Personal Computer

MZ-80B

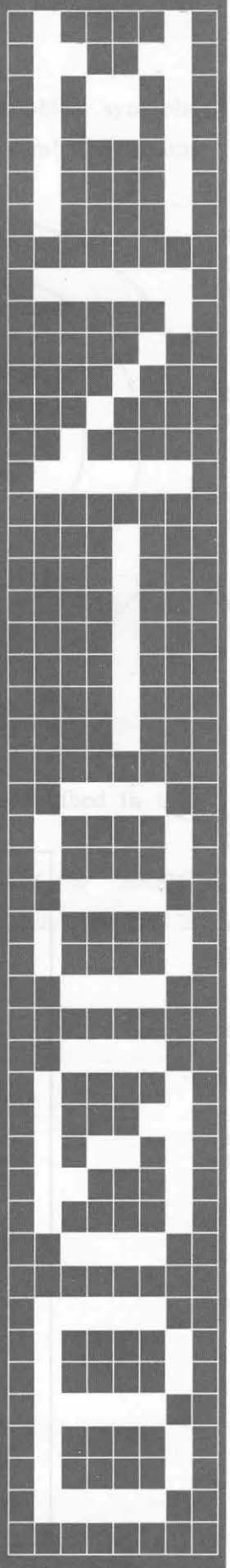
SYSTEM PROGRAM MANUAL

SHARP

Personal Computer
MZ-80B

SYSTEM PROGRAM MANUAL

MARCH 1982



080281-050382

NOTICE

This manual is based on the system programs for the MZ-80B personal computer. It describes text editor SB-2102, assembler SB-2202, linker SB-2301, symbolic debugger SB-2401, PROM formatter SB-2501 and K-B converter SB-2601.

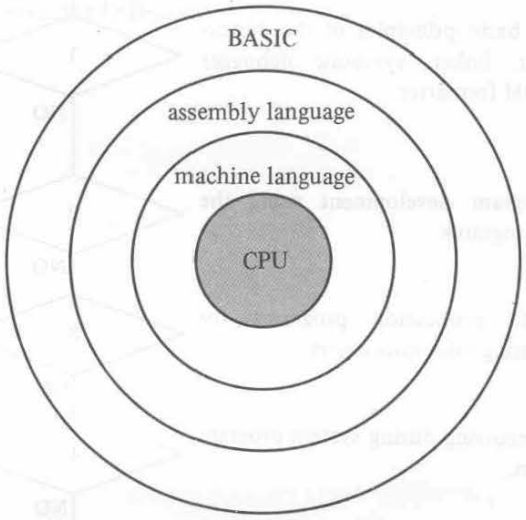
The MZ-80B general-purpose personal computer is supported by system software which is filed in software packs (cassette tapes or diskettes). All system software is subject to revision without prior notice, therefore, you are requested to pay special attention to file version numbers.

This manual has been carefully prepared and checked for compelteness, accuracy and clarity. However, in the event that you should encounter any errors or ambiguities, please feel free to contact your local SHARP representative for clarification.

PREFACE

This manual describes the system programs, i.e., the editor-assembler, symbolic debugger, linker and PROM formatter, which assist in preparation of assembler programs for the MZ-80B personal computer.

Computer programming languages such as BASIC, assembly language, and machine language are classified hierarchically as shown at right.

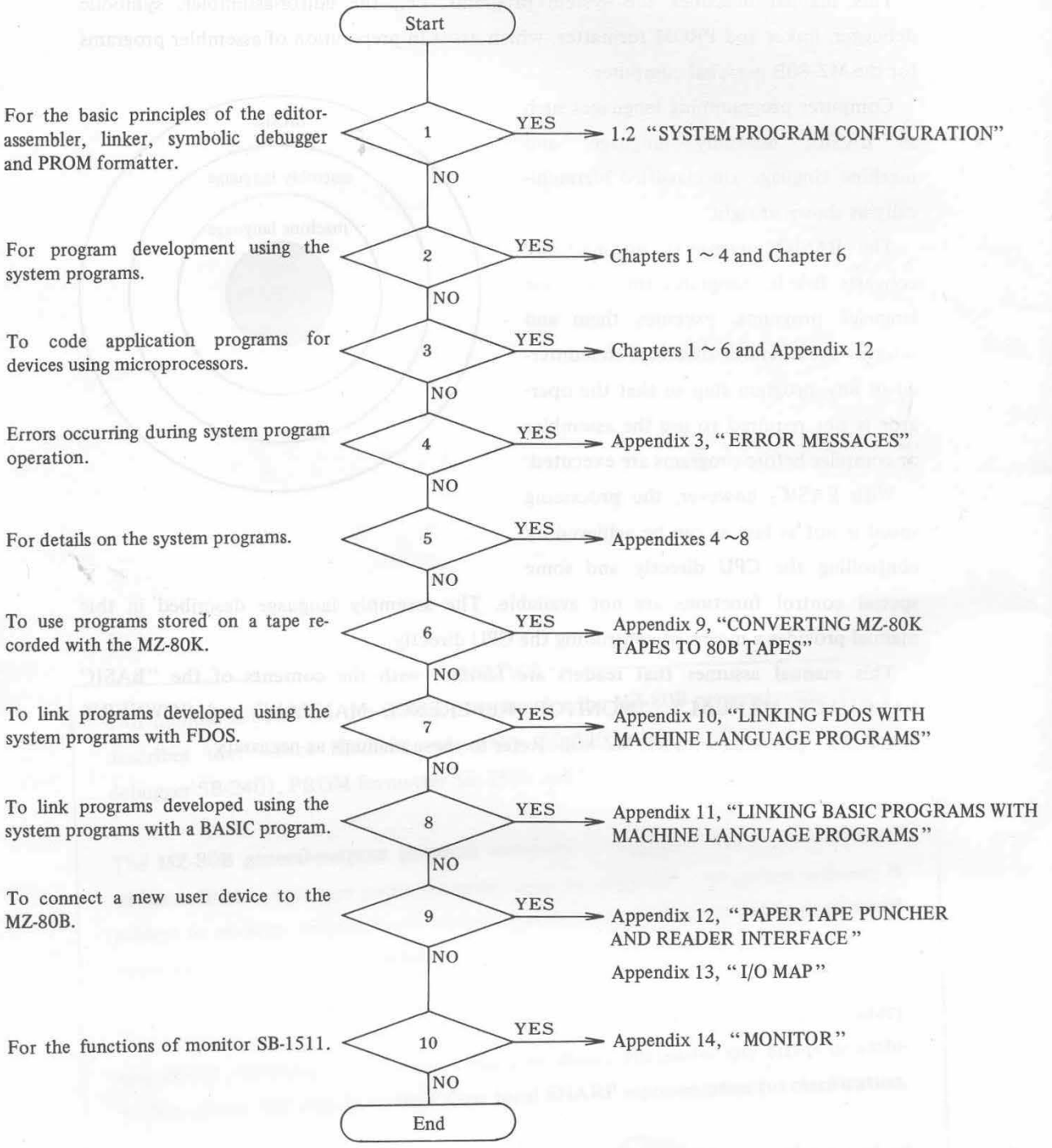


The BASIC interpreter automatically converts BASIC programs into machine language programs, executes them and informs the operator of errors encountered in any program step so that the operator is not required to use the assembler or compiler before programs are executed.

With BASIC, however, the processing speed is not as fast as can be achieved by controlling the CPU directly and some special control functions are not available. The assembly language described in this manual provides a means of controlling the CPU directly.

This manual assumes that readers are familiar with the contents of the "BASIC LANGUAGE MANUAL", "MONITOR REFERENCE MANUAL" and "OWNER'S MANUAL" provided with the MZ-80B. Refer to these manuals as necessary.

Guide to use of this manual



Product Guide

The following materials are included in this group of products.

SYSTEM PROGRAM MANUAL

Z-80 PROGRAMMING MANUAL

System Program Filed in 4 Cassette Tapes

- **Editor-Assembler (SB-2102, SB-2202), K-B converter (SB-2601)**
- **Linker (SB-2301)**
- **Symbolic debugger (SB-2401)**
- **PROM formatter (SB-2501)**

CONTENTS

CHAPTER 1
OUTLINE OF SYSTEM PROGRAM

1.1 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.

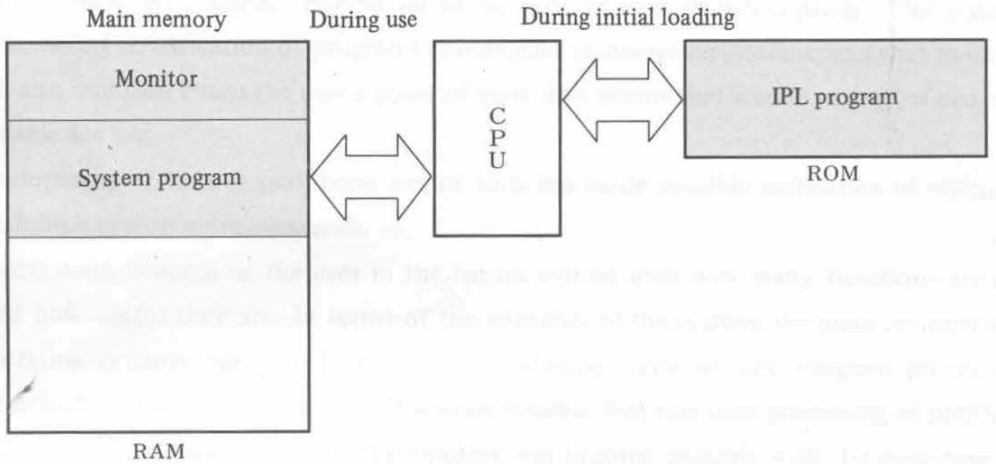
Although the name "clean computer" has been given to the MZ-80 series, computers are basically clean in principle. As the field of personal computers opens, the concept of embedding a single language, BASIC, in ROM has become a hindrance to use of full computer capacity. Out of consideration for the 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. We think that the software technology and utilization procedures applied in this system will open a new world of possibilities for personal computers.

1.2 SYSTEM PROGRAM CONFIGURATION

In keeping with the concept that the MZ-80B is a clean computer, all system software is supported by external files; these constitute the system which starts the various software elements.

To use the MZ-80B, first set the power switch to ON; when this is done, a program called the IPL (initial program loader) is started automatically. As the name indicates, this program loads the system software from the cassette tape file or diskette file. After loading is completed, control is passed to the system software and system activation is complete.
As is shown in the figure below, the IPL program is located in ROM at an address which is outside of that of the main memory. The IPL program controls the CPU only while the system software is being loaded after the power is turned on.



As is shown in the figure above, a program called the monitor is read into main memory when the IPL is started. The monitor includes functions for monitoring operation of the MZ-80B's various types of software, subroutines for performing the various logical operations, and subroutines for controlling input/output of the MZ-80B's hardware devices. The monitor is a program which has been prepared in machine language, and is provided with commands for active system control of data preparation and file input/output.

After the monitor program is loaded, the system programs are read into main memory. The system programs are the software which is used by the Z-80 CPU for assembler programming, and consists of an editor-assembler, linker, symbolic debugger, and PROM formatter. Although direct use of machine language as instruction language is possible in programming, instructions are difficult to grasp directly and addresses cannot be made relocatable. Therefore, a method is used in which the programmer uses machine code mnemonics to describe the program, together with arbitrarily selected symbols to express the data, addresses, and so forth which are referenced by the program's instructions. The system programs support the sequence of operations necessary to convert programs written in this manner into mahcine language.

1.2.1. System program organization

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

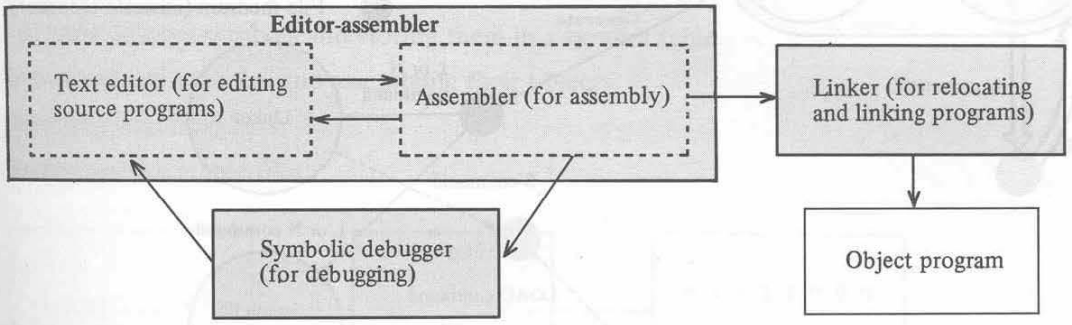


Fig. 1-1 Outline of the assembly process

Figure 1.1 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 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 1.2** 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 accomodate 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 Appendix 2.

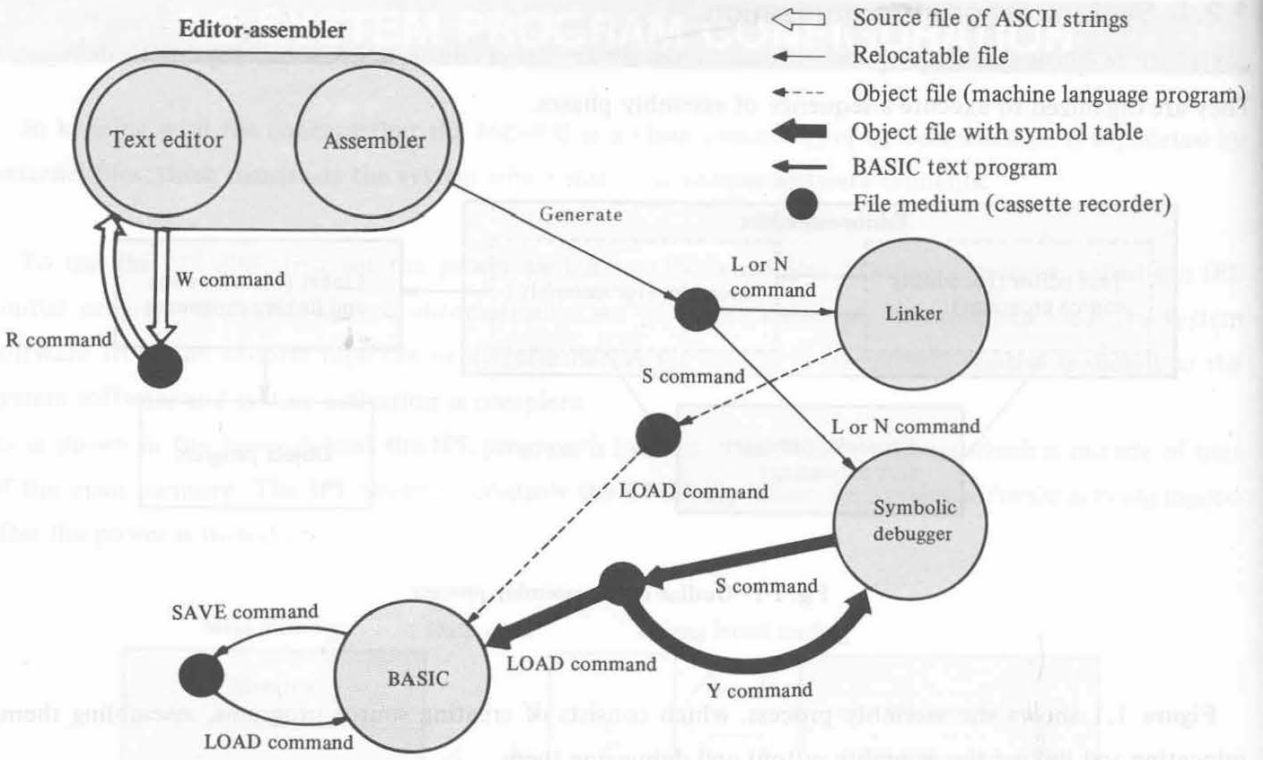


Fig. 1-2 File handling among the tape based system programs

1.2.2. Functions of the text editor

The primary functions of the text editor include those used for making insertions, deletions, and modifications in source programs. As is shown in **Figure 1-3**, the contents of source programs are displayed on a CRT screen and can then be modified/edited conversationally. This makes it possible to perform these tasks with a minimum of effort. Further, introduction of the concept of a character pointer (referred to as CP below) makes it possible to edit source programs with even less effort.

The command format used in this system is compatible with the NOVA editor program manufactured by the Data General Corporation, perfected over a period of many years by many users.

The figure below shows the general flow of processing performed by the text editor.

①Source program read into the edit buffer of the text editor from cassette tape.

②While watching the CRT screen, the user moves the CP around as necessary and makes insertions, deletions, and modifications. The source program in the edit buffer is revised concurrently.

③After all modifications have been made, the source program is written onto cassette tape from the edit buffer.

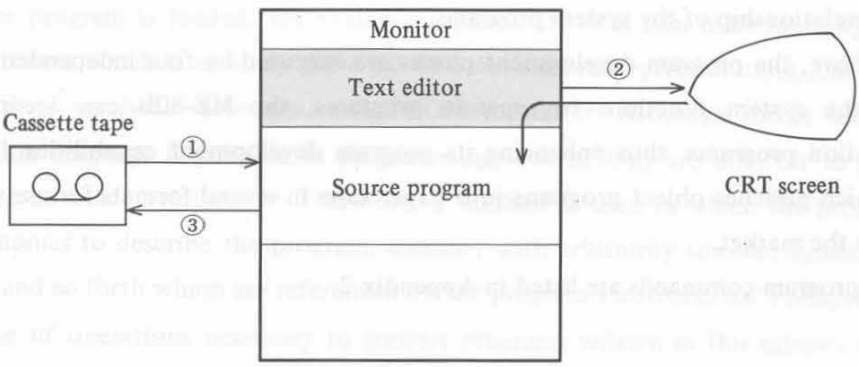


Fig. 1-3 Processing performed using the text editor

1.2.3 Functions of the assembler

The assembler converts programs written in assembly language into machine language. In other words, source programs composed of ASCII code which are prepared using the text editor are read and used to prepare relocatable programs composed of arrays of binary numbers.

This process can be broadly divided into four steps, as follows.

(1) Identifying label symbols and storing them in a symbol table.

(2) Identifying nmemonics and assembling their objects.

(3) Preparing assembly lists.

(4) Preparing relocatable files.

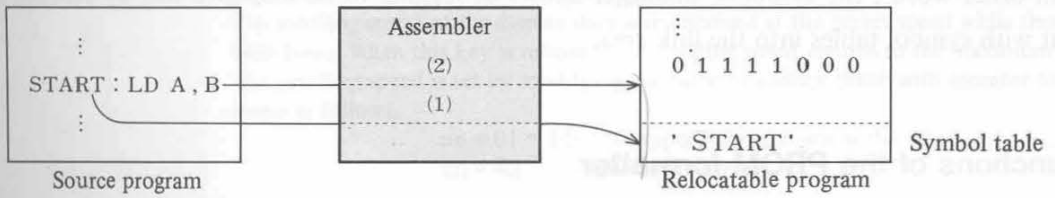


Fig. 1-4 Functions of the assembler

1.2.4 Functions of the linker

The linker reads relocatable programs output by the assembler, converts them into the format in which they are actually executed, then outputs them as object files. It is also capable of linking relocatable program units to produce a single object file.

In other words, relocatable programs output by the assembler are normally organized around addresses which are relative to address 0000. When such programs are executed, no problem results if they are loaded starting at 0000; however, this is not normally the case. Thus, it is necessary to reorganize such programs around the addresses into which they are loaded for execution. This is the function of the linker.

Also, in some cases a program references symbols which are defined in another program. Another function of the linker is to link such programs and to ensure that external program references are made properly.

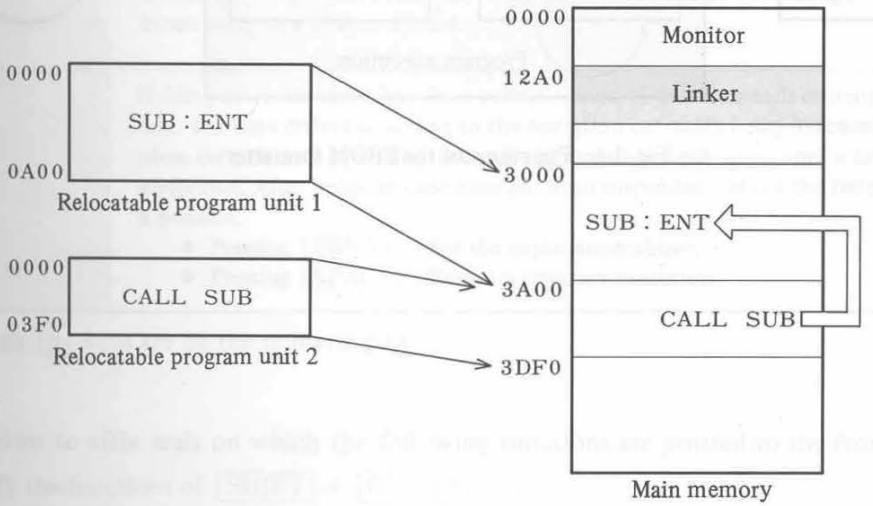


Fig. 1-5 Functions of the linker

1.2.5 Functions of the symbolic debugger

The functions of the symbolic debugger are similar to those of the linker, except that the symbolic debugger operates on the premise that there is already an object program in executable format in the link area. Debugging is then performed by actually executing the program.

Debugging is performed using break points. These break points are set at appropriately determined locations in the program, and program execution stops at these points to allow the status of the system to be determined.

The symbolic debugger is also capable of outputting object programs being debugged along with their symbol tables as object files. This makes debugging easier when program units are linked to form a single program. In other words, the symbolic debugger allows debugging to be reopened just by reading object files output with symbol tables into the link area.

1.2.6 Functions of the PROM formatter

The PROM formatter is the system program which controls the tape puncher used by the PROM writer; it also controls the paper tape reader, and is equipped with functions which are identical to those of the symbolic debugger.

There are many paper tape output formats; those which are provided are as shown below.

(1) BNPF format
(2) B10F format
(3) Hexadecimal format
(4) Binary format

The functions of the PROM formatter are as shown in **Figure 1-6.**

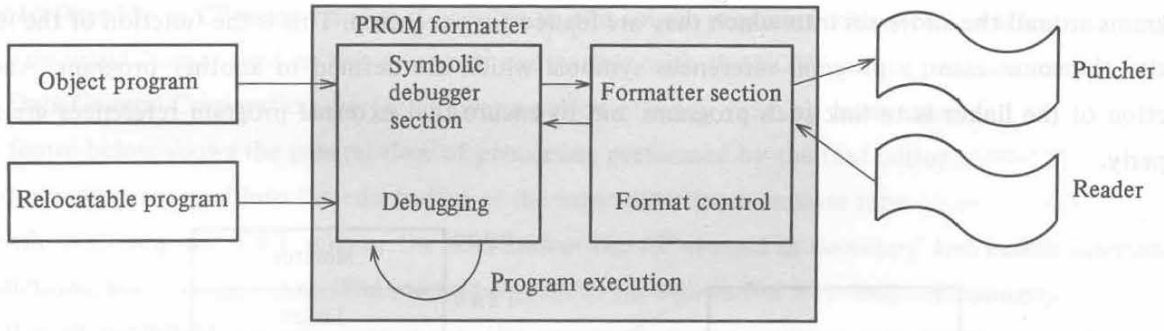


Fig. 1-6 Functions of the PROM formatter

1.3 CONTROL KEYS OPERATIONS

This section explains the functions and use of the special control keys which are used in common by the system programs.

1.3.1 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 by modifying the value of address 000F with monitor M command as follows. 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 [CR] terminates insertion.
[BREAK]	Terminates the program currently being executed, dispalys the message " Break " and awaits entry of a new command.
[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 [BREAK] : See the explanation above. • Pressing [SPACE] : 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 TO EOL] [SETTAB] [CLRTAB] [CLR ALL TAB] [CHR40] [CHANGE] [CHR80]

1.3.2 Automatic repeat function

The auto repeat speed and the amount of time which keys must be depressed before auto repeat operates can be modified using the monitor M command as follows.

＊M	← Enter M while the monitor is in the command wait state.
M–adr . $ 0 0 0 D	← Specify address 000D.
000D 2 0 s s [CR]	← ss: Auto repeat speed; speed decreases as ss is increased from 01 to FF.
000E 1 0 t t [CR]	← ss＊tt: The amount of time keys must be depressed before auto repeat operates; becomes longer as tt is increased from 01 to FF.
000F 4 0 [BREAK]	← Press [BREAK] to terminate the M command.

1.3.3 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]	H	05H	Moves the cursor to the home position.
[GRPH] + [CLR]	C	06H	Clears the screen and moves the cursor to the home position.
[SHIFT] + [TAB]	▩	1FH	Delimiter.

1.4 PROCEDURE FOR USING THE SYSTEM PROGRAMS TO DEVELOP OBJECT PROGRAMS

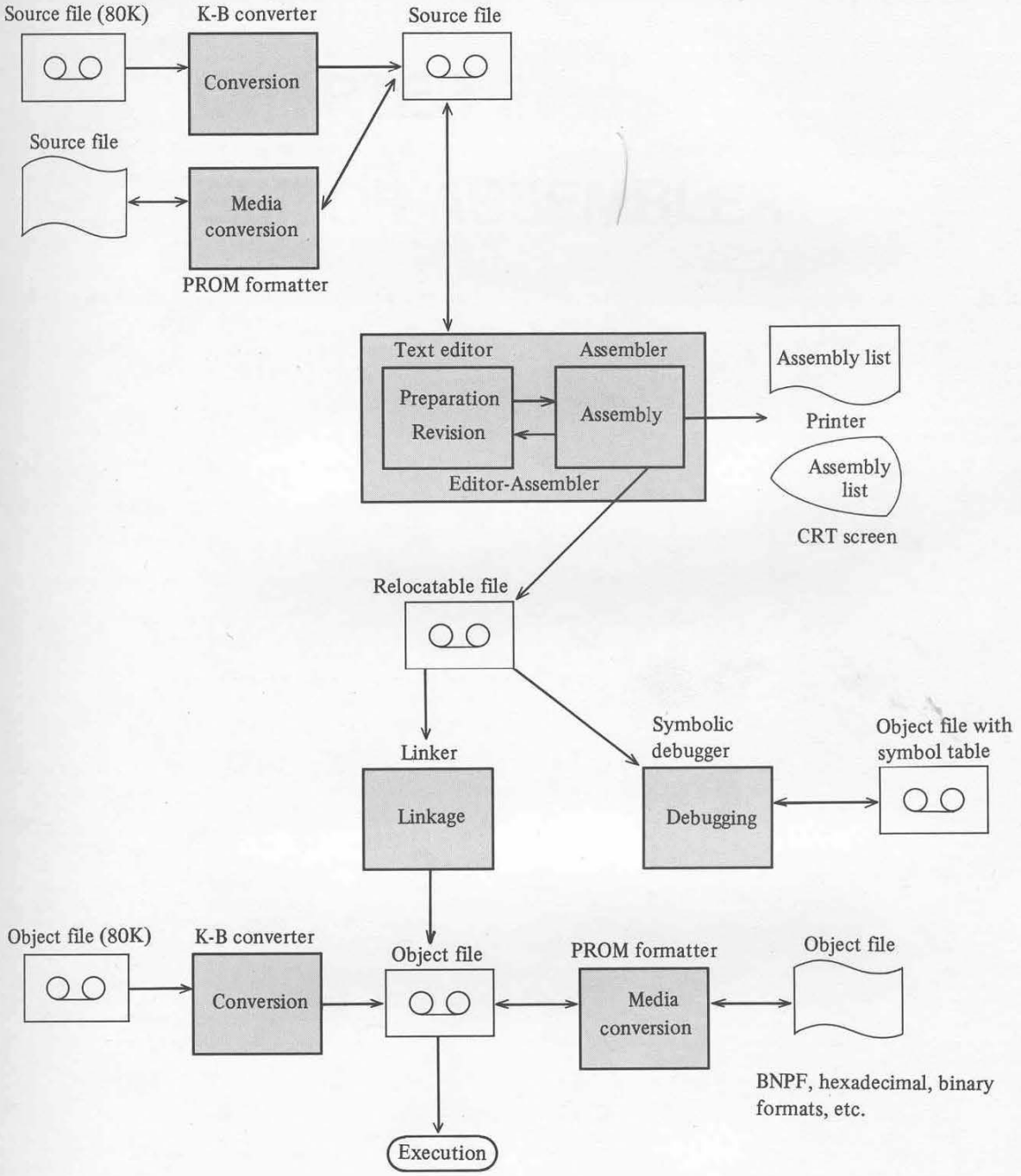


Fig. 1-7

CHAPTER 2
EDITOR-ASSEMBLER

2.1 OUTLINE OF THE EDITOR-ASSEMBLER

As its name indicates, the editor-assembler is the system program which includes both the text editor and the assembler. This section discusses the editor-assembler in outline; see sections 2.2 and 2.3 for details.

Control is transferred between the text editor and the assembler as indicated below.

Text editor → **Assembler : "X" command**
Assembler → **Text editor: "BREAK" key**

The reason for combining the text editor and the assembler in this manner is to eliminate the need to change tapes when control is transferred between the two. That is, combining the text editor and the assembler makes it possible to edit and assemble programs in one sitting by allowing the assembly list to be reviewed and errors in the source program to be corrected immediately. For example, it is normal for several errors to be made in keying and symbols during source program preparation; if it were necessary to replace the tape each time an error was corrected, a great amount of time would be consumed. The text editor eliminates this requirement and makes it possible to both edit the source program and check it at the same time.

In the photo below, the editor-assembler is first loaded by the IPL, then three text lines are prepared using text editor SB-2102 (which is activated first); then the **X command** is executed to shift to assembler SB-2202; finally, the **BREAK key** is pressed to return control to the text editor from the assembler and the **T command** is executed.

① : Editor-assembler loaded by IPL program.
② : Number of usable edit buffer bytes displayed.
③ : Three lines of text prepared using the text editor "I" command.
④ : "X" command executed to transfer control to the assembler.
⑤ : Instruction entered in response to question from the assembler.
⑥ : Control returned to the text editor by pressing the "BREAK" key and the command wait state entered.
⑦ : "T" command entered and text lines displayed. The CP remains in the position it was in before control is transferred to the assembler.

2.2 TEXT EDITOR

2.2.1 Outline of the text editor

The text editor is used to prepare source programs for the assembler and files (such as data files) which consist of strings of ASCII characters. It is also used to read in and correct or edit such programs and files and to output edited source files.

The following functions are provided for making modifications and revisions.

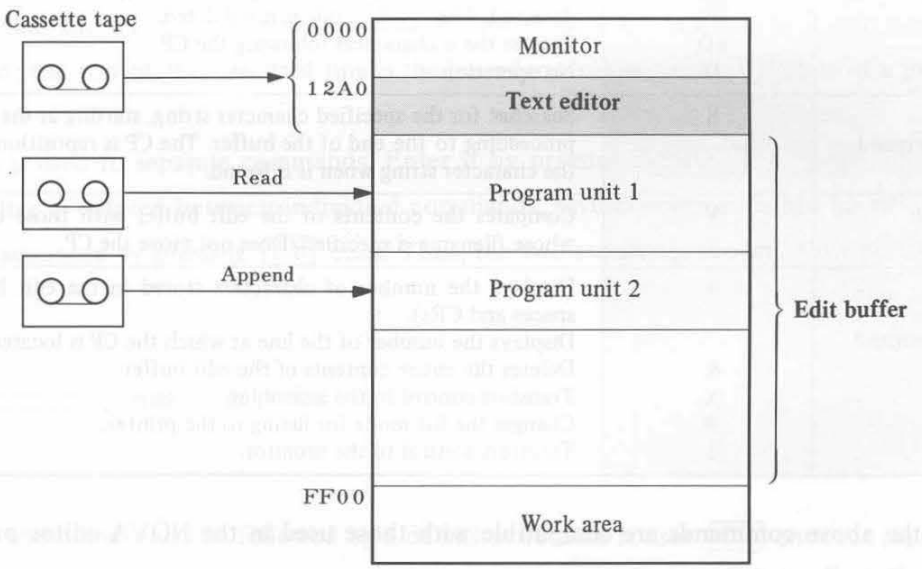
1. **Insertion**
2. **Deletion**
3. **Change**

Data input into the edit buffer is organized two dimensionally in lines and columns. A number which is referred to as the **line number** is assigned to each line in sequence, starting with the first line in the edit buffer.

Locations within the edit buffer which are to be modified are usually specified by means of a pointer (which is referred to as the **character pointer**, hereafter referred to as **CP**). Insertions, deletions, and changes are made by moving the CP to the appropriate line and executing the appropriate command.

Revisions and modifications can be made in units of either lines or words. It is also possible to search for or exchange character strings in character string units.

When the text editor is used, the memory is organized as shown in the figure below.



Editor commands are listed in the following table. Commands are 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.
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.
	Z	Moves the CP to the end of the text in the edit buffer.
Correction command	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.
	D	No operation.
Search command	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.
Comparison command	V	Compares the contents of the edit buffer with those of the input file whose filename is specified. Does not move the CP.
Special 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.
	X	Transfers control to the assembler.
	#	Changes the list mode for listing to the printer.
	!	Transfers control to the monitor.

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

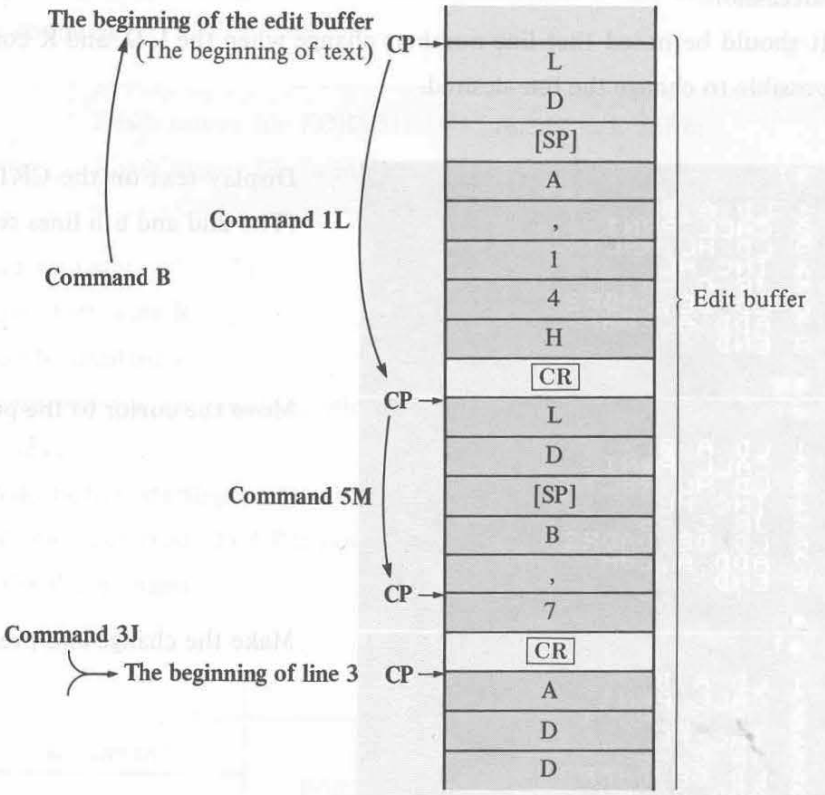
2.2.2. Character pointer and delimiter

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.

The delimiter is used to separate commands. Enter it by pressing [SHIFT] + [TAB] simultaneously. When the delimiter is entered between individual commands, several commands can be entered together and executed in sequence by pressing [CR] once. Thus, the two sequences shown below perform the same function.

```
B [CR]
10L [CR]   ⟺   B▩10L▩1K [CR]
1K [CR]
```

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.

```
3J▩2M▩1D▩IC▩ [CR]      or      B▩CADD▩ADC [CR]
```

— Screen editing—

Date can be changed or modified directly on the CRT screen.

After the data has been displayed using the T, C, Q, or S commands, the cursor is moved to lines displayed on the screen and the data is rewritten. The line in which the cursor is positioned is changed when [CR] is pressed, and the CP is positioned to the end of that line. It is also possible to change multiple lines in succession.

It should be noted that line numbers change when the I, D, and K commands are used; this can make it impossible to change the line desired.

```
*T

1 LD A,B
2 CALL PPNT
3 LD HL,RSLT
4 INC HL
5 JR C,-6
6 LD DE,DADR3
7 ADD A,20H
8 END

*
```

Display text on the CRT screen with the "T" command. (The 2nd and 6th lines require revision.)

```
*T

1 LD A,B
2 CALL PRNT
3 LD HL,RSLT
4 INC HL
5 JR C,-6
6 LD DE,DADR3
7 ADD A,20H
8 END

*
```

Move the cursor to the point to be modified.

```
*T

1 LD A,B
2 CALL PRNT
3 LD HL,RSLT
4 INC HL
5 JR C,-6
6 LD DE,DADR3
7 ADD A,20H
8 END

*
```

Make the change and press [CR] .

```
*T

1 LD A,B
2 CALL PRNT
3 LD HL,RSLT
4 INC HL
5 JR C,-6
6 LD DE,DADR1
7 ADD A,20H
8 END

*
```

Move the cursor to the next line to be modified, make the change, and press [CR] .

```
*T

1 LD A,B
2 CALL PRNT
3 LD HL,RSLT
4 INC HL
5 JR C,-6
6 LD DE,DADR1
7 ADD A,20H
8 END

*T
```

Return the text editor to the command wait state by moving the cursor to a blank line and pressing [CR] ; or, position the cursor immediately after " * " and enter the next command immediately.

2.2.3. Text editor commands

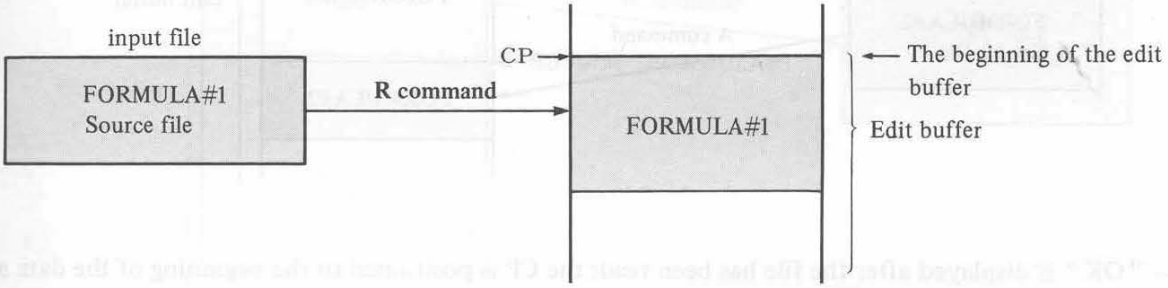
—Input commands—

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.

＊ **RFORMULA#1** [CR]	Reads source file **FORMULA#1** into the edit buffer.
＊ **R** [CR]	Reads source file found first into the edit buffer.

— Key in **R** while in the command wait state (" ＊ ").
— Specify the filename immediately following R.
(The filename specification may be omitted.)
— The text editor locates the specified file and reads it when [CR] is pressed. If the filename is not specified, the first file found is read.
— The file read is stored in the edit buffer, starting at the edit buffer's beginning. (See the figure below.)
— " OK " is displayed after the file has been read; the CP is positioned to the beginning of the edit buffer.
— Press [BREAK] to terminate the R command.



— " Check sum error " is displayed if an error occurs while a file is being read.
— The message " Full buffer " is displayed when the buffer becomes full. In such cases, only part of the input file has been read.

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.

