directive is missing from the source program.    |                                                                                                                                                                                                                                           |

Note: Refer to the System Error Messages in the System Command manual for other system errors.

## —Symbolic Debugger Commands—

| Command type         | Command name         | Function                                                                                                                                                  |
|----------------------|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Symbol table command | <b>T</b>             | Displays the contents of the symbol table; i.e., the label symbol name, its absolute address and the definition status for each table entry. (Table Dump) |
| Debugging commands   | <b>B<sup>†</sup></b> | Displays, sets or alters a breakpoint. (Breakpoint)                                                                                                       |
|                      | <b>&amp;</b>         | Clears all breakpoints set. (Clear Breakpoints)                                                                                                           |
|                      | <b>M<sup>†</sup></b> | Displays the contents of the specified block in the link area in hexadecimal representation or alters them. (Memory Dump)                                 |
|                      | <b>D<sup>†</sup></b> | Displays the contents of the specified block in the link area in hexadecimal representation with one instruction on a line. (Memory List Dump)            |
|                      | <b>W<sup>†</sup></b> | Writes hexadecimal data, starting at the specified address in the link area. (Write)                                                                      |
|                      | <b>G<sup>†</sup></b> | Executes the program at the specified address. (GOTO)                                                                                                     |
|                      | <b>I</b>             | Executes the program at the address designated by PC with the register buffer data set to the CPU internal registers. (Indicative Start)                  |
|                      | <b>A</b>             | Displays the contents of registers A, F, B, C, D, E, H and L in hexadecimal representation or alters them. (Accumulator)                                  |
|                      | <b>C</b>             | Displays the contents of complementary registers A', F', B', C', D', E', H' and L' in hexadecimal representation or alters them. (Complementary)          |
|                      | <b>P</b>             | Displays the contents of registers PC, SP, IX, IY and I in hexadecimal representation or alters them. (Program Counter)                                   |
|                      | <b>R</b>             | Displays the contents of all registers in hexadecimal representation. (Register)                                                                          |
|                      | <b>X</b>             | Transfers the specified memory block to the specified address. (Transfer)                                                                                 |
| File I/O commands    | <b>S</b>             | Saves the object program in the link area in an output file with the specified name. (Save)                                                               |
|                      | <b>Y</b>             | Reads the object program from the object file with the specified file name into memory. (Yank)                                                            |
| Special commands     | <b>\</b>             | Executes the specified FDOS built-in command.                                                                                                             |
|                      | <b>#</b>             | Switches the printer list mode for listing printout.                                                                                                      |
|                      | <b>!</b>             | Transfers control to FDOS.                                                                                                                                |

Note: Commands marked by a dagger permit symbolic operations.

## —Symbolic Debugger Error Messages—

| Error message | Description                                                                                                                                                                                                                                                                                                                                       | Related commands                   |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------|
| ???           | <ul style="list-style-type: none"> <li>○ The command operand fields does not match the 4-digit hexadecimal format.</li> <li>○ A symbolic label is missing.</li> <li>○ A data defining symbol is used as a label.</li> </ul>                                                                                                                       | M, D, W, B, G                      |
| Error         | <ul style="list-style-type: none"> <li>○ An invalid number of digits was entered when altering register or memory contents, or a key other than 0 through 9 or A through F was pressed.</li> </ul>                                                                                                                                                | A, C, P, M                         |
| DJNZ?         | A breakpoint was set for a DJNZ instruction.                                                                                                                                                                                                                                                                                                      | B                                  |
| CALL?         | A breakpoint was set for a CALL instruction.                                                                                                                                                                                                                                                                                                      | B                                  |
| RST 6?        | A breakpoint was set for a RST 6 instruction.                                                                                                                                                                                                                                                                                                     | B                                  |
| Over          | An attempt was made to set more than 9 breakpoints.                                                                                                                                                                                                                                                                                               | B                                  |
| ?             | <ul style="list-style-type: none"> <li>○ An attempt was made to access outside the link area.</li> <li>○ The starting address is greater than the ending address.</li> <li>○ An attempt was made to clear an undefined breakpoint.</li> <li>○ The breakpoint counter was set to F (the maximum permissible value is E in hexadecimal).</li> </ul> | M, D, W, B, G, X<br>M, D<br>B<br>B |

Note: Refer to the System Error Messages in the System Command manual for other system error messages.

## —PROM Formatter Commands—

| COMMAND                        |                                                                           | OPERATION                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------------------------------|---------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| File Input/<br>Output commands | Y (Yank)<br>S (Save)<br>CY (Yank disk)                                    | Loads a program (data) from the diskette into the free area.<br>Saves the program (data) in the free area on diskette.<br>Loads data in 256-byte units from the specified sector(s) of the specified track on the diskette into RAM.                                                                                                                                                                        |
|                                | CS (Save disk)                                                            | Saves data in 256-byte units from RAM memory in the specified sector(s) of the specified track of the diskette.                                                                                                                                                                                                                                                                                             |
| Format commands                | P (Punch)<br>R (Read)                                                     | Punches the specified contents of the free area in the specified format.<br>Reads in a paper tape punched in the format specified.                                                                                                                                                                                                                                                                          |
| Other commands                 | M (Memory)<br>V (Verify)<br>\ (FDOS)<br>#<br>& (Clear)<br>?<br>! (Return) | Displays and modifies data in the free area.<br>Reads data from the paper tape reader and compares it with the contents of the RAM free area.<br>Executes the specified built-in FDOS command.<br>Switches the list mode for listing on a printer.<br>Buries all data in the free area in hexadecimal code FFH.<br>Displays the starting and ending addresses of the free area.<br>Returns control to FDOS. |

| Error message      | Error content                                        | Related command  |
|--------------------|------------------------------------------------------|------------------|
| memory protection  | An address outside of the free area was specified.   | Y, S, P, R, M, V |
| il command         | The command was not entered correctly.               |                  |
| il data            | The format specified does not match the format read. | R, V             |
| check sum          | Check sum error.                                     | R, V             |
| \$ LPT : not ready | The printer is not ready.                            | #                |
| \$ PTP : not ready | The paper tape punch is not ready.                   | P                |
| \$ PTR : not ready | The paper tape reader is not ready.                  | R, V             |

See the "System Error Messages" in System Command for other error messages.

### Caution:

Entry of characters other than S, Y, CS, CY, P, R, M, V, \, &, #, ? or ! will cause a return to the command wait state after the command table is displayed.

If a character other than A~H is input while "format?" is displayed and format entry awaited, the format table will be displayed, after which the format entry wait state will be reentered. A return can be made to the command wait state at this time by pressing **BREAK**.