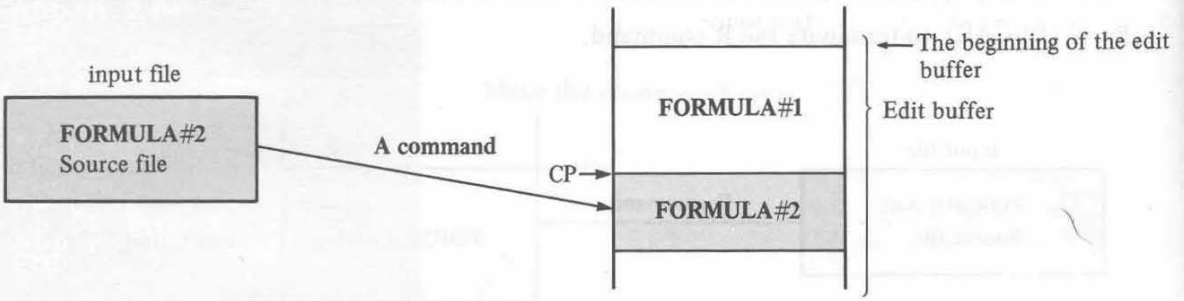
＊ **AFORMULA#2** CR	Appends source file **FORMULA#2** to the contents of the edit buffer starting at the CP position.
＊ **A** CR	Appends the first file found to the contents of the edit buffer starting at the CP position.

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

— Specify the filename immediately following A.
(The filename specification may be omitted.)

— The text editor locates the specified file and reads it when CR is pressed. If the filename is not specified, the first file found is read.

— The file read is stored in the edit buffer, starting at the position of the CP. **Use Z in order to position the CP to the end of the text when an addition is to be made to its end.** The figure below shows addition of input file "FORMULA#2" to the end of text "FORMULA#1".)



— "OK" is displayed after the file has been read; the CP is positioned to the beginning of the data added.

— Press BREAK to terminate the A command.

— "Check sum error" is displayed if an error occurs while a file is being read.

— The message "Full buffer" is displayed when the buffer becomes full. In such cases, only part of the input file has been read.

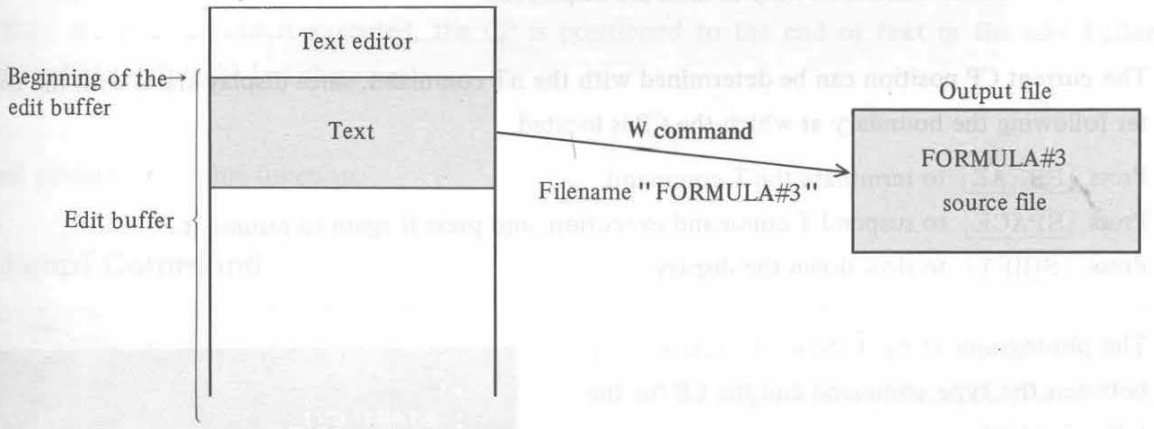
–Output command–

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.

＊ **WFORMULA#3** [CR]	Specifies "**FORMULA#3**" as the filename for the file created in the edit buffer and outputs it to that file.
＊ W [CR]	Outputs the file created in the edit buffer without specifying a filename.

— Key in W while in the command wait state ("＊").
— Specify the filename immediately following W.
— The filename specification may be omitted.
— The text editor begins outputting the text to the file when [CR] is pressed.
— "OK" is displayed after output to the file has been completed. The text editor then enters the command wait state. The file output is a source file.



— The CP position is not affected by execution of the W command.
— Press [BREAK] to terminate the W command.

—Type 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]	Assigns line numbers to lines, starting at the CP position and continuing to the line specified by n, then displays them. (Same as 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 ≧ 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.

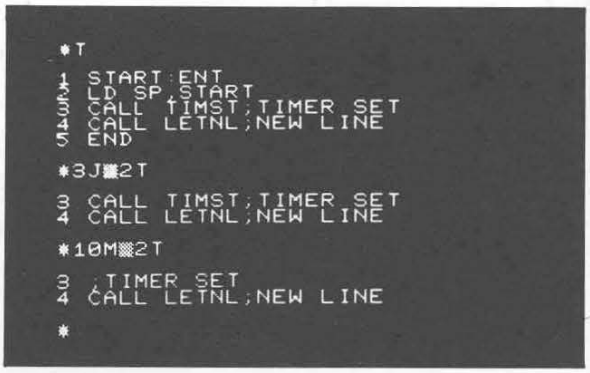
— Press [BREAK] to terminate the T command.

Press [SPACE] to suspend T command execution, and press it again to resume it.

Press [SHIFT] to slow down the display.

— 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	Moves the CP to the end of text in the edit buffer.

— Key in **Z** while in the command wait state.
— Press CR.
— When the Z command is executed, the CP is positioned to the end of text in the edit buffer (to immediately after the last character).

— 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.

∗ **nL** CR	Moves the **CP** to the beginning of the **n**th line from the line at which it is currently located.
∗ **L** CR	Moves the **CP** to the beginning of the line at which it is currently located.

— Key in **the 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 ≧ 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| ≧ ℓ − 1 (where ℓ 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.

∗ **nM** CR	Moves the **CP** to the position which is **n** characters from its current position.

— Key in **the 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** CR	Searches for the character string specified by **string 1**, starting at the current **CP** position and proceeding toward the end of the edit buffer; replaces the string with the one specified by **string 2** when it is found.
＊ **Cstring 1** CR	Deletes the character string specified by **string 1**.

— 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 CR 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.
— String 1 and string 2 need not be of the same length.

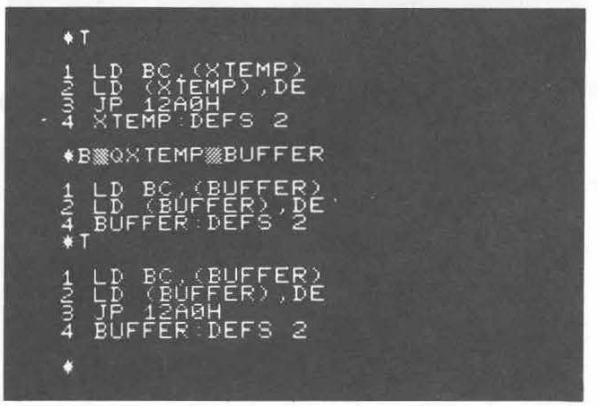
Q (Queue) Command

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** CR	Causes the function of the C command to be executed repeatedly.
＊ **Qstring 1** CR	Deletes all occurrences of the character string specified by **string 1.**

— 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 12A0H
4 XTEMP:DEFS 2
```



I (Insert) Command

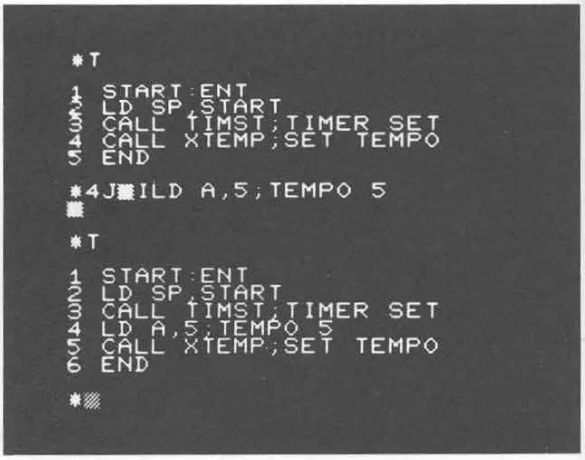
This command inserts the specified string at the CP position. A carriage return is performed on the CRT screen if one is included in the string.

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

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

— Key in **I** while in the command wait state.
— Key in the string to be inserted.

— Characters keyed in are inserted starting at the CP position. Therefore, the edit buffer contents following the CP are automatically shifted toward the end of the edit buffer.
— When a [CR] is keyed in, it is inserted as a carriage return code.
— Key in a delimiter ▩ after all the strings have been keyed in.
— Press [CR] 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.

＊ nK CR	Deletes the **n** lines preceding or following the CP from the edit buffer. If the **CP** is located in the middle of a line, the characters preceding the **CP** are not deleted if **n > 0** and the characters following the **CP** are not deleted if **n = < 0.**
＊ K CR	Deletes characters preceding the **CP** position until a CR is detected. CR 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 to illustrate operation of the commad; it has no meaning in assembly language.)


```
1 AABBCC
2 DDEEFF
3 GGHHII
4 JJKKLL
```

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. 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.

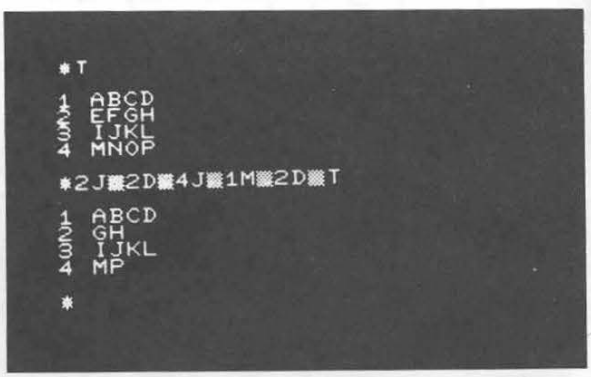
n > 0 :	Deletes the n characters following the CP from the edit buffer. A [CR] code is counted as a character.
n < 0 :	Deletes the | n | characters preceding the CP from the edit buffer. A [CR] code is counted as a character.
n = 0 or not specified	No operation results.

— 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.

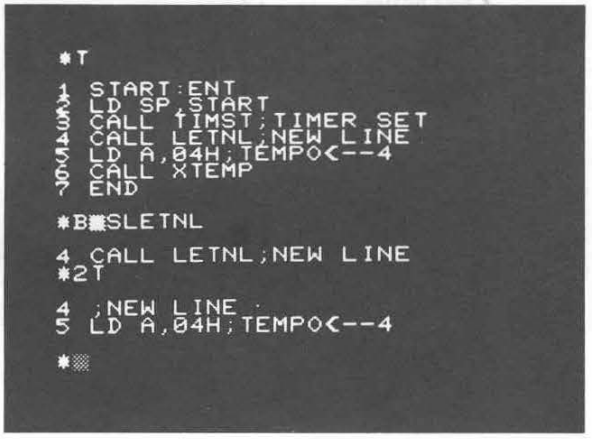
＊ S string [CR]	Searches for the specified character **string**, starting at the current **CP** position; the **CP** is repositioned to the end of the character **string** when it is found.

— 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 containing 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
```



—Verify command—

V (Verify) Command

This commands verifies the contents of the edit buffer with the contents of the file whose filename is specified.

＊ **VFORMULA#3** [CR]	Verifies the contents of the edit buffer with the contents of file **FORMULA#3.**
＊ **V** [CR]	Verifies the contents of the edit buffer with the contents of the input file.

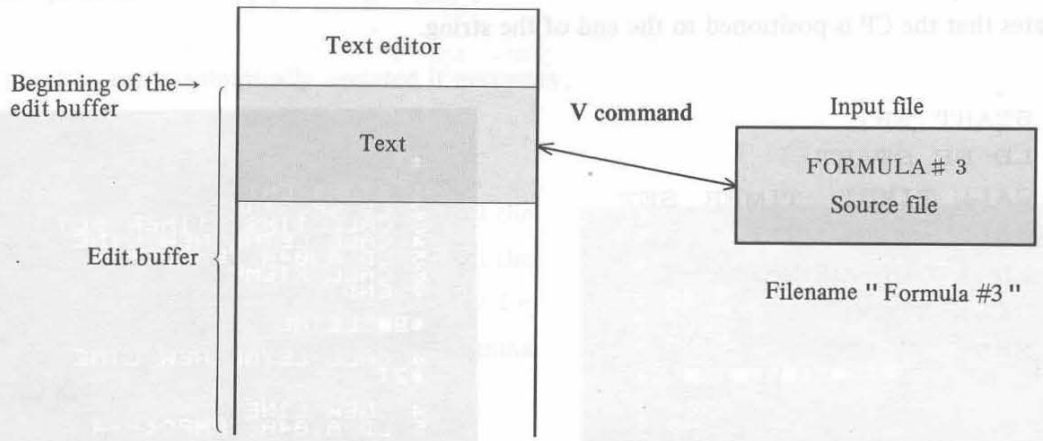
— Key in **V** while in the command wait state.

— Key in the filename of the file whose contents are to be verified. (This step may be omitted.)

— Press [CR] ; the system then looks for the specified file and starts verification.
When no filename is specified, the contents of the file first encountered are verified with the edit buffer contents.

— "OK" is displayed when the input file contents are the same as the edit buffer contents, otherwise "Check sum error" is displayed.

— The CP position is not affected by execution of this command.



–Special commands–

= Command

＊ = CR	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 CR; the total number of characters stored in the edit buffer is displayed.

. Command

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

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

& Command

＊ & CR	Clears the contents of the edit buffer.

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

X (TRANSfer) Command

＊ X CR	Transfers control to the assembler.

— Key in **X** while in the command wait state.
— Press CR ; control is then trasferred to the assembler and an assembler message is displayed.

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 ;
2 ;***  EDITOR LIST SAMPLE  ***
3 ;
4 START:ENT
5 MAIN1:ENT
6 LD SP,START ;INITIAL STACK POINTER
7 CALL LETNL
8 LD A,5
9 CALL XTEMP ;SET TEMPO TO 5
10 CALL CLTBL ;CLEAR TABLE
11 XOR A
12 LD (?TABP),A ;INITIAL I/O #1
13 RET
14 ?TABP:DEFS 1
15 END
```

! Command

＊ ! CR	Transfers control to the monitor.

— Key in ! **(exclamation mark)** while in the command wait state.

— Press CR ; the following message is then displayed.

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

Pressing the M key transfers control to the monitor.

Pressing the B key transfers control to the IPL.

Pressing the C key cancels the ! command and returns the text editor to the command wait state.

— There are two methods of returning control to the editor from the monitor.

Jump to address 12A0: The edit buffer contents are cleared. (cold start)

Jump to address 12A3: The edit buffer contents are not cleared. (warm start)

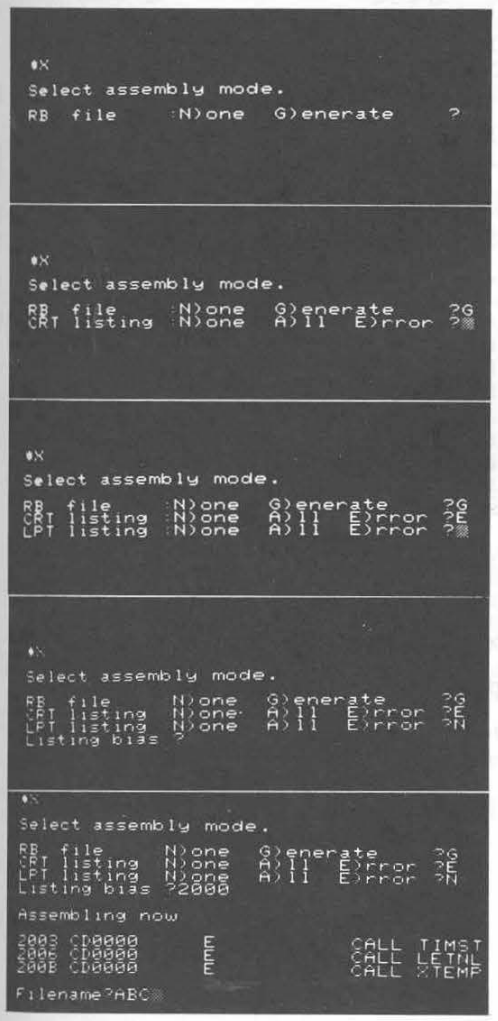
2.3 ASSEMBLER

2.3.1 Outline of the assembler

The assembler is a system program which assembles source files prepared and edited using the text editor and outputs relocatable files (relocatable binary files). Relocatable files are the stage which is between source files and object files, and are organized in such a manner as to be relocatable and linkable. In other words, the fact that final determination of addresses is made by the linker means that the linker can be used to combine various program units through the use of label symbol entry declarations (ENT directive).

Source files consist of assembly language (in other words, label symbols, mnemonic symbols of instruction codes, directives, comments, and end statements) which must be coded in accordance with the assembler rules. Source programs edited with the text editor are output as ASCII code. The assembler interprets the syntax of such output to produce relocatable files; messages are displayed at this time to indicate the status of symbol address (data) definitions and syntax errors. These messages are output in the assembly list message column of either the CRT screen or the printer.

–Starting the assembler–



Control is transferred from the text editor to the assembler by entering the X command.
First, select whether or not a relocatable file is to be prepared.

No → None
Yes → Generate

Next, indicate what is to be displayed on the CRT screen.

Nothing → None
Everything → All
Error information only → Error

Next, indicate what is to be printed on the printer.

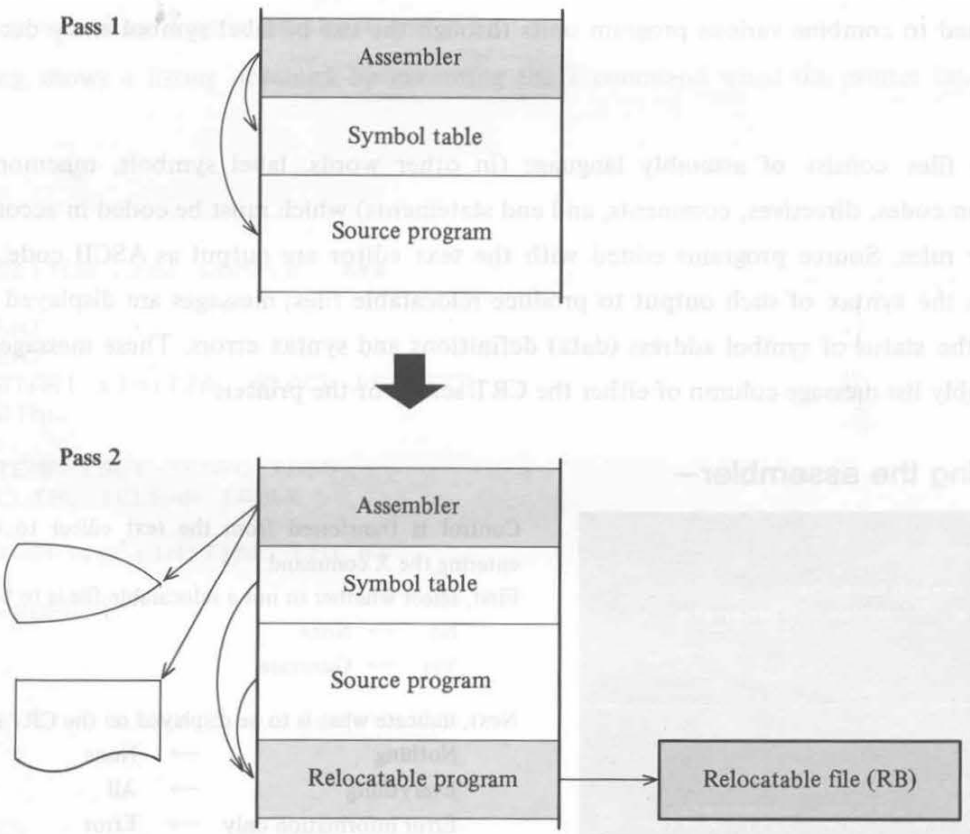
Nothing → None
Everything → All
Error information only → Error

Next, enter the listing bias (4 hexadecimal numerals; to be discussed later).

When a relocatable file is to be prepared, enter the filename.

—2 pass system—

The assembler basically uses a 2 pass system. An assembler pass is the process of reading the source program once from its beginning to end. Since the text editor must be started before the assembler, the source program in memory has already been developed using the text editor; thus the assembler does not read a external file but the source program in memory. The figure below shows assembler operation for the 2 pass system.



In pass 1, the assembler stores label symbols permitted under the assembler rules in the symbol table. This is not only to express data and addresses in decimal or hexadecimal representation, but to make the task of programming easier.

In pass 2, the relocatable program is prepared with reference to the symbol table and the assembly list is output (either on the CRT screen or the printer). The specified filename is then assigned to the relocatable program prepared and it is output as a relocatable file.

–Listing bias and ORG directive–

In the sample listing below, the "**ORG**" directive at the beginning starts relative addressing at address "2000". The assembly listing can be started at an appropriate address in the same manner to make it easier to read.

This is the idea of the "**listing bias**" which was mentioned earlier. For example, a listing which is the same as the one shown below can be obtained even without the ORG directive if a listing bias of 2000 is specified.

Unlike the ORG directive, however, listing bias is effective only when a listing is being produced and has no effect on the relocatable file created. Also, the ORG directive has priority when it is used even if listing bias is specified.

The simple program shown below is provided for the purpose of helping to explain the functions of the assembler; it has no meaning in execution.

```
   **   Z80  ASSEMBLER SB-2202 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    'CHH'
14 2010 0304
15 2012 7E                  LD      A, (HL)
16 2013 7E                  LD      A,M             ; M may be used instead of (HL)
17 2014             ;
18 2014             XYZ:    EQU     10
19 2014 C32120              JP      ABC+XYZ         ; Address definition symbol ± EQU definition symbol
20 2017 C30A00      ABC:    JP      XYZ
21 201A C31420              JP      ABC-3
22 201D C30A00              JP      10              ; Absolute address 10
23 2020 C32A20              JP      +10             ; Relative address 2AH (20H+10)
24 2023 2100D0              LD      HL,D000         ; Interpreted as a hexadecimal number
25 2026 213930              LD      HL,12345
26 2029 212120              LD      HL,ABC+XYZ
27 202C 3E0D                LD      A, XYZ+3        ; EQU definition label ± numerical data
28 202E 3EFF                LD      A, -1           ; Negative number is converted to one's complement
29 2030 21FFFF              LD      HL, -1
30 2033 21F0FF              LD      HL, -10H
31 2036 C33520              JP      -1
32 2039             ;
33 2039 CD4A20              CALL    ZZZ
34 203C CD5420              CALL    ZZZ+10
35 203F CD4B20              CALL    ZZZ+XXX
36 2042 21FFFF              LD      HL, -XXX
37 2045 21FEFF              LD      HL, -XXX-XXX
38 2048 4920                DEFW    ZZZ-XXX
39 204A 00          ZZZ:    NOP
40 204B  P          XXX:    EQU     1
41 204B                     END
   **   Z80 ASEMBLER SB-2202 PAGE 02 **
ABC      0017   XYZ    000A                        ; Shows the contents of the symbol table
```

2.3.2. 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, sepecial 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 [SP] F0H [SP]; A<–X0          }
                XYZ : PUSH [SP] AF               } Editor list
                ADD [SP] HL, BC [SP] ; BC = COUNT}
                          ⬇
                OR      F0H      ; A<–X0         }
          XYZ:  PUSH    AF       ;               } Assembly listing
                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 performes 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:  ;                                  }
          ; SAMPLE PROGRAM                   } All lines are comments.
          ;                                  }
          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 instruction statements.

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 alphanumerics 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 40.)

—Label symbols—

All characters other than special symbols may be used for label symbols, but generally alphanumerics 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 ←				These 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.

Multiple label symbols may be defined as relocatable instruction addresses as follows.

```
Example:  ABCD:   ENT
          EFGH:   ENT
          IJK     LD     A, B
```

Label symbols ABCD, EFGH and IJK are all defined as relocatable addresses of LD A, B. ABCD and EFGH are also defined as global symbols.

```
          ABCD:
          EFGH:
          IJK:    LD     A, B
```

Same as the above, except that ABCD and EFGH are not global symbols.

—Constants—

There are two types of constants: decimal and hexadecimal. + and − signs can be attached to these. A character string which is defined as a label symbol is treated as a label symbol even if it satisfies the requirements for a constant.

The assembler treats a constant as a decimal constant when it consists of numerics only or it consists of numerics followed by D.

Example: 23 999 +3 −62 16D 0003D

16 3

The assembler treats a constant as a hexadecimal constant when it consists of 0~9, A, B, C, D, E and / or F followed by H.

Example: 2AH CDH +01H −BH 0010H 00ADH 00H

A constant used in the operand of a JP, JR, DJNZ or CALL instruction represents an absolute address when it has no sign and a location relative to the current address when it has a sign. In other cases, constants without signs and those with a + sign represent numerics, while those with a − sign are converted into two's complement.

2.3.3. Assembly listing and assembler messages

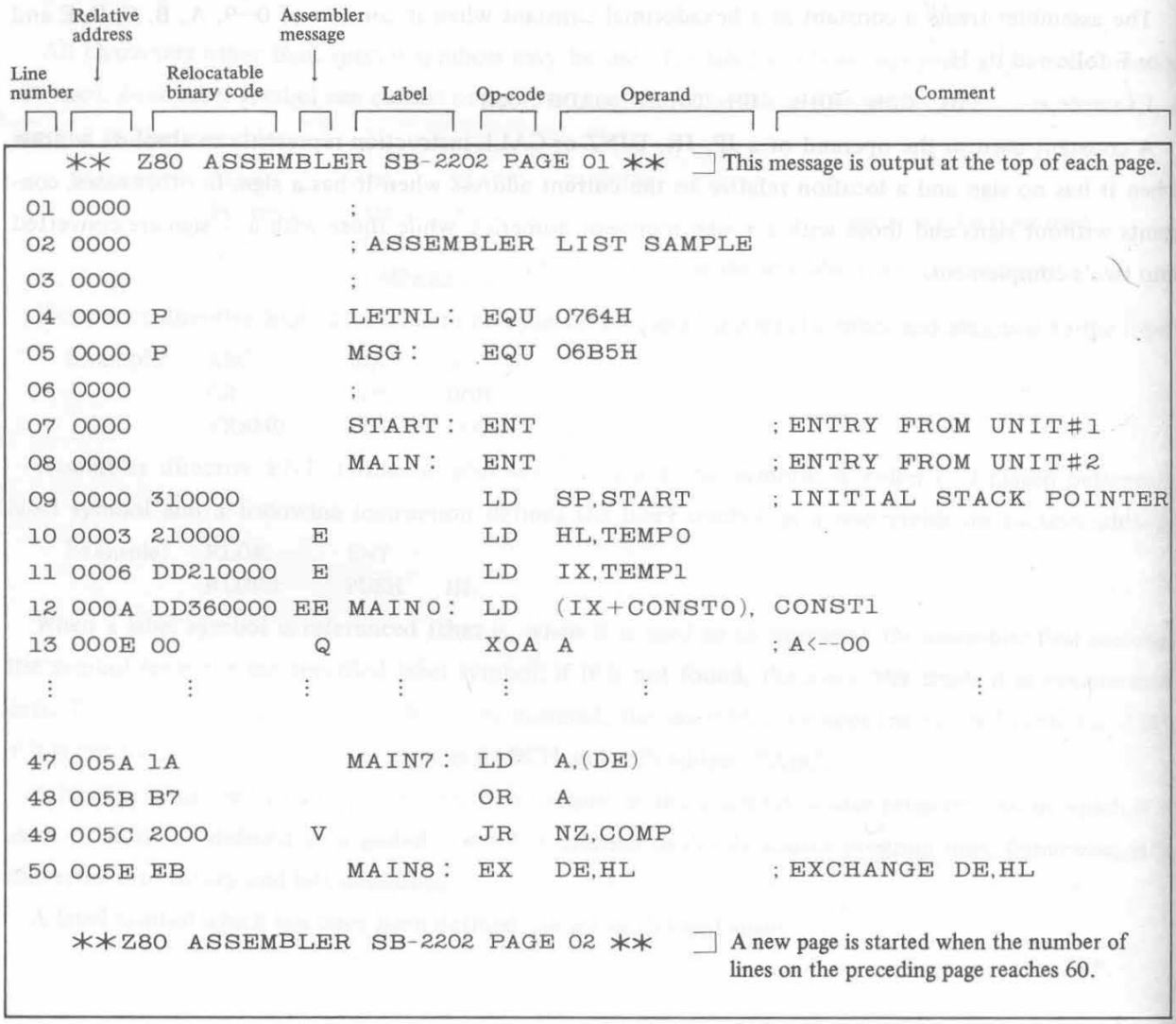
When " A " is entered in response to " CRT? " or " LPT? " from the assembler, the assembly listing is output on the CRT screen and/or printer. Examining the assembly listing is one of the most important procedures in assembly programming since this is when a check is made for errors in the source program.

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 paged 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.

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 43.)

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.
```

U (Undefined parameter)

This message indicates that a parameter was not defined when a macro instruction was called.

```
(Example)      U    JP   Z , @3
```

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
```

2.3.4. 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
Address undefined           :                  MAIN0
  CD0000  E                 CALL  CMPLX
     E message              :
                            CALL  CMPLX+2    ← No offset can be added to a label symbol
                            :                  which is defined externally.
                            END              ← END is always required at the end of a
                                               program unit.
```

Program unit 2
"GAUSS-SR"

```
                    ; GAUSS-SR
                    ;
                    CMPLX:  ENT              ← Entry definition of label symbol
                            :                  CMPLX
                            RET
Address undefined           :
  C30000  E                 JP    MAIN0
     E message              :
                            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-2202 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 assignment 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.

```
   **  Z80 ASSEMBLER SB-2202  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             ; SEPARATOR = " , " ?
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-2202  PAGE 02  **
.TYPE   2000    BUFFER  2064    SWTBL  2062    TYPE0  2014    TYPE10  203C
TYPE20  2048    TYPEER  2058
```

—IF ~ ENDIF—

This assembler directive instructs the assembler as to whether or not assembler codes following it should be assembled. If the value of the label symbol specified in its operand is zero, the assembler assembles following instruction codes up to the next ENDIF assembler directive; otherwise the assembler ignores them.

The label symbol specified in the operand must be defined on the preceding line. The operand of the IF directive may be specified by adding or subtracting numeric data to/from a label symbol.

```
01 0000                  COND :  EQU     0
02 0000                          IF      COND
03 0000 86 ]                     ADD     A,(HL)
04 0001 12 ]  Assembled because  LD      (DE),A
05 0002 23 ]  COND = 0.          INC     HL
06 0003                          END     IF
07 0003                          IF      COND+1
08 0003 ]                        SUB     (HL)
09 0003 ]  Not assembled         LD      (DE),A
10 0003 ]  because COND+1 ≠ 0    DEC     HL
11 0003                          ENDIF
```

—MACRO ~ ENDM—

This assembler directive defines the macro instruction whose label is specified in the MACRO assembler directive and executes instructions between the MACRO and ENDM assembler directives. Parameters are indicated by serial numbers preceded by @, e.g., @1 and @2. The maximum number of parameters is 7.

To call a macro instruction, use its label as a mnemonic code and specify operands corresponding to each of its parameters in succession. The assembler assembles the instruction codes which correspond to the macro instruction. The assembly list is printed out with the specified operands substituted for the corresponding parameters.

Macro instructions can be defined anywhere in a program, but must be made before the macro instruction is called. A macro instruction is similar to a subroutine, but control is not transferred to the macro instruction and the instruction codes corresponding to the macro instruction are inserted during assembly instead of the macro instruction.

```
12 0003          *              MACRO   INT
13 0003                         LD      A,(@1)
14 0003                         LD      B,00H
15 0003                 @3:     SUB     @2
16 0003                         JR      C,+5
17 0003                         INC     B
18 0003                         JR      @3
19 0003                         LD      A,B
20 0003                         LD      (@4),A
21 0003          *              ENDM
22 0003          *              MACRO   ZZZ
23 0003                         @1@2@3
24 0003          *              ENDM
25 0003          *              ZZZ     ABC,DE,F
26 0003 00       Q              ABCDEF
27 0004          *              ENDM
28 0004          *              INT     DIV,04H,LABEL,ANS
29 0004 3A2100                  LD      A,(DIV)
30 0007 0600                    LD      B,00H
31 0009 D604            LABEL:  SUB     04H
32 000B 3803                    JR      C,+5
33 000D 04                      INC     B
34 000E 18F9                    JR      LABEL
35 0010 78                      LD      A,B
36 0011 322200                  LD      (ANS),A
37 0014          *              ENDM
38 0014          *              INT     DE,(HL),LOOP
39 0014 1A                      LD      A,(DE)
40 0015 0600                    LD      B,00H
41 0017 96              LOOP:   SUB     (HL)
42 0018 3803                    JR      C,+5
43 001A 04                      INC     B
44 001B 18FA                    JR      LOOP
45 001D 78                      LD      A,B
46 001E 320000   U E            LD      (@4),A
47 0021          *              ENDM
48 0021 51              DIV:    DEFB    51H
49 0022                 ANS:    DEFS    2
50 0024                         END
```

A macro instruction is defined whose label is INT. This macro instruction uses 4 parameters.

A macro instruction is defined whose label is ZZZ.

Code corresponding to macro ZZZ is inserted.

The instruction codes corresponding to macro INT are inserted. The operands are substituted for the parameters.

The instruction codes corresponding to macro INT are inserted. Error message "U" is displayed because the number of operands specified is less than the number of parameters defined for the macro instruction.

As shown above, the relocatable binary code columns of the MACRO and ENDM directives and macro call instructions are filled with "＊".

—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 MSG (06B5H) can be used to output the message.

```
13  1FF3   B7          E          OR     A
14  1FF4   CA0000      E          JP     Z, READY
15  1FF7   110020                 LD     DE, MESG0
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               MESG0:     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               MESG0:     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          MESG1:     DEFB   '©'
29  2007   03                     DEFB   '⇨'
30  2008   0D                     DEFB   0DH
31  2009   27          MESG2:     DEFB   "'"   ]
32  200A   22                     DEFB   '"'   ]   Be sure to use single and
                                                   double quotation marks correctly.
```

-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                  MESG0 :  ENT          ; "ERROR"
22 2000  4552524F                DEFM  'ERROR'
23 2004  52
24 2005  0D                      DEFB  0DH
25 2006  06034142        MESG1 :  DEFM  '©⇨AB'
26 200A  0D                      DEFB  0DH
27 200B  41274227        MESG2 :  DEFM  "A'B'C'"
28 200F  4327                    DEFB  0DH
29 2011  0D
```

—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                TEMP0 :   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   0AH
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.

2.4 ERROR MESSAGES OF THE EDITOR-ASSEMBLER

2.4.1. Text editor error messages

Error Messages	Meaning	Relevant commands
Full buffer	Edit buffer is full.	R, A
? ? ?	n < 0 in an nT or nJ command.	T, J
Large	n greater than 65535 was specified.	T, J, L, M, K, D, B, Z
Not found	The string (or string1) specified in Sstring, Cstring1 ⊠ string2, or Qstring1 ⊠ string2 was not found following the CP.	S, C, Q
Invalid	An illegal command was entered or an incorrect format was used. Ex.) ＊H [CR] : There is no H command. ＊S [CR] : A string should be specified.	any case
Check sum error	When the V command was executed, it was found that the contents of the edit buffer differed from the contents of the input buffer; or, an error occured while a file was being read.	V, R, A

2.4.2. Assembler messages

Definition status message	Meaning	Example
E (External)	Indicates that a label symbol is being referenced externally; that is, the label is not defined in the current source program unit.	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.
P (Phase)	Defines a label symbol with a constant assigned. This message is also output when a label symbol is encountered during pass 2 which was not encountered during pass 1.	P LETNL : EQU 0762H P DATA1 : EQU 3 └—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 message	Meaning	Example
C (illegal Character error)	Indicates that an illegal character is used in the operand.	C JP +1000–3
F (Format error)	Indicates that the instruction format is incorrect.	
N (Non label error)	Indicates that no label symbol is specified for ENT or EQU.	N EQU 0012H └—No label symbol
L (erroneous Label error)	Indicates that an illegal label symbol is used.	L JR XYZ └—XYZ is not defined in the current program. 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.
M (Multiple label error)	Indicates that a label symbol is defined two or more times.	M ABC : LD DE, BUFFER ≀ M ABC : ENT └—ABC is defined twice.
O (erroneous Operand)	Indicates that an illegal operand is specified.	
Q (Qestionable mnemonic)	Indicates that the mnemonic code is incorrect.	Q CAL XYZ CALL XYZ is correct.
S (String error)	Indicates that single or double quotation mark(s) are omitted.	S DEFM GAME OVER DEFM 'GAME OVER' is correct.
U (Undefined parameter)	The number of operands specified in a macro call instruction was less than the number of parameters defined for the macro instruction.	U JP Z, @3
V (Vaule over)	Indicates that the value of the operand is out of the prescribed range.	V LD A, FF8H V SET 8, A V JR –130
END?	Indicates that the END directive is missing from the source program.	

CHAPTER 3

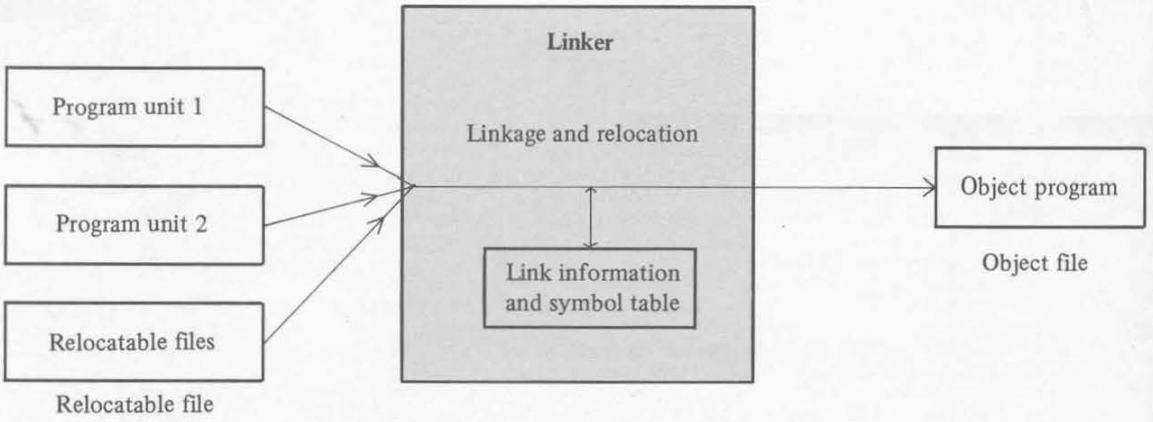
LINKER

3.1 OUTLINE OF THE LINKER

The linker inputs relocatable files generated by the assembler and generates an object file. Relocatable files are files which contain program units in a form that can not be executed by the CPU, but which contains relocation information to make the program units relocatable. They contain global symbols which are declared to link multiple program units in ASCII code.

The linker receives this information and generates the object (machine language) program in the link area while relocating each program unit by adding the programmer-specified assembly bias to each relative address referenced in the program unit. With one or more relocatable files, subsequent relocatable files are appended to preceding files during link/load operation, once the linking address is specified in the first file.

The object file is output with a loading address and execution address specified.



The linker commands are listed below.

Command name	Function
L (relocate Load)	Loads a program.
N (Next file)	Appends a program to a preceding program.
H (Height)	Displays the current assembly bias and load address.
T (Table dump)	Displays the contents of the symbol table.
S (Save)	Saves the object program in memory in a file.
V (Verify)	Compares the contents of the object file generated by the S command with the contents of the object program in memory.
X (TRANSfer)	Moves the specified memory block to the specified memory area.
＊ (clear table)	Clears the symbol table and resets the assembly bias and link address to 0000.
# (change printer mode)	Switches the printer mode.
! (go to monitor)	Transfers control to the monitor.