## —FILE Mode—

| File mode | Meanings                                                                                                                                                                                      |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| .ASC      | ASCII file. A source file generated by the text editor or a file containing ASCII character strings generated by a BASIC interpreter.                                                         |
| .RB       | Relocatable file. A file containing pseudo-machine language code (relocatable binary code) which can be loaded into any location in memory. It is generated by the assembler or the compiler. |
| .OBJ      | Object file. A file containing Z-80 machine language codes.                                                                                                                                   |
| .LIB      | Library file. A file into which FDOS links multiple relocatable files.                                                                                                                        |
| .SYS      | System file. A file containing a system program runs under FDOS and which contains relocatable binary codes (such as the text editor and the assembler).                                      |

## —I/O Devices Handled by FDOS—

\$KB : MZ-80B system keyboard  
\$CRT : MZ-80B system display unit  
\$FD1 :  
\$FD2 : } Floppy disk drives (MZ-80FB or MZ-80FBK)  
\$FD3 : }  
\$FD4 :  
\$CMT : System cassette tape deck  
\$LPT : System printer (MZ-80P4 or MZ-80P5)  
\$MEM : A part of MZ-80 main memory  
\$PTR : Paper tape reader  
\$PTP : Paper tape punch  
\$SIA : Serial input port A  
\$SIB : Serial input port B  
\$SOA : Serial output port A  
\$SOB : Serial output port B  
\$USR1 :  
\$USR2 : } User devices 1 ~ 4  
\$USR3 : }  
\$USR4 :  
\$CMT1 : Cassette tape deck for MZ-80K

## —File Attributes—

File attributes are information pertaining to file protection. There are four types of file attribute: 0, R, W and P. File attribute 0 indicates that a file is not protected. The other attributes inhibit the use of specific commands as indicated below.

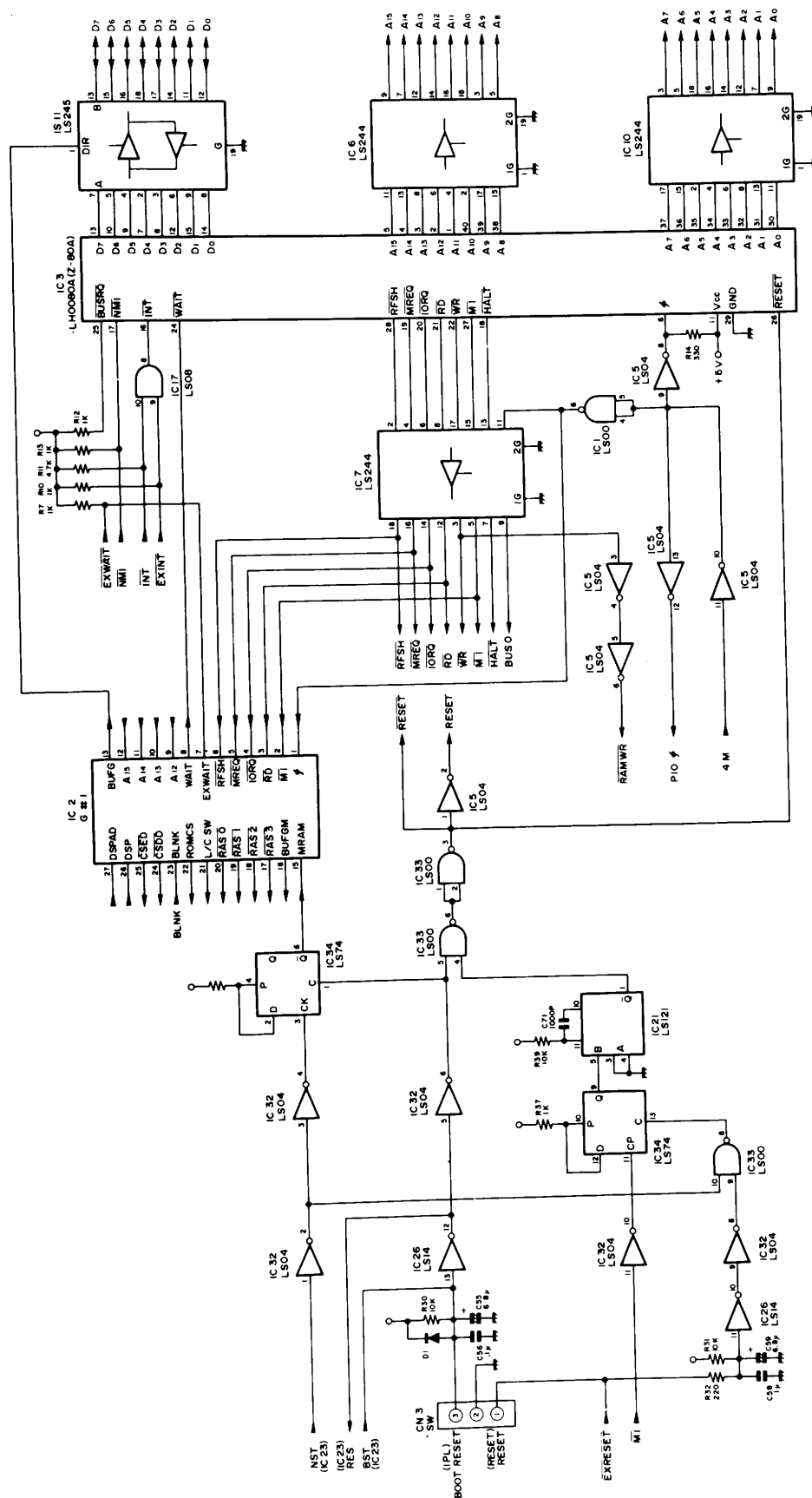
| File attribute             | R                                                             | W                | P                                                                                 |                                                                                                 |
|----------------------------|---------------------------------------------------------------|------------------|-----------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|
| Inhibited FDOS commands    | TYPE<br>XFER<br>EDIT<br>ASM<br>LINK<br>DEBUG<br>PROM<br>BASIC | DELETE<br>RENAME | TYPE<br>XFER<br>EDIT<br>ASM<br>LINK<br>DEBUG<br>PROM<br>BASIC<br>DELETE<br>RENAME | 0: No file protection<br>R: Read-inhibited file<br>W: Write-inhibited file<br>P: Permanent file |
| Inhibited BASIC statements | ROPEN#<br>INPUT#( )                                           | PRINT#( )        | ROPEN#<br>INPUT#( )<br>PRINT#( )                                                  |                                                                                                 |

# ASCII CODE TABLE

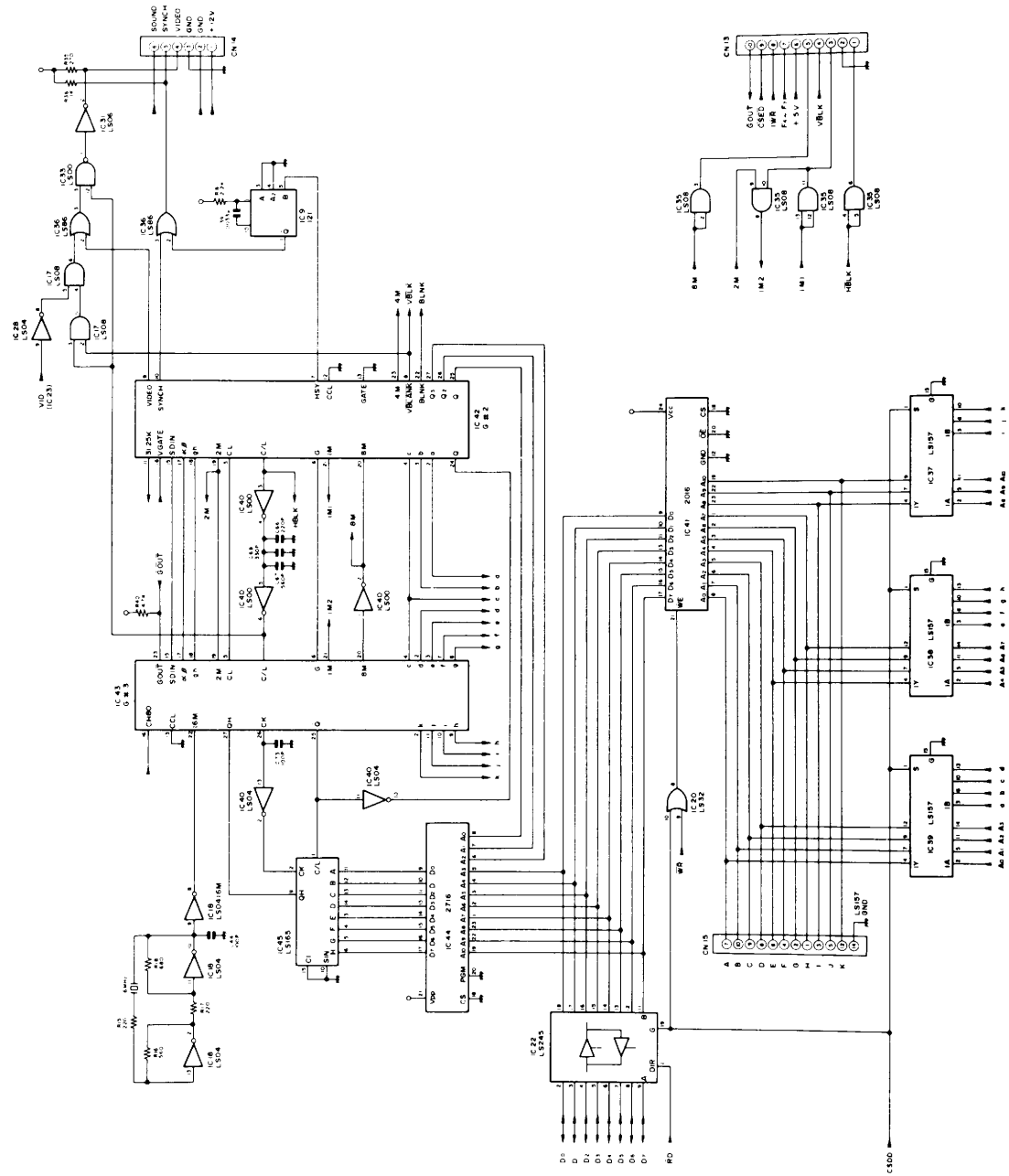
|              |   | UPPER 4 BITS |                 |    |   |   |   |   |   |   |   |    |   |   |   |   |   |
|--------------|---|--------------|-----------------|----|---|---|---|---|---|---|---|----|---|---|---|---|---|
|              |   | 0            | 1               | 2  | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A  | B | C | D | E | F |
| LOWER 4 BITS | 0 | NULL         | f <sub>1</sub>  |    | O | @ | P | ` | p |   | = |    | O | @ | P | ` | p |
|              | 1 | ↓            | f <sub>2</sub>  | !  | I | A | Q | a | q | ↓ | ¥ | !  | I | A | Q | a | q |
|              | 2 | ↑            | f <sub>3</sub>  | "  | 2 | B | R | b | r | ↑ | £ | "  | 2 | B | R | b | r |
|              | 3 | →            | f <sub>4</sub>  | #  | 3 | C | S | c | s | → | ● | #  | 3 | C | S | c | s |
|              | 4 | ←            | f <sub>5</sub>  | \$ | 4 | D | T | d | t | ← | ○ | \$ | 4 | D | T | d | t |
|              | 5 | HOME         | f <sub>6</sub>  | %  | 5 | E | U | e | u | ♠ | ☐ | %  | 5 | E | U | e | u |
|              | 6 | CLR          | f <sub>7</sub>  | &  | 6 | F | V | f | v | ♥ | ☐ | &  | 6 | F | V | f | v |
|              | 7 | DEL          | f <sub>8</sub>  | '  | 7 | G | W | g | w | ♦ | ☐ | '  | 7 | G | W | g | w |
|              | 8 | INST         | f <sub>9</sub>  | (  | 8 | H | X | h | x | ♣ | ☐ | (  | 8 | H | X | h | x |
|              | 9 | GRPH         | f <sub>10</sub> | )  | 9 | I | Y | i | y | ⚡ | ⚡ | )  | 9 | I | Y | i | y |
|              | A | SFT LOCK     | 00              | *  | : | J | Z | j | z | ≡ |   | *  | : | J | Z | j | z |
|              | B | BREAK        | TAB             | +  | ; | K | [ | k | { | ≡ | — | +  | ; | K | [ | k | { |
|              | C | RVS          |                 | ,  | < | L | \ |   |   | ≡ | ≡ | ,  | < | L | \ |   |   |
|              | D | CR           |                 | —  | = | M | ] | m | } | ≡ | ≡ | —  | = | M | ] | m | } |
|              | E | L SCRIPT     |                 | .  | > | N | ^ | n | ~ | ≡ | ≡ | .  | > | N | ^ | n | ~ |
|              | F | RVS CANCEL   | ☐               | /  | ? | O | — | o | ↵ | ≡ | ≡ | /  | ? | O | — | o | π |