3.2 SYMBOL TABLE

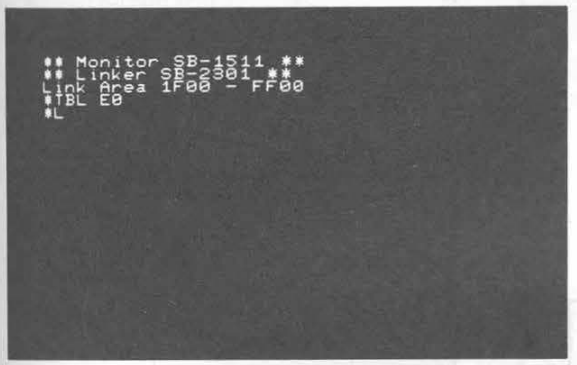
Symbols referred to by the linker and symbolic debugger are label symbols which are globally declared in a source program; that is, label symbols defined with assembler directive ENT or EQU. They are stored in ASCII code in relocatable files for use during program linkage.

When the linker inputs a relocatable file, it enters each label symbol that it encounters into the symbol table. The symbol table is located at the end of the link area that follows the linker. The higher order two digits of the starting address of the symbol table must be specified by the user. For example, when the user enters:

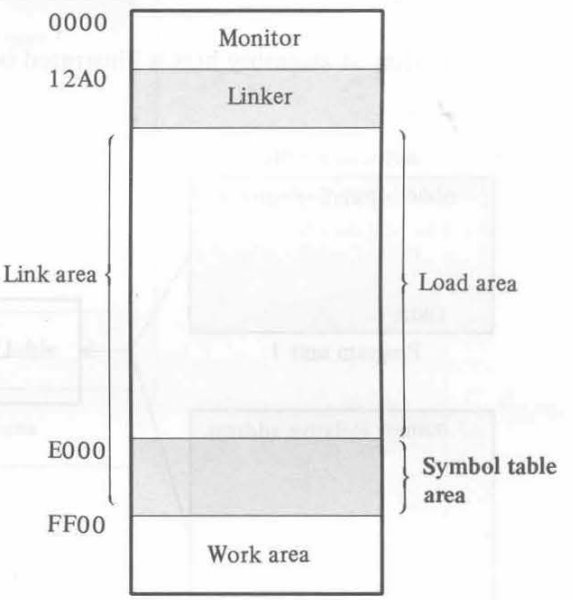
＊TBL E0

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

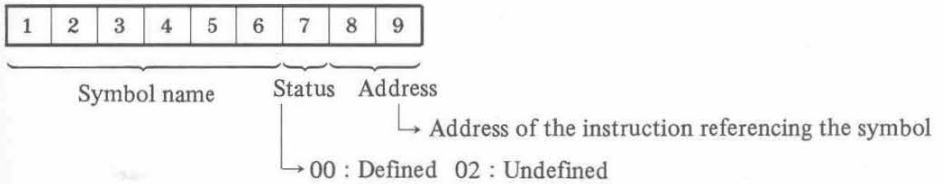
The photo below shows the display when the linker is started and the figure on the right is a memory map for the linker.



Symbol table starting address is set to E000.



Each symbol table entry is 9 bytes wide and has the format shown in the figure below. When a new symbol is encountered, the linker loads the table entry with the information shown in the figure. Refer to Appendix 6 for the manner in which the linker links relocatable files using the symbol table information.



3.3 LINKER BIAS AND ADDRESSES

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

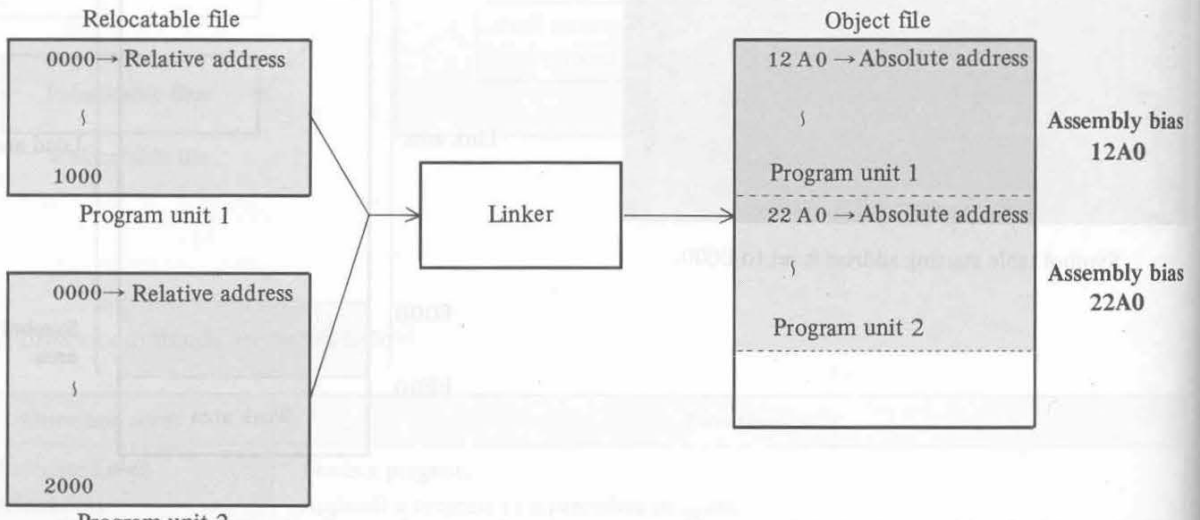
These addresses determine some of the characteristics of the object program, and cannot be determined arbitrarily; attention must be paid to their interrelationship. These addresses are described below.

—Assembly bias—

Assembly bias is used to convert the addressing mode of the object program to absolute addressing. It is added to all relative addresses of relocatable files to form absolute addresses.

Each relocatable file generated by the assembler uses relative addresses starting at 0000H. To convert the object program so that it starts at address 12A0H, for example, the user specifies 12A0H to the linker as the assembly bias.

The function of assembly bias is illustrated below.

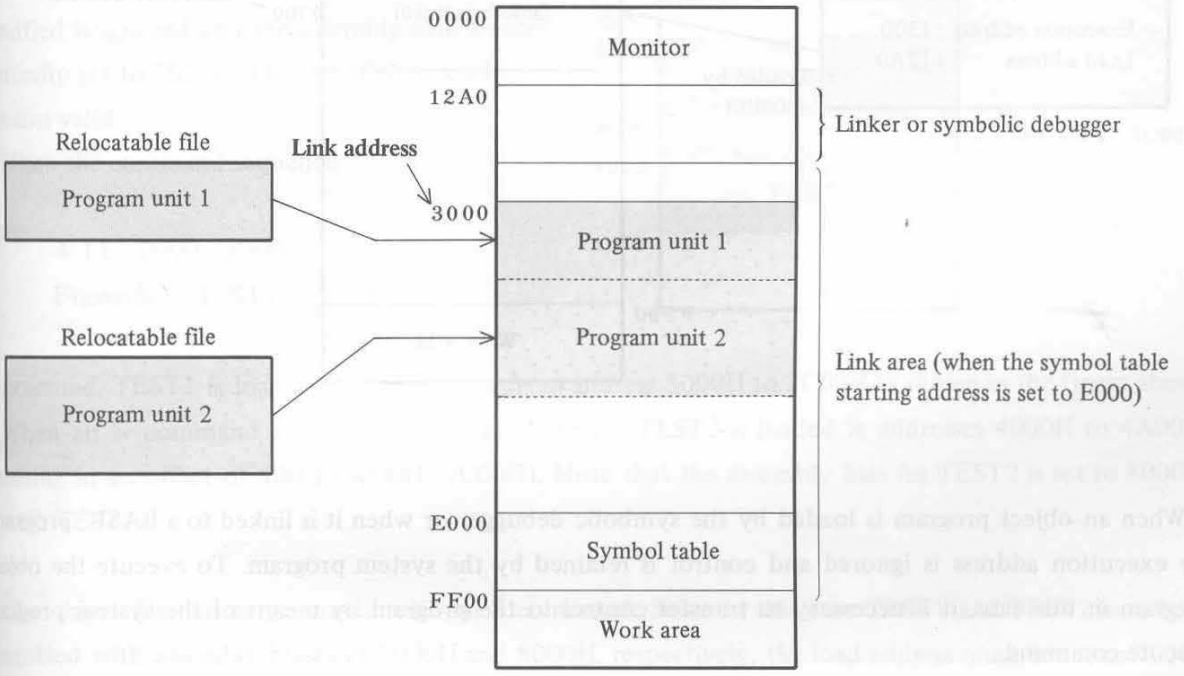


–Link address–

The link address specifies the starting address of a relocatable file in the link area. Generally, the address in memory where an object program is to be loaded in the link area does not match the address where it is to be loaded for execution. Since the link area is used to hold relocatable files only temporarily, arbitrary addresses are selected to link relocatable files; therefore, link addresses are selected arbitrarily.

Note: The symbolic debugger has commands which can be executed immediately. To use them, the object program must be in an executable form. In some cases, the link address for such programs is automatically determined when the assembly bias is specified.

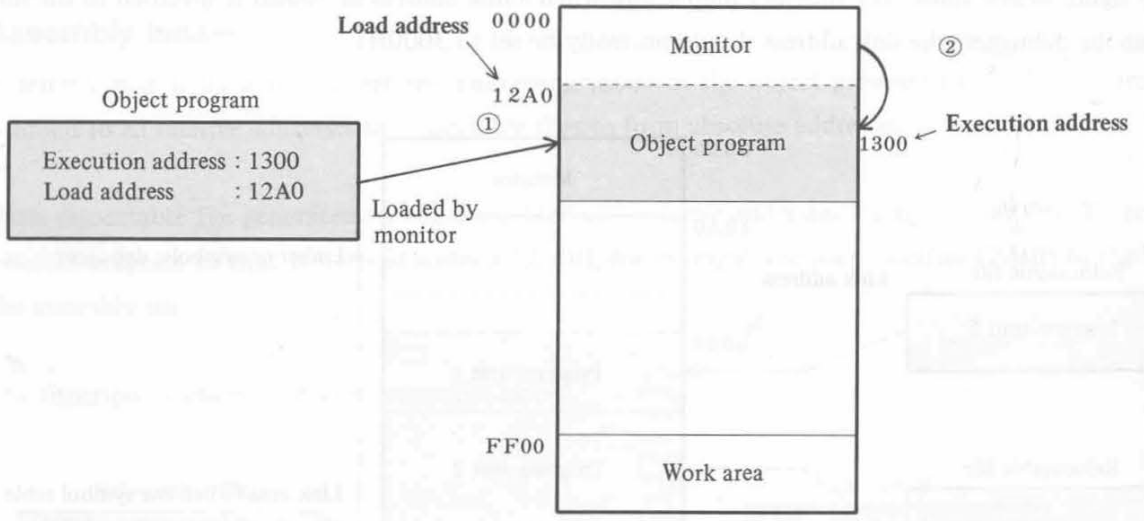
The figure below shows the memory map set up when a link address of 3000H is specified to the linker or symbolic debugger (the link address should normally be set to 3000H).



—Execution and load addresses—

The execution and load addresses must be specified when an object file is to be generated by the linker. They cannot be specified arbitrarily, but must be determined according to how the object program is created in memory. These addresses are stored as information data in the object file.

The load address specifies the starting address of an object program as it is loaded through the monitor. The execution address determines the value to be set up in the program counter in the CPU after the object program has been loaded. The figure below shows how an object program is loaded and given control when a load address of 12A0H and an execution address of 1300H are specified.



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

3.4 RELATIONSHIP BETWEEN THE ORG DIRECTIVE AND THE FOUR ADDRESSES

The load address can be specified by the linker or by the ORG assembler directive. This section describes the relationship between the ORG directive and the assembly bias, link address, load address, and execution address.

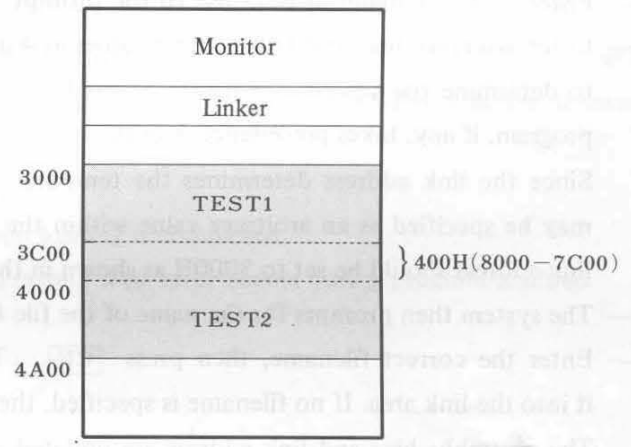
Assume two programs TEST1 and TEST2, whose starting addresses and program size are as follows:

TEST1 : ORG 7000H specified, occupies 7000H to 7C00H
TEST2 : ORG 8000H specified, occupies 8000H to 8A00H

When loading TEST1 with an L command, it is necessary to specify the assembly bias and link address. In the example, any assembly bias value specified is ignored and the assembly bias is automatically set to 7000H. The specified link address remains valid.

When the command sequence

＊LL 0000 3000
Filename? TEST1

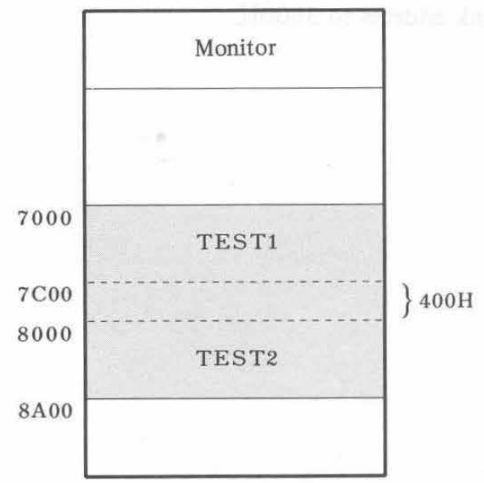


is executed, TEST1 is loaded in the link area from address 3000H to 3C00H as shown in the figure above.

When an N command is entered to read in TEST2, TEST2 is loaded in addresses 4000H to 4A00H, resulting in an offset of 400H (8000H~7C00H). Note that the assembly bias for TEST2 is set to 8000H, as with TEST1.

The object file can be generated with an S command. Since in this case TEST1 and TEST2 have been assembled with assembly biases of 7000H and 8000H, respectively, the load address must be set to 7000H for the object program to run properly.

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



3.5 LINKER COMMANDS

L (relocate Load) Command

The L command loads relocatable files into the link area. Absolute addresses in the object file are determined by specifying the assembly bias in this command.

＊ **LL 12A0 3000**	Loads a relocatable file while converting relative addresses to absolute addresses. The assembly bias is set to 12A0H and the link address is set to 3000H.

— Enter an **L** command in response to the prompt " ＊L " (" ＊L " is a prompt for a linker command).

— Enter assembly bias and link address values as 4-digit hexadecimal numbers. The assembly bias is used to determine the absolute addresses of the label symbols in the object file. The ORG directive in the program, if any, takes precedence over the assembly bias; that is, the assembly bias is ignored.
Since the link address determines the temporary location of the object program in the link area, it may be specified as an arbitrary value within the link area, excluding the symbol table. Normally, the link address should be set to 3000H as shown in the above example.

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

— Enter the correct filename, then press [CR] . The system searches for the specified file and reads it into the link area. If no filename is specified, the system reads in the first file encountered.

— The assembly bias and link address are updated when a file is read in. For example, if a relocatable file which is 100H in size is loaded with the above command, the assembly bias and link address are updated to 13A0 and 3100H, respectively.

— The system displays " OK " when the specified file has been read.

— The system displays the message " Check sum error " when an error occurs during the read.

— Press [BREAK] to terminate the file read.

— The photo at right shows how relocatable file RELOC is loaded into the link area with the L command. The assembly bias is set to 12A0H and the link address to 3000H.

N (Next file) Command

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

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

— Enter an **N** command in response to the prompt "＊L."
— The system then prompts for the name of the file to be read in with the message "**Filename?.**"
— Enter the required filename and press [CR] . The system then searches for the specified file and reads it into the link area. If no filename is specified, the system reads in the first relocatable file encountered.
— The system uses the assembly bias and link address values set up immediately before the N command is executed when loading the relocatable file. If the source program contains an ORG directive, it takes precedence over the assembly bias value.
— Programs are appended and linked during loading.
— The system displays message "OK" when program loading is completed.
— The system issues the error message "Check sum error" if an error occurs during program loading.
— Press [BREAK] to terminate program loading.

H (Height) Command

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

— Enter a **H** command in response to the prompt "＊L."
— The system then displays two 4-digit hexadecimal numbers indicating the current assembly bias and link address.

T (Table dump) Command

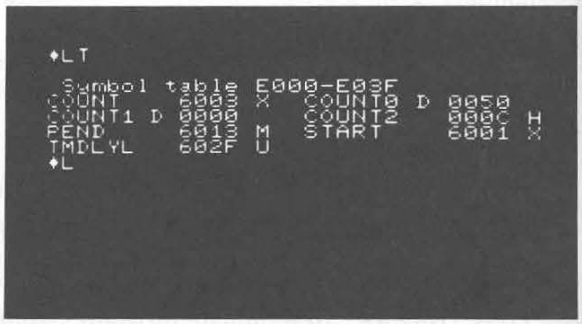
The T command displays the contents of the symbol table. Each symbol table entry consists of a label symbol name, its absolute address, and its definition status.

＊ LT	Displays the contents of the symbol table.

— Enter a T command in response to the prompt "＊L."

— The system displays the label symbol name, absolute address (in hexadecimal), and definition status for each symbol table entry. The operator can find invalid symbol definitions by examining the definition status for each symbol.

— The photo at right shows an example of execution of the T command following an L command, to check symbol definitions for validity. Note that undefined symbols are identified by the character "U."



— Messages pertaining to the symbol definition status are listed in the table blow.

Examples of link messages are given at next page.

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

No message is issued for symbols defined. Messages U, M, X, and H are error messages.

— Link message examples —

First program unit loaded (UNIT—#1)

```
TMDLYH :   LD      HL,START
COUNT :    ENT
           DEC     HL
           LD      A,H
           CP      COUNTO
           JR      NZ,COUNT
           LD      A,L
           CP      COUNT1
           JR      NZ,COUNT
           CP      COUNT2
           JR      NZ,COUNT
           RET
PEND :     ENT
           DEFM    'TMDLYH'
           DEFB    0DH
COUNT 1:   EQU     00H
COUNTO:    EQU     50H
           END
```

Second program unit loaded (UNIT—#2)

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


           END
```

Third program unit loaded (UNIT—#3)

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

           END
```

"START" X

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

"TMDLYL" U

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

S (Save) Command

The S command saves the object program generated in the link area by the linker with its filename, execution address, and load address in an output file.

＊LS **Filename? SAMPLE** [CR] **From? 3000 To ? 3A00** **Load ? 12A0 Execute ? 12A0**	Saves the object program in the link area block starting at address 3000H and ending at address 3A00H. The filename is "SAMPLE," the load address is 12A0H, and the execution address is 12A0H.

— Enter an **S** command in response to the prompt "＊L."

— The system displays the message "**Filename?**" on the next line to prompt for the name of file to be created.

— Enter the filename and press [CR].

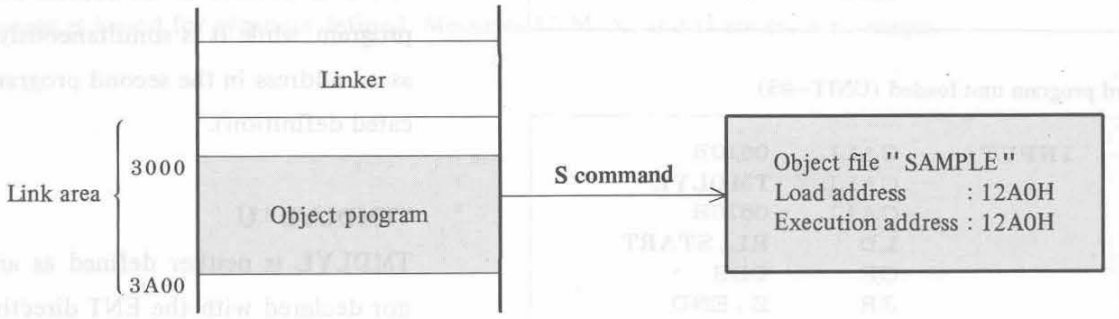
— The system displays the message "**From?**" on the next line and waits for the operator to enter the starting address of the object program in the link area with a 4-digit hexadecimal number. The system then prompts for the ending address with the message "**To?.**"

— After the block of memory to be saved is specified, the system prompts for the load and execution addresses.

The load address is the address in memory at which the object program is to be loaded for execution, and the execution address is the address to which control is to be transferred (i.e., the value that the program counter is to assume) after the object program is loaded.

```
*LS
Filename?SAMPLE
From? 3000 To? 3A00
Load? 12A0 Execute? 12A0
Writing SAMPLE
OK
*L
```

— The system starts saving the object program after the execution address is specified.



— The system displays the message "OK" after the object program has been saved.

— Press [BREAK] to interrupt the save operation.

V (Verify) Command

The V command compares the contents of the object file whose filename is specified with the contents of the link area.

＊LV	Compares the contents of the object file with the contents of the link area.

— Enter a V command in response to the prompt "＊L."

— The system then prompts for the name of the object file to be compared with the message "**Filename?.**"

— Enter the filename and press [CR].

— The system displays the starting and ending addresses which were specified in the S command when the object file was generated, followed by the message "**Top?.**" This message asks the operator to specify the address in the link area at which comparison is to start. Specify the address that was specified in the "From" clause of the S command.

When no filename is specified, the system performs the same operations for the first object file that is encountered.

— After the starting address is specified, the system starts comparing the contents of the link area and the object file.

— The system displays the message "OK" if the contents of the link area and object file match, and the error message "Check sum error" if they do not match.

— Press [BREAK] to terminate the compare operation.

— The photo on the right shows how the object file "SAMPLE" is verified.

X (data TRANSfer) Command

The X command moves the specified memory block to another specified memory area.

＊LX

From? 0000 To? 12A0 Top? 3000 To 42A0

Moves memory block 0000H ~ 12A0H to the memory area starting at 3000H.

— Enter an X command in response to the prompt "＊L."

— The system displays the message "**From?**" and waits for the operator to enter the starting address (a 4-digit hexadecimal number) of the memory block to be moved. The system then displays the message "**To?**" to prompt for the ending address (a 4-digit hexadecimal number).

The system then prompts for the starting address (a 4-digit hexadecimal number) of the memory area to which data is to be transferred with the message " **Top?.** " When the operator enters the starting address of the destination area, the system computes and displays the ending address of the destination area and starts the data transfer. The system returns to the command wait state after the data transfer is completed.

— The destination memory area must be within the link area.

＊ (CLEAR bias and table) Command

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

— Enter a ＊ command in response to the prompt " ＊L."

— The system resets the assembly bias and link address to 0000H and clears the symbol table. But the starting address of the symbol table is not affected.

Command

＊L#	Switches the printer list mode on or off.

— Enter a # command in response to the prompt " ＊L. "

— The system switches the printer list mode on or off. The printer list mode is disabled when the linker is invoked. The # command switches the list mode each time it is issued. In the printer list mode, all output is directed to both the CRT display and the printer.

! Command

＊L!	Transfers control to the monitor.

— Enter a ! command in response to the prompt " ＊L " and press CR .

— The system displays the message:

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

Enter M to transfer control to the monitor.

Enter B to transfer control to the IPL program.

Enter C to cancel the ! command and return control to the linker.

— There are two methods of returning to the linker mode from the monitor:

1) **Jump to address 12A0H : The link area is cleared (cold start).**
2) **Jump to address 12A3H : The link area is not cleared (warm start).**

3.6 ERROR MESSAGES OF THE LINKER

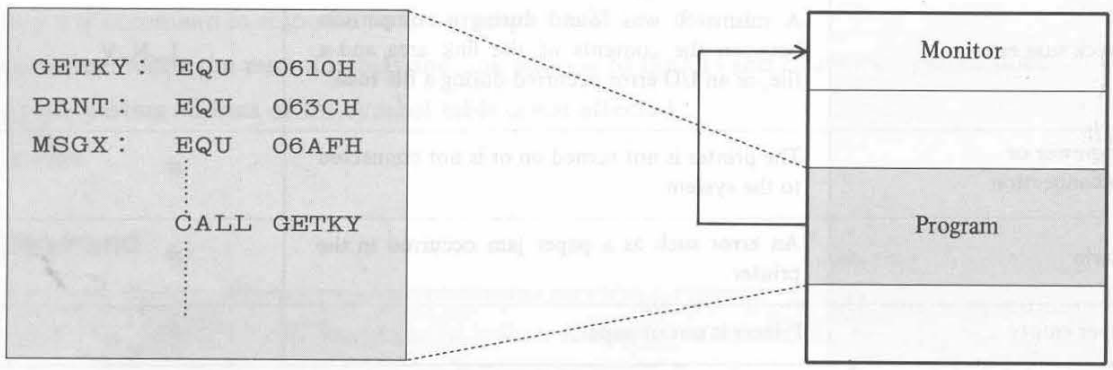
Error message	Meaning	Relevant commands
? ? ?	The specified address was outside the link area or the load address value was updated beyond the link area during a load operation.	L, N, S, X
Invalid	The format of the specified command is invalid. (Examples) * LL 12A0 [CR] The link address is missing. * LL 12 [CR] Fewer digits than required were specified.	L, S, V, X
Check sum error	A mismatch was found during a comparison between the contents of the link area and a file, or an I/O error occurred during a file read.	L, N, V
No power or no connection	The printer is not turned on or is not connected to the system.	#
Alarm	An error such as a paper jam occurred in the printer.	#
Paper empty	Printer is out of paper.	#

3.7 HOW TO USE MONITOR SUBROUTINES

The subroutines in the monitor program may be used to construct programs in assembly language. There are two methods of using monitor subroutines.

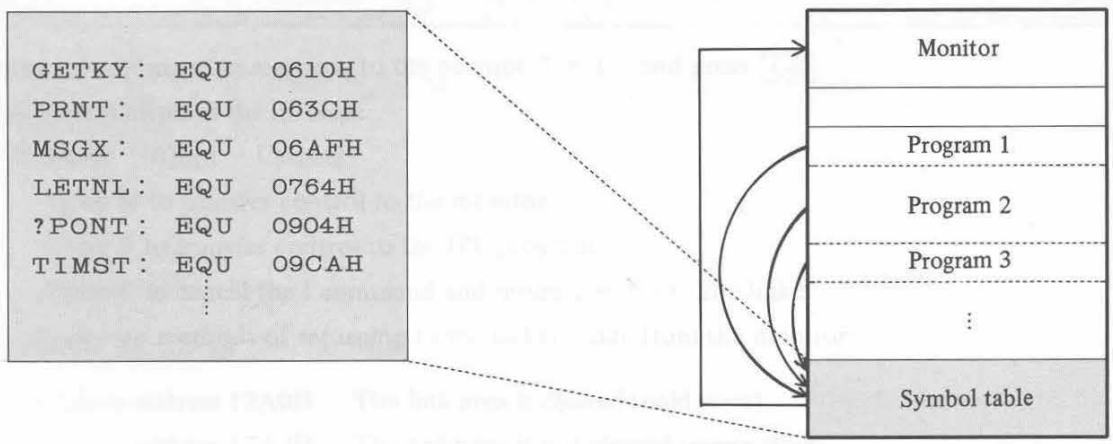
(1) When there are only a few programs to be linked

Declare all monitor subroutines (e.g., GETKY, PRNT, etc.) at the beginning of the programs in which they are to be referenced and call them by name; for example, "CALL GETKY" (see the figure below). Although using addresses instead of subroutine names does not cause errors, it is recommneded that monitor subroutines be referenced by name as shown below to improve readability and maintainability.



(2) When there is a substantial number of programs to be linked

Using method (1) in this situation will be inefficient because of duplications and redundancies. One way of alleviating this inefficiency is to extract all monitor subroutine declarations from all programs referencing them to form a program consisting of monitor subroutine declarations, then to link this program to the others. In this case, it is necessary to link the program unit containing EQU directives first (actually, such a program is not linked as an independent program but is absorbed into the symbol table as shown below).

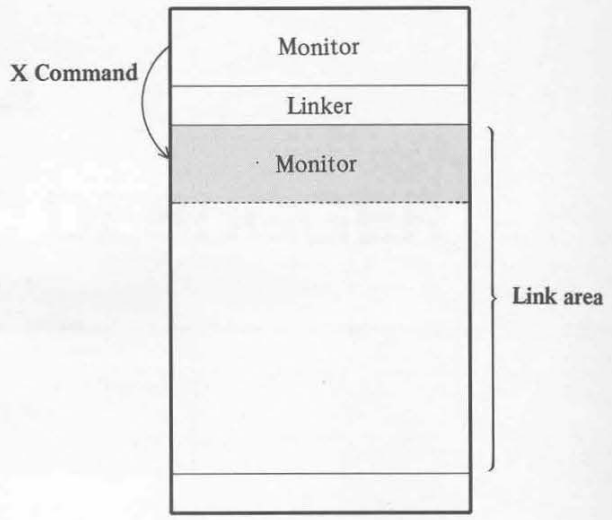


–Linking procedure–

The procedure for linking the monitor with a program using monitor subroutines and generating a file that can be loaded by the IPL is given below.

As an example, consider the problem of linking three program units with the monitor to form an object program.

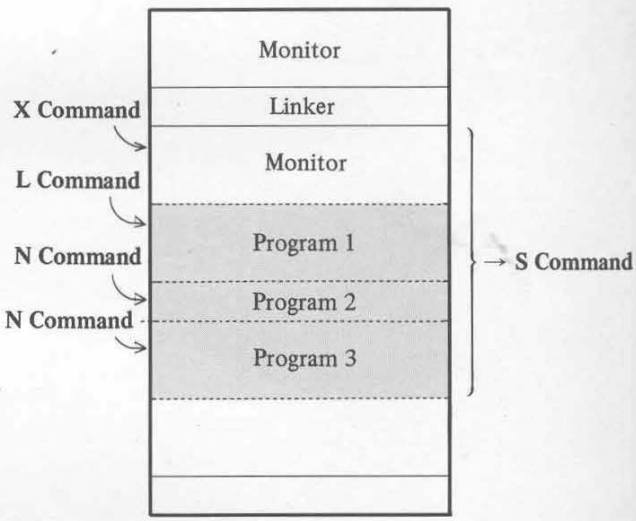
First, move the monitor (from addresses 0000H to 129FH) to the link area using the linker **X command.**



＊LX
From? **0000** To? **129F** Top? **3000**

Note that the current link address remains unchanged (confirm this with the H command).

Link and load program 1 with the **L command.** Specify 12A0H as the assembly bias when loading the program immediately following the monitor program. Also specify the link address which is equal to the starting address to which the monitor program has been moved plus 12A0H (normally, all that is required is to enter "L 12A0 42A0"). Then link and load programs 2 and 3 using **N commands.**



Finally, execute an **S command** to save the link area block from the beginning of the moved monitor program to the end of program 3 into an object file.

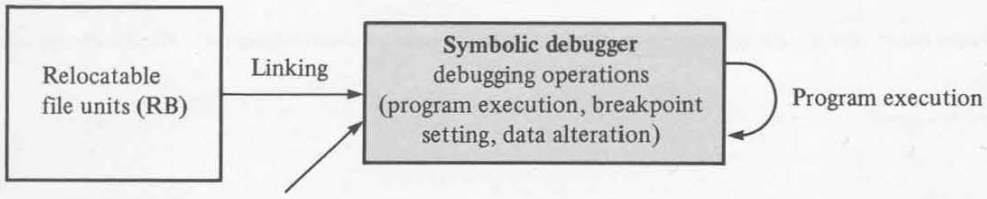
If the load address and execution address are specified as 0000H in this case, the program in the object file will be loaded into memory starting at address 0000H when it is loaded. When it is loaded, control is transferred to address 0000H; that is, to the beginning of the monitor. The monitor then initializes the monitor area transfers control to address 12A0H; that is, to the beginning of program 1. Program 1 execution then starts.

Care should be taken when the JP 0000H instruction (transferring control to the monitor) is used in a program. When the monitor is loaded in memory, the instruction "JP ST" at address 00AE has been resolved to "JP 12A0H" (see Appendix 14.3). Consequently, when control reaches address 00AEH after the monitor receives control, control is again passed to the beginning of program 1. To return to the monitor and enter the monitor command mode, it is necessary to replace this instruction with one which is appropriate so that execution continues at address 00B1H (for example, change C3A012 to 01A012 〈LD BC, 12A0H〉).

CHAPTER 4
SYMBOLIC DEBUGGER

4.1 OUTLINE OF THE SYMBOLIC DEBUGGER

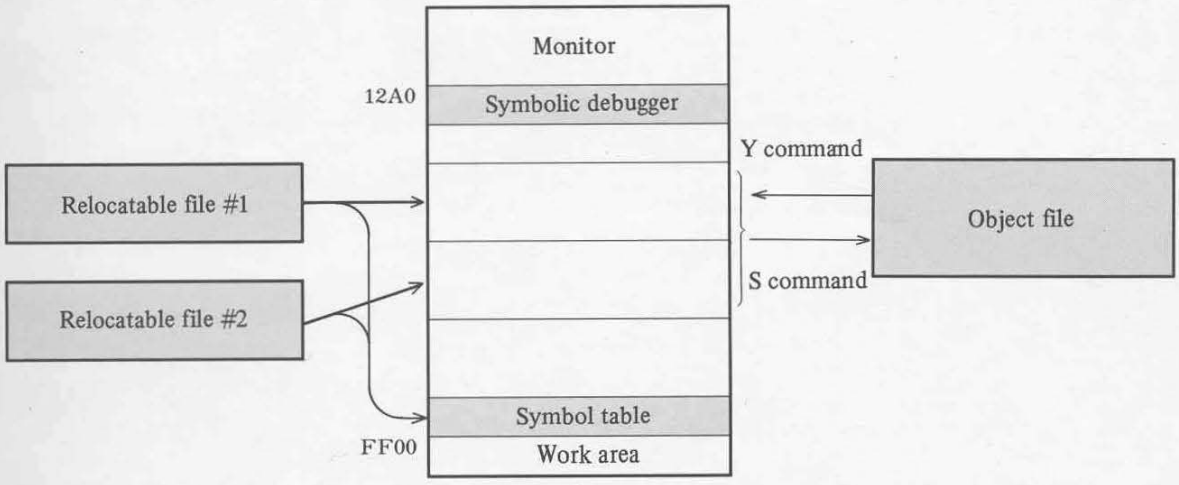
The SHARP MZ-80B symbolic debugger links and loads one or more program units from relocatable files to form an object program in memory in an immediately executable form and runs the object program for debugging. It provides the programmer with facilities for taking a memory dump of the object program 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).



Debugging with the symbolic debugger

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

When errors are detected during program debugging, it is necessary to reedit the source program after the debugging session. After the debugging of all source program units is completed, the final object program can be obtained using the linker. The symbol table can be set up in the same manner as with the linker.



Symbolic debugger file processing

–Symbolic debugger command table–

Command type	Command name	Function
Link/load and symbol table commands	L	Loads a relocatable file into the link area. The program in the relocatable file is loaded to form an object program through relocation at the location designated by the assembly bias and link address (relocate Load).
	N	Appends a relocatable file to the end of the preceding program in the link area (Next file).
	H	Displays the current values of the assembly bias and link address (Height).
	T	Displays the contents of the symbol table. Each table entry consists of a label symbol name, its absolute address, and its definition status (Table dump).
	*	Clears the symbol table and current assembly bias and link address values to 0000H (CLEAR bias and table).
Debugging commands	B†	Displays, sets or alters a breakpoint. (Breakpoint)
	&	Clears all breakpoints set. (CLEAR breakpoint)
	M†	Displays the contents of the specified block in the link area in hexadecimal representation or alters them. (Memory dump)
	D†	Displays the contents of the specified block in the link area in hexadecimal representation with one instruction on a line. (memory list Dump)
	W†	Writes hexadecimal data, starting at the specified address in the link area. (data Write)
	G†	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 filename into memory. (Yank)
	V	Compares the file whose filename is specified with the contents of the link area. (Verify)
Special commands	#	Switches the printer list mode for listing printout.
	!	Transfers control to the monitor.

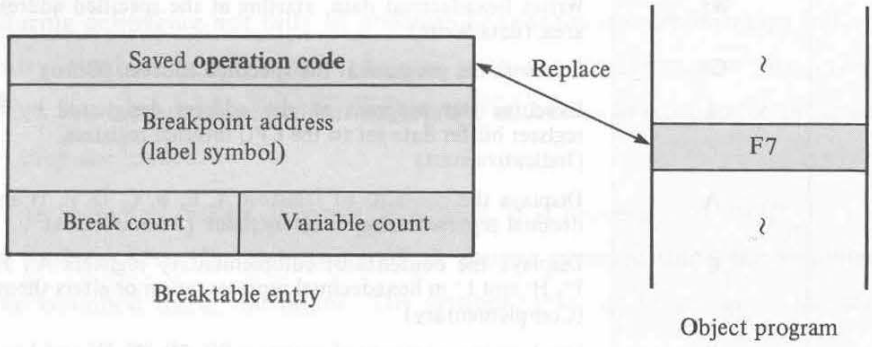
Note: Commands marked by a dagger permit symbolic operations.

4.2 BREAKPOINTS

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

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

When a breakpoint is set in a program, the debugger saves the operation code at that location (address) in the break table and replaces it with code 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.

4.3 SYMBOLIC DEBUGGER COMMANDS

–Link/load commands–

L (relocate Load) Command

The L command loads a relocatable file generated by the assembler into the link area in memory. The operator must specify the assembly bias in this command, taking into consideration the fact that the symbolic debugger presumes that any program loaded under symbolic debugger control is executable.

＊DL 3000 3000	Loads a relocatable file into memory with an assembly bias of 3000H and a link address of 3000H.

— Enter an L command in response to the prompt "＊D" ("＊D" is a prompt for a debugger command).

— Enter assembly bias and link address values as 4-digit hexadecimal numbers. The debugger will create an immediately executable object program in the link area as explained in Section 3.3. Normally, the assembly bias and link address should be set to **the same address** in the link area as shown above. If the corresponding source program contains an ORG directive, the specified assembly bias is ignored.

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

— Enter the correct filename, then press [CR] . The system searches for the specified file and reads it into the link area. If no filename is specified, the system reads in the first file that is encountered.

— The system displays the message "Check sum error" if an error occurs during the read.

— Press [BREAK] to terminate the file read.

— The photo at right shows how relocatable file "FORMULA#1" is loaded into the link area at address 3000H.

N (Next file) Command

The N command links and loads the next relocatable file in the location determined by the current assembly bias and link address values (which can be verified with the H command).

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

— Enter an N command in response to the prompt "＊D."
— The system then prompts for the name of the file to be read in with the message " **Filename?.** "
— Enter the required filename and press [CR] . The system searches for the specified file and reads it into the link area. If no filename is specified, the system reads in the first relocatable file that is encountered.
— The system uses the assembly bias and link address values set up immediately before the N command is executed when loading the relocatable file. If the source program contains an ORG directive, it takes precedence over the assembly bias value.
— The program is appended and linked during loading.
— The system issues the error message " Check sum error " if an error occurs during program loading.
— Press [BREAK] to terminate program loading.

H (Height) Command

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

— Enter a H command in response to the prompt "＊D."
— The system then displays two 4-digit hexadecimal numbers which indicate the current values of the assembly bias and link address. These values connot be manually changed.