ASCII Codes of characters and control codes

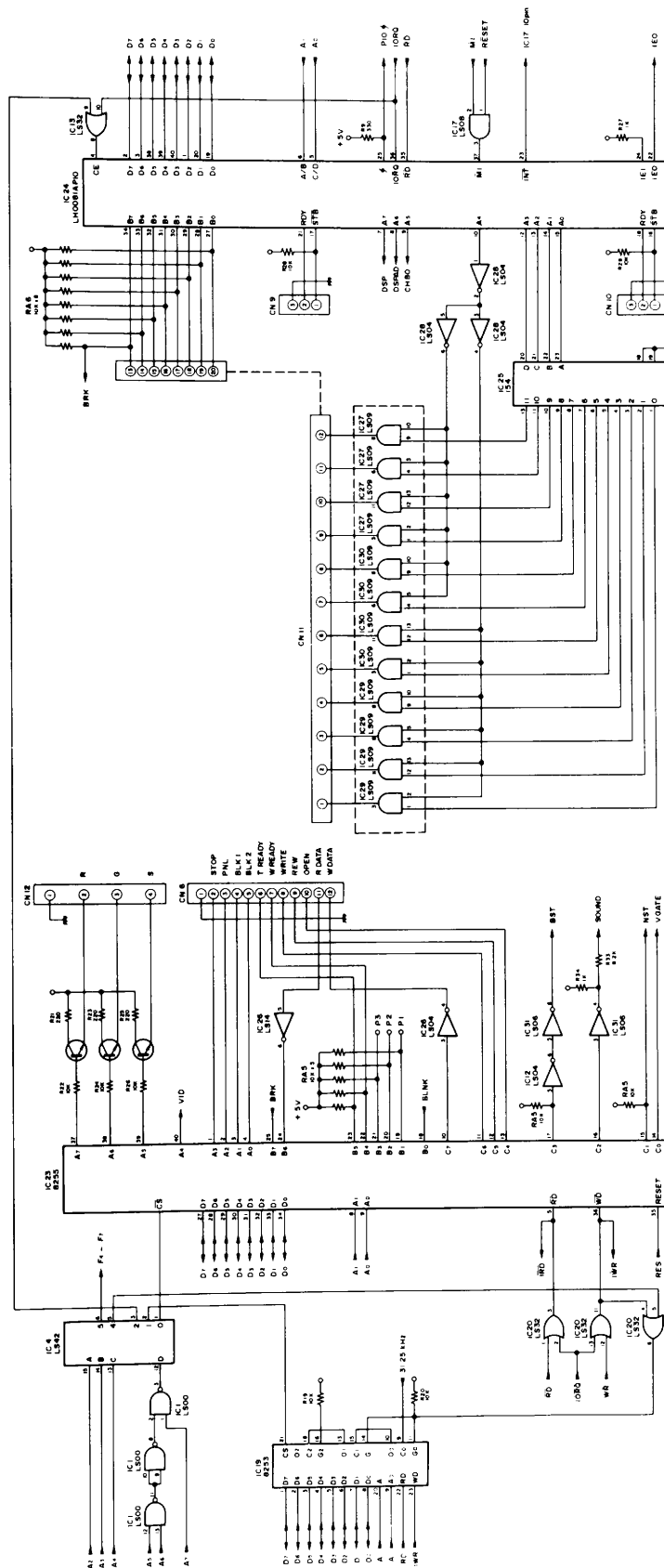
## MZ-80B CIRCUIT DIAGRAMS



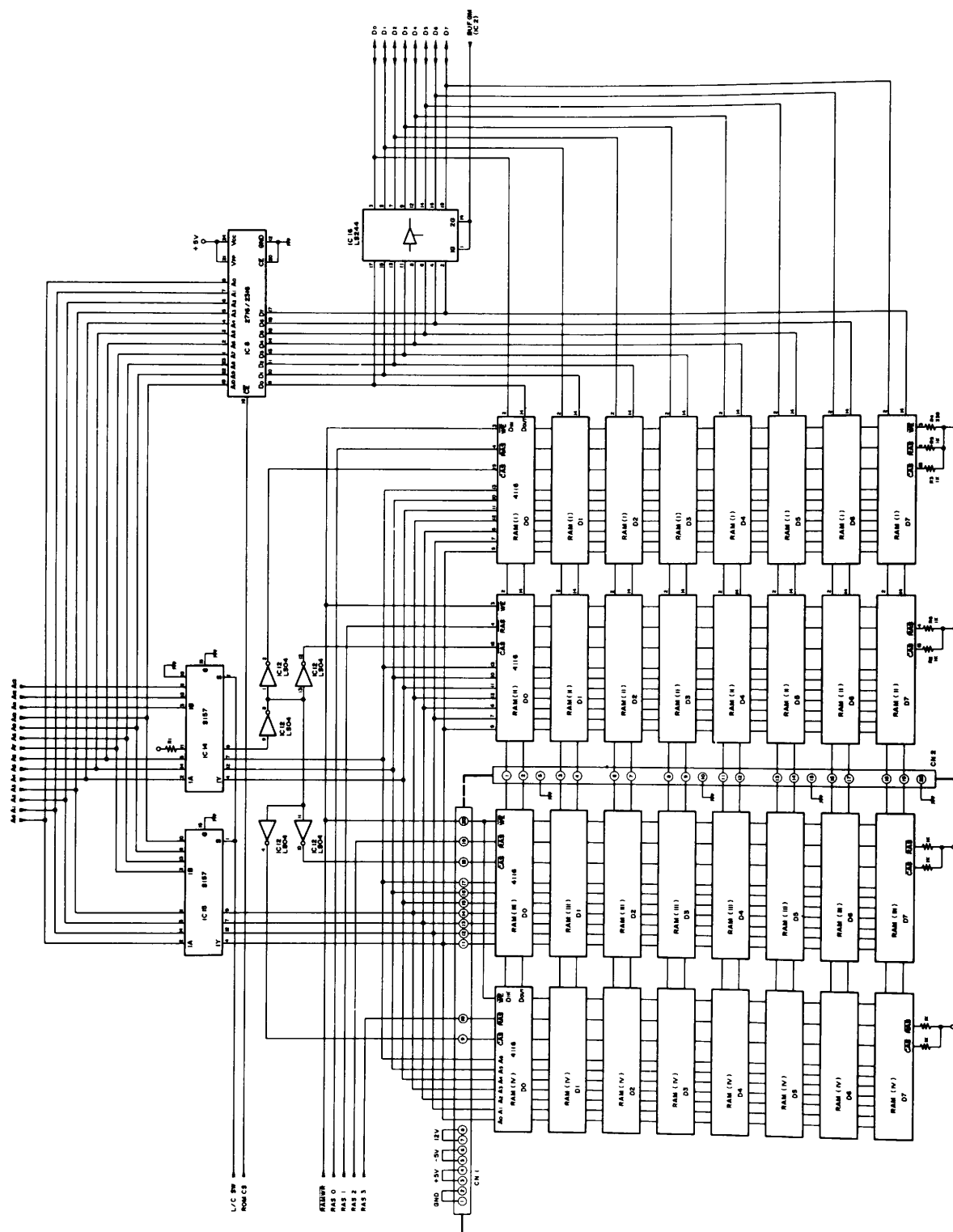
**CPU board, block 1 : CPU signal system**



CPU board, block 2



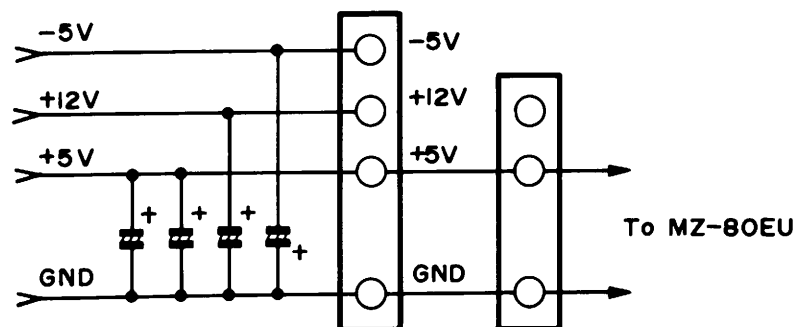
CPU board, block 3 : 8255 signal system



CPU board, block 4 : RAM signal system

CN5 → MZ-80GM

| CN 4, 5      40P |                              |    |                            |
|------------------|------------------------------|----|----------------------------|
| 1                | A15                          | 2  | A14                        |
| 3                | A13                          | 4  | A12                        |
| 5                | A11                          | 6  | A10                        |
| 7                | A9                           | 8  | A8                         |
| 9                | GND                          | 10 | A7                         |
| 11               | A6                           | 12 | A5                         |
| 13               | A4                           | 14 | A3                         |
| 15               | A2                           | 16 | A1                         |
| 17               | A0                           | 18 | GND                        |
| 19               | D7                           | 20 | D6                         |
| 21               | D5                           | 22 | D4                         |
| 23               | D3                           | 24 | D2                         |
| 25               | D1                           | 26 | D0                         |
| 27               | GND                          | 28 | $\overline{\text{NMI}}$    |
| 29               | $\overline{\text{EX WAIT}}$  | 30 | $\overline{\text{EX INT}}$ |
| 31               | $\overline{\text{EX RESET}}$ | 32 | RESET                      |
| 33               | IEO                          | 34 | $\overline{\text{HALT}}$   |
| 35               | $\overline{\text{MREQ}}$     | 36 | $\overline{\text{IOREQ}}$  |
| 37               | $\overline{\text{RD}}$       | 38 | $\overline{\text{WR}}$     |
| 39               | $\overline{\text{MT}}$       | 40 | BUS0                       |



CPU board, block 5

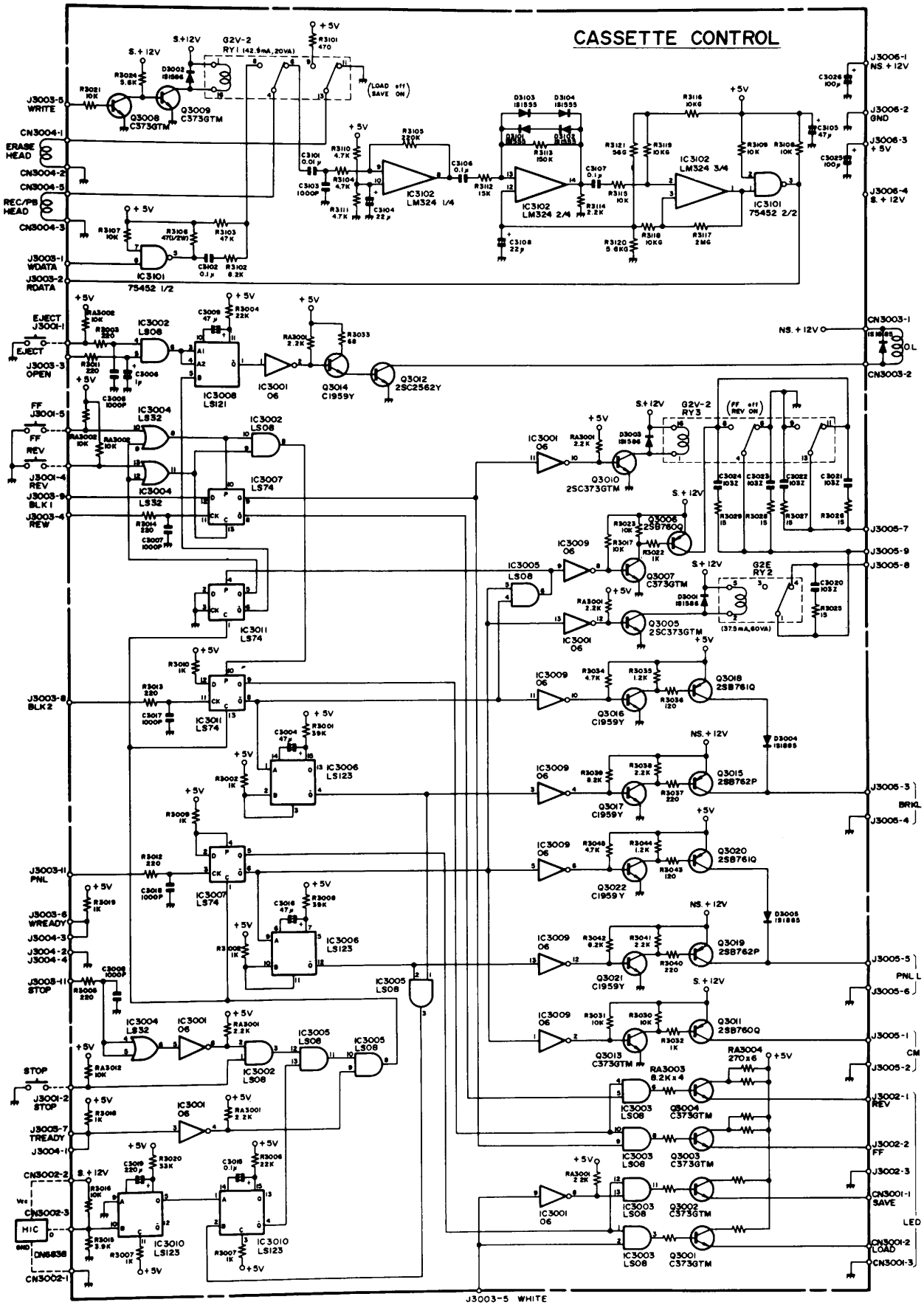
VS P0080P-16YA  
32 ohm

Q2002 2SC1514  
Q2003 2SC1213  
Q2004 2SC1213  
Q2005 2SC1213  
Q2006 2SC1213  
Q2007 2SC1213  
Q2008 2SC1213  
Q2009 2SC1213  
Q2010 2SC1213  
Q2011 2SC1213  
Q2012 2SC1213  
Q2013 2SC1213  
Q2014 2SC1213  
Q2015 2SC1213  
Q2016 2SC1213  
Q2017 2SC1213  
Q2018 2SC1213  
Q2019 2SC1213  
Q2020 2SC1213  
Q2021 2SC1213  
Q2022 2SC1213  
Q2023 2SC1213  
Q2024 2SC1213  
Q2025 2SC1213  
Q2026 2SC1213  
Q2027 2SC1213  
Q2028 2SC1213  
Q2029 2SC1213  
Q2030 2SC1213  
Q2031 2SC1213  
Q2032 2SC1213  
Q2033 2SC1213  
Q2034 2SC1213  
Q2035 2SC1213  
Q2036 2SC1213  
Q2037 2SC1213  
Q2038 2SC1213  
Q2039 2SC1213  
Q2040 2SC1213  
Q2041 2SC1213  
Q2042 2SC1213  
Q2043 2SC1213  
Q2044 2SC1213  
Q2045 2SC1213  
Q2046 2SC1213  
Q2047 2SC1213  
Q2048 2SC1213  
Q2049 2SC1213  
Q2050 2SC1213  
Q2051 2SC1213  
Q2052 2SC1213  
Q2053 2SC1213  
Q2054 2SC1213  
Q2055 2SC1213  
Q2056 2SC1213  
Q2057 2SC1213  
Q2058 2SC1213  
Q2059 2SC1213  
Q2060 2SC1213  
Q2061 2SC1213  
Q2062 2SC1213  
Q2063 2SC1213  
Q2064 2SC1213  
Q2065 2SC1213  
Q2066 2SC1213  
Q2067 2SC1213  
Q2068 2SC1213  
Q2069 2SC1213  
Q2070 2SC1213  
Q2071 2SC1213  
Q2072 2SC1213  
Q2073 2SC1213  
Q2074 2SC1213  
Q2075 2SC1213  
Q2076 2SC1213  
Q2077 2SC1213  
Q2078 2SC1213  
Q2079 2SC1213  
Q2080 2SC1213  
Q2081 2SC1213  
Q2082 2SC1213  
Q2083 2SC1213  
Q2084 2SC1213  
Q2085 2SC1213  
Q2086 2SC1213  
Q2087 2SC1213  
Q2088 2SC1213  
Q2089 2SC1213  
Q2090 2SC1213  
Q2091 2SC1213  
Q2092 2SC1213  
Q2093 2SC1213  
Q2094 2SC1213  
Q2095 2SC1213  
Q2096 2SC1213  
Q2097 2SC1213  
Q2098 2SC1213  
Q2099 2SC1213  
Q2100 2SC1213