–Symbol table command–

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 and 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.

＊(CLEAR bias and table) Command

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

— Enter a ＊ command in response to the prompt "＊D."

— The system resets the assembly bias and link address to 0000H and clears the symbol table. But the starting address of the symbol table is not affected.

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
```

Second program unit loaded (UNIT–#2)

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

          END
```

Third program unit loaded (UNIT–#3)

```
INPUT:    CALL    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
```

"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).

"TMDLYL" U

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

B (Breakpoint) Command

The B command sets or changes a breakpoint. 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 displacement applied to the label symbol (" +5L " in example 3 and " −9 " in example 4 below) must be a decimal number from 1 to 65535 in line or from ±1 to ±65535 in byte.

＊ DB addr count	Sets a breakpoint.
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 a **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.

— **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 attemps 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 [DEL] while setting breakpoints. When [CR] is pressed, the debugger is returned to the command wait state.

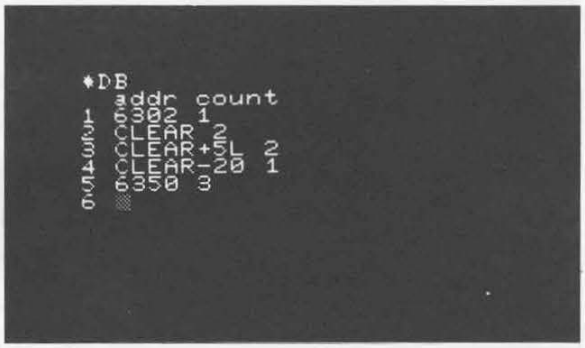
& (CLEAR breakpoint) Command

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

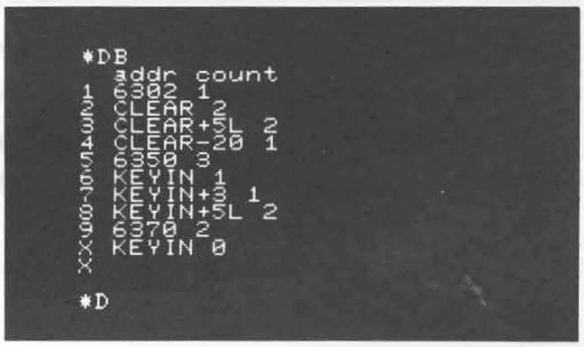
— Enter the & command in response to the prompt "＊D."

— The debugger clears all breakpoints set and waits for entry of a new command.

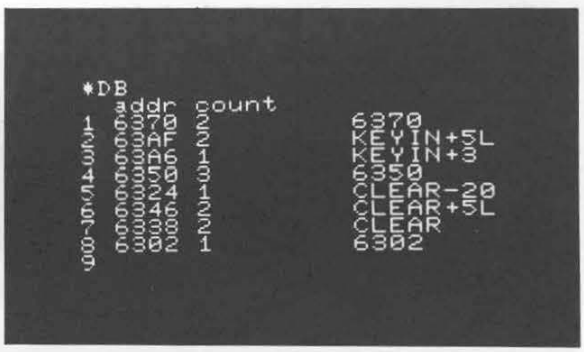
— The photo at right shows an example of setting breakpoints. The breakpoints are set with a 4-digit hexadecimal number (absolute address), a global label symbol, a label symbol plus a line specification and a label symbol plus a byte displacement.



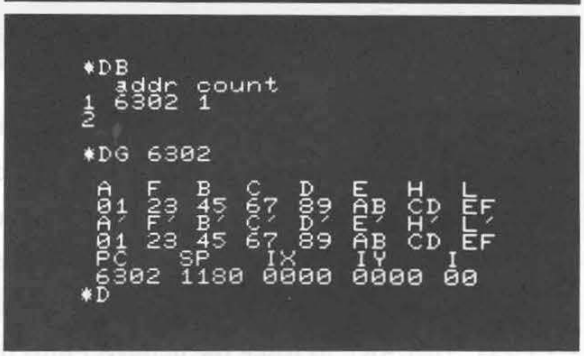
— The photo at right shows that breakpoint "KEYIN" has been cleared on the line identified by "X".



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



— The photo at right shows that a break occurred immediately when the program was executed from address 6302 with a G command with a breakpoint at 6302 and a count of 1. As soon as a breakpoint was taken, an R command was executed to display the status of the CPU registers.



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 an **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 SPACE to suspend display of the memory dump. To resume display, press SPACE again. Press SHIFT to slow down the speed of display.

— Press BREAK 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.

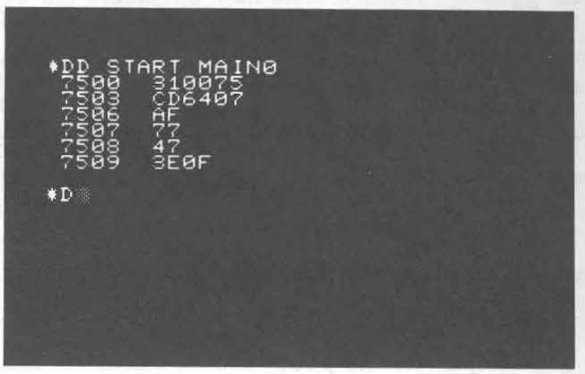
＊ DD 7800␣7850 CR	Displays the contents of the memory block from addresses 7800 to 7850 with one instruction on a line.
＊ DD START␣MAIN0 CR	Displays the contents of the memory block from the addresses identified by "START" to the address identified by "MAIN0" with one instruction on a line.
＊ DD 7500␣START+12L 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 a 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 CR 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 [SPACE]. Press [SHIFT] to slow down the speed of display.

— The D command does not allow memory alteration; after the memory dump is completed, the debugger is returned to the command wait state.

— Press [BREAK] 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 a W command in response to the prompt "＊ D."

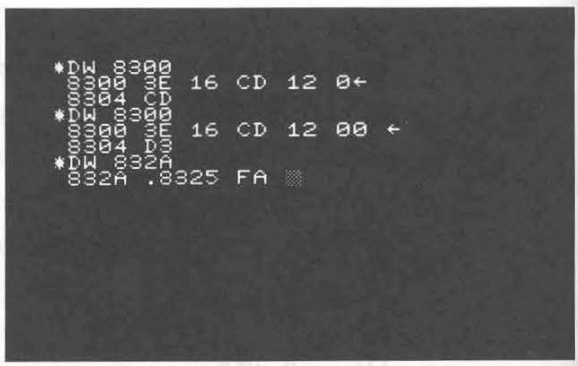
— The debugger displays the cursor with a space between it and the letter W, then waits for the programmer to enter the starting address of the memory area to be written.
The programmer may specify the memory block starting address with a 4-digit hexadecimal number or a global symbol.

— 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 [CR] 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 [←] to return the cursor to the byte of data just entered and correct it. The photo at right shows an example. As the photo shows, when [←] 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.

— To execute the program and return to the symbolic debugger at a certain point, use the following command.

JP 12A3H

Address 12A3 is the warm start address for the debugger; at this address, "＊D" is displayed to prompt for command entry without the contents of the link area being lost. (If a start is made from address 12A0, it is a cold start and the link area, symbol table, and bias are cleared.)

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

— Press [BREAK] 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
  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? ■
```

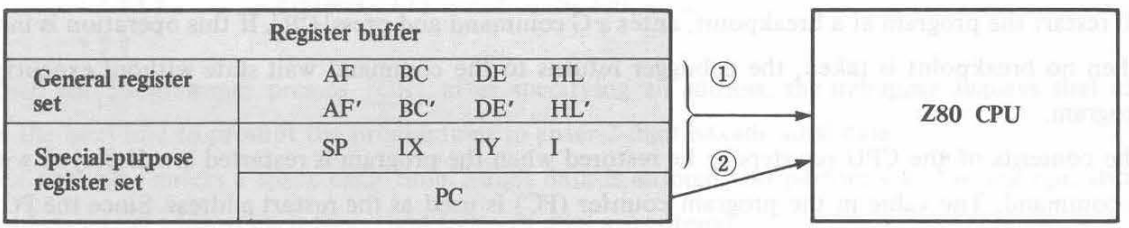
Executes the program at the address designated by the program counter with the data shown on the screen loaded in the CPU registers.

— Enter an 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.

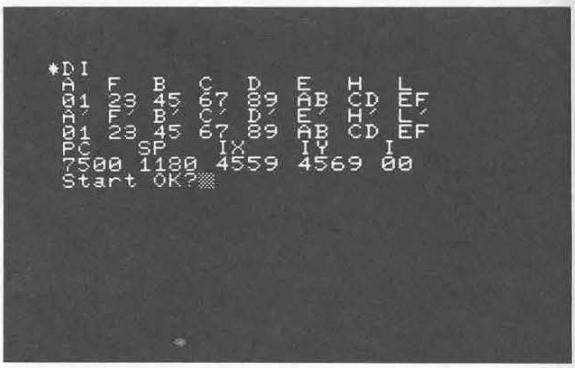
— The debugger then displays message "Start OK?." To start the program in this environment, press CR. **The debugger then executes the program, starting at the address designated in the program counter.** To change register values or terminate the I command, press BREAK ; the debugger then returns to 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.)



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 an 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 hexadeciaml 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.

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

The register values displayed 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] ; 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 a 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 [CR] ; 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
  PC     SP     IX     IY     I
  78AB   1FEA   5F70   5F50   00
```

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

— Enter a **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.

Register values displayed or altered through cursor manipulation are restored to 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] ; 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
  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
```

Displays the contents of all CPU registers.

— Enter an **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.

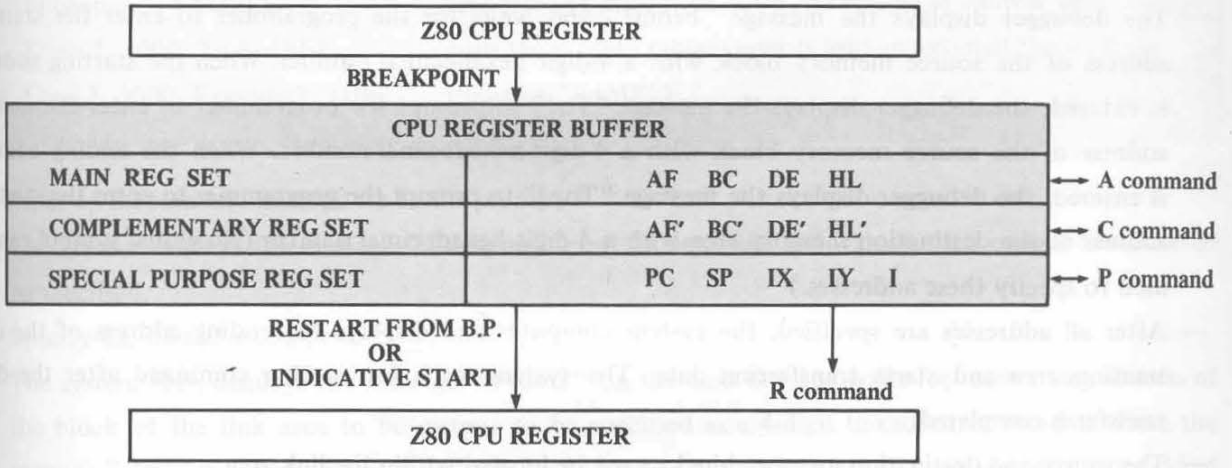
— The debugger enters the command wait state after displaying all the register contents.

— The above display is on 1 line in the 80 characters per line mode.

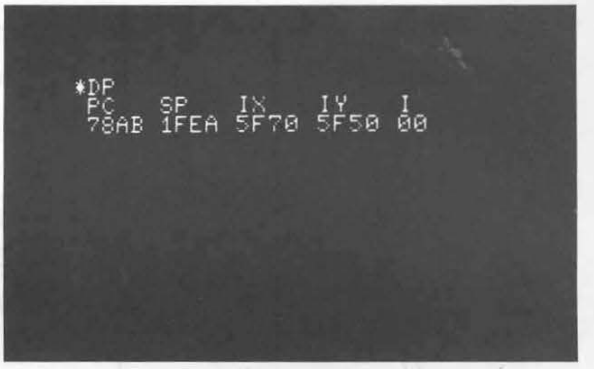
Using register commands A, C, P and R

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.

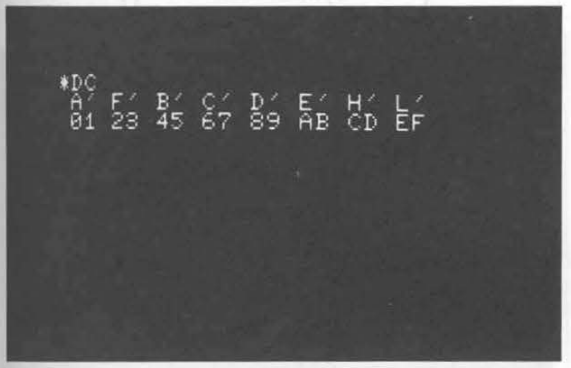
The figure below shows the relationship between the CPU registers and the register commands; the photos show examples of use of the register commands.



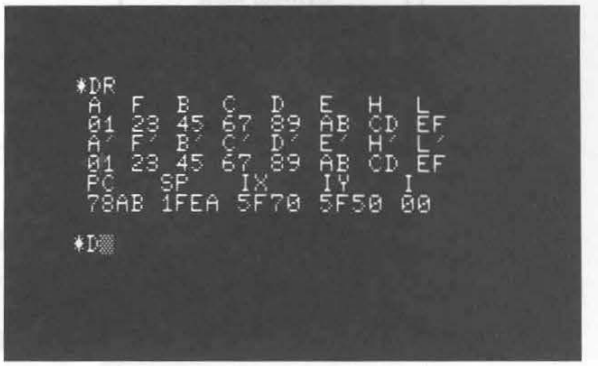
A command



P command



C command



R command

X (data TRANSfer) Command

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

＊ DX **From? 7500 To? 811F Top? 9000 To 9C1F**	Transfers the contents of the memory block from addresses **7500** to **811F** to the memory area starting at address **9000.**

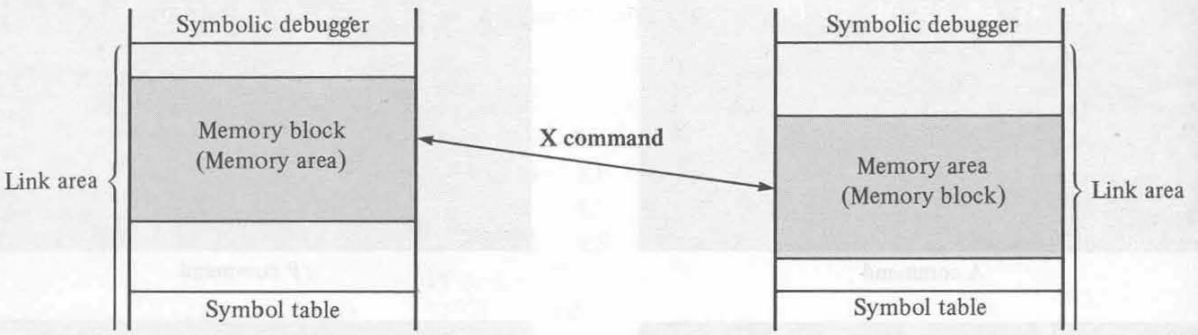
— Enter an **X** command in response to the prompt " ＊ D."

— 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 (No global symbol can be used to specify these addresses.)

— After all addresses are specified, the system computes and displays the ending address of the destination area and starts transferring data. The system waits for another command after the data transfer is completed.

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

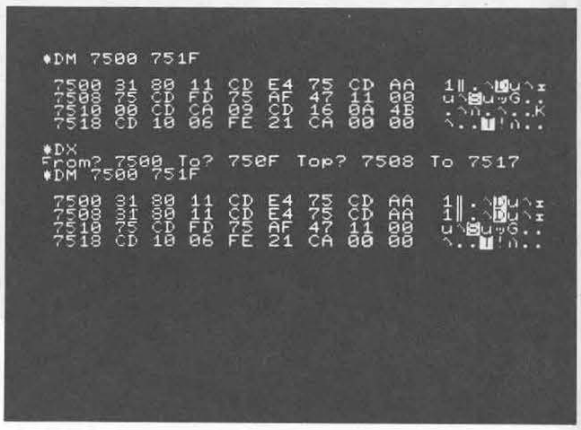
— Data trasfer is accomplished successfully even if the source and destination memory blocks overlap as shown below. The memory block shown in the figure at left may be transferred to the memory block shown in the figure at right and vice versa.



— 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.



–Object file I/O command–

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.

＊ DS Filename ? SAMPLE From ? 3000 To ? 4BFF Load ? 3000 Execute ? 3100	Saves the immediately executable object program in the link area starting at address 3000 and ending at address 4BFF along with the symbol table contents into an external file with the filename "SAMPLE."

— Enter an S command in response to the prompt " ＊ D."

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

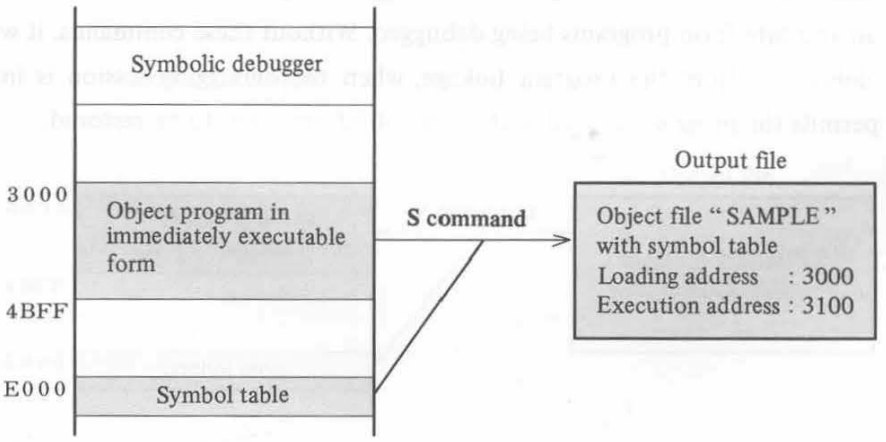
— Specify the filename and press [CR] .

— The system then displays the message " **From?** " on the next line and waits for the starting address of the block of the link area to be output to be specified as a 4-digit hexadecimal number. Then the message " **To?** " is displayed and the system waits for the end address of the same block to be specified as a 4-digit hexadecimal number.

— The system then displays the message " **Load?** " on the next line and waits for the loading address of the block to be specified as a 4-digit hexadecimal number. Next, the message " **Execute?** " is displayed and the system waits for the execution address to be specified in the same manner. (For details, see page 60.)

— After the four addresses indicated above have been specified, the specified memory block and symbol table contents are output to the external file (object file with symbol table).

— The figure below shows how the S command is used to specify the block from 3000 to 4BFF for output to its output file with the filename "SAMPLE," a loading address of 3000, and an execution address of 3100 specified.



Y (Yank) Command

The Y command reads a named object file with a symbol table into the link area.

＊DY	Clears the link area and reads an object file with a symbol table into the location determined when the file is created.

— Enter a **Y** command in response to the prompt "＊D."

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

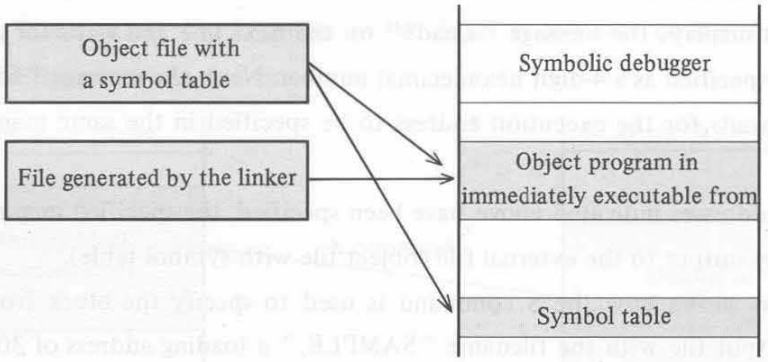
— Enter the name of the required file and press [CR]. The system then searches for the specified file and reads it into memory. If no filename is specified, the system reads the first object file with a symbol table that is encountered.

— The program in the object file is loaded as is between the starting and ending addresses that were specified when the file was saved with the S command. The symbol table in the object file is reproduced in memory under the conditions set up when the S command was executed, except that it is placed at the beginning of the area set up with the ＊TBL command. The debugger will read in only object files generated by the linker since they contain no symbol table.

— The system displays the message "OK" when the file read is completed.

— The system issues the error message "Check sum error" if an error occurs during the file read.

— Press [BREAK] to terminate the read.



— The S, V, and Y commands of the symbolic debugger are provided to facilitate creation of external files from an absolute-form programs being debugged. Without these commands, it would be necessary to restart debugging from the program linkage, when the debugging session is interrupted. The Y command permits the program, as well as the symbol information, to be restored.

V (Verify) Command

The V command compares the contents of the specified object file with symbol table with the contents of the link area.

＊ DV **Filename? SAMPLE**	Compares the contents of the object file with symbol table indicated by filename "SAMPLE" with the contents of the link area.

— Enter a **V** command in response to the prompt "＊ D."

— The system then prompts for the name of the object file to be compared with the message "**Filename?.**"

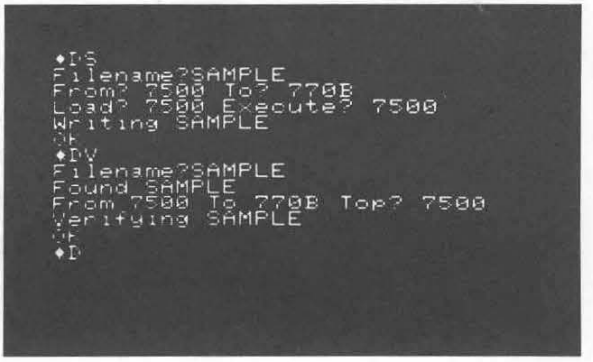
— Enter the required filename and press [CR] .

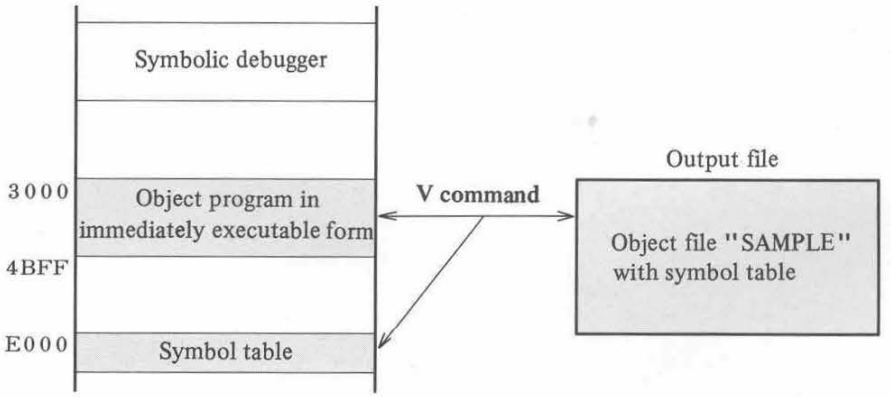
— The system displays the starting and ending addresses which were specified in the S command when the file was generated, followed by the message "**Top?.**" This message asks the operator to specify the address in the link area at which comparison is to start. Specify a 4-digit hexadecimal number. When no filename is specified, the system performs the same operations on the first object file that is encountered.

— After the starting address is specified, the system starts comparing the contents of the link area and object file.

— The system displays the message "OK" if the contents of the link area and object file match, and the error message "Check sum error" if they do not match.

— The photo at right shows how the object file "SAMPLE" with symbol table is compared with the contents of the link area. The "OK" message indicates that the contents of the link area have been copied into an output file without any errors.





Command

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

— Enter a # 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).

— The system issues the error message "No power or no connection (Printer)" if the printer is not turned on or is not connected to the system.

— The system issues the error message "Alarm" if an error occurs in the printer.

— The system issues the message "Paper empty" when the paper has run out.

! Command

＊D!	Transfers control to the monitor.

— Enter an ! command in response to the prompt " ＊ D."

— The system displays the message:

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

Enter M to transfer control to the monitor.

Enter B to transfer control to the IPL program.

Enter C to cancel the ! command and return control to the symbolic debugger command mode.

— There are two methods of returning to the symbolic debugger from the monitor:

1) Jump to address 12A0H : The link area is cleared (cold start).

2) Jump to address 12A3H : The link area is not cleared (warm start).

4.4 ERROR MESSAGES OF THE SYMBOLIC DEBUGGER

Error message	Meaning	Relevant commands
? ? ?	An attempt was made to access a location outside the link area.	B , W , X , S , V
Error	An incorrect number of digits was specified or a digit other than a hexadecimal digit was entered during execution of a register (or memory) change command.	M , A , C , P
RST6?	A break point was set at an RST6 instruction.	B
Over	More than nine breakpoints were set.	B
Invalid	The format of the entered command is incorrect.	X , S , V
?	○ An invalid symbol (undefined label symbol or nonlabel symbol) was specified. ○ An attempt was made to clear a break point which was not set. ○ An attempt was made to set the break counter more than 14 (E in hexadecimal) times. ○ The format of the specified address is incorrect. ○ The starting address is not smaller than or equal to the ending address. ○ The destination and source blocks overlap.	B , W B B M , D , G M , D , W X
Check sum error	○ A mismatch was found between the contents of the link area and the object file being verified. ○ An error occurred while a file was being read.	V L , N , Y
No power or no connection	The printer is not turned on or is not connected to the system.	#
Alarm	An error such as paper jam occurred in the printer.	#
Paper empty	Printer is out of paper.	#

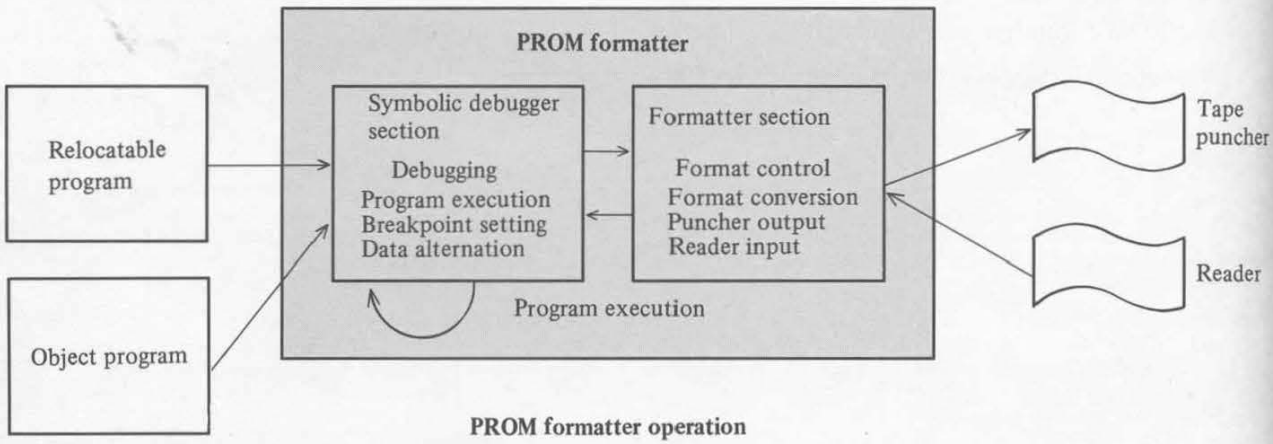
CHAPTER 5
PROM FORMATTER

5.1 OUTLINE OF THE 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 puncher after converting it to PROM writer format.



PROM formatter operation

A PROM writer is required to write programs into PROM devices. There are a number of PROM writers on the market (e.g., Takeda Riken, Minato Electronics, etc.). Those PROM writers, however, use different formats (in which the PROM writer reads input data from the tape reader). Programs debugged and completed by the symbolic debugger* section of the PROM formatter are converted to a format suitable for the PROM writer used and output to the tape puncher by the formatter section.

*: See Chapter 4 for the symbolic debugger.

The PROM formatter can also read in programs written in one format on paper tape and output them in a different format. During this format conversion, it is also possible to debug the programs and alter their data using the symbolic debugger section (except program execution and symbolic debugging).

The following formats are provided in the PROM formatter:

1. BNPF

- Britronics
- Intel
- Takeda Riken

2. B1OF

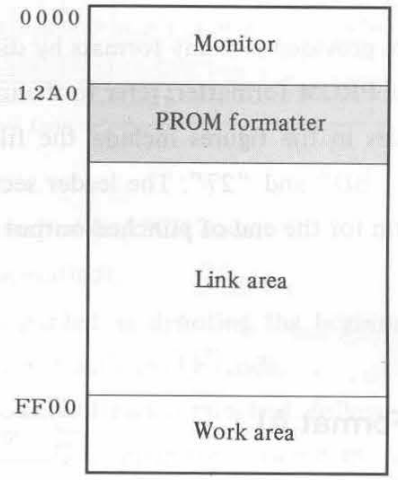
- Takeda Riken

3. HEXADECIMAL

- Britronics
- Takeda Riken
- Minato Electronics

4. BINARY

- Britronics



PROM formatter memory map

The PROM formatter commands are listed below. In addition to these commands, it is possible to use the symbolic debugger commands under the PROM formatter program.

Command name	Function
FP (Format Punch)	Punches a specified link area block on paper tape.
FC (parity Form Change)	Changes the parity of the input or output tape.
FR (Format Read)	Reads a formatted program from paper tape into the link area.
FM (Format Message)	Displays a list of the formats available for the PROM formatter.

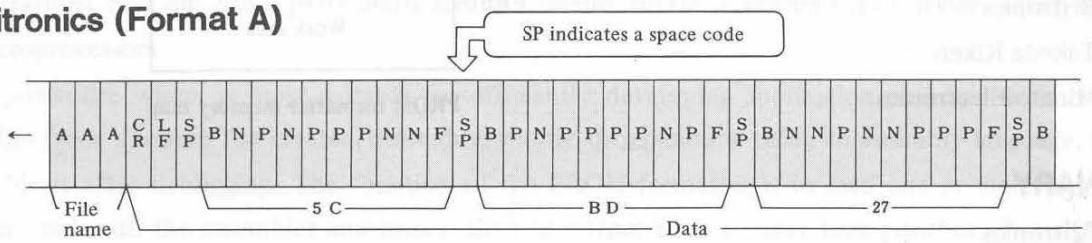
5.2 PROM WRITER FORMATS

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 filename "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.)

5.2.1 BNPF

a) Britronics (Format A)



— The filename is punched in ASCII code (if one is specified). (Using the character "B" as a filename 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.

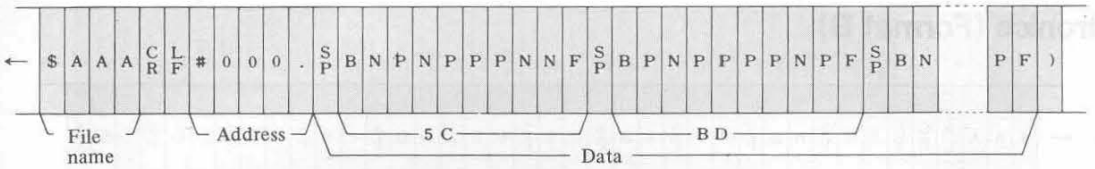
— [CR] and [LF] are punched after each 6 items of data are punched in the BNPF format.

— Punching is performed in BNPF format up to the address specified for "To? ."

b) 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.

c) Takeda Riken (Format E)



— The " $ " mark, which denotes the filename, is punched in ASCII code.

— The filename is punched in ASCII code (if one is specified).

— [CR] and [LF] are punched. The "$" mark is regarded as denoting the beginnning 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.

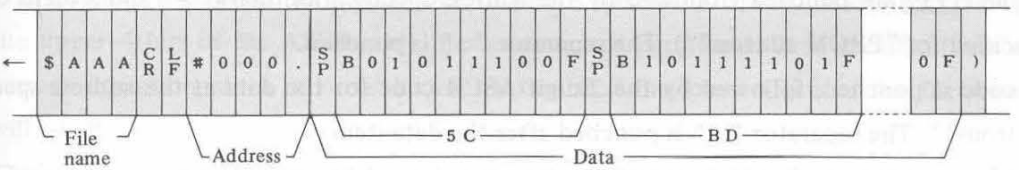
— [CR] and [LF] are punched after each 6 items of data are punched in the BNPF format.

— 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 filename is specified. (Otherwise, incorrect operation will result.)

5.2.2 B10F

a) Takeda Riken (Format F)

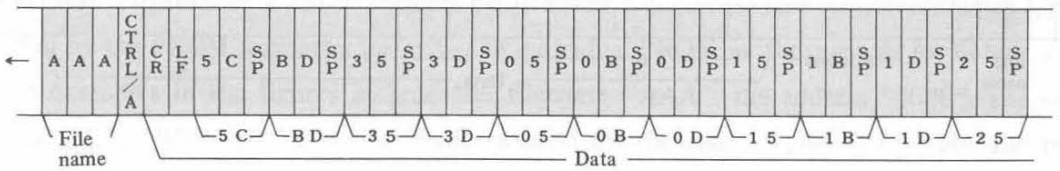


— Except for the NP section, this is the same as Takeda Riken's BNPF format.

— The B10F format corresponds to the BNPF format in that 1 = P and 0 = N.

5.2.3 HEXADECIMAL

a) Britronics (Format B)



— The filename is punched in ASCII code (if one is specified).

— The "CTRL/A" mark (01H) indicating the beginning of data is punched.

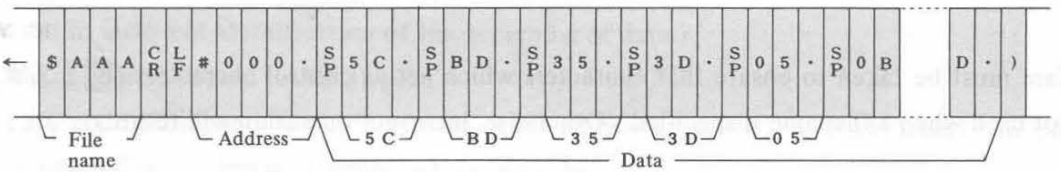
— [CR] and [LF] are punched.

— The data item at the address specified for "RAM from?" is punched as a 2-digit ASCII code, then a space code is punched.

— [CR] and [LF] are punched after 16 bytes of data have been punched.

— Data is punched up to the address specified for "To?."

b) Takeda Riken (Format G)



— The "$" mark, which denotes the filename, is punched in ASCII code.

— The filename is punched in ASCII code (if one is specified).

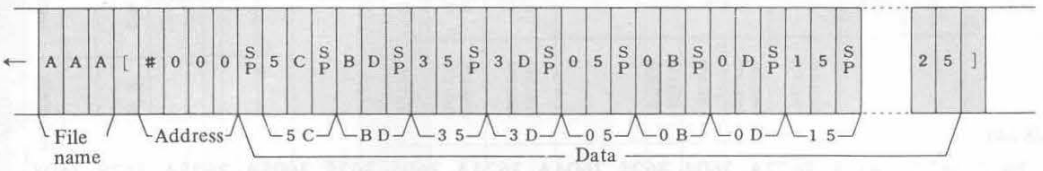
— After [CR] and [LF] are punched (followed by the address specification mark "#" and 3 digits of the address specified for "PROM address?"). The separator " . " is punched.

— The space code is punched, followed by the 2-digit ASCII code for the data at the address specified for "RAM from?." The separator " . " is punched after the data item.

— [CR] and [LF] are punched after 16 bytes of data have been punched.

— Data is punched up to the address specified for "To?", at which point the tape leader stop mark ")" is punched.

c) Minato Electronics (Format H)



— The filename is punched in ASCII code when filename 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?."

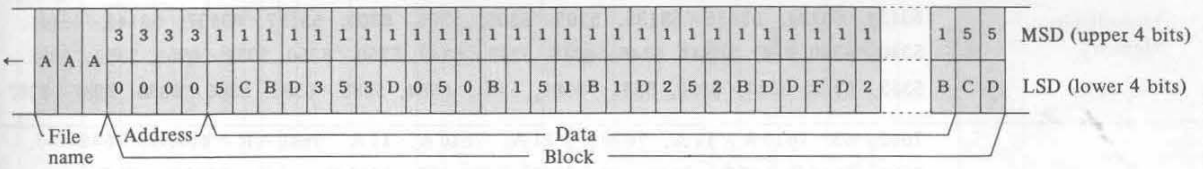
— A space code is punched, then the data at the address specified for "RAM from?" is converted to a 2-digit ASCII code and punched.

— 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?."

5.2.4 BINARY

a) 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 filename is punched in ASCII code (if one is specified). Specifications which result in a " 3 " in the upper 4 bits of the ASCII code filename are not permitted. Such specifications will result in incorrect operation, since incorrect determination that the lower 4 bits of the filename 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.

5.2.5. 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	3602A／22A, 3604A／24A, 3604AL／24L, 3605／25, 3608／28
Intersil	5600／10, 5603A／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, 7610A／11A, 7620A／21A, 7640A／41A, 7640AR／41AR, 7642／43, 7644, 7646R／47R, 7648／49, 7608, 7680／81, 7680R／81R, 7680P／81P, 7680RP／81RP, 7683, 7684／85, 7684P／85P, 7686／87, 7686R／87R, 7686P／87P, 7686RP／87RP
Fairchild	93417／27, 93436／36, 93438／48, 93452／52
National Semiconductor	54／74S387, 54／74S287, 54／74S470, 54／74S471, 54／74S570, 54／74S571, 77／87S295, 77／87S296, 54／74S473, 54／74S472, 54／74S572, 54／74S573
NEC	403D, 406D
Raytheon	29660／61, 29600／01, 29612／13
Signetics	82S114／115, 82S126／127, 82S130／131, 82S140／141, 82S136／137, 82S180／181, 82S2708, 82S184／185, 82S190／191
Texas Instruments	54／7488A, 54／74S／88, 54／74S288, 54／74S470, 54／74S71, 54／74S73, 54／74S72, 54／74S75

c) Minato Electronics

Adaptable to all PROMs.

d) Takeda Riken

MOS Type

Element	Bit configuration, capacity (words x bits = capacity)	Maker name											
		Oki Denki	Toshiba	NEC	Hitachi	Fujitsu	Mitsubishi	Intersil	Intel	Texas Instruments	Fairchild	AMD	SIGNE
MOS	256 x 8 = 2048			μPD454D⑩	HN351702A⑦	MB8503 MB8513	M5L1702⑦		1602A ⑦ 1702A			AM1702A⑦	1702A ⑦
	512 x 4 = 2048		TMM121C TMM121C-1										
	512 x 8 = 4096								2704 ⑧				
	1024 x 8 = 8192	MSM3758⑧	*TMM322C⑧	μPD458D⑩	HN462708⑧	MB8518⑧	M5L2708⑧		2708 ⑧ 2758 ⑨	TMS2708⑧	F2708 ⑧	AM2708 ⑧	2708 ⑧
	2048 x 8 = 16384		TMM323C⑨	μPD2716D⑨	HN462716⑨	MB8516⑨	M5L2716⑨		2716 ⑨	TMS2716⑧ TMS2516⑨			
	4096 x 8 = 32768								2732	TMS2532			
CMOS	512 x 8 = 4096							1M6654④					
	1024 x 4 = 4096							1M6653④					
	1024 x 8 = 8192						M58460 S						
Compound MOS									8755				

Bipolar Type

Element	Bit configuration, capacity (words x bits = capacity)	Maker name												
		NEC	Hitachi	Fujitsu	Mitsubishi	Intersil	Intel	Texas Instruments	Harris	Fairchild	Raytheon	MMI	AMD	SIGNE
Bipolar	32 x 8 = 256			MB7051 MB7056	M54730	1M5600 1M5610		SN74188A SN74S188 SN74S288	HM7602 HM76LS03 HM7603			6330-1 6331-1	AM27S18 AM27S19	82S23 82S123
	256 x 4 = 1024	μPB403D②		MB7052② MB7057②	M54700	1M5603A 1M5623	3601 3621	SN74S287 SN74S387	HM7610A HM7610 HM7611A HM7611	93417 93427	29662 29663 29660 29661	6300-1 6301-1	AM27S20 AM27S21	82S126 82S129
	256 x 8 = 2048							SN74S470 ③ SN74S471	HM7625R HM7629		29600 29601	6308-1 6309-1 6335-1 6336-1 63135-1		82S114
	512 x 4 = 2048			MB7053② MB7058②		1M5604 1M5624	3602 3622		HM7620 HM7620A HM7621 HM7621A	93436 93446	29612 29613	6305-1 6306-1	AM27S12 AM27S13	82S130 82S131
	512 x 8 = 4096	μPB405D μPB425D				1M5605 1M5625	3604A 3624A	SN74S472 SN74S473③ SN74S474 SN74S475	HM7640 H M7640A HM7641 HM7641A HM7640AR HM7641AR HM7647R HM7648 HM7649	93438 93448		6341-1 6340-1 6348-1 6349-1	AM27S15 AM27S26 AM27S27	82S115 82S140 82S141
	1024 x 4 = 4096	μPB406D μPB426D⑤		MB7054⑤ MB7059⑤		1M5606 1M5626	3605A 3625A		HM7642 HM7642A HM7643 HM7643A HM7644 HM7644A HM7642P HM7643P	93452 93453		6350-1 6351-1 6352-1 6353-1	AM27S32 AM27S33	82S136 82S137
	1024 x 8 = 8192	μPB417D μPB427D⑥		MB7055⑥ MB7060⑥			3608A 3628A		HM7680 HM7681 HM7680P HM7681P HM7680R HM7681R HM7680RP HM7681RP HM7608 HM7683	93450 93451		6380-1 6381-1 6384-1 6385-1 6386-1 6387-1 63133-1		82S180 82S181 82S2708
	2048 x 4 = 8192								HM7684 HM7685 HM7684P HM7685P HM7686 HM7687 HM7686P HM7687P HM7686R HM7687R HM7686RP HM7687RP					82S184 82S185
	2048 x 8 = 16384						3636		HM7616 HM76160 HM76161					82S190 82S191

Elements annotated with the same figures in circles can be used with the same performance boards.

5.3 PROM FORMATTER COMMANDS

The PROM formatter is a system program which is based on the symbolic debugger. See Chapter 4 for the commands available with the PROM formatter symbolic debugger section. This section describes the commands available for the formatter section.

—Formatting commands—

FP (Format Punch) Command

The FP command punches the contents of the specified relocatable program or data block from the link area onto paper tape in the specified format. When this command is to be used for a relocatable program, all breakpoints set in the specified block must be cleared.

＊ **PFP** **no parity** **Filename? PROM #1** **Port? FC** **Format? C** **From? 5000 To? 5100** **PROM address? 0000**	Outputs the contents of the object program from addresses **5000H** to **5100H** of the link area onto paper tape in format C. The PROM load address is specified as **0000H** and the filename as **PROM#1**.

— Enter an **FP** command in response to the prompt "＊ P " ("＊ P" represents the PROM formatter).

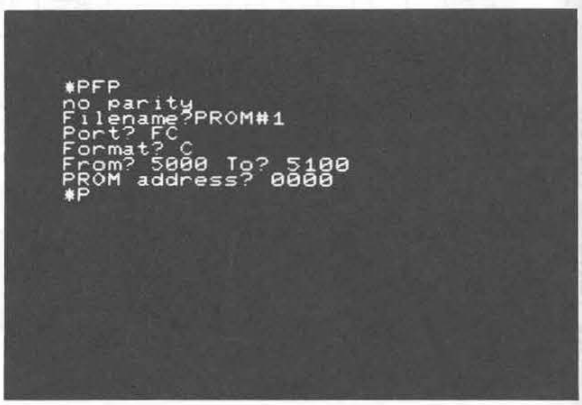
— The system prompts for the name of the file to be output with the message " Filename?." Press [CR] when no filename is required.

— Enter the required filename and press [CR] .

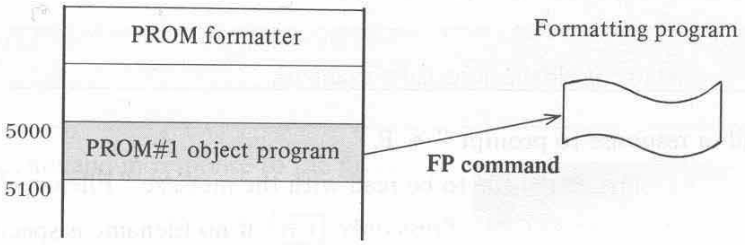
— The system displays the message "Port?" and waits for the port number of the interface to which the paper tape puncher is connected. Specify it as a 2-digit hexadecimal number. The port must be a output data Port. The associated control port number is the data port number plus 1. The PROM formatter accepts only even port numbers to prevent entry of invalid port numbers.

— The system then dispalys the message " Format?" and waits for format command A to H. Enter the required format command and press [CR] . The format commands are described later.

— The system displays the message " From?" and waits for the starting address of the object program to be punched out. Specify a 4-digit hexadecimal number. The system then displays the message "To?" and waits for the ending address.



— Depending on the format command specified, the system prompts for a PROM load address with the message "PROM address?." Specify the required address with a 3- or 4-digit hexadecimal number.
— After completing the punch operation, the system waits for another PROM formatter command with the prompt "＊P."
— Press BREAK to interrupt the punch operation.
— The photo shows that the program block from addresses 5000H to 5100H has been punched out through port FCH in format C with the filename "PROM#1."



— The formatter program will return to the command wait state after examining the punch status if no punch is connected, if the puncher is not turned on, or if an invalid port is specified. Press BREAK if control is not restored within a few seconds.
— It is recommended that the contents of the output file be verified with the FR command.

FC (parity Form Change) Command

The FC command changes the parity of the tape to be read with the FR command and of the tape to be punched. There are three types of parity specifications (no parity, even parity, and odd parity).

＊PFC even parity	Specifies that even parity is to be used for parity checking.

— Enter an **FC** command in response to the prompt "＊P."
— The system displays the current parity scheme, which switches cyclically (in the order even parity, odd parity, and no parity) each time the FC command is entered.
— The system then displays a list of format commands A to H and returns to the command wait state.

FR (Format Read) Command

The FR command reads a program on a formatted tape into the link area from the reader. After the program read is completed, the ending address of the program is displayed.

＊ **PFR** **no parity** **Filename? PROM#2** **Port? FC** **Format? C** **From? 5000**	Clears the link area and reads program "**PROM#2**", stored on tape in format **C**, from the reader into the link area starting at address **5000H**.

— Enter an **FR** command in response to prompt "＊P."

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

— Enter the required filename and press [CR] . Press only [CR] if no filename is specified.

— The system then displays the message "Port?" and waits for the port number of the interface to which the reader is connected. Enter it as a 2-digit hexadecimal number. The port must be a input data port. The associated control port number is the data port number plus 1. The PROM formatter accepts only even port numbers to prevent entry of invalid port numbers.

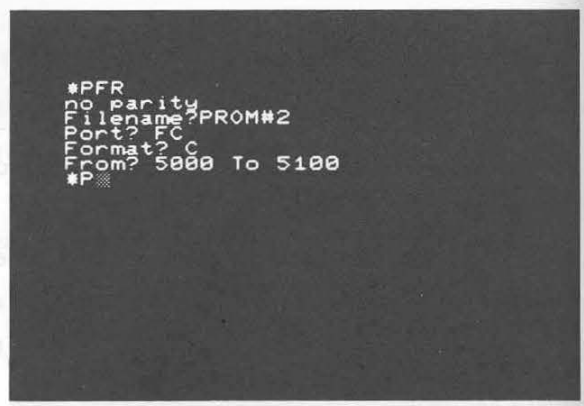
— The system then displays the message "Format?" and waits for a format command. Enter a format command (A to H) and press [CR] . If an incorrect format command is specified, the system displays the message "Format" and returns to the command wait state after having read the specified file.

— The system then displays the message "From?" and waits for the address at which the program is to be stored. Enter this as a 4-digit hexadecimal number; the system then starts reading the specified file. After completing the read, the system dispalys the message "To" followed by the ending address, then returns to the PROM formatter command wait state.

— Press [BREAK] to interrupt the file read operation.

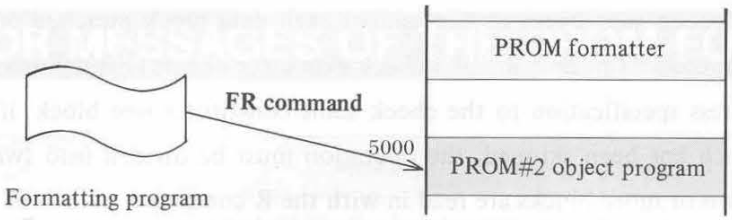
— The read program can be debugged using the symbolic debugger section of the PROM formatter. Symbolic debugging and program execution, however, are not allowed because no symbols are loaded in the symbol table.

— The photo at right shows how a program stored on tape in format C with the filename "PROM #2" is read into the link area (starting at address 5000H) from the reader through data port FC.



— The system returns to the PROM formatter command wait state if no reader is connected or no paper tape is loaded.

— Tapes punched with other than this PROM formatter program cannot be read because the PROM writer formats supported by this program contain a degree of redundancy.



FM (Format Message) Command

The FM command displays a list of formats provided by the PROM formatter.

＊ PFM	Displays a list of available formats.

— Enter an **FM** command in response to the prompt "＊ P."

The system then displays a list of available formats, as shown in the photo at right.

–Format commands–

Format commands are specified by the operator in response to the prompt message "Format?" during execution of the FP or FR command. Selecting one of these commands during execution of the FP 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 FR command will result in failure to correctly read the program into the link area.

A Command

— Used to specify the Britronics BNPF format. The control character "B" may not be used when the filename 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 filename is specified.

— During execution of the FP command, the system displays the message "PROM address?" to prompt for a PROM load address. Specify a 4-digit hexadecimal number.

— The system writes check sum bytes at the end of each data block punched out (FP command), or displays the error message " Check sum " if a check sum error occurs (FR 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 filename.

E Command

— This command is used to specify the Takeda Riken BNPF format. The character " B " may be used in the filename.
— A file is a block which begins with " $ " and ends with ")."
— During execution of the FP command, the system displays the message " PROM address? " to prompt for a PROM load address. Specify the low order 3 digits of the required load address.
— 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 filename.
— A file is a block which begins with " $ " and ends with ")."
— During execution of the FP command, the system displays the message " PROM address? " to prompt for a PROM load address. Specify the low order 3 digits of the required load address.
— 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 ")."
— During execution of the FP command, the system displays the message " PROM address? " to prompt for a PROM load address. Specify the low order 3 digits of the required load address.
— 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 filename.
— During execution of the FR command, the system displays the message " PROM address? " to prompt for a PROM load address. Specify the low order 3 digits of the required load address.
— Denote the end of data with the symbol "] ".

5.4 ERROR MESSAGES OF THE PROM FORMATTER

Error message	Meaning	Relevant commands
? ? ?	An attempt was made to access a location outside the link area.	FP , FR
?	The specified starting address is not smaller than or equal to the ending address.	FP
Not found	The specified file was not found.	FR
Format?	The format of the paper tape to be read does not match the specified format command.	FR
Parity?	The parity scheme of the paper tape to be read does not match the specified parity scheme.	FR
Check sum	A check sum error occurred while a paper tape in format C was being read.	FR (with format command C only)
Invalid	A command was specified in an invalid format.	FP , FR

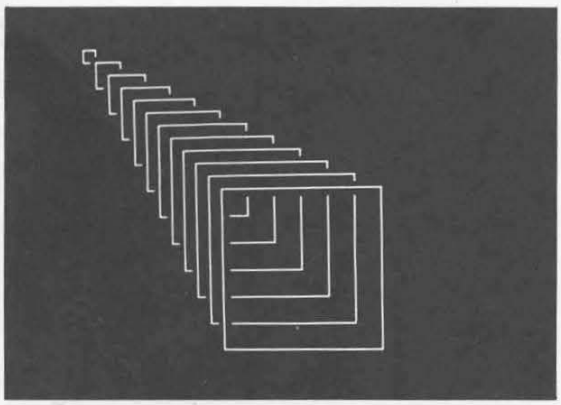
(Note) The error messages associated with the symbolic debugger section are identical to those associated with the symbolic debugger.

CHAPTER 6

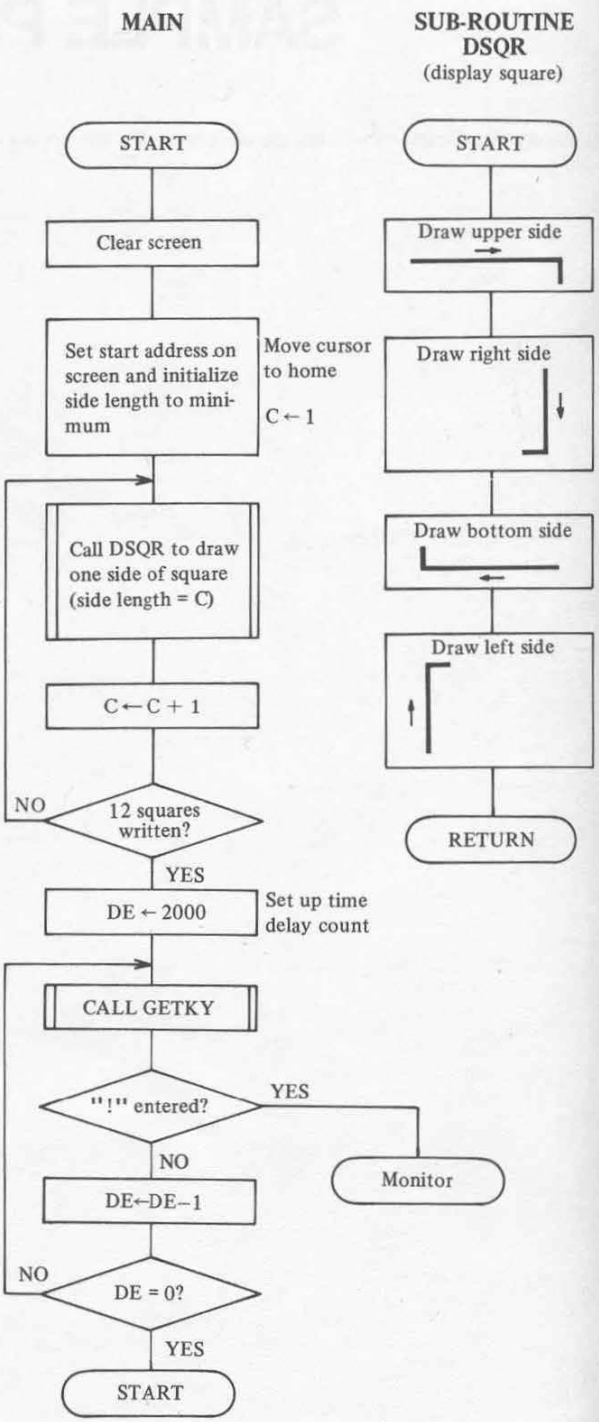
SAMPLE PROGRAM

6.1 DRAWING AN APPROACHING SQUARE

Let us prepare a program which draws squares which get bigger and bigger as they approach. We want a small square drawn at the upper left corner of the screen to appear as if it were approaching us. This can be accomplished by drawing squares with sides of increasing length one over another with a slight displacement.



The photo above shows a square which apparently approaches us as it grows bigger and bigger.