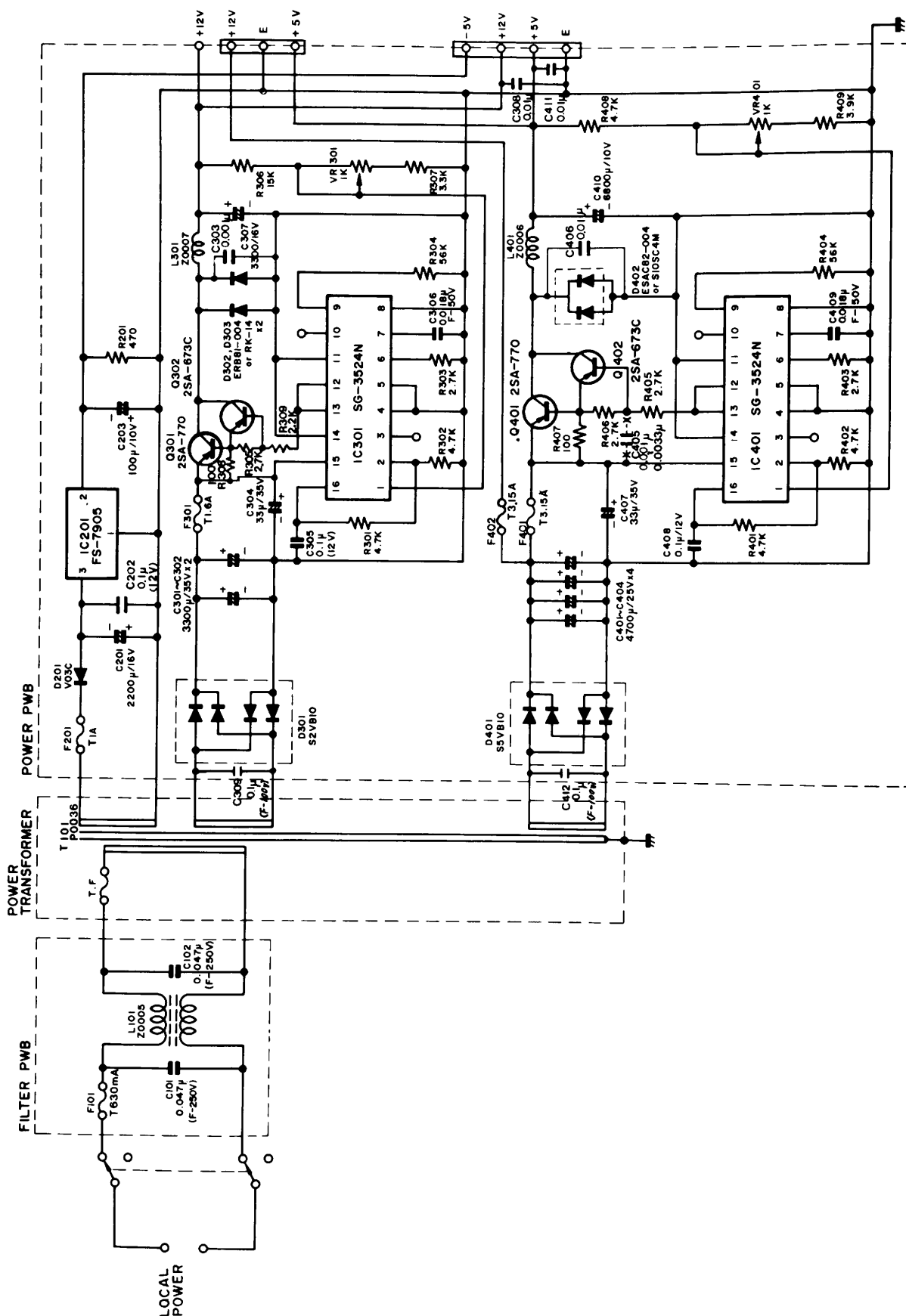
R2001 10K  
R2002 10K  
R2003 10K  
R2004 10K  
R2005 10K  
R2006 10K  
R2007 10K  
R2008 10K  
R2009 10K  
R2010 10K  
R2011 10K  
R2012 10K  
R2013 10K  
R2014 10K  
R2015 10K  
R2016 10K  
R2017 10K  
R2018 10K  
R2019 10K  
R2020 10K  
R2021 10K  
R2022 10K  
R2023 10K  
R2024 10K  
R2025 10K  
R2026 10K  
R2027 10K  
R2028 10K  
R2029 10K  
R2030 10K  
R2031 10K  
R2032 10K  
R2033 10K  
R2034 10K  
R2035 10K  
R2036 10K  
R2037 10K  
R2038 10K  
R2039 10K  
R2040 10K  
R2041 10K  
R2042 10K  
R2043 10K  
R2044 10K  
R2045 10K  
R2046 10K  
R2047 10K  
R2048 10K  
R2049 10K  
R2050 10K  
R2051 10K  
R2052 10K  
R2053 10K  
R2054 10K  
R2055 10K  
R2056 10K  
R2057 10K  
R2058 10K  
R2059 10K  
R2060 10K  
R2061 10K  
R2062 10K  
R2063 10K  
R2064 10K  
R2065 10K  
R2066 10K  
R2067 10K  
R2068 10K  
R2069 10K  
R2070 10K  
R2071 10K  
R2072 10K  
R2073 10K  
R2074 10K  
R2075 10K  
R2076 10K  
R2077 10K  
R2078 10K  
R2079 10K  
R2080 10K  
R2081 10K  
R2082 10K  
R2083 10K  
R2084 10K  
R2085 10K  
R2086 10K  
R2087 10K  
R2088 10K  
R2089 10K  
R2090 10K  
R2091 10K  
R2092 10K  
R2093 10K  
R2094 10K  
R2095 10K  
R2096 10K  
R2097 10K  
R2098 10K  
R2099 10K  
R2100 10K

C2001 100P  
C2002 100P  
C2003 100P  
C2004 100P  
C2005 100P  
C2006 100P  
C2007 100P  
C2008 100P  
C2009 100P  
C2010 100P  
C2011 100P  
C2012 100P  
C2013 100P  
C2014 100P  
C2015 100P  
C2016 100P  
C2017 100P  
C2018 100P  
C2019 100P  
C2020 100P  
C2021 100P  
C2022 100P  
C2023 100P  
C2024 100P  
C2025 100P  
C2026 100P  
C2027 100P  
C2028 100P  
C2029 100P  
C2030 100P  
C2031 100P  
C2032 100P  
C2033 100P  
C2034 100P  
C2035 100P  
C2036 100P  
C2037 100P  
C2038 100P  
C2039 100P  
C2040 100P  
C2041 100P  
C2042 100P  
C2043 100P  
C2044 100P  
C2045 100P  
C2046 100P  
C2047 100P  
C2048 100P  
C2049 100P  
C2050 100P  
C2051 100P  
C2052 100P  
C2053 100P  
C2054 100P  
C2055 100P  
C2056 100P  
C2057 100P  
C2058 100P  
C2059 100P  
C2060 100P  
C2061 100P  
C2062 100P  
C2063 100P  
C2064 100P  
C2065 100P  
C2066 100P  
C2067 100P  
C2068 100P  
C2069 100P  
C2070 100P  
C2071 100P  
C2072 100P  
C2073 100P  
C2074 100P  
C2075 100P  
C2076 100P  
C2077 100P  
C2078 100P  
C2079 100P  
C2080 100P  
C2081 100P  
C2082 100P  
C2083 100P  
C2084 100P  
C2085 100P  
C2086 100P  
C2087 100P  
C2088 100P  
C2089 100P  
C2090 100P  
C2091 100P  
C2092 100P  
C2093 100P  
C2094 100P  
C2095 100P  
C2096 100P  
C2097 100P  
C2098 100P  
C2099 100P  
C2100 100P

Q2001 2SC1213  
Q2002 2SC1213  
Q2003 2SC1213  
Q2004 2SC1213  
Q2005 2SC1213  
Q2006 2SC1213  
Q2007 2SC1213  
Q2008 2SC1213  
Q2009 2SC1213  
Q2010 2SC1213  
Q2011 2SC1213  
Q2012 2SC1213  
Q2013 2SC1213  
Q2014 2SC1213  
Q2015 2SC1213  
Q2016 2SC1213  
Q2017 2SC1213  
Q2018 2SC1213  
Q2019 2SC1213  
Q2020 2SC1213  
Q2021 2SC1213  
Q2022 2SC1213  
Q2023 2SC1213  
Q2024 2SC1213  
Q2025 2SC1213  
Q2026 2SC1213  
Q2027 2SC1213  
Q

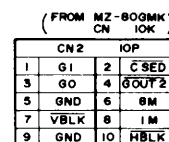


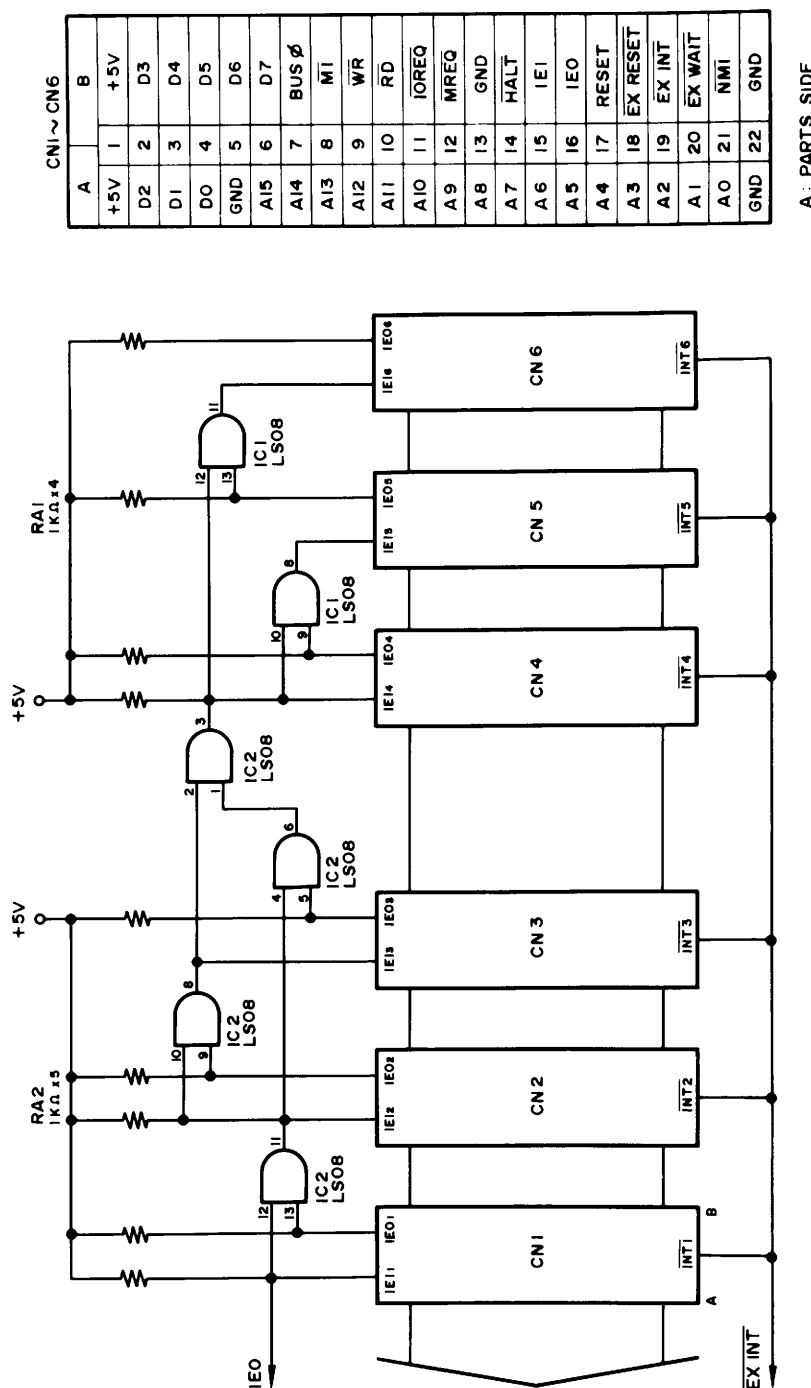
Cassette tape deck control



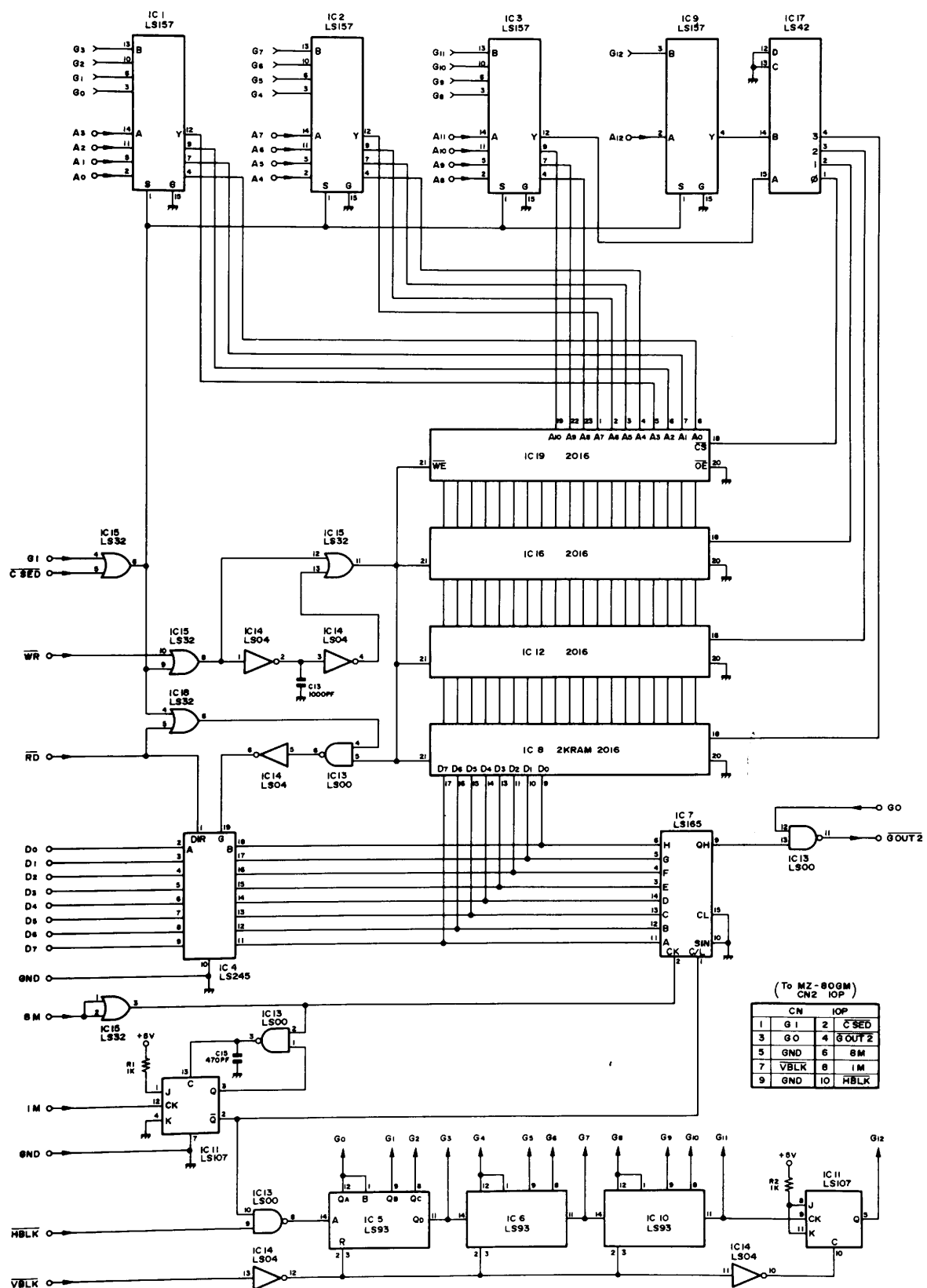
Power supply

| (To CPU BOARD)<br>CN13 IOP |      |    |                           |
|----------------------------|------|----|---------------------------|
| 41                         | GOUT | 42 | CS $\overline{\text{ED}}$ |
| 43                         | IWR  | 44 | F $_4 \sim F_7$           |
| 45                         | +5V  | 46 | 8M                        |
| 47                         | VLK  | 48 | 1M                        |
| 49                         | GND  | 50 | HBLK                      |



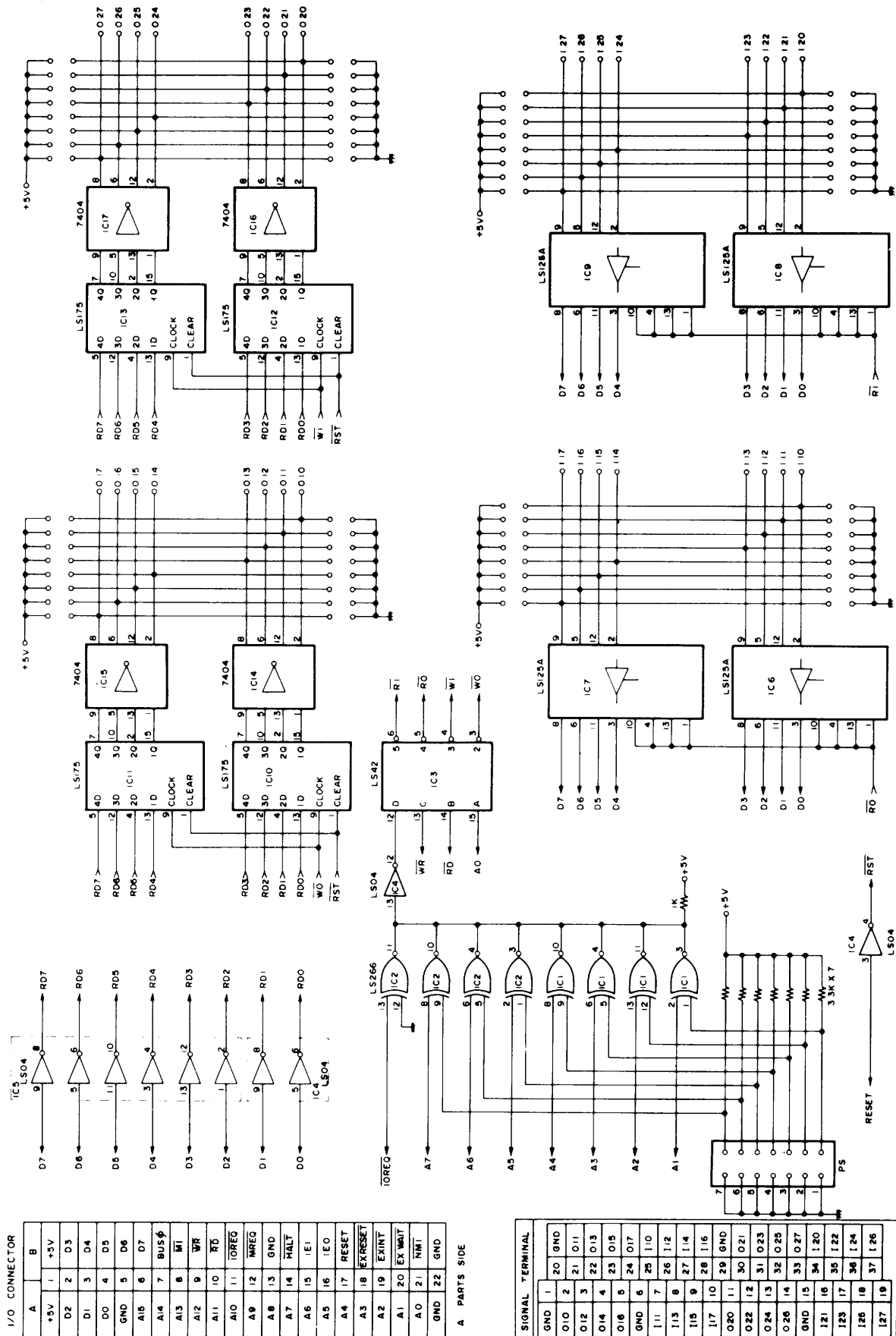


| (To CPU BOARD)<br>CN4 40P |          | CN7.8 |        | 40P |  |
|---------------------------|----------|-------|--------|-----|--|
| 1                         | A15      | 2     | A14    |     |  |