```
   **  Z80 ASSEMBLER SB-2202   PAGE 01  **

01 0000                    ;
02 0000                    ; APPROACHING SQUARE
03 0000                    ;
04 0000  P                 CHR1:   EQU     9BH
05 0000  P                 CHR2:   EQU     95H
06 0000  P                 CHR3:   EQU     9AH
07 0000  P                 CHR4:   EQU     96H
08 0000  P                 CHR5:   EQU     98H
09 0000  P                 CHR6:   EQU     97H
10 0000                    MNTR:   EQU     0000H
11 0000  P                 GETKY:  EQU     0610H
12 0000  P                 PRNT:   EQU     063CH
13 0000                    ;
14 0000 119A00             3DG0:   LD      DE,DSP1
15 0003 CD9100                     CALL    CPRNT
16 0006 0E01                       LD      C,1
17 0008 41                 3DG1:   LD      B,C
18 0009 CD3E00                     CALL    DSQR
19 000C CD8800                     CALL    TMDLY
20 000F 3E03                       LD      A,03H
21 0011 CD3C06                     CALL    PRNT
22 0014 3E01                       LD      A,01H
23 0016 CD3C06                     CALL    PRNT
24 0019 0C                         INC     C
25 001A 79                         LD      A,C
26 001B FE0D                       CP      13
27 001D 20E9                       JR      NZ,3DG1
28 001F 110020                     LD      DE,2000H
29 0022 1B                 3DG2:   DEC     DE
30 0023 CD1006                     CALL    GETKY
31 0026 FE21                       CP      '!'
32 0028 CA3100                     JP      Z,MNTR1
33 002B 7A                         LD      A,D
34 002C B3                         OR      E
35 002D 20F3                       JR      NZ,3DG2
36 002F 18CF                       JR      3DG0
37 0031 21AE00             MNTR1:  LD      HL,00AEH
38 0034 3601                       LD      (HL),01H
39 0036 C30000                     JP      MNTR
40 0039                    ;
41 0039                    ; SUB ROUTINE
42 0039                    ;
43 0039 3E9B               DSQR0:  LD      A,CHR1
44 003B CD3C06                     CALL    PRNT
45 003E 10F9               DSQR:   DJNZ    DSQR0
46 0040 3E95                       LD      A,CHR2
47 0042 41                         LD      B,C
48 0043 1802                       JR      +4
49 0045 3E9A               DSQR1:  LD      A,CHR3
50 0047 CD3C06                     CALL    PRNT
51 004A 3E01                       LD      A,01H
52 004C CD3C06                     CALL    PRNT
53 004F 3E04                       LD      A,04H
54 0051 CD3C06                     CALL    PRNT
55 0054 10EF                       DJNZ    DSQR1
56 0056 3E96                       LD      A,CHR4
57 0058 41                         LD      B,C
58 0059 1802                       JR      +4
59 005B 3E9B               DSQR2:  LD      A,CHR1
60 005D CD3C06                     CALL    PRNT
```

Defines graphic characters for drawing squares.

Symbol	Character
CHR1	⊟
CHR2	◱
CHR3	◫
CHR4	◰
CHR5	◳
CHR6	◲

Defines monitor subroutine addresses.

Clears screen and positions cursor at initial position.

Determines side length of square to be drawn first.

Draws one square whose side length is specified in C, then determines next position after a delay.

Increments C (which contains the side length). Exits this loop when C reaches 13.

Gets character with a delay.
Returns to monitor if "!" is entered; otherwise, returns to beginning of the program.

Draws top side of square.

Draws right side.

Draws bottom side.

```
   **  Z80 ASSEMBLER SB-2202   PAGE 02  **

01 0060 3E04                    LD      A,04H
02 0062 CD3C06                  CALL    PRNT
03 0065 3E04                    LD      A,04H
04 0067 CD3C06                  CALL    PRNT
05 006A 10EF                    DJNZ    DSQR2
06 006C 3E98                    LD      A,CHR5
07 006E 41                      LD      B,C
08 006F 1802                    JR      +4
09 0071 3E9A            DSQR3:  LD      A,CHR3
10 0073 CD3C06                  CALL    PRNT
11 0076 3E02                    LD      A,02H
12 0078 CD3C06                  CALL    PRNT
13 007B 3E04                    LD      A,04H
14 007D CD3C06                  CALL    PRNT
15 0080 10EF                    DJNZ    DSQR3
16 0082 3E97                    LD      A,CHR6
17 0084 CD3C06                  CALL    PRNT
18 0087 C9                      RET
19 0088                 ;
20 0088 110020          TMDLY:  LD      DE,2000H
21 008B 1B                      DEC     DE
22 008C 7A                      LD      A,D
23 008D B3                      OR      E
24 008E 20FB                    JR      NZ,-3
25 0090 C9                      RET
26 0091                 ;
27 0091                 CPRNT:  ENT
28 0091 1A                      LD      A,(DE)
29 0092 B7                      OR      A
30 0093 C8                      RET     Z
31 0094 CD3C06                  CALL    PRNT
32 0097 13                      INC     DE
33 0098 18F7                    JR      CPRNT
34 009A                 ;
35 009A 06              DSP1:   DEFB    06H
36 009B 01                      DEFB    01H
37 009C 03                      DEFB    03H
38 009D 00                      DEFB    00H
39 009E                         END
```

Draws left side and returns.

Loads repeat count (2000H) into DE register pair.

Prints message designated by DE register pair.

Data area.

6.2 SORTING DATA

Consider the problem of sorting a block of data (e.g., A0H bytes of data), starting at the beginning of the monitor area, into a ascending order. We first move the contents of that data block to the memory area starting at address 3000H, then sort them using the bubble sort method. In the bubble sort method, two adjacent data items are compared and immediately interchanged if they are out of order. This is the simplest sorting method. As bubble sorting is applied to the block from beginning to the end, the largest data item is bubbled to the end of the block. This first sorting process is called pass 1. N data items can be sorted with a maximum of n passes. In this example, A0 (in hexadecimal) passes are required.

Data sorting plays an important role in data base management, as well as data retrieval. There are a variety of sorting methods whose efficiency differs depending on the data type and volume.

The program below moves A0 (in hexadecimal) bytes of data from the block starting at address 0000H (the monitor area) to the memory block starting at address 3000H, sorts the data, and displays the sorted data on the CRT display.

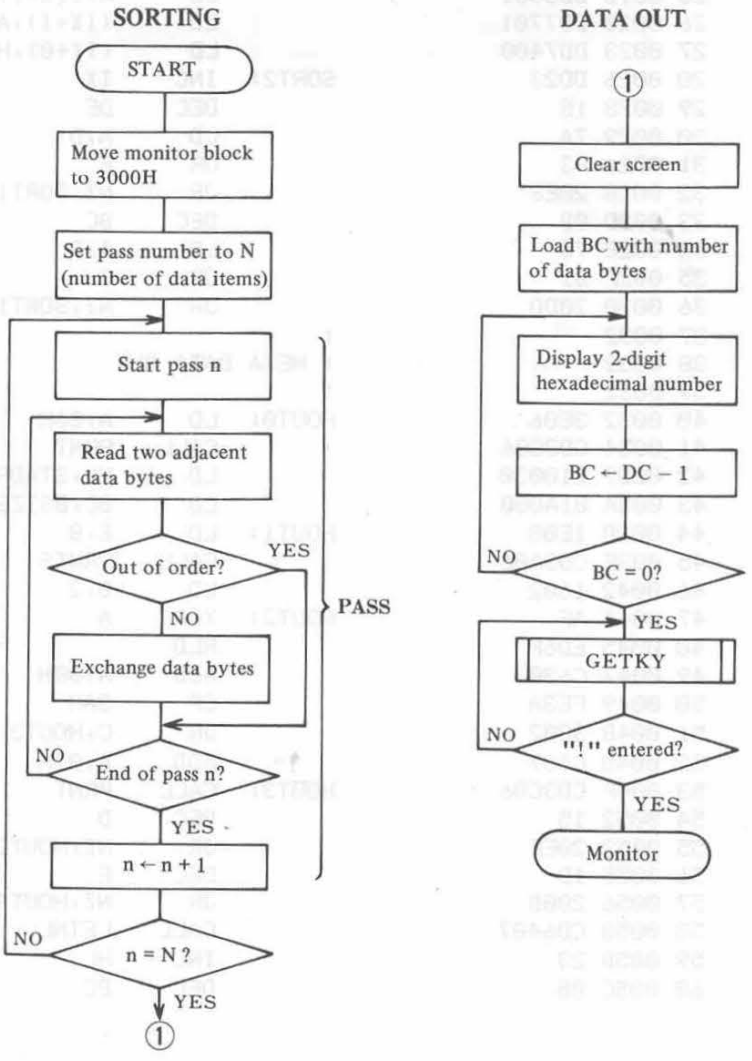
Bubble sort passes

Unsorted data	Pass 1	Pass 2	.....
2 A	2 A	0 3	
3 5	0 3	2 A	
0 3	3 3	3 3	.....
3 3	3 5	0 2	
4 1	0 2	1 F	
0 2	1 F	3 5	
1 F	4 1	4 1	

The figure above shows how data items are interchanged during sorting passes. It can be seen that after pass n, at least n data items are placed in their proper positions, starting with the largest data item.

This program references the following monitor subroutines:

PRNT
PRNTS
LETNL
GETKY



```
   **  Z80 ASSEMBLER SB-2202   PAGE 01  **

01 0000                    ;
02 0000                    ; SORTING
03 0000                    ;
04 0000                    RDADR:  EQU     0000H
05 0000  P                 STADR:  EQU     3000H
06 0000  P                 BSIZE:  EQU     00A0H
07 0000                    MNTR:   EQU     0000H
08 0000  P                 GETKY:  EQU     0610H
09 0000  P                 PRNTS:  EQU     063AH
10 0000  P                 PRNT:   EQU     063CH
11 0000  P                 LETNL:  EQU     0764H
12 0000                    ;
13 0000 210000             SORT0:  LD      HL,RDADR
14 0003 110030                     LD      DE,STADR
15 0006 01A000                     LD      BC,BSIZE
16 0009 EDB0                       LDIR
17 000B 01A000                     LD      BC,BSIZE
18 000E 0B                         DEC     BC
19 000F DD210030           SORT1:  LD      IX,STADR
20 0013 50                         LD      D,B
21 0014 59                         LD      E,C
22 0015 DD7E00                     LD      A,(IX+0)
23 0018 DDBE01                     CP      (IX+1)
24 001B 3809                       JR      C,SORT2
25 001D DD6601                     LD      H,(IX+1)
26 0020 DD7701                     LD      (IX+1),A
27 0023 DD7400                     LD      (IX+0),H
28 0026 DD23               SORT2:  INC     IX
29 0028 1B                         DEC     DE
30 0029 7A                         LD      A,D
31 002A B3                         OR      E
32 002B 20E8                       JR      NZ,SORT1+6
33 002D 0B                         DEC     BC
34 002E 78                         LD      A,B
35 002F B1                         OR      C
36 0030 20DD                       JR      NZ,SORT1
37 0032                    ;
38 0032                    ; HEXA DATA OUT
39 0032                    ;
40 0032 3E06               HOUT0:  LD      A,06H
41 0034 CD3C06                     CALL    PRNT
42 0037 210030                     LD      HL,STADR
43 003A 01A000                     LD      BC,BSIZE
44 003D 1E08               HOUT1:  LD      E,8
45 003F CD3A06                     CALL    PRNTS
46 0042 1602                       LD      D,2
47 0044 AF                 HOUT2:  XOR     A
48 0045 ED6F                       RLD
49 0047 C630                       ADD     A,30H
50 0049 FE3A                       CP      3AH
51 004B 3802                       JR      C,HOUT3
52 004D C607                       ADD     A,07H
53 004F CD3C06             HOUT3:  CALL    PRNT
54 0052 15                         DEC     D
55 0053 20EF                       JR      NZ,HOUT2
56 0055 1D                         DEC     E
57 0056 200B                       JR      NZ,HOUT4
58 0058 CD6407                     CALL    LETNL
59 005B 23                         INC     HL
60 005C 0B                         DEC     BC
```

Defines source address, destination address, and data block length.

Defines monitor subroutine addresses.

Moves monitor data to block indicated by STADR.

Loads BC register with number of passes.

Uses index register IX for sorting.

Exchanges if following data is smaller than preceding data.

Tests pass number.

Clears screen.

Prints 8 bytes on a line.

Loads 4 high order bits of memory location indicated by HL register into Acc.

Converts hexadecimal code to ASCII code and prints it (All sorted data bytes are displayed).

```
   **  Z80 ASSEMBLER SB-2202   PAGE 02  **

01 005D 78                     LD      A,B
02 005E B1                     OR      C
03 005F 20DC                   JR      NZ,HOUT1
04 0061 1806                   JR      HOUT5
05 0063 23             HOUT4:  INC     HL
06 0064 0B                     DEC     BC
07 0065 78                     LD      A,B
08 0066 B1                     OR      C
09 0067 20D6                   JR      NZ,HOUT1+2
10 0069 CD1006         HOUT5:  CALL    GETKY
11 006C FE21                   CP      '!'
12 006E 20F9                   JR      NZ,HOUT5
13 0070 21AE00                 LD      HL,00AEH
14 0073 3601                   LD      (HL),01H
15 0075 C30000                 JP      MNTR
16 0078                        END
```

Checks character input and returns to monitor if "!" is entered.

Run the program with various data sizes.

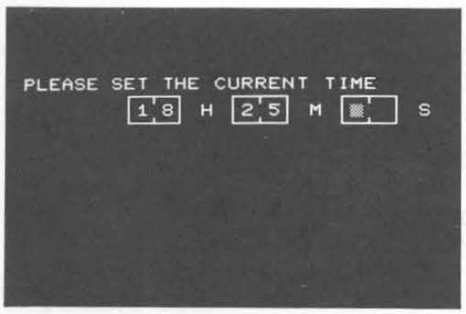
Consider how to modify the program to sort data in descending order.

6.3 MAKING A DIGITAL CLOCK

Let us construct a 24-hour digital clock using the built-in timer facility. To read the built-in timer, use the monitor subroutine TIMRD (Time Read). The timer can be started using the monitor subroutine TIMST (Time Start).

The program below uses the built-in timer only to detect lapses of one second. The program also contains BELL subroutines to produce (or stop) a beeper tone each second.

Setting time



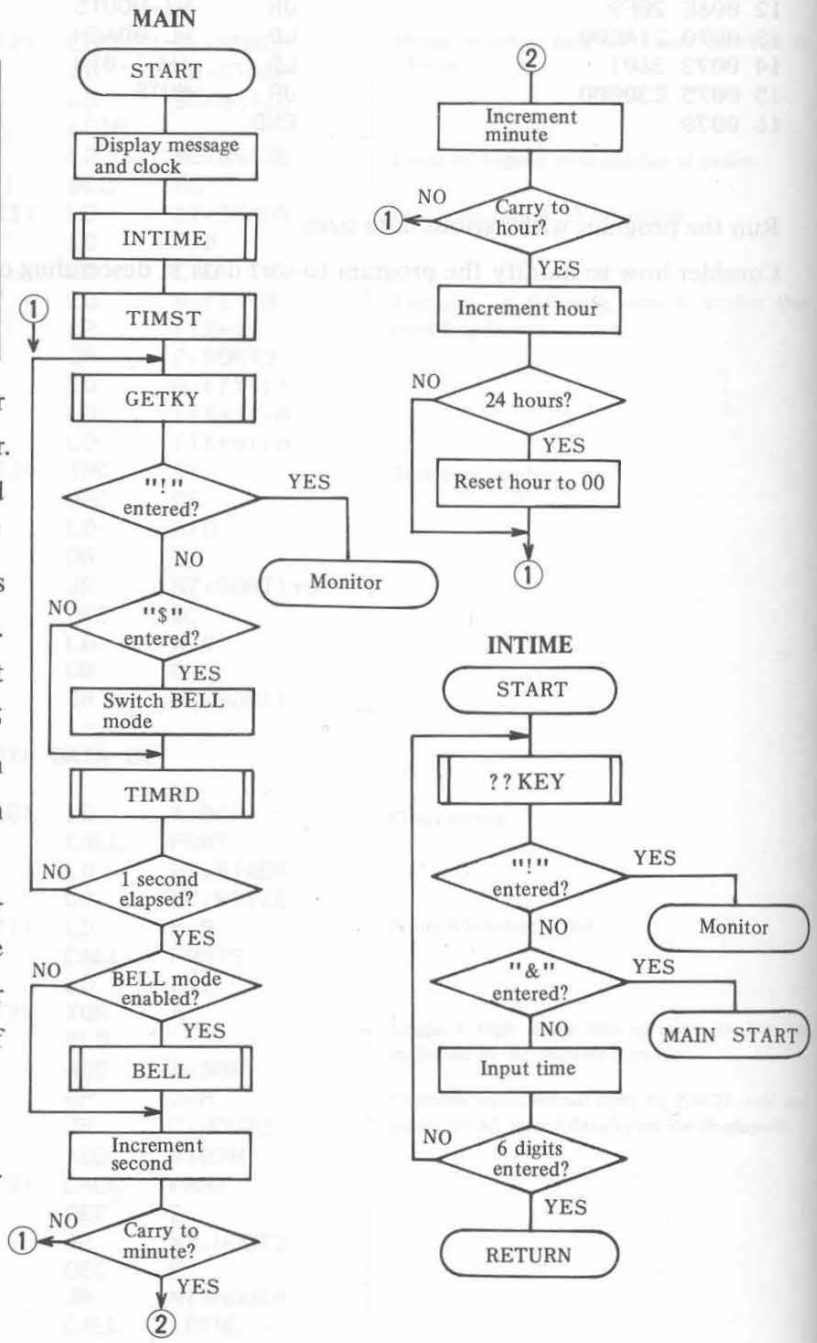
When started, the program asks for the current time by flashing the cursor. Enter the time, minute, and second with 2-digit numbers. Press the "&" key if you enter an invalid value. Press the "!" key to return to the monitor.

Note that this program does not check for invalid time values (e.g., 25 hours 70 minutes). Also the program does not accept characters other than numeric character.

After the timer is started, it displays the current time and sounds the beeper tone every second. The beeper (BELL) mode is switched on or off each time the "$" key is pressed.

This program references the following monitor subroutines:

TIMST	LETNL
TIMRD	MSG
GETKY	PRNT
BELL	? ? KEY



```
   **  Z80 ASSEMBLER SB-2202   PAGE 01  **

01 0000                    ;
02 0000                    ; DIGITAL CLOCK
03 0000                    ;
04 0000                    MNTR:   EQU     0000H
05 0000  P                 GETKY:  EQU     0610H
06 0000  P                 PRNT:   EQU     063CH
07 0000  P                 MSG:    EQU     06B5H
08 0000  P                 LETNL:  EQU     0764H
09 0000  P                 TIMST:  EQU     09CAH
10 0000  P                 TIMRD:  EQU     0A16H
11 0000  P                 BELL:   EQU     0A80H
12 0000  P                 ??KEY:  EQU     0D77H
13 0000  P                 IBUFE:  EQU     1180H
14 0000                    ;
15 0000 318011             START:  LD      SP,IBUFE
16 0003 CDEC00                     CALL    MESG0
17 0006 CDB200                     CALL    INTIME
18 0009 CD0501                     CALL    MESG1
19 000C AF                         XOR     A
20 000D 47                         LD      B,A
21 000E 110000                     LD      DE,0000H
22 0011 CDCA09                     CALL    TIMST
23 0014 CD160A             CLK0:   CALL    TIMRD
24 0017 4B                         LD      C,E
25 0018 CD1006                     CALL    GETKY
26 001B FE21                       CP      '!'
27 001D CAAA00                     JP      Z,MNTR1
28 0020 FE26                       CP      '&'
29 0022 28DF                       JR      Z,START+3
30 0024 FE24                       CP      '$'
31 0026 2804                       JR      Z,+6
32 0028 0600                       LD      B,0H
33 002A 180C                       JR      CLK1
34 002C A0                         AND     B
35 002D 2009                       JR      NZ,CLK1
36 002F 3A1202                     LD      A,(PIPPI)
37 0032 2F                         CPL
38 0033 321202                     LD      (PIPPI),A
39 0036 06FF                       LD      B,FFH
40 0038 CD160A             CLK1:   CALL    TIMRD
41 003B 7B                         LD      A,E
42 003C B9                         CP      C
43 003D 28D9                       JR      Z,CLK0+4
44 003F 3A1202                     LD      A,(PIPPI)
45 0042 B7                         OR      A
46 0043 2003                       JR      NZ,+5
47 0045 CD800A                     CALL    BELL
48 0048 CD2601                     CALL    CSINIT
49 004B 216901                     LD      HL,DTS0
50 004E 7E                         LD      A,(HL)
51 004F 3C                         INC     A
52 0050 FE3A                       CP      3AH
53 0052 2806                       JR      Z,CLK2
54 0054 77                 JRCLK0: LD      (HL),A
55 0055 CD3C06                     CALL    PRNT
56 0058 18BA                       JR      CLK0
57 005A CD1301             CLK2:   CALL    SETZR
58 005D 7E                         LD      A,(HL)
59 005E 3C                         INC     A
60 005F FE36                       CP      '6'
```

Defines monitor subroutine addresses.

Displays message and clock frame and waits for time to be specified.

Starts built-in timer
Set B to 00 to indicate it possible to switch BELL mode.

Reads timer value and stores it in C.

Gets character .
Returns to monitor if "!" is entered, switch BELL mode if "$" is entered, and returns to start of program if "$" is entered.

Enables BELL mode switching.

Disables mode switching if BELL mode has been switched (B = FFH)

Switches BELL mode
Complements data (00 to FFH or FFH to 00) in (PIPPI)

Loads B with BELL mode switching status.

Checks whether timer value is changed (1 second has elapsed).

Rings the bell if (PIPPI) = 00

Reads one's digit of second value and increments it.
If there is a carry, sets it to 0 and increments the ten's digit by 1.
If there is no carry, returns to timer read subroutine (CLK0).

```
01 0061 20F1              JR      NZ,JRCLK0
02 0063 3E30              LD      A,30H
03 0065 77                LD      (HL),A
04 0066 CD3C06            CALL    PRNT
05 0069 CD4001            CALL    CSL7B
06 006C 2B                DEC     HL
07 006D 7E                LD      A,(HL)
08 006E 3C                INC     A
09 006F FE3A              CP      3AH
10 0071 20E1              JR      NZ,JRCLK0
11 0073 CD1301            CALL    SETZR
12 0076 7E                LD      A,(HL)
13 0077 3C                INC     A
14 0078 FE36              CP      '6'
15 007A 20D8              JR      NZ,JRCLK0
16 007C 3E30              LD      A,30H
17 007E 77                LD      (HL),A
18 007F CD3C06            CALL    PRNT
19 0082 CD4001            CALL    CSL7B
20 0085 2B                DEC     HL
21 0086 7E                LD      A,(HL)
22 0087 3C                INC     A
23 0088 FE34              CP      '4'
24 008A 200E              JR      NZ,CLK3
25 008C 2B                DEC     HL
26 008D 7E                LD      A,(HL)
27 008E FE32              CP      '2'
28 0090 2013              JR      NZ,CLK4
29 0092 23                INC     HL
30 0093 CD1301            CALL    SETZR
31 0096 3E30              LD      A,30H
32 0098 18BA              JR      JRCLK0
33 009A FE3A        CLK3: CP      3AH
34 009C 20B6              JR      NZ,JRCLK0
35 009E CD1301            CALL    SETZR
36 00A1 34                INC     (HL)
37 00A2 7E                LD      A,(HL)
38 00A3 18B0              JR      JRCLK0+1
39 00A5 23          CLK4: INC     HL
40 00A6 34                INC     (HL)
41 00A7 7E                LD      A,(HL)
42 00A8 18AB              JR      JRCLK0+1
43 00AA 21AE00      MNTR1: LD     HL,00AEH
44 00AD 3601              LD      (HL),01H
45 00AF C30000            JP      MNTR
46 00B2                   ;
47 00B2                   ; SUB-ROUTINE
48 00B2                   ;
49 00B2 216401      INTIME: LD    HL,DTH1
50 00B5 CD2601            CALL    CSINIT
51 00B8 CD4C01            CALL    CSL18B
52 00BB CDC700            CALL    INPUT
53 00BE CD5801            CALL    CSL4F
54 00C1 CDC700            CALL    INPUT
55 00C4 CD5801            CALL    CSL4F
56 00C7                   ;
57 00C7 1E02        INPUT: LD     E,02H
58 00C9 CD770D            CALL    ??KEY
59 00CC FE21              CP      '!'
60 00CE CAAA00            JP      Z,MNTR1
```

Increments minute and resets second to 00 when second reaches 60.

Checks whether there is a carry in one's digit of minute value and increment ten's digit by 1 if there is a carry.

Increments hour by 1 when minute reaches 60.

Resets hour to 00 when it reaches 24.

Returns to monitor with patch on it .

Gets hour, minute, and second values.

Loads E with 2 to get 2 numeric characters.

Returns to monitor if "!" is entered.

```
00D1 FE26                CP      '&'
00D3 CA0300              JP      Z,START+3
00D6 FE30                CP      30H
00D8 38EF                JR      C,INPUT+2
00DA FE3A                CP      3AH
00DC 30EB                JR      NC,INPUT+2
00DE 77                  LD      (HL),A
00DF CD3C06              CALL    PRNT
00E2 3E03                LD      A,03H
00E4 CD3C06              CALL    PRNT
00E7 23                  INC     HL
00E8 1D                  DEC     E
00E9 20DE                JR      NZ,INPUT+2
00EB C9                  RET
00EC                ;
00EC 116A01         MESG0:  LD      DE,DATA1
00EF CD0C01              CALL    NLMSG
00F2 119001              LD      DE,DATA2
00F5 CD0C01              CALL    NLMSG
00F8 11AE01              LD      DE,DATA3
00FB CD0C01              CALL    NLMSG
00FE 11CE01              LD      DE,DATA4
0101 CD0C01              CALL    NLMSG
0104 C9                  RET
0105                ;
0105 11EC01         MESG1:  LD      DE,DATA5
0108 CD0C01              CALL    NLMSG
010B C9                  RET
010C                ;
010C CD6407         NLMSG:  CALL    LETNL
010F CDB506              CALL    MSG
0112 C9                  RET
0113                ;
0113 F5             SETZR:  PUSH    AF
0114 3E30                LD      A,30H
0116 77                  LD      (HL),A
0117 CD3C06              CALL    PRNT
011A 2B                  DEC     HL
011B 0603                LD      B,03H
011D 3E04                LD      A,04H
011F CD3C06              CALL    PRNT
0122 10F9                DJNZ    -5
0124 F1                  POP     AF
0125 C9                  RET
0126                ;
0126 F5             CSINIT: PUSH    AF
0127 3E05                LD      A,05H
0129 CD3C06              CALL    PRNT
012C 061B                LD      B,1BH
012E 3E03                LD      A,03H
0130 CD3C06              CALL    PRNT
0133 10F9                DJNZ    -5
0135 060B                LD      B,0BH
0137 3E01                LD      A,01H
0139 CD3C06              CALL    PRNT
013C 10F9                DJNZ    -5
013E F1                  POP     AF
013F C9                  RET
0140                ;
0140 F5             CSL7B:  PUSH    AF
```

Returns to start of program if "&" is entered.
Only numeric characters (ASCII code) 0 to 9 are allowed; number input is displayed on the screen.

Displays message and clock frame.

Displays message indicated by DATA5.

Prints newline and message.

Sets "0" in location indicated by HL, displays "0" on screen, and moves cursor to next position.

Moves cursor to one's place of second value.

Moves cursor 7 columns to left.

```
01 0141 0607                    LD      B,07H
02 0143 3E04                    LD      A,04H
03 0145 CD3C06                  CALL    PRNT
04 0148 10F9                    DJNZ    -5
05 014A F1                      POP     AF
06 014B C9                      RET
07 014C                 ;
08 014C F5              CSL18B: PUSH    AF
09 014D 0612                    LD      B,12H
10 014F 3E04                    LD      A,04H
11 0151 CD3C06                  CALL    PRNT
12 0154 10F9                    DJNZ    -5
13 0156 F1                      POP     AF
14 0157 C9                      RET
15 0158                 ;
16 0158 F5              CSL4F:  PUSH    AF
17 0159 0604                    LD      B,04H
18 015B 3E03                    LD      A,03H
19 015D CD3C06                  CALL    PRNT
20 0160 10F9                    DJNZ    -5
21 0162 F1                      POP     AF
22 0163 C9                      RET
23 0164                 ;
24 0164                 ; DATA
25 0164                 ;
26 0164                 DTH1:   DEFS    1
27 0165                 DTH0:   DEFS    1
28 0166                 DTM1:   DEFS    1
29 0167                 DTM0:   DEFS    1
30 0168                 DTS1:   DEFS    1
31 0169                 DTS0:   DEFS    1
32 016A                 ;
33 016A 0601            DATA1:  DEFW    0106H
34 016C 0101                    DEFW    0101H
35 016E 0101                    DEFW    0101H
36 0170 0101                    DEFW    0101H
37 0172 0101                    DEFW    0101H
38 0174 B9DEDDBB                DEFM    'PLEASE SET THE CURRENT TIME'
39 0178 DEB220C9
40 017C 20BCDEBA
41 0180 B820A620
42 0184 BEAFC420
43 0188 BCC3B8C0
44 018C DEBBB2
45 018F 0D                      DEFB    0DH
46 0190                 ;
47 0190 20202020        DATA2:  DEFM    '       ┌─┐     ┌─┐     ┌─┐'
48 0194 20202020
49 0198 979B9D9B
50 019C 95202020
51 01A0 979B9D9B
52 01A4 95202020
53 01A8 979B9D9B
54 01AC 95
55 01AD 0D                      DEFB    0DH
56 01AE                 ;
57 01AE 20202020        DATA3:  DEFM    '       |   | H |   | M |   | S'
58 01B2 20202020
59 01B6 9A202020
60 01BA 9A204820
```

Moves cursor 18 columns to left.

Moves cursor 4 columns to right.

Data areas for storing hour, minute, and time values.

Cursor control codes for moving cursor down 9 lines after returning cursor to home position and clearing screen.

```
   **  Z80 ASSEMBLER SB-2202   PAGE 05  **

01 01BE 9A202020
02 01C2 9A204D20
03 01C6 9A202020
04 01CA 9A2053
05 01CD 0D                      DEFB    0DH
06 01CE                 ;
07 01CE 20202020        DATA4:  DEFM    '    ␣␣␣    ␣␣␣    ␣␣␣'
08 01D2 20202020
09 01D6 989B9C9B
10 01DA 96202020
11 01DE 989B9C9B
12 01E2 96202020
13 01E6 989B9C9B
14 01EA 96
15 01EB 0D                      DEFB    0DH
16 01EC                 ;
17 01EC 0501            DATA5:  DEFW    0105H   ┐ Cursor control codes for moving cursor down 9
18 01EE 0101                    DEFW    0101H   │ lines after returning cursor to home position.
19 01F0 0101                    DEFW    0101H   │
20 01F2 0101                    DEFW    0101H   │
21 01F4 0101                    DEFW    0101H   ┘
22 01F6 C0C0DEB2                DEFM    'THE CURRENT TIME IS    '
23 01FA CF20C920
24 01FE BCDEBAB8
25 0202 20CA2020
26 0206 20202020
27 020A 20202020
28 020E 202020
29 0211 0D                      DEFB    0DH
30 0212                 ;
31 0212 00              PIPPI:  DEFB    00H     ┐ Data area for storing BELL mode status.
32 0213                         END
```

[Application 1]

Add a timer function to this clock program. Modify the program so that if "25 hours 63 minutes 80 seconds" is entered, for example, it is converted to "02 hours 04 minutes 20 seconds."

[Application 2]

BASIC TI$ is a string variable which contains 6 characters indicating the hour, minute, and time read into the DE register through the TIMRD monitor subroutine. Construct a program which displays the 6-character time each time the operator enters "PRINT TI$" as in BASIC.

6.4 MULTIPLYING HEXADECIMAL NUMBERS

Let us prepare a program for multiplying 8-digit hexadecimal numbers. This is to be done by expressing the numbers to be multiplied in binary format and shifting them to the left the number of times necessary to achieve the product. For example, when the 2-digit hexadecimal numbers 23H and 14H are multiplied, the equivalent binary expression would be as follows.

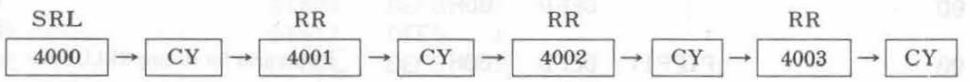
00100011 x 00010100

Multiplication is performed by shifting the "1's" the applicable number of places to the left, then adding the numbers.

```
  10001100 ..... numbers shifted to the left 2 places.
+) 1000110000 ..... Numbers shifted to the left 4 places.
  1010111100
```

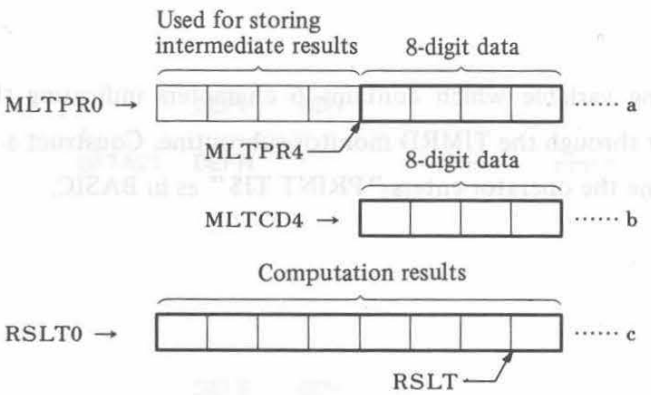
Thus, the answer is 2BC.

Further, the SRL and RR instructions are to be used to read in the numbers to be multiplied. For example, the numbers are shifted and rotated as shown below if stored in addresses 4000 to 4003.

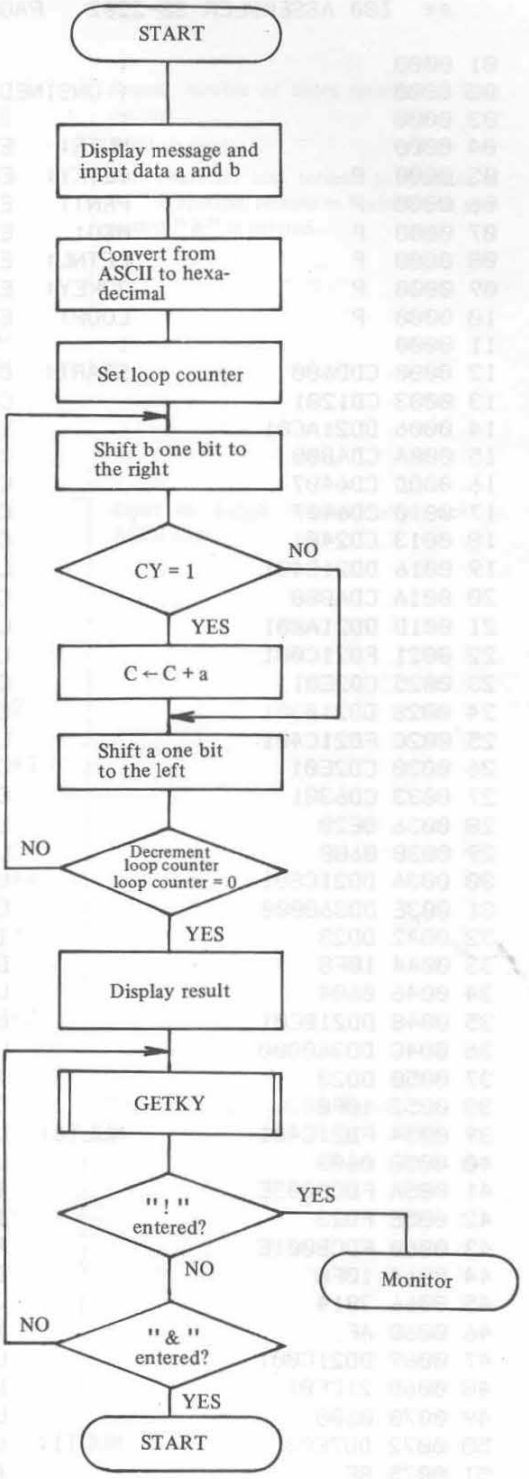


At this point, all that is required is to check whether CY for the last RR instruction executed (i.e., the one on the far right) is "1" or "0".

The memory areas used for multiplication are allocated as follows:



The results of multiplication are displayed on the screen as shown in the photo below. The program flowchart is shown at right.



```
    **  Z80 ASSEMBLER SB-2202   PAGE 01  **

01 0000                    ;
02 0000                    ; UNSINED 8 BYTES BINARY MULTIPLY
03 0000                    ;
04 0000                    MNTR:   EQU     0000H
05 0000  P                 GETKY:  EQU     0610H
06 0000  P                 PRNT:   EQU     063CH
07 0000  P                 MSG:    EQU     06B5H
08 0000  P                 LETNL:  EQU     0764H
09 0000  P                 ??KEY:  EQU     0D77H
10 0000  P                 LOOP:   EQU     20H
11 0000                    ;
12 0000 CDD600             START:  CALL    DSPM
13 0003 CD1201                     CALL    CSLIN
14 0006 DD21AC01                   LD      IX,MLTPR
15 000A CDAB00                     CALL    KEYIN
16 000D CD6407                     CALL    LETNL
17 0010 CD6407                     CALL    LETNL
18 0013 CD2401                     CALL    CSLFW
19 0016 DD21B401                   LD      IX,MLTCD
20 001A CDAB00                     CALL    KEYIN
21 001D DD21AB01                   LD      IX,MLTPR-1
22 0021 FD21C001                   LD      IY,MLTPR4
23 0025 CD2E01                     CALL    CONV84
24 0028 DD21B301                   LD      IX,MLTCD-1
25 002C FD21C401                   LD      IY,MLTCD4
26 0030 CD2E01                     CALL    CONV84
27 0033 CD6301                     CALL    DSANS
28 0036 0E20                       LD      C,LOOP
29 0038 0608                       LD      B,08H
30 003A DD21C801                   LD      IX,RSLT0
31 003E DD360000                   LD      (IX+0),00H
32 0042 DD23                       INC     IX
33 0044 10F8                       DJNZ    -6
34 0046 0604                       LD      B,04H
35 0048 DD21BC01                   LD      IX,MLTPR0
36 004C DD360000                   LD      (IX+0),00H
37 0050 DD23                       INC     IX
38 0052 10F8                       DJNZ    -6
39 0054 FD21C401           MULT0:  LD      IY,MLTCD4
40 0058 0603                       LD      B,03H
41 005A FDCB003E                   SRL     (IY+0)
42 005E FD23                       INC     IY
43 0060 FDCB001E                   RR      (IY+0)
44 0064 10F8                       DJNZ    -6
45 0066 3014                       JR      NC,MULT2
46 0068 AF                         XOR     A
47 0069 DD21C001                   LD      IX,MLTPR4
48 006D 21CF01                     LD      HL,RSLT
49 0070 0608                       LD      B,08H
50 0072 DD7E03             MULT1:  LD      A,(IX+3)
51 0075 8E                         ADC     A,(HL)
52 0076 77                         LD      (HL),A
53 0077 DD2B                       DEC     IX
54 0079 2B                         DEC     HL
55 007A 10F6                       DJNZ    MULT1
56 007C DD21C001           MULT2:  LD      IX,MLTPR4
57 0080 0607                       LD      B,07H
58 0082 DDCB0326                   SLA     (IX+3)
59 0086 DD2B                       DEC     IX
60 0088 DDCB0316                   RL      (IX+3)
```

Defines monitor subroutine addresses.

Loop count.

Displays message.

Input a (MLTPR).

Input b (MLTCD).

Converts a to hexadecimal → MLTPR4

Converts b to hexadecimal → MLTCD4

Initialize.

Shifts b one bit to the right.

Check CY.

c ← c + a

Shifts a one bit to the left.

```
008C 10F8                    DJNZ    -6
008E 0D                      DEC     C
008F 20C3                    JR      NZ,MULT0
0091 CD7E01                  CALL    DPRST
0094 CD1006          MULT3:  CALL    GETKY
0097 FE21                    CP      '!'
0099 CAA300                  JP      Z,MNTR1
009C FE26                    CP      '&'
009E CA0000                  JP      Z,START
00A1 18F1                    JR      MULT3
00A3 21AE00          MNTR1:  LD      HL,00AEH
00A6 3601                    LD      (HL),01H
00A8 C30000                  JP      MNTR
00AB                 ;
00AB                 ; SUBROUTINE
00AB                 ;
00AB 1E08            KEYIN:  LD      E,08H
00AD CD770D                  CALL    ??KEY
00B0 FE21                    CP      '!'
00B2 CAA300                  JP      Z,MNTR1
00B5 FE26                    CP      '&'
00B7 CA0000                  JP      Z,START
00BA FE30                    CP      '0'
00BC 38EF                    JR      C,KEYIN+2
00BE FE47                    CP      'G'
00C0 30EB                    JR      NC,KEYIN+2
00C2 FE41                    CP      'A'
00C4 3004                    JR      NC,+6
00C6 FE3A                    CP      3AH
00C8 30E3                    JR      NC,KEYIN+2
00CA DD7700                  LD      (IX+0),A
00CD CD3C06                  CALL    PRNT
00D0 DD23                    INC     IX
00D2 1D                      DEC     E
00D3 20D8                    JR      NZ,KEYIN+2
00D5 C9                      RET
00D6                 ;
00D6 11D001          DSPM:   LD      DE,MSG1
00D9 CD0801                  CALL    MSGNL2
00DC 11F201                  LD      DE,MSG2
00DF CD0101                  CALL    MSGNL
00E2 110602                  LD      DE,MSG3
00E5 CD0101                  CALL    MSGNL
00E8 11F201                  LD      DE,MSG2
00EB CD0801                  CALL    MSGNL2
00EE 111A02                  LD      DE,MSG4
00F1 CD0801                  CALL    MSGNL2
00F4 112702                  LD      DE,MSG5
00F7 CD0801                  CALL    MSGNL2
00FA 113B02                  LD      DE,MSG6
00FD CDB506                  CALL    MSG
0100 C9                      RET
0101                 ;
0101 CDB506          MSGNL:  CALL    MSG
0104 CD6407                  CALL    LETNL
0107 C9                      RET
0108                 ;
0108 CDB506          MSGNL2: CALL    MSG
010B CD6407                  CALL    LETNL
010E CD6407                  CALL    LETNL
```

Repeat number of times specified in loop counter.

Displays result.

Gets character and returns to monitor if "!" is entered; returns to beginning of program if "&" is entered.

Input an 8-digit hexadecimal number in ASCII code.

Displays message on CRT screen.

```
01 0111 C9                      RET
02 0112                 ;
03 0112 3E05            CSLIN:  LD      A,05H
04 0114 CD3C06                  CALL    PRNT
05 0117 0608                    LD      B,08H
06 0119 3E01                    LD      A,01H
07 011B CD3C06                  CALL    PRNT
08 011E 10F9                    DJNZ    -5
09 0120 CD2401                  CALL    CSLFW
10 0123 C9                      RET
11 0124                 ;
12 0124 060B            CSLFW:  LD      B,0BH
13 0126 3E03                    LD      A,03H
14 0128 CD3C06                  CALL    PRNT
15 012B 10F9                    DJNZ    -5
16 012D C9                      RET
17 012E                 ;
18 012E 1600            CONV84: LD      D,00H
19 0130 0E04                    LD      C,04H
20 0132 DD23                    INC     IX
21 0134 DD7E00                  LD      A,(IX+0)
22 0137 5F                      LD      E,A
23 0138 E6F0                    AND     F0H
24 013A FE30                    CP      30H
25 013C 2805                    JR      Z,+7
26 013E 7B                      LD      A,E
27 013F C609                    ADD     A,09H
28 0141 1801                    JR      +3
29 0143 7B                      LD      A,E
30 0144 E60F                    AND     0FH
31 0146 14                      INC     D
32 0147 280D                    JR      Z,CONV1
33 0149 0604                    LD      B,04H
34 014B CB27                    SLA     A
35 014D 10FC                    DJNZ    -2
36 014F FD7700                  LD      (IY+0),A
37 0152 16FF                    LD      D,FFH
38 0154 18DC                    JR      CONV84+4
39 0156 FDB600          CONV1:  OR      (IY+0)
40 0159 FD7700                  LD      (IY+0),A
41 015C 0D                      DEC     C
42 015D C8                      RET     Z
43 015E FD23                    INC     IY
44 0160 18D0                    JR      CONV84+4
45 0162 C9                      RET
46 0163                 ;
47 0163 3E05            DSANS:  LD      A,05H
48 0165 CD3C06                  CALL    PRNT
49 0168 060D                    LD      B,0DH
50 016A 3E01                    LD      A,01H
51 016C CD3C06                  CALL    PRNT
52 016F 10F9                    DJNZ    -5
53 0171 114F02                  LD      DE,MSG7
54 0174 CD0801                  CALL    MSGNL2
55 0177 115802                  LD      DE,MSG8
56 017A CDB506                  CALL    MSG
57 017D C9                      RET
58 017E                 ;
59 017E 21C801          DPRST:  LD      HL,RSLT0
60 0181 1E08                    LD      E,08H
```

Positions cursor at initial position.

Moves cursor 11 columns to right.

Converts 8-digit ASCII code to 4-digit hexadecimal number.

Displays result message.

Displays result.

```
01 0183 1600                    LD      D,00H
02 0185 7E              DPRST1: LD      A,(HL)
03 0186 14                      INC     D
04 0187 280D                    JR      Z,DPRST2
05 0189 0604                    LD      B,04H
06 018B CB3F                    SRL     A
07 018D 10FC                    DJNZ    -2
08 018F CDA001                  CALL    APRNT
09 0192 16FF                    LD      D,FFH
10 0194 18EF                    JR      DPRST1
11 0196 E60F            DPRST2: AND     0FH
12 0198 CDA001                  CALL    APRNT
13 019B 1D                      DEC     E
14 019C C8                      RET     Z
15 019D 23                      INC     HL
16 019E 18E5                    JR      DPRST1
17 01A0                 ;
18 01A0 FE0A            APRNT:  CP      0AH
19 01A2 3802                    JR      C,+4
20 01A4 C607                    ADD     A,07H
21 01A6 C630                    ADD     A,30H
22 01A8 CD3C06                  CALL    PRNT
23 01AB C9                      RET
24 01AC                 ;
25 01AC                 ; DATA AREA
26 01AC                 ;
27 01AC                 MLTPR:  DEFS    8
28 01B4                 MLTCD:  DEFS    8
29 01BC                 MLTPR0: DEFS    4
30 01C0                 MLTPR4: DEFS    4
31 01C4                 MLTCD4: DEFS    4
32 01C8                 RSLT0:  DEFS    7
33 01CF                 RSLT:   DEFS    1
34 01D0 06              MSG1:   DEFB    06H
35 01D1 554E5349                DEFM    'UNSIGNED 8 BYTES BINARY MULTIPLY'
36 01D5 474E4544
37 01D9 20382042
38 01DD 59544553
39 01E1 2042494E
40 01E5 41525920
41 01E9 4D554C54
42 01ED 49504C59
43 01F1 0D                      DEFB    0DH
44 01F2 20202020        MSG2:   DEFM    '       **************'
45 01F6 20202A2A
46 01FA 2A2A2A2A
47 01FE 2A2A2A2A
48 0202 2A2A2A
49 0205 0D                      DEFB    0DH
50 0206 20202020        MSG3:   DEFM    '       * C = A * B *'
51 020A 20202A20
52 020E 43203D20
53 0212 41202A20
54 0216 42202A
55 0219 0D                      DEFB    0DH
56 021A 504C4541        MSG4:   DEFM    'PLEASE INPUT'
57 021E 53452049
58 0222 4E505554
59 0226 0D                      DEFB    0DH
60 0227 20202020        MSG5:   DEFM    '         A = ********'
```

Converts hexadecimal number to ASCII code and displays on CRT screen.

Reserved for computation.

```
   **  Z80 ASSEMBLER SB-2202   PAGE 05  **

01 022B 20202041
02 022F 203D202A
03 0233 2A2A2A2A
04 0237 2A2A2A
05 023A 0D                      DEFB   0DH
06 023B 20202020        MSG6:   DEFM   '       B = ********'
07 023F 20202042
08 0243 203D202A
09 0247 2A2A2A2A
10 024B 2A2A2A
11 024E 0D                      DEFB   0DH
12 024F 414E5345        MSG7:   DEFM   'ANSWER IS'
13 0253 52204953
14 0257 0D                      DEFB   0DH
15 0258 20202020        MSG8:   DEFM   '       C = '
16 025C 20202043
17 0260 203D20
18 0263 0D                      DEFB   0DH
19 0264                         END
```

[Application]

Modify the program so that it can multiply decimal numbers.

Expand the program so that it can perfrom all basic (4) arithmetic operations on decimal numbers.

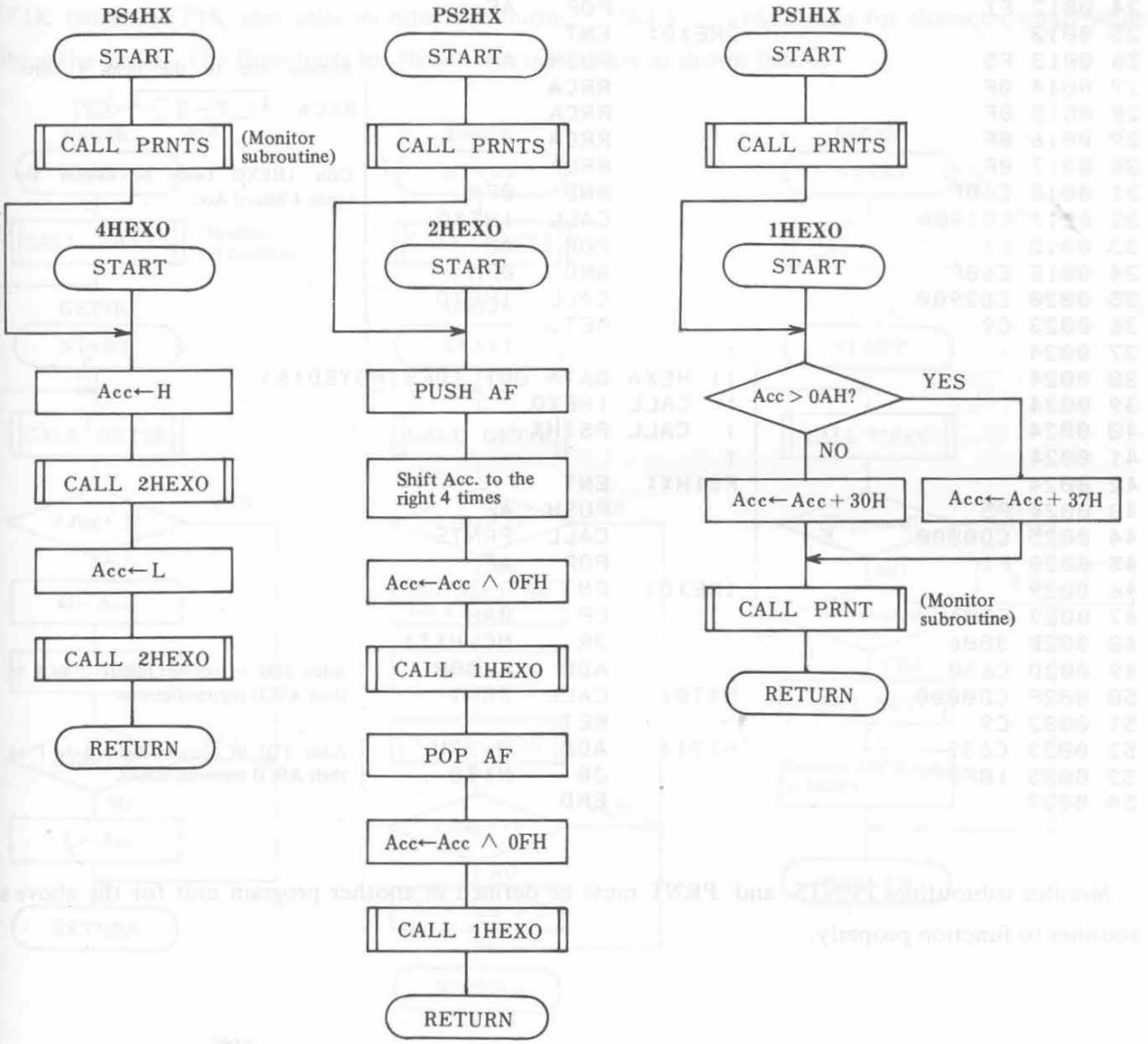
6.5 DISPLAYING BINARY DATA IN HEXADECIMAL REPRESENTATION

Let us construct a subprogram to display binary data in hexadecimal. The subprogram must display the contents of the HL register pair as a 4-digit hexadecimal number, the contents of the accumulator as a 2-digit hexadecimal number, and the lower 4 bits of the accumulator as a 1-digit hexadecimal number. The subprogram must also place a space before the displayed number.

The subprogram has six entry points as follows:

CALL 4HEXO	(4hexa data out)	: Displays the HL contents.
CALL PS4HX	(print space, 4hexa data out)	: Displays a space and the HL contents.
CALL 2HEXO	(2hexa data out)	: Dispalys the Acc. contents.
CALL PS2HX	(print space, 2hexa data out)	: Displays a space and the Acc. contents.
CALL 1HEXO	(1hexa data out)	: Displays the lower 4 bits of Acc.
CALL PS1HX	(print space, 1hexa data out)	: Displays a space and the lower 4 bits of Acc.

The above subprograms are closely related to one another; 4HEXO calls 2HEXO twice and 2HEXO calls 1HEXO twice. The program flows are as shown below.

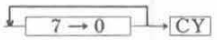


```
   **  Z80 ASSEMBLER SB-2202   PAGE 01  **

01 0000                        ;
02 0000                        ;4 HEXA DATA OUT (DESTROYED:A)
03 0000                        ;  CALL 4HEXO
04 0000                        ;  CALL PS4HX (PRINT SPACE)
05 0000                        ;
06 0000                        PS4HX:  ENT
07 0000 F5                             PUSH   AF
08 0001 CD0000      E                  CALL   PRNTS
09 0004 F1                             POP    AF
10 0005                        4HEXO:  ENT
11 0005 7C                             LD     A,H
12 0006 CD1300                         CALL   2HEXO
13 0009 7D                             LD     A,L
14 000A CD1300                         CALL   2HEXO
15 000D C9                             RET
16 000E                        ;
17 000E                        ;2 HEXA DATA OUT (DESTROYED:A)
18 000E                        ;  CALL 2HEXO
19 000E                        ;  CALL PS2HX
20 000E                        ;
21 000E                        PS2HX:  ENT
22 000E F5                             PUSH   AF
23 000F CD0000      E                  CALL   PRNTS
24 0012 F1                             POP    AF
25 0013                        2HEXO:  ENT
26 0013 F5                             PUSH   AF
27 0014 0F                             RRCA
28 0015 0F                             RRCA
29 0016 0F                             RRCA
30 0017 0F                             RRCA
31 0018 E60F                           AND    0FH
32 001A CD2900                         CALL   1HEXO
33 001D F1                             POP    AF
34 001E E60F                           AND    0FH
35 0020 CD2900                         CALL   1HEXO
36 0023 C9                             RET
37 0024                        ;
38 0024                        ;1 HEXA DATA OUT (DESTROYED:A)
39 0024                        ;  CALL 1HEXO
40 0024                        ;  CALL PS1HX
41 0024                        ;
42 0024                        PS1HX:  ENT
43 0024 F5                             PUSH   AF
44 0025 CD0000      E                  CALL   PRNTS
45 0028 F1                             POP    AF
46 0029                        1HEXO:  ENT
47 0029 FE0A                           CP     0AH
48 002B 3006                           JR     NC,HXT1
49 002D C630                           ADD    A,30H
50 002F CD0000      E          HXT0:   CALL   PRNT
51 0032 C9                             RET
52 0033 C637                   HXT1:   ADD    A,37H
53 0035 18F8                           JR     HXT0
54 0037                                END
```

Calls 2HEXO with H and L data in Acc.

Rotates Acc to the right 4 times.

RRCA 

Calls 1HEXO twice to display the lower 4 bits of Acc.

Adds 30H to convert digits 0 to 9 to their ASCII representations.

Adds 37H to convert digits A to F to their ASCII representations.

Monitor subroutines PRNTS and PRNT must be defined in another program unit for the above subroutines to function properly.

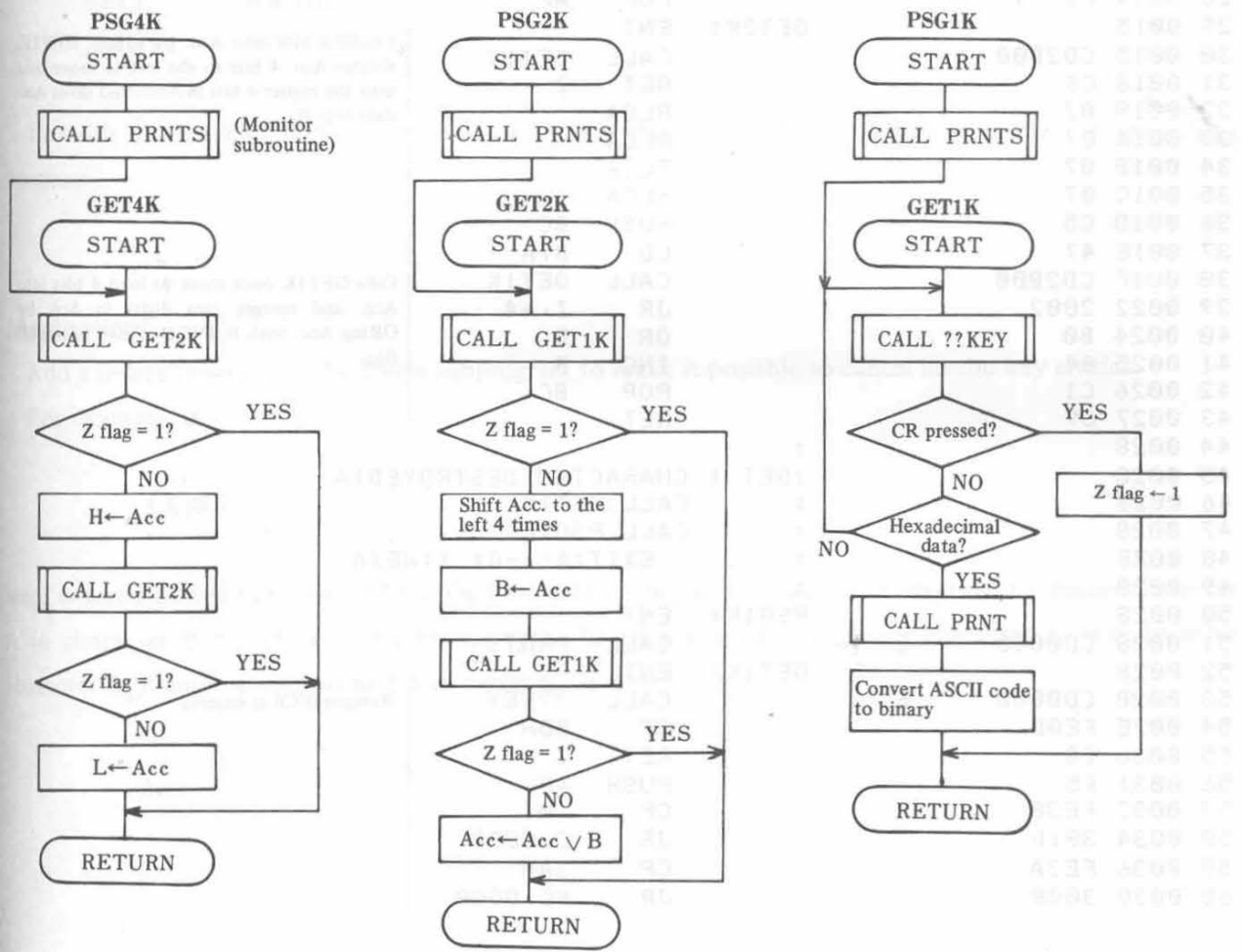
6.6 ENTERING HEXADECIMAL DATA

Let us construct a subprogram to read hexadecimal data from the keyboard, with the cursor to blink when prompting for data. Data is to be entered as one, two, or four digits, and the cursor is to flash until the required number of digits have been entered. A subprogram is to generate a beeper tone when an invalid code is entered, and the subprogram is to return with the Z flag set when a carriage return is entered.

The subprogram has six entry points as follows:

CALL GET4K	(get 4hexa data)	: Enters a 4-digit hexadecimal number into the HL register pair.
CALL PSG4K	(print space, get 4hexa data)	: Prints a space, then enters a 4-digit hexadecimal number into the HL register pair.
CALL GET2K	(get 2hexa data)	: Enters a 2-digit hexadecimal number into Acc.
CALL PSG2K	(print space, get 2hexa data)	: Prints a space, then enters a 2-digit hexadecimal number into Acc.
CALL GET1K	(get 1hexa data)	: Enters a 1-digit hexadecimal number into the lower 4 bits of Acc.
CALL PSG1K	(print space, get 1hexa data)	: Prints a space, then enters a 1-digit hexadecimal number into lower 4 bits of Acc.

The above subprograms are related to one another; GET4K calls GET2K twice and GET2K calls GET1K twice. GET1K also calls monitor subroutine "??KEY", which waits for character input while flashing the cursor. The flowcharts for these subprograms are as shown below.



```
   **  Z80 ASSEMBLER SB-2202   PAGE 01  **

01 0000                        ;
02 0000                        ;GET 4 CHARACTER(DESTROYED:A,H,L)
03 0000                        ;       CALL GET4K
04 0000                        ;       CALL PSG4K
05 0000                        ;         EXIT:HL<--XXXX X:HEXA
06 0000                        ;
07 0000                        PSG4K:  ENT
08 0000 F5                             PUSH   AF
09 0001 CD0000      E                  CALL   PRNTS
10 0004 F1                             POP    AF
11 0005                        GET4K:  ENT
12 0005 CD1500                         CALL   GET2K
13 0008 C8                             RET    Z
14 0009 67                             LD     H,A
15 000A CD1500                         CALL   GET2K
16 000D C8                             RET    Z
17 000E 6F                             LD     L,A
18 000F C9                             RET
19 0010                        ;
20 0010                        ;GET 2 CHARACTER(DESTROYED:A)
21 0010                        ;       CALL GET2K
22 0010                        ;       CALL PSG2K
23 0010                        ;         EXIT:A<--XX X:HEXA
24 0010                        ;
25 0010                        PSG2K:  ENT
26 0010 F5                             PUSH   AF
27 0011 CD0000      E                  CALL   PRNTS
28 0014 F1                             POP    AF
29 0015                        GET2K:  ENT
30 0015 CD2B00                         CALL   GET1K
31 0018 C8                             RET    Z
32 0019 07                             RLCA
33 001A 07                             RLCA
34 001B 07                             RLCA
35 001C 07                             RLCA
36 001D C5                             PUSH   BC
37 001E 47                             LD     B,A
38 001F CD2B00                         CALL   GET1K
39 0022 2802                           JR     Z,+4
40 0024 B0                             OR     B
41 0025 04                             INC    B
42 0026 C1                             POP    BC
43 0027 C9                             RET
44 0028                        ;
45 0028                        ;GET 1 CHARACTER(DESTROYED:A)
46 0028                        ;       CALL GET1K
47 0028                        ;       CALL PSG1K
48 0028                        ;         EXIT:A<--0X X:HEXA
49 0028                        ;
50 0028                        PSG1K:  ENT
51 0028 CD0000      E                  CALL   PRNTS
52 002B                        GET1K:  ENT
53 002B CD0000      E                  CALL   ??KEY
54 002E FE0D                           CP     0DH
55 0030 C8                             RET    Z
56 0031 F5                             PUSH   AF
57 0032 FE30                           CP     '0'
58 0034 381D                           JR     C,GGG2
59 0036 FE3A                           CP     3AH
60 0038 3008                           JR     NC,GGG0
```

Calls GET2K twice.

Loads 4 bits into Acc. by calling GET1K, rotates Acc. 4 bits to the left to move data into the higher 4 bits in Acc., and saves Acc. data into B.

Calls GET1K once more to load 4 bits into Acc. and merges two digits in Acc. by ORing Acc. with B; INC B is used to reset Z flag.

Returns if CR is entered.