| 3                         | A13      | 4     | A12    |     |  |
| 5                         | A11      | 6     | A10    |     |  |
| 7                         | A9       | 8     | A8     |     |  |
| 9                         | GND      | 10    | A7     |     |  |
| 11                        | A6       | 12    | A5     |     |  |
| 13                        | A4       | 14    | A3     |     |  |
| 15                        | A2       | 16    | A1     |     |  |
| 17                        | A0       | 18    | GND    |     |  |
| 19                        | D7       | 20    | D6     |     |  |
| 21                        | D5       | 22    | D4     |     |  |
| 23                        | D3       | 24    | D2     |     |  |
| 25                        | D1       | 26    | D0     |     |  |
| 27                        | GND      | 28    | NMi    |     |  |
| 29                        | EX WAIT  | 30    | EX INT |     |  |
| 31                        | EX RESET | 32    | RESET  |     |  |
| 33                        | 1EO      | 34    | HALT   |     |  |
| 35                        | MREQ     | 36    | IOREQ  |     |  |
| 37                        | RD       | 38    | WR     |     |  |
| 39                        | MI       | 40    | BUSØ   |     |  |



Graphic Memory 2 card (optional) MZ-80GMK

# UNIVERSAL I/O CARD CIRCUIT DIAGRAM



UNIVERSAL I/O CARD