```
   **  Z80 ASSEMBLER SB-2202   PAGE 02  **

01 003A CD0000      E          CALL   PRNT
02 003D F1                     POP    AF
03 003E D630                   SUB    30H
04 0040 180E                   JR     GGG1
05 0042 FE41              GGG0: CP     'A'
06 0044 380D                   JR     C,GGG2
07 0046 FE47                   CP     47H
08 0048 3009                   JR     NC,GGG2
09 004A CD0000      E          CALL   PRNT
10 004D F1                     POP    AF
11 004E D637                   SUB    37H
12 0050 FEF0              GGG1: CP     F0H
13 0052 C9                     RET
14 0053 F1                GGG2: POP    AF
15 0054 CD0000      E          CALL   BELL
16 0057 18D2                   JR     GET1K
17 0059                        END
```

Checks whether input data is a hexadecimal character; if so, converts it to binary and loads converted data into the lower 4 bits of Acc.

Returns with Z flag reset.

Generates a beeper tone if an invalid code is input.

This subprogram references the following monitor subroutines:

PRNTS	063AH
PRNT	063CH
BELL	0A80H
??KEY	0D77H

For this subprogram to be used as a subroutine, the above addresses must be defined in the calling program.

[Application]

Add a delete function to the above subprogram to make it possible to cancel invalid key entries.

For example, if

3A■

were entered during execution of the GET4K subroutine, all that would be required to change character A to character B would be to backspace with the DEL key once to delete character A, then to enter character B. Prepare a subroutine which performs as stated above.

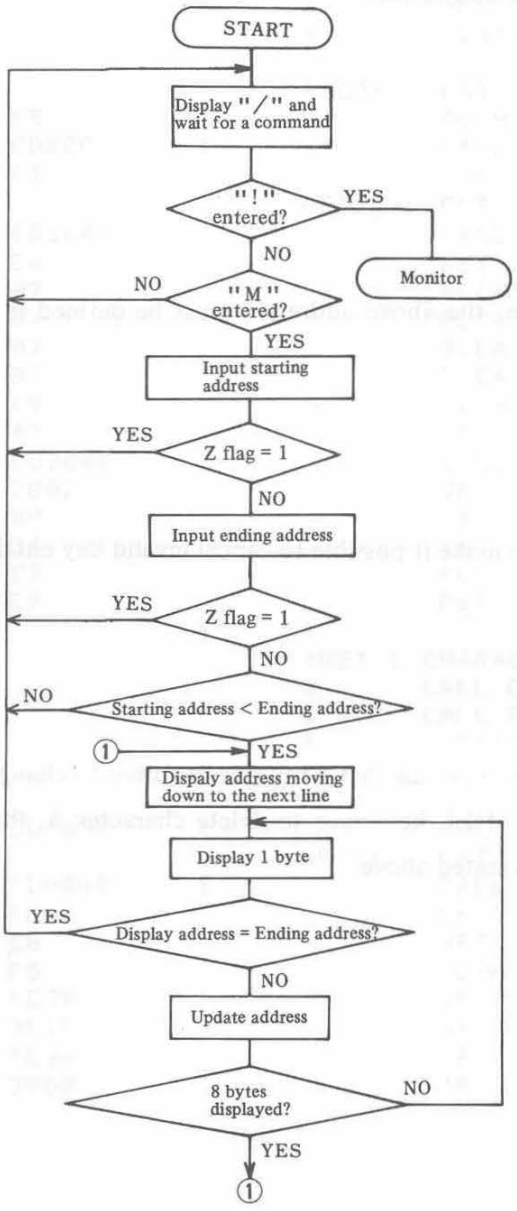
6.7 DISPLAYING A MEMORY BLOCK

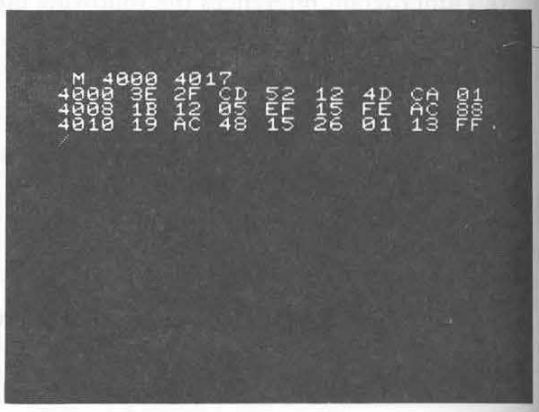
Let us construct a program which uses the hexadecimal data input and output subrpograms described above to display the contents of a specified memory block. The memory block must be specified in the same format as the M symbolic debugger command.

At the beginning of this command, the cursor is to blink to prompt for a command. There are to be two commands: M and !.

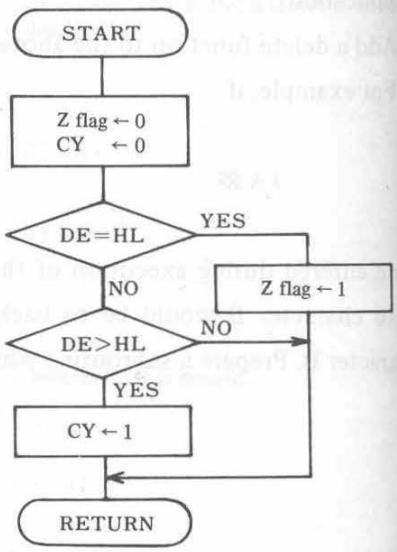
The memory dump is to be started with the M command and control is to be returned to the monitor by the ! command.

When the program is started with the M command, it is to wait for the starting address (a 4-digit hexadecimal number) at which the memory dump is to start. After the starting address is specified, the program is to wait for the ending address after printing a space. After the ending address is specified, the program is to start the memory dump, then return to the command wait state when the memory dump is completed.





Subroutine COMPR



```
   **  Z80 ASSEMBLER SB-2202   PAGE 01  **

01 0000                        ;
02 0000                        ; MEMORY DUMP
03 0000                        ;   M:START
04 0000                        ;   !:GOTO MONITOR
05 0000                        ;
06 0000                        MEMRY:  ENT
07 0000 CD0000      E                  CALL    NL
08 0003 3E2F                           LD      A,2FH
09 0005 CD0000      E                  CALL    PRNT
10 0008 CD0000      E                  CALL    ??KEY
11 000B FE21                           CP      '!'
12 000D CA0000      E                  JP      Z,MNTR
13 0010 FE4D                           CP      'M'
14 0012 2805                           JR      Z,+7
15 0014 CD0000      E          MEMR0:  CALL    BELL
16 0017 18E7                           JR      MEMRY
17 0019 CD0000      E                  CALL    PRNT
18 001C CD0000      E                  CALL    PSG4K
19 001F 28F3                           JR      Z,MEMR0
20 0021 EB                             EX      DE,HL
21 0022 CD0000      E                  CALL    PSG4K
22 0025 28ED                           JR      Z,MEMR0
23 0027 CD4300                         CALL    COMPR
24 002A 38E8                           JR      C,MEMR0
25 002C EB                             EX      DE,HL
26 002D CD0000      E          MEMR1:  CALL    NL
27 0030 CD0000      E                  CALL    4HEXO
28 0033 0608                           LD      B,8
29 0035 7E                     MEMR2:  LD      A,(HL)
30 0036 CD0000      E                  CALL    PS2HX
31 0039 CD4300                         CALL    COMPR
32 003C 28D6                           JR      Z,MEMR0
33 003E 23                             INC     HL
34 003F 10F4                           DJNZ    MEMR2
35 0041 18EA                           JR      MEMR1
36 0043                        ;
37 0043                        ; COMPARE DE,HL(DESTROYED:A)
38 0043                        ;   CALL COMPR
39 0043                        ;     EXIT:DE=HL  Z=1
40 0043                        ;          DE>HL  C=1
41 0043                        ;
42 0043                        COMPR:  ENT
43 0043 7C                             LD      A,H
44 0044 92                             SUB     D
45 0045 C0                             RET     NZ
46 0046 7D                             LD      A,L
47 0047 93                             SUB     E
48 0048 C9                             RET
49 0049                                END
```

Displays " / " after moving to the next line and waits for a command.

Returns to monitor if " ! " is entered and starts the memory dump if " M " is entered.

Returns to the command wait state if an invalid command is entered.

Input starting and ending addresses of the memory block to be dumped.
Returns to the command wait state if the starting address is greater than tne ending address.

Displays address after moving down to the next line.

Displays the memory dump (8 bytes on a line) until the ending address is reached.

Returns with Z flag set if DE = HL, and with C flag set if DE > HL.

6.8 WRITING DATA INTO A MEMORY AREA

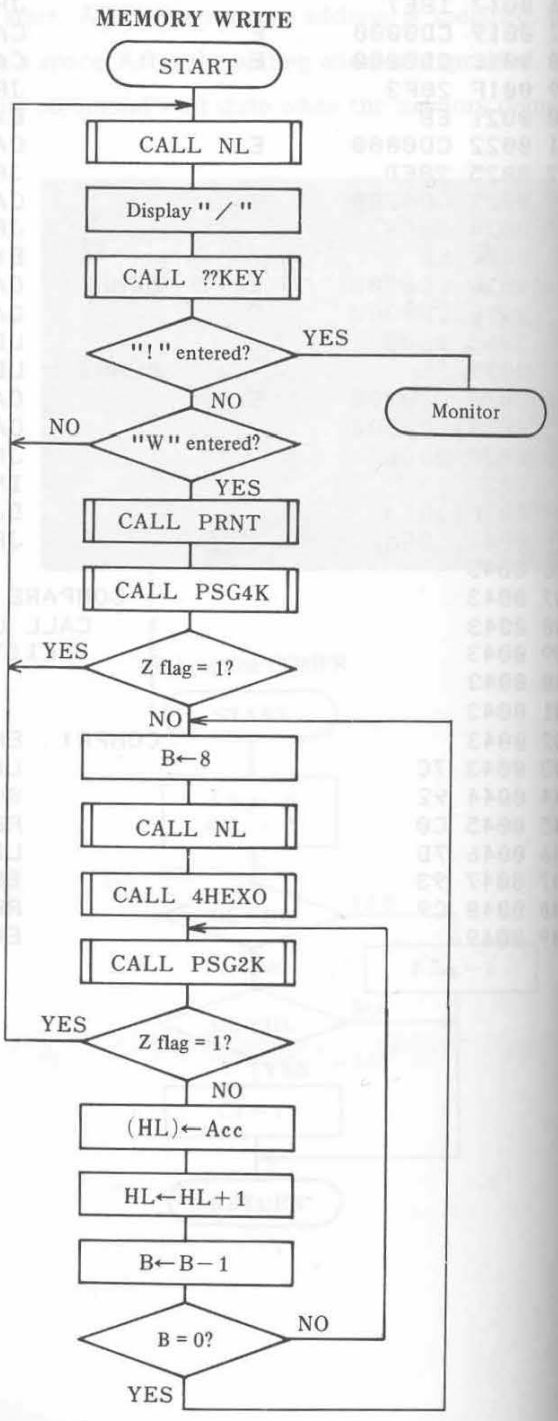
Let us construct a program to write 2-digit hexadecimal numbers into a memory block, starting at a specified address. The memory block is to be specified in the same format as with the W command.

At the beginning of this program, the program is to flash the cursor while waiting for command entry. Memory write is to be started with the W command and control is to be returned to the monitor with the ! command.

When the program is started with the W command, it is to prompt for the starting address (a 4-digit hexadecimal number) of the memory block at which the memory write is to start. After the starting address is specified, the program is to display the specified address on a new line and wait for the operator to enter 2-digit hexadecimal numbers.

Data entry is to be terminated with [CR] .

```
 W 4070
4070 41 38 1C BD 79 EF 15 20
4078 01 AF 31
```



```
   **  Z80 ASSEMBLER SB-2202   PAGE 01  **

01 0000                    ;
02 0000                    ; MEMORY WRITE
03 0000                    ;   W:START
04 0000                    ;   !:GOTO MONITOR
05 0000                    ;
06 0000                    WRITE:  ENT
07 0000 CD0000      E              CALL   NL
08 0003 3E2F                       LD     A,2FH
09 0005 CD0000      E              CALL   PRNT
10 0008 CD0000      E              CALL   ??KEY
11 000B FE21                       CP     '!'
12 000D CA0000      E              JP     Z,MNTR
13 0010 FE57                       CP     'W'
14 0012 2805                       JR     Z,+7
15 0014 CD0000      E      WRITE0: CALL   BELL
16 0017 18E7                       JR     WRITE
17 0019 CD0000      E              CALL   PRNT
18 001C CD0000      E              CALL   PSG4K
19 001F 28F3                       JR     Z,WRITE0
20 0021 0608                WRITE1: LD     B,8
21 0023 CD0000      E              CALL   NL
22 0026 CD0000      E              CALL   4HEXO
23 0029 CD0000      E      WRITE2: CALL   PSG2K
24 002C 28E6                       JR     Z,WRITE0
25 002E 77                         LD     (HL),A
26 002F 23                         INC    HL
27 0030 10F7                       DJNZ   WRITE2
28 0032 18ED                       JR     WRITE1
29 0034                            END
```

Displays " / " on a new line and waits for command entry.

Returns to the monitor if " ! " is entered and starts the memory write.

Returns to the beginning of program and sounds a beeper tone if an invalid key is pressed.

Input the starting address at which memory write is to start.

Input 8 bytes on a line and continues the memory write while displaying write addresses.
Data entry is terminated with [CR] .

This program references the following monitor and external subroutines:

PRINT	063CH	Monitor subroutines
NL	0757H	
BELL	0A80H	
??KEY	0D77H	
4HEXO	(4hexa data out)	See 6.5
PSG2K	(print space, get 2hexa data)	See 6.6
PSG4K	(print space, get 4hexa data)	See 6.6

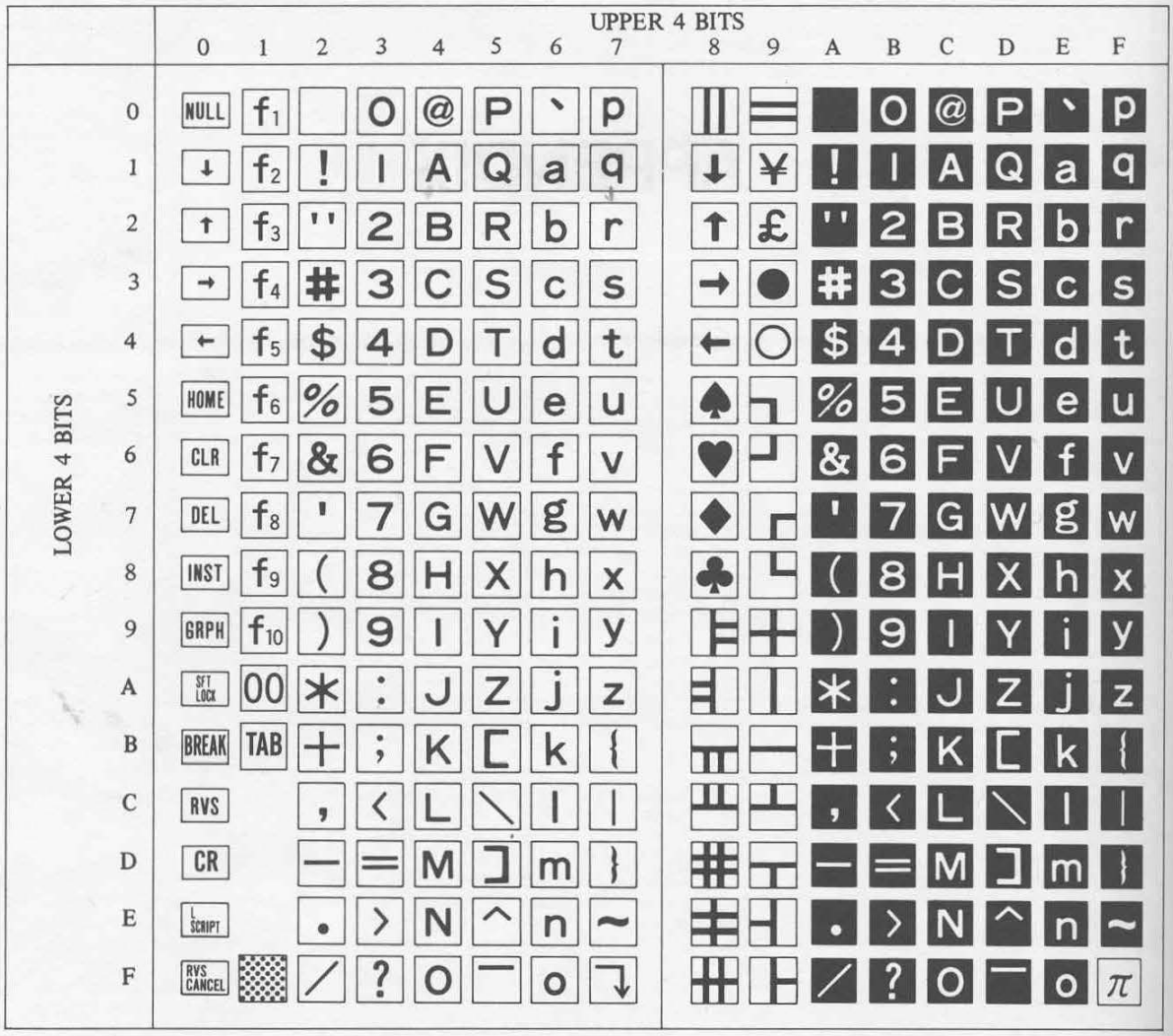
[Application]

Construct a machine-language monitor program which executes the W, M, and ! commands described above, as well as additional execution command G. The G command must input the starting address using the GET4K subroutine and execute the program (by loading the starting address into the program counter (PC)).

/W XXXX	memory write	
/M XXXX YYYY	memory dump	
/G XXXX	goto XXXX	..... Load XXXX into program counter.
/!	goto monitor	..... Jump to address 0000H.

APPENDIX

1. ASCII CODE TABLE



ASCII Codes of characters and control codes

Note: The column numbers shown above represent the first digit of hexadecimal numbers corresponding to characters, and the line numbers represent the second digit. For example, the ASCII code for the character A is represented in hexadecimal as 41H.

In the table above, codes 01H to 06H are used for cursor control. For example, if 05H is stored in register A, the cursor is moved to the home position when CALL PRNT is executed.

2. SYSTEM PROGRAM COMMANDS

2.1 Text editor commands

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.
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.
	Z	Moves the CP to the end of the text in the edit buffer.
Correction command	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.
	D	No operation.
Search command	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.
Comparison command	V	Compares the contents of the edit buffer with those of the input file whose filename is specified. Does not move the CP.
Special 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.
	X	Transfers control to the assembler.
	#	Changes the list mode for listing to the printer.
	!	Transfers control to the monitor.

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

2.2 Linker commands

Command name	Function
L (relocate Load)	Loads a program.
N (Next file)	Appends a program to a preceding program.
H (Height)	Displays the current assembly bias and load address.
T (Table dump)	Displays the contents of the symbol table.
S (Save)	Saves the object program in memory in a file.
V (Verify)	Compares the contents of the object file generated by the S command with the contents of the object program in memory.
X (TRANSfer)	Moves the specified memory block to the specified memory area.
* (clear table)	Clears the symbol table and resets the assembly bias and link address to 0000.
# (change printer mode)	Switches the printer mode.
! (go to monitor)	Transfers control to the monitor.

2.3 PROM formatter commands

The PROM formatter commands are listed below. In addition to these commands, it is possible to use the symbolic debugger commands under the PROM formatter program.

Command name	Function
FP (Format Punch)	Punches a specified link area block on paper tape.
FC (parity Form Change)	Changes the parity of the input or output tape.
FR (Format Read)	Reads a formatted program from paper tape into the link area.
FM (Format Message)	Displays a list of the formats available for the PROM formatter.

2.4 Symbolic debugger commands

Command type	Command name	Function
Link/load and symbol table commands	L	Loads a relocatable file into the link area. The program in the relocatable file is loaded to form an object program through relocation at the location designated by the assembly bias and link address (relocate Load).
	N	Appends a relocatable file to the end of the preceding program in the link area (Next file).
	H	Displays the current values of the assembly bias and link address (Height).
	T	Displays the contents of the symbol table. Each table entry consists of a label symbol name, its absolute address, and its definition status (Table dump).
	*	Clears the symbol table and current assembly bias and link address values to 0000H (Clear bias and table).
Debugging commands	B†	Displays, sets or alters a breakpoint. (Breakpoint)
	&	Clears all breakpoints set. (Clear breakpoints)
	M†	Displays the contents of the specified block in the link area in hexadecimal representation or alters them. (Memory dump)
	D†	Displays the contents of the specified block in the link area in hexadecimal representation with one instruction on a line. (memory list Dump)
	W†	Writes hexadecimal data, starting at the specified address in the link area. (Write)
	G†	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 filename into memory. (Yank)
	V	Compares the file whose filename is specified with the contents of the link area. (Verify)
Special commands	#	Switches the printer list mode for listing printout.
	!	Transfers control to the monitor.

Note: Commands marked by a dagger permit symbolic operations.

3. ERROR MESSAGES

3.1 Text editor error messages

Error Message	Meaning	Relevant commands
Full buffer	Edit buffer is full.	R, A
? ? ?	n < 0 in an nT or nJ command.	T, J
Large	n greater than 65535 was specified.	T, J, L, M, K, D, B, Z
Not found	The string (or string1) specified in Sstring, Cstring1 ▩ string2, or Qstring1 ▩ string2 was not found following the CP.	S, C, Q
Invalid	An illegal command was entered or an incorrect format was used. Ex.) ＊H [CR] : There is no H command. ＊S [CR] : A string should be specified.	any case
Check sum error	When the V command was executed, it was found that the contents of the edit buffer differed from the contents of the input buffer; or, an error occured while a file was being read.	V, R, A

3.2 Assembler messages

Definition status message	Meaning	Example
E (External)	Indicates that a label symbol is being referenced externally; that is, the label is not defined in the current source program unit.	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.
P (Phase)	Difines a label symbol with a constant assigned. This message is also output when a label symbol is encountered during pass 2 which was not encountered during pass 1.	P LETNL : EQU 0762H P DATA1 : EQU 3 └── 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 message	Meaning	Example
C (illegal Character error)	Indicates that an illegal character is used in the operand.	C JP +1000–3
F (Format error)	Indicates that the instruction format is incorrect.	
N (Non label error)	Indicates that no label symbol is specified for ENT or EQU.	N EQU 0012H └── No label symbol
L (erroneous Label error)	Indicates that an illegal label symbol is used.	L JR XYZ └── XYZ is not defined in the current program. 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.
M (Multiple label error)	Indicates that a label symbol is defined two or more times.	M ABC : LD DE, BUFFER ⋮ M ABC : ENT └── ABC is defined twice.
O (erroneous Operand)	Indicates that an illegal operand is specified.	
Q (Qestionable mnemonic)	Indicates that the mnemonic code is incorrect.	Q CAL XYZ CALL XYZ is correct.
S (String error)	Indicates that single or double quotation mark(s) are omitted.	S DEFM GAME OVER DEFM 'GAME OVER' is correct.
V (Value over)	Indicates that the value of the operand is out of the prescribed range.	V LD A, FF8H V SET 8, A V JR –130
U (Undefined parameter)	The number of operands specified in a macro call instruction was less than the number of parameters difined for the macro instruction.	U JP Z, @3
END?	Indicates that the END directive is missing from the source program.	

3.3 Linker messages

Error message	Meaning	Relevant commands
? ? ?	The specified address was outside the link area or the load address value was updated beyond the link area during a load operation.	L, N, S, X
Invalid	The format of the specified command is invalid. (Examples) * LL 12A0 [CR] The link address is missing. * LL 12 [CR] Fewer digits than required were specified.	L, S, V, X
Check sum error	A mismatch was found during a comparison between the contents of the link area and a file, or an I/O error occurred during a file read.	L, N, V
No power or no connection	The printer is not turned on or is not connected to the system.	#
Alarm	An error such as a paper jam occurred in the printer.	#
Paper empty	Printer is out of paper.	#

—Messages regarding the status of symbol definition (common to the linker and symbolic debugger)—

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

No message is issued for symbols defined. Messages U, M, X, and H are error messages.

3.4 Symbolic debugger error messages

Error message	Meaning	Relevant commands
? ? ?	An attempt was made to access a location outside the link area.	B , W , X , S , V
Error	An incorrect number of digits was specified or a digit other than a hexadecimal digit was entered during execution of a register (or memory) change command.	M , A , C , P
RST6?	A break point was set at an RST6 instruction.	B
Over	More than nine breakpoints were set.	B
Invalid	The format of the entered command is incorrect.	X , S , V
?	○ An invalid symbol (undefined label symbol or nonlabel symbol) was specified. ○ An attempt was made to clear a break point which was not set. ○ An attempt was made to set the break counter more than 14 (E in hexadecimal) times. ○ The format of the specified address is incorrect. ○ The starting address is not smaller than or equal to the ending address. ○ The destination and source blocks overlap.	B , W B B M , D , G M , D , W X
Check sum error	○ A mismatch was found between the contents of the link area and the object file being verified. ○ An error occurred while a file was being read.	V L , N , Y
No power or no connection	The printer is not turned on or is not connected to the system.	#
Alarm	An error such as paper jam occurred in the printer.	#
Paper empty	Printer is out of paper.	#

3.5 PROM formatter error messages

Error message	Meaning	Relevant commands
? ? ?	An attempt was made to access a location outside the link area.	FP , FR
?	The specified starting address is not smaller than or equal to the ending address.	FP
Not found	The specified file was not found.	FR
Format?	The format of the paper tape to be read does not match the specified format command.	FR
Parity?	The parity scheme of the paper tape to be read does not match the specified parity scheme.	FR
Check sum	A check sum error occurred while a paper tape in format C was being read.	FR (with format command C only)
Invalid	A command was specified in an invalid format.	FP , FR

(Note) The error messages associated with the symbolic debugger section are identical to those associated with the symbolic debugger.

4. 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 4-1**. 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 commands are characteristic of interline pointer movement commands.

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.

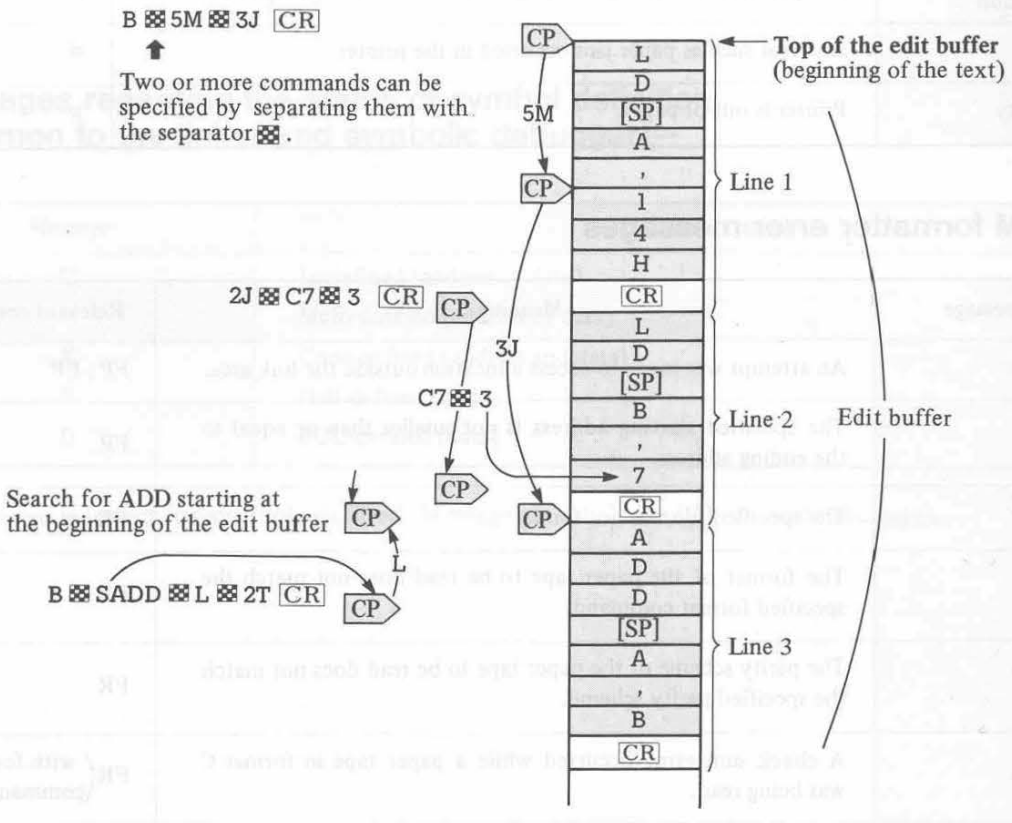


Fig. 4-1 Character pointer movement

5. ASSEMBLY PROCEDURES

Currently, many microprocessors other than the Z-80 (such as the 6800 and the 8048) are in use. However, the architecture and operating principles of these are similar in many respects. An obvious point is that this makes development of a general purpose assembler for these possible. This section describes the concept which serves as the starting point for such assemblers; this is the concept which is employed in the MZ-80 assembler.

The basic operation of any assembler is the interpretation of statements. It is therefore important to establish a proper statement coding format. **Figure 5-1** 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 5-2**) and a 2-dimensional operation table (**Table 5-1**) are introduced.

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 5-1**, 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 5-3**. 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.

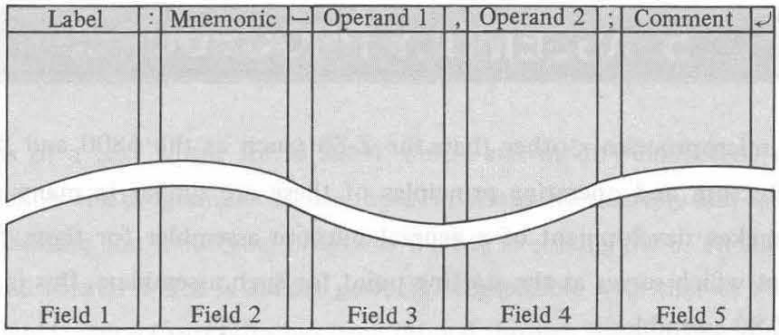


Fig. 5-1 Assembler coding format

```
01  0000                    :
02  0000                    : INSTRUCTION LIST
03  0000                    :
04  0000                    SYMP :    ENT
05  0000  4C442023                    DFFM    'LD #, #'          ;LIKE LD B, C
06  0004  2C23
07  0006  F1                          DFFB    F1H          F delimits the instruction pattern. 1 indicates the length of
                                                               the instruction in bytes.
08  0007  40                          DFFB    40H          Main portion of the mnemonic code
09  0008  01                          DFFB    01H          Atom type
10  0009  4C442023                    DFFM    'LD #, (IX$)'      ; LIKE  LD A, (IX+15)
11  000D  2C284958
12  0011  2429
13  0013  F3                          DFFB    F3H          3 indicates the length of the instruction in bytes.
14  0014  DD46                        DFFW    46DDH        DD4600 is the main portion of the mnemonic code.
15  0016  00                          DFFB    00H
16  0017  03                          DFFB    03H          Atom type
17  0018  4C442023                    DFFM    'LD #, (IY$)'      ; LIKE  LD B, (IY+AFC)
18  001C  2C284959
19  0020  2429
20  0022  F3                          DFFB    F3H
21  0023  FD46                        DFFW    46FDH
22  0025  00                          DFFB    00H
23  0026  03                          DFFB    03H
24  0027  4C442028                    DFFM    'LD  (IX$), #'     ; LIKE  LD  (IX+23), A
25  002B  49582429
26  002F  2C23
27  0031  F3                          DFFB    F3H
28  0032  DD70                        DFFW    70DDH
29  0034  00                          DFFB    00H
30  0035  04                          DFFB    04H
```

Fig. 5-2 Instruction list (part)

Table 5-1 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					

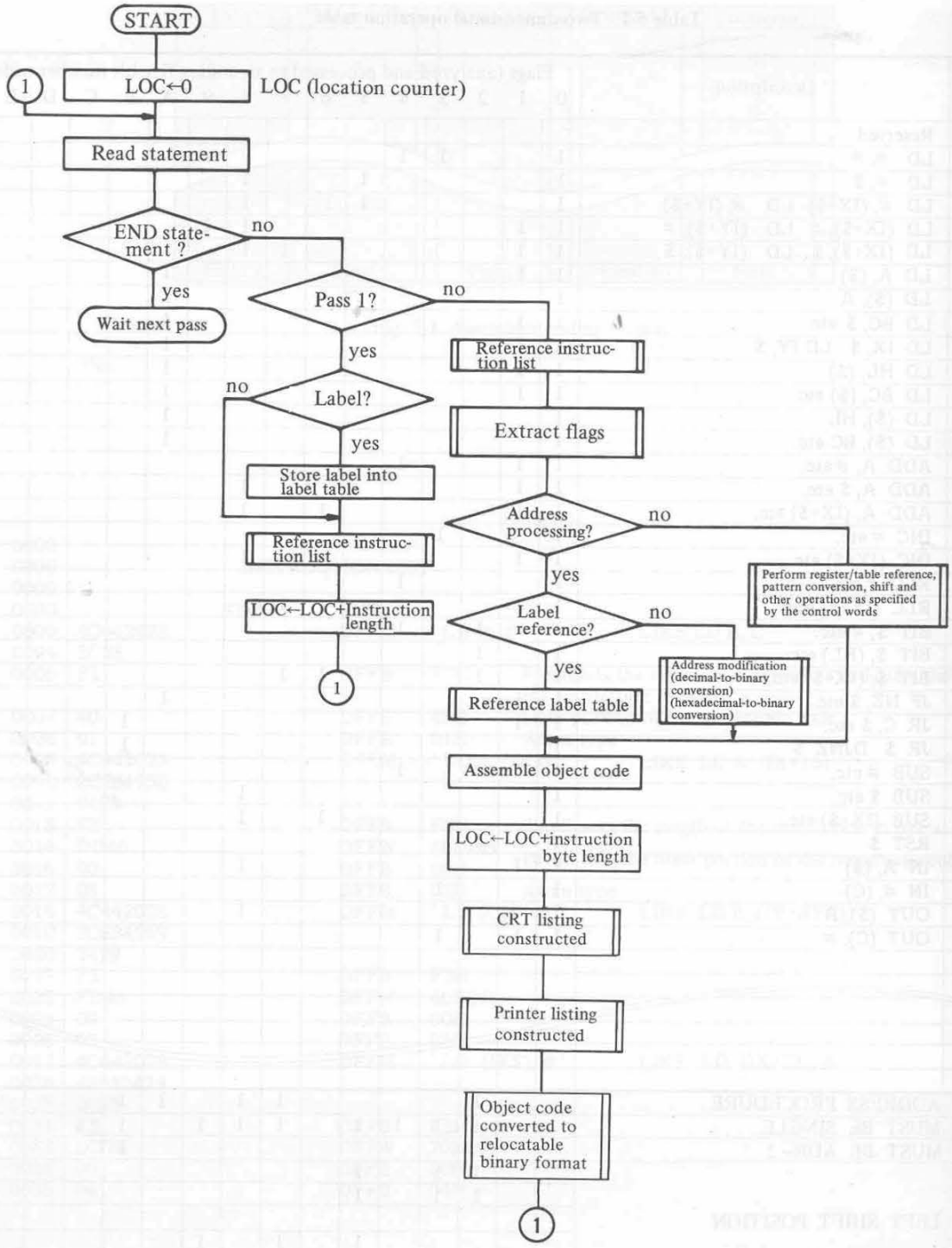


Fig. 5-3 General assembly flow (excluding assembler directive processing)

6. LINKER FUNCTIONS

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 6-2**). 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 6-3**).

When the first CALL SUB1 instruction is enccuntered 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 6-3(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 6-3(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 6-3(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.

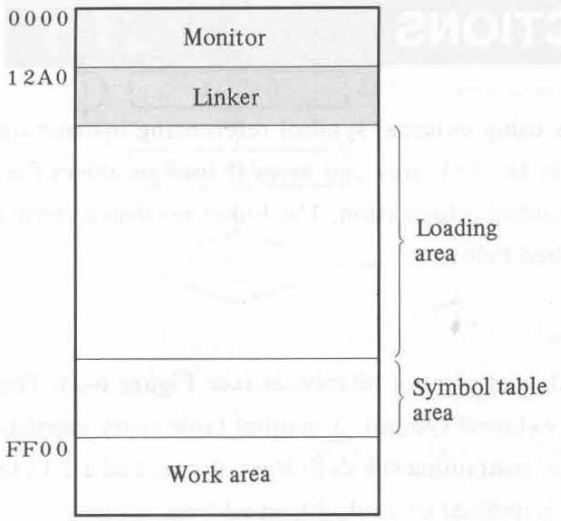


Fig. 6-1 Memory map for the linker

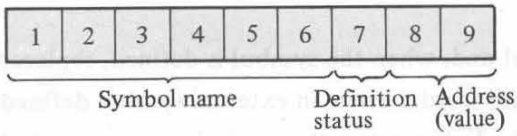
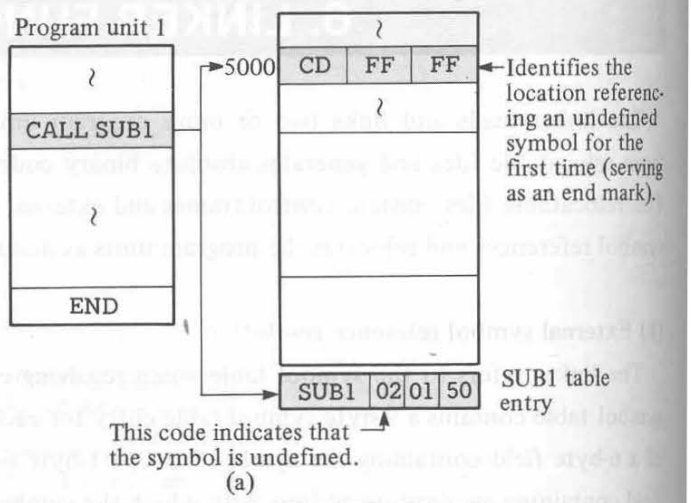
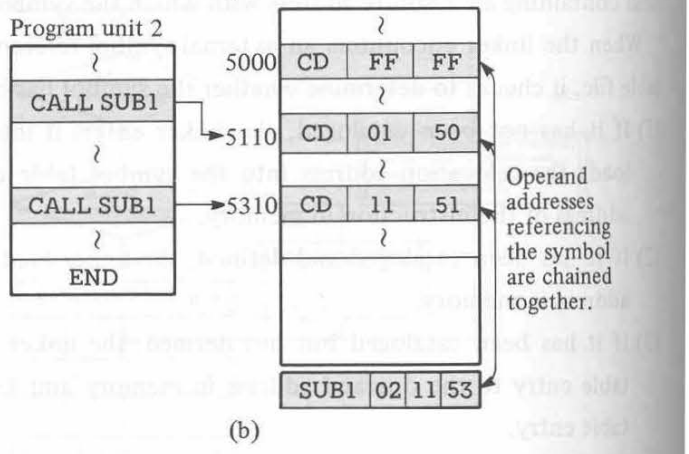


Fig. 6-2 Symbol table entry format





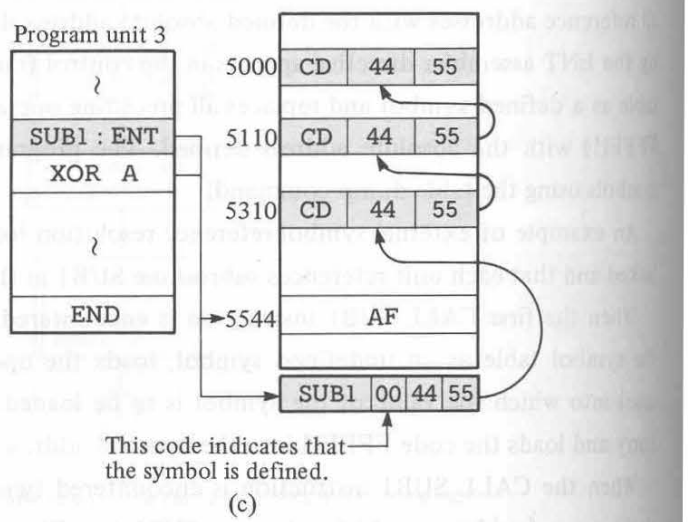


Fig. 6-3 Example of external symbol reference chaining

(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.

7. SYMBOLIC DEBUGGER FUNCTIONS

The symbolic debugger inputs relocatable files under the same input conditions as the linker except that it presumes that absolutable 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 7-1**).

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.

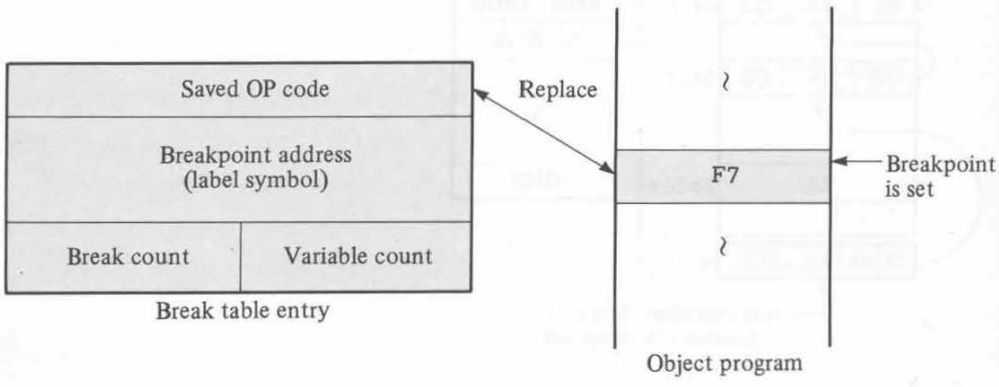
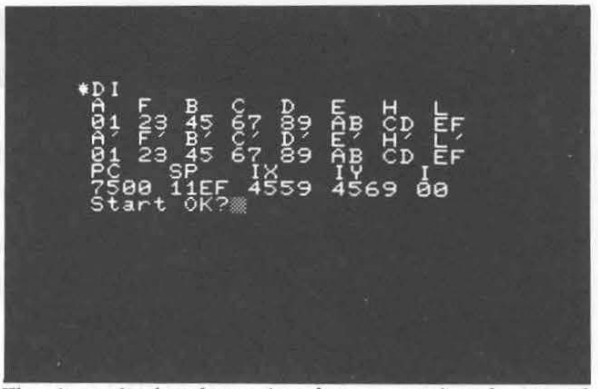


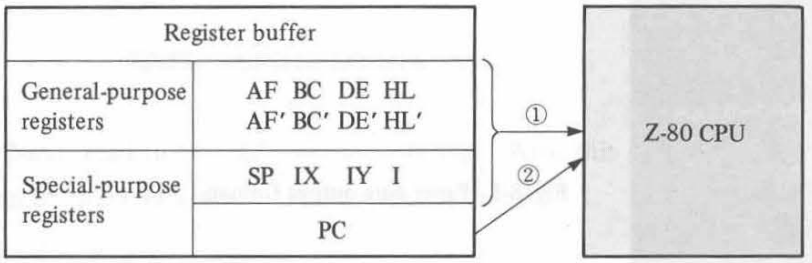
Fig. 7-1 Breakpoint setting and breakpoint table format

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 7-2** appears on the screen. When the programmer presses [CR] after confirming the CPU register contents, the debugger initiates an indicative start as shown in **Figure 7.3.**



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

Fig. 7-2 I command example



The debugger restores the contents of the general-purpose registers and special-purpose registers SP, IX, IY and I (①), then the value of the PC (②) and initiates program execution.

Fig. 7-3 I command operation

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 7-4.** 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 65535_{10} lines away from the address represented by the symbol

(3) An address ±1 to 65535_{10} 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 7-4** 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 7-4.**

```
START : ENT
        LD    SP , START
        CALL  LETNL
        XOR   A
        LD    (HL), A
        LD    B , A
MAINO : ENT
        LD    A , OFH
```

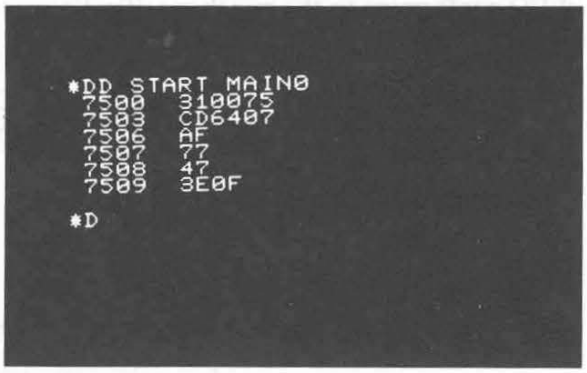


Fig. 7-4 D Command

8. PROM FORMATTER FUNCTIONS

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 8-1**):

(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.

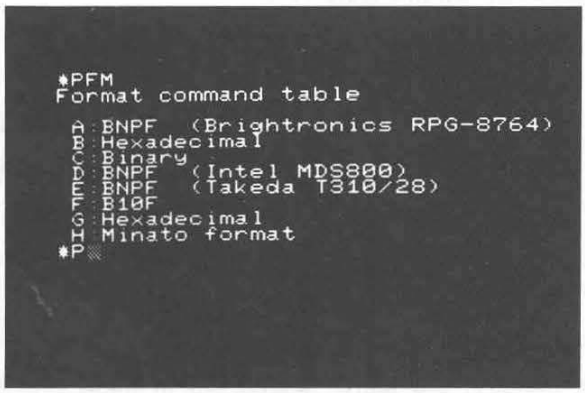


Fig. 8-1 Paper tape output formats

The PROM formatter is made up of format, the PTP and the PRT controls (See **Figure 8-2**). 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 FC (parity Form Change) command.

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-80IO2). 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.

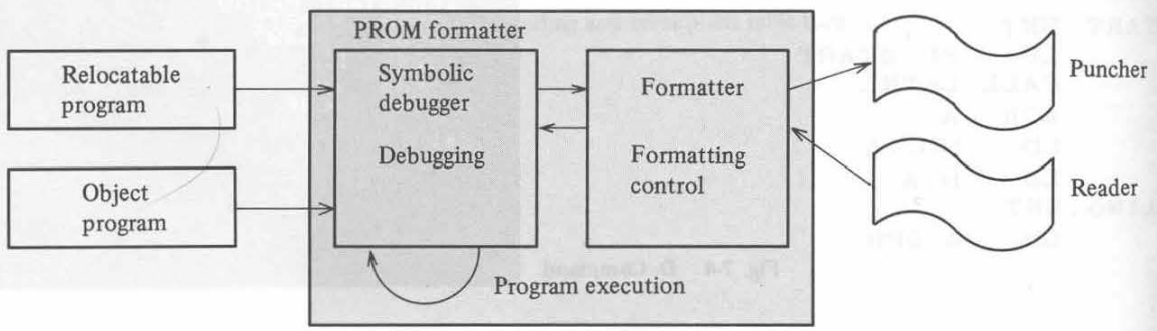


Fig. 8-2 Operation of the PROM formatter

9. CONVERTING MZ-80K TAPES TO 80B TAPES

The format of cassette tapes prepared using the MZ-80K system programs and FDOS must be converted for use with the MZ-80B. There are two methods of doing this as described below. However, the only two types of tapes which are convertable are those which contain source files (with file mode . ASC) or object files (with file mode . OBJ). Relocatable files (files with file mode . RB) and object files with symbol tables cannot be converted.

9.1 When FDOS is available —The Floppy Disk Based System—

First, execute the following command to link MZ-80K cassette tape I/O control routine $CMT1 with FDOS.

EXEC $FD1 ; LOADAUX

Next, read in the MZ-80K cassette tape. With this command, the filename must consist of a combination of characters which are permitted under FDOS.

XFER $CMT1 , $FDn (n = 1 ~ 4)

If the filename does not consist of a combination of characters which are permitted under FDOS, the file must be renamed as follows.

XFER $CMT1 , $FDn ; filename (n = 1 ~ 4)

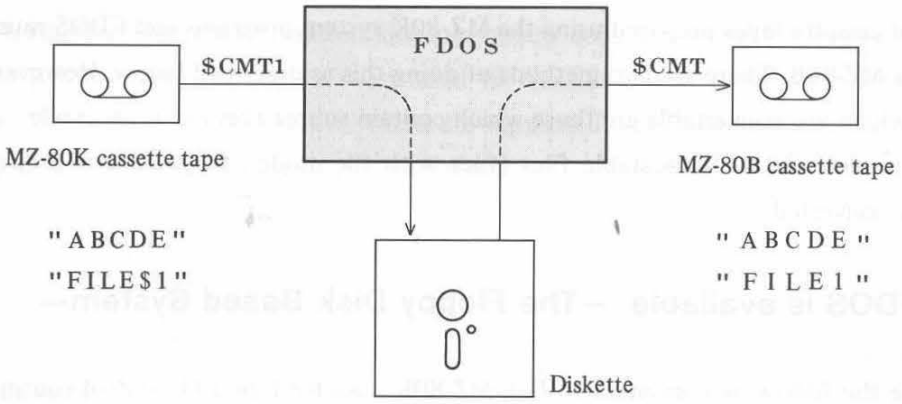
If the source file transferred is one which has been prepared for input to the MZ-80K assembler, check for any REL directives. This may be done by reading the applicable file with the text editor and conducting a search with the S command. Any REL directives found must be deleted. The reason for this is that the specifications of the MZ-80B assembler, linker, and symbolic debugger have been changed so that the REL directive is no longer required.

After deleting all REL directives, the required object file can be produced by assembling the file and producing the relocatable file.

Next, install the cassette tape for writing the file and execute the next command.

XFER $FDn ; filename , $CMT

The operations make it possible to produce an MZ-80B cassette tape with a filename which is either the same as that of the MZ-80K cassette tape or reassigned.



9.2 With a tape based system

Use the K-B converter, which is stored toward the rear end of the editor-assembler (around count 75).

— Load the K-B converter.

— The system displays the message " **K-source?.** " Set the K cassette tape, specify the filename, and press [CR] to locate and read in the applicable file. If no filename is specified, the first file encountered will be read in.

— " OK " is displayed if the file is read in without errors.

— The message " Check sum error " is displayed if an error occurs during the file read.

— Next, the system displays the message " **B-destination?.** "
As this point, set the cassette tape for writing the file, specify the filename, and press [CR] . If no filename is specified, that specified when the message " **K-source?** " is displayed will be used.

— When the write is normally completed, the message " **K-source?** " will be displayed again. Repeat the sequence if another tape is to be converted; otherwise, enter " ! " to terminate; the message shown below will then be displayed.

M)onitor B)oot C)ancel?

Pressing the M key transfers control to the monitor.
Pressing the B key transfers control to the IPL.
Pressing the C key returns to the beginning of the K-B converter.

10. LINKING FDOS WITH MACHINE LANGUAGE PROGRAMS

There are three methods of executing machine language programs which have been prepared with the tape based system under FDOS. These are discussed below.

10.1. Execution after transfer to a diskette with the XFER command

With this method, the program is transferred from cassette tape to a diskette in the manner described in section 9.1.

However, if the cassette tape has been prepared using the MZ-80B, $CMT1 is replaced with $CMT. In other words, for an MZ-80K cassette tape, execute

XFER $CMT1 , $FDn (; filename) (n = 1~4)

and with MZ-80B cassette tape, execute

XFER $CMT , $FDn (; filename) (n = 1~4)

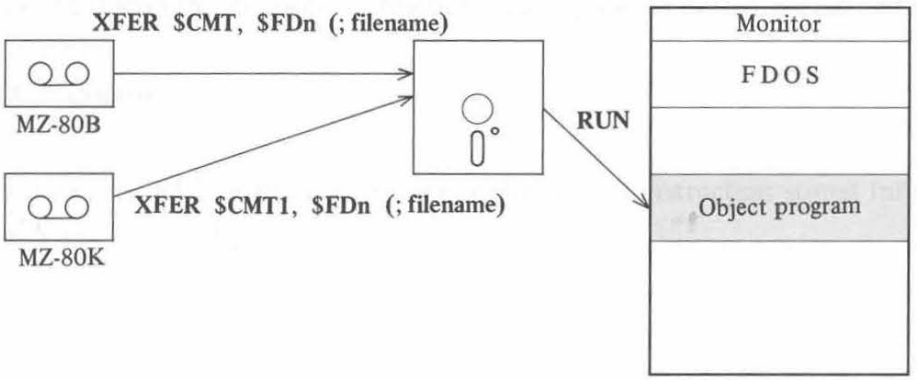
Further, when an object file prepared with the cassette based system programs is transferred to a diskette, it is executed as follows.

RUN $FDn ; filename

If the specified object file has loading addresses which would destroy the FDOS area, the following message is dispalyed on the CRT screen.

destroy FDOS?

If execution is desired even though the FDOS area will be destroyed, press [Y] . If destruction of the FDOS area is to be avoided, press [N] to return to the FDOS command wait state.



10.2. Direct execution using the RUN command

With this procedure, the program is loaded directly from the cassette tape and executed without transferring it to a diskette. The command is entered as follows.

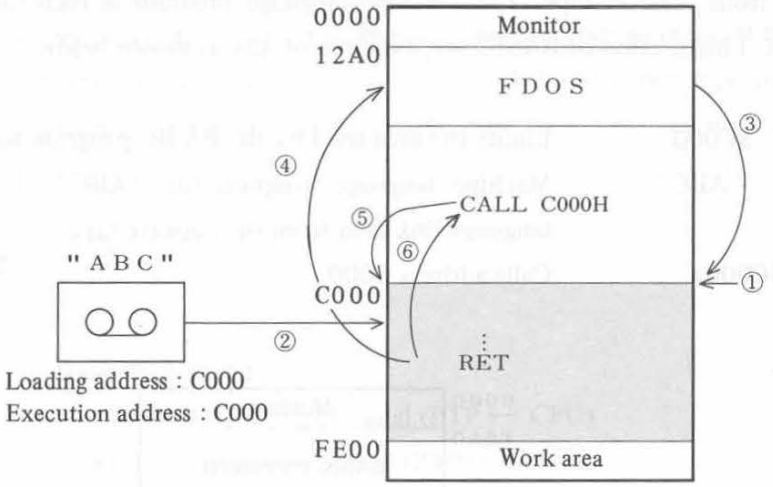
RUN	**$CMT (; filename)**	(for MZ-80B cassette tape)
RUN	**$CMT1 (; filename)**	(for MZ-80K cassette tape)

If an object program with the filename "ABC" is to be read in under FDOS, any of the following commands are effective.

RUN	**$CMT**	(In this case, file "ABC" must be the first one on the cassette tape.)
RUN	**$CMT ; ABC**	
RUN	**$CMT ; ABC . OBJ**	

10.3. Execution using the LIMIT command

By limiting FDOS memory management with the LIMIT command and loading the program into the excluded area with the LOAD command, the program can be executed. This procedure and the linking performed are shown in broad outline below.



①LIMIT $C000

Prepares a free area outside of FDOS management

②LOAD $CMT ; ABC

Loads the object program with the filename " ABC " from the cassette tape.

③RUN $C000

Transfers control to address C000.

④Control is returned to FDOS by the RET instruction.

It is also possible to execute the program outside of the FDOS area by loading it from cassette tape in advance, then calling it with the following instruction from a program executed under FDOS control.

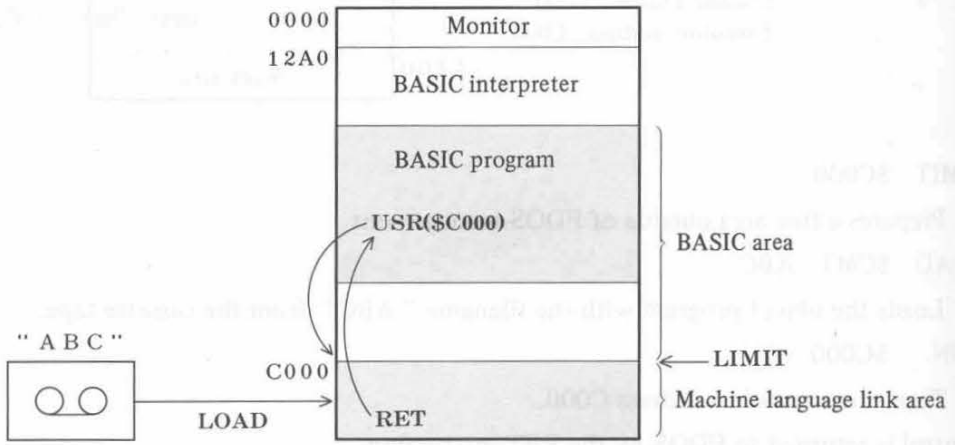
CALL C000H

When this is done, the RET instruction returns control to the instruction stored following the CALL instruction. (See 5 and 6 in the figure above.)

11. LINKING BASIC PROGRAMS WITH MACHINE LANGUAGE PROGRAMS

As is the case with FDOS, BASIC programs are linked with machine language programs by reserving a machine language program area with the LIMIT instruction of BASIC, then loading the machine language program from cassette tape. The machine language program is then called as a subroutine by means of the USR( ) instruction of BASIC. An example of this is shown below.

100 LIMIT $C000	Limits the area used by the BASIC program to address $BFFF.
200 LOAD " ABC "	Machine language program file "ABC" is read into the machine language link area from the cassette tape.
300 USR ($C000)	Calls address C000.



(Note) The monitor included in the BASIC interpreter is the SB-1510, but that included in the system programs is the SB-1511. When a machine language program is to be linked with the monitor of the BASIC interpreter, it is necessary to change the address of the monitor subroutine to be referenced.

12. PAPER TAPE PUNCHER AND READER INTERFACE

Normally, paper tape is used for PROM writer data input/output. Therefore, reformatted programs are output onto paper tape. To enable this, interfaces with the puncher and reader are required.

The method for controlling the paper tape puncher and reader is not standardized. A paper tape puncher 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 puncher and reader manufactured by Nada Electronics Laboratory. (For details, refer to the manual included with the paper tape puncher and reader.)

12.1 Signal name

<Puncher>

$\overline{DT_1} \sim \overline{DT_8}$	:	Data (PTP ← CPU)
$\overline{MI}$*	:	Motor ON/OFF control signal (PTP ← CPU)
$\overline{ST}$	:	START/STOP control signal (PTP← CPU)
$\overline{TO}$	:	Timing signal (PTP → CPU)
$(\overline{RDY})$**	:	Ready state signal (PTP → 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 → CPU)
$\overline{STA}$	:	START/STOP control signal (PTR ← CPU)
$\overline{SPR}$	:	Sprocket signal (PTR → CPU)
$\overline{RB}$	:	Tape end signal (abnormal stop signal) (PTR → 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.

12.2 I/O ports

Port FCH is used for data by both the puncher and the reader. Port FDH is used for control signals. Care is required when designing interfaces since the control signals for the puncher and the reader both share the same port. As is apparent from Table 12-1, the control signal for the reader is supplied to the MSD side of the control signal port, while that for the puncher is supplied to the LSD side.

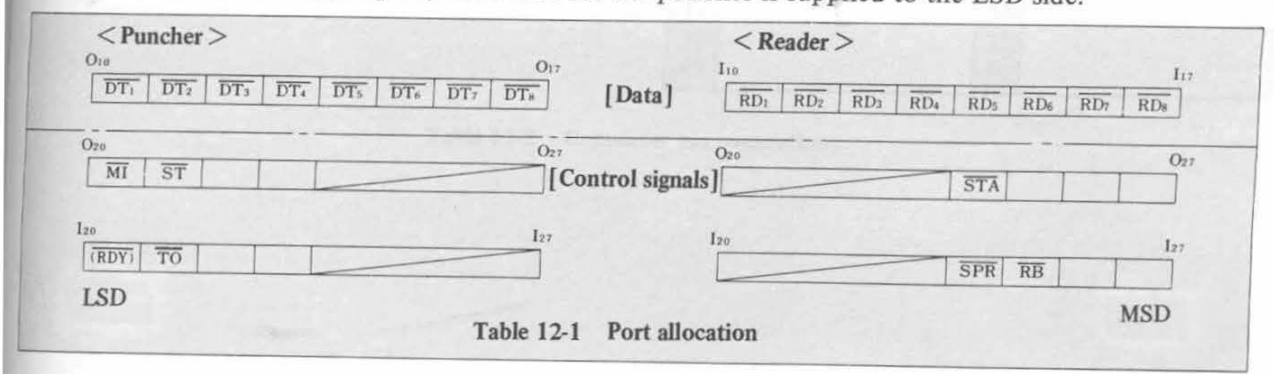


Table 12-1 Port allocation

12.3 Timing chart

Puncher

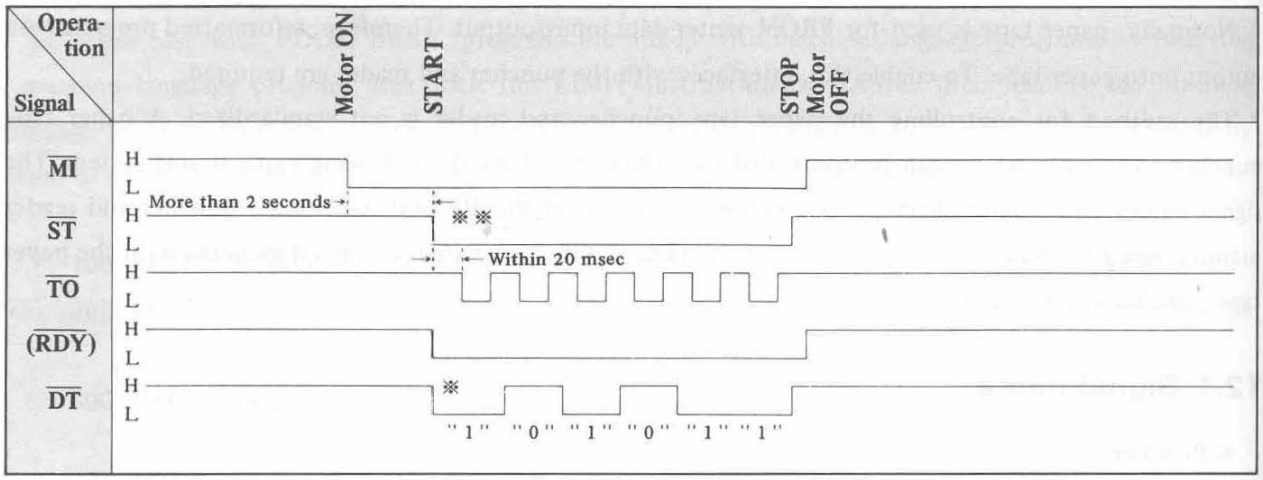


Figure 12-1 Puncher timing chart

* 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

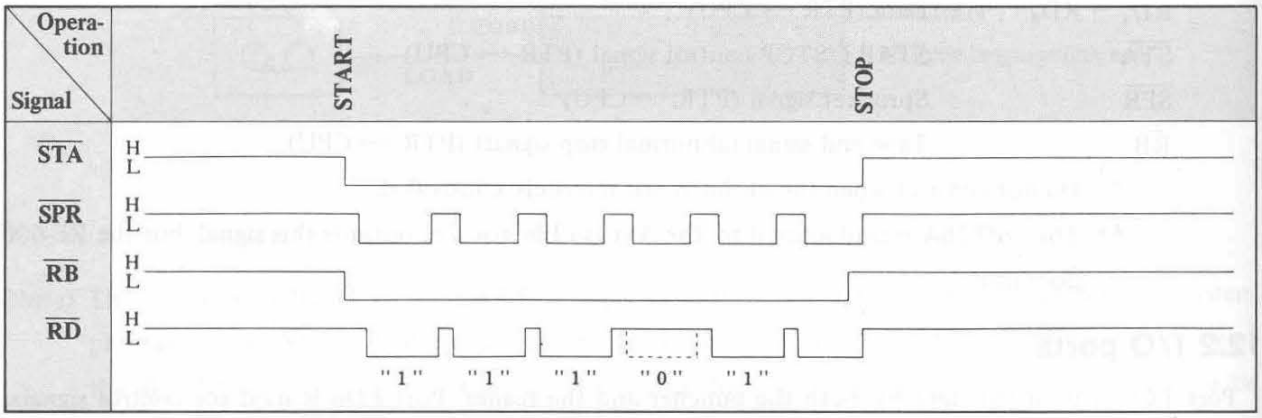


Figure 12-2 Reader timing chart

12.4 Preparing a paper tape puncher/reader I/O card

It is convenient to use a universal I/O card (MZ-80IO2) for preparing a paper tape puncher and reader I/O interface circuit. Markings such as O_{10} or O_{17} in the port allocation table on page 171 match those on the universal I/O card. See page 174 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).

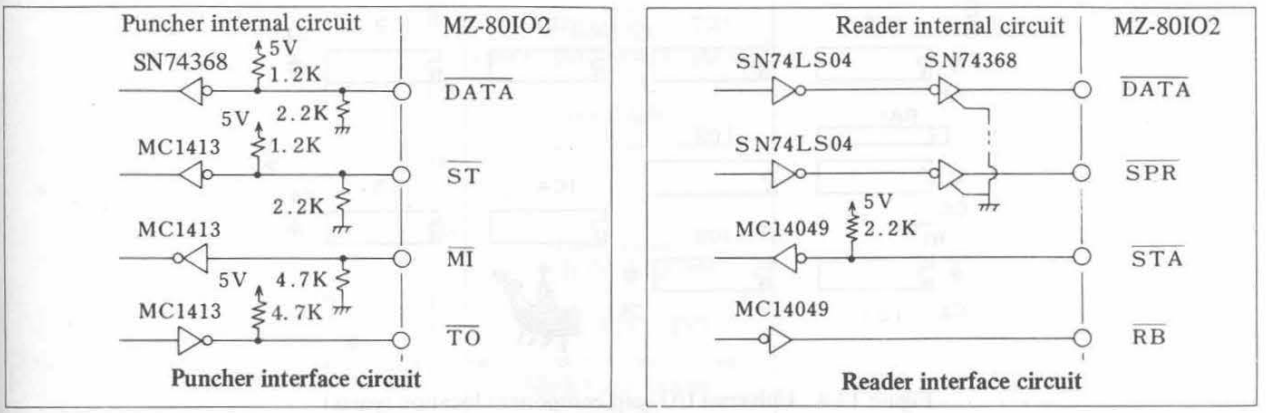


Figure 12-3 Interface circuit (RP-600)

Puncher 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	$\overline{DT}_5$ Data	18	
6	$\overline{DT}_6$	19	
7	$\overline{DT}_7$	20	
8		20	
9	$\overline{DT}_8$	21	
10		22	$\overline{MI}$ Motor ON/OFF
11	$\overline{TO}$ Timing signal		signal
12	GND	23	$\overline{ST}$ START/STOP signal
13		24	FG Frame ground

Reader I/O connector

Pin	Signal	Pin	Signal
1	$\overline{RD}_1$	20	
2	$\overline{RD}_2$	21	
3	$\overline{RD}_3$	22	
4	$\overline{RD}_4$ Data	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
10			signal
11		29	$\overline{RB}$ Operating state
12	GND	30	FG Frame ground
13		31	
14		32	
15		33	
16		34	
17		34	
18		35	
19		36	

Table 12-2 Connector pin connections

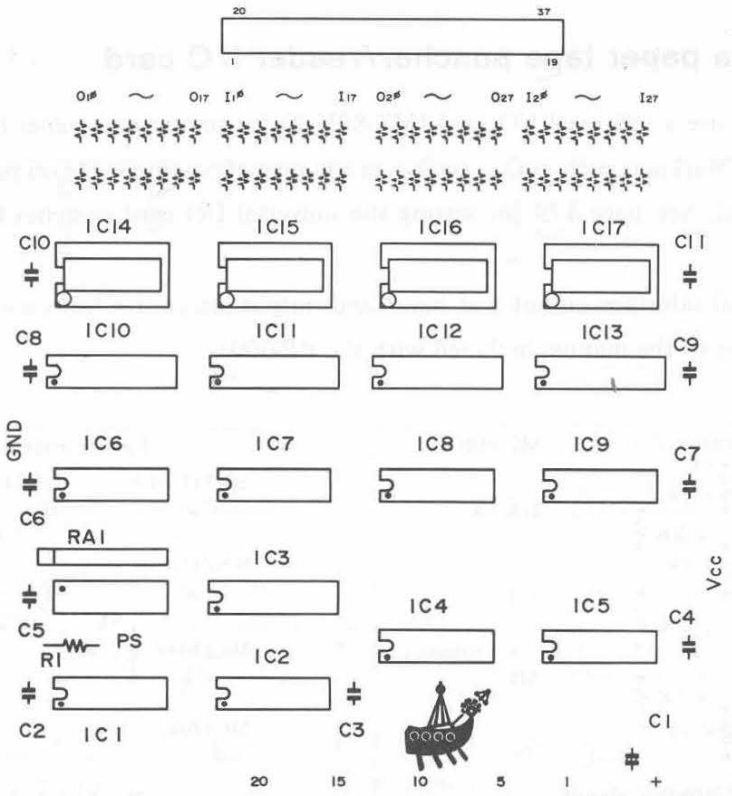


Figure 12-4 Universal I/O card component location (parts)

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, the MZ-80B uses those from B0H on.)

The input port for $I_{10} \sim I_{17}$ is set to an even address.

The input port for $I_{20} \sim I_{27}$ is set to an odd address.

The output port for $O_{10} \sim O_{17}$ is set to an even address.

The output port for $O_{20} \sim O_{27}$ 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".

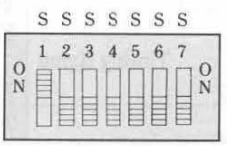
Switch No.	7	6	5	4	3	2	1
Address bit	A_7	A_6	A_5	A_4	A_3	A_2	A_1

Example: Setting the PS switches as shown below sets the port address to FC_H.

1	1	1	1	1	1	0	0
↑	↑	↑	↑	↑	↑	↑	
S_7	S_6	S_5	S_4	S_3	S_2	S_1	

When the PS switches are set as shown above, ports FC_H and FD_H are used for this card.

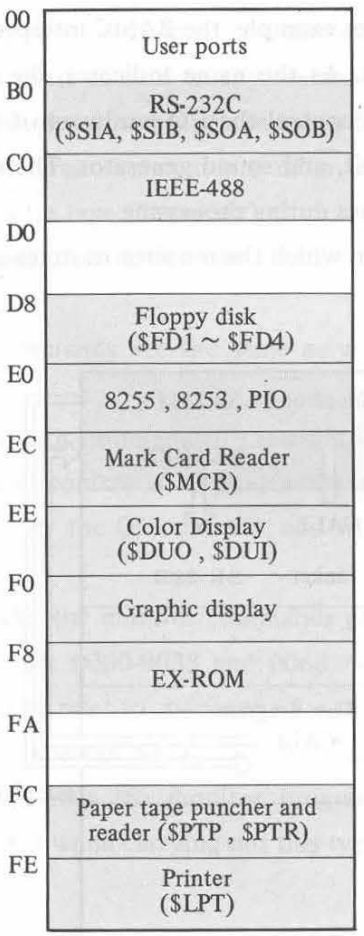
S_7 ..... OFF
S_6 ..... OFF
S_5 ..... OFF
S_4 ..... OFF
S_3 ..... OFF
S_2 ..... OFF
S_1 ..... ON



Caution: Installing two or more interface cards which have the same port address settings will result in destruction of ICs.

13. 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.



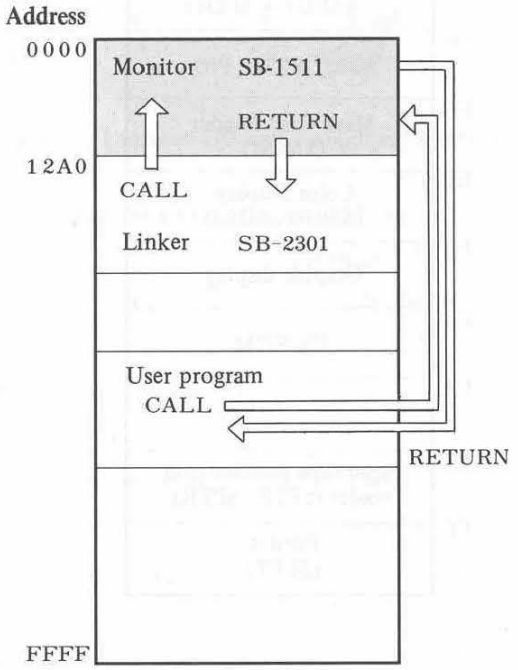
14. MONITOR

14.1 Functions of monitor SB-1511

The monitor program is one of the most basic system programs of the MZ-80B personal computer. Consideration of its functions can be divided into several phases.

First, all MZ-80B system software (for example, the BASIC interpreter, tape based system program, and FDOS) employ the monitor functions. As the name indicates, the monitor contains subroutines which perform the basic logic operations and control the I/O hardware of the MZ-80B; e.g., its keyboard, CRT display, cassette tape deck, timer circuit, and sound generator. The main system programs of the system software constantly call these subroutines during processing.

The figure below shows the manner in which the monitor routines are referenced in the linker.



Monitor subroutine calls

The reason for providing a single, independent monitor with general applicability for each of the MZ-80B processing systems (rather than include separate monitors in each main system program such as BASIC) is to make the system more orderly. By doing this, various controls which are directly linked to the hardware are centrally consolidated in the monitor; since this is located between the hardware and the software, it is referred to as firmware.

The size of monitor SB-1511 is about 4.5K bytes; as is shown in the figure above, it is stored in main memory starting at address 0000.

The second function of the monitor is to act as a convenient machine language program monitor when control is transferred to the monitor command level control. In other words, the monitor is not simply a collection of subroutines which are used in common by system and user programs; system control itself can be transferred from the system programs to the monitor and the following commands used to create programs, generate data, and actively control system oprations such as file input/output.

M command ..... Rewrites the contents of the memory (memory correction).
D command ..... Displays the contents of a memory block (memory dump).
J command ..... Transfers control to any desired address and executes programs (jump).
S command ..... Saves the contents of a memory block onto cassette tape (save).
V command ..... Compares the contents of a memory block with the contents of a cassette tape file (verify).
L command ..... Loads cassette tape files (load).

The functions of all of these commands are the same as with monitor SB-1510. See the Monitor Reference Manual for details.

All system programs are provided with cammands for transferring control to the monitor program. For example, with the BASIC interpreter, control is transferred by means of the MON command; with the PASCAL interpreter, it is transferred by the Q/command; and with the editor-assembler, it is transferred by means of the " ! " command.*

Since the monitor operates in RAM, the monitor commands can be used to change the monitor itself. For example, the contents of addresses 0000-0038 and 0066, which are called in response to the CPU restart instruction and interrupts, can be reset to change the functions of the monitor subroutines.

Further, the fact that the cassette tape can be freely written and read out means that an independent machine language program which includes the monitor program itself can be filed. See the monitor program assembly listing in section 14.3 when carrying out this type of operation.

* See the manuals for BASIC, PASCAL, the editor-assembler, FDOS, and so forth.

14.2 Monitor subroutines

The subroutines of monitor SB-1511 are shown in Table 14-1. The subroutine names shown in this table are the labels appearing in the monitor program assembly listing in section 14.3. These labels symbolically express the functions of the various subroutines.

The contents of registers preserved are not changed upon return from a monitor subroutine call, but the contents of other registers generally are changed. Consideration must be given to this fact when calling monitor subroutines.

To call a monitor subroutine, use machine language instruction CALL or the USR function of BASIC as shown below.

CALL subroutine address

USR ($ subroutine address)

The subroutine address is specified as a 4-digit hexadecimal number.

For example, to start a new line by calling monitor subroutine LETNL, execute CALL 0764H (CD6407 (hex) in machine language) or USR ($0764).

Table 14-1 Subroutines of monitor SB-1511

Subroutine name (hexadecimal address)	Function	Registers preserved
CALL WRINF (021DH)	Writes the current contents of a certain part of the header buffer (described later) onto the tape, starting at the current tape position. Return conditions C flag = 0 No error occurred. C flag = 1 BREAK was pressed.	All registers except AF
CALL WRDAT (024EH)	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. Return conditions C flag = 0 No error occurred. C flag = 1 BREAK was pressed.	All registers except AF
CALL RDINF (025FH)	Reads the first CMT header found starting at the current tape position into a certain part of the header buffer. Return conditions C flag = 0 No error occurred. C flag = 1, A = FFH A check sum error occurred. C flag = 1, A ≠ FFH BREAK was pressed.	All registers except AF
CALL RDDAT (027DH)	Reads in the CMT data block according to the current contents of a certain part of the header buffer. Return conditions C flag = 0 No error occurred. C flag = 1, A = FFH A check sum error occurred. C flag = 1, A ≠ FFH BREAK was pressed.	All registers except AF
CALL VERFY (0286H)	Compares the first CMT header found starting at the current tape position with the contents of the memory area indicated by the header. Return conditions C flag = 0 No error occurred. C flag = 1, A = FFH A match was not obtained. C flag = 1, A ≠ FFH BREAK was pressed.	All registers except AF
CALL BRKEY (0527H)	Checks whether BREAK was pressed. Z flag is set if it was pressed, and Z flag is reset if it was not.	All registers except AF
CALL PRTHL (0568H)	Displays the contents of the register pair HL on the display screen as a 4-digit hexadecimal number.	All registers except AF
CALL PRTHX (056DH)	Displays the contents of the A register on the display screen as a 2-digit hexadecimal number.	All registers except AF
CALL ASCI (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
CALL HEX (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. When C flag = "0" upon return A ← hexadecimal When C flag = "1" upon return A is not assured.	All registers except AF
CALL HLHEX (05A2H)	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. ∴ DE ← starting address of the ASCII string (e.g., "3" "1" "A" "5") DE CALL HLHEX C flag = 0 HL ← hexadecimal number (e.g., HL = 31A5H) C flag = 1 HL is not assured.	All registers except AF and HL

Table 14-1 Subroutines of monitor SB-1511 (Continued)

Subroutine name (hexadecimal address)	Function	Registers preserved
CALL 2HEX (05B1H)	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. DE ← starting address of the ASCII string (e.g., "3" "A") ↑——DE CALL 2HEX C flag = 0 A ← hexadecimal number (e.g., A = 3AH) C flag = 1 A is not assured.	All registers except AF and DE
CALL GETKY (0610H)	Takes one character only into the A register from the keyboard. For example, when this subroutine is called with [B] held down, ASCII code 24H, 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
CALL PRNTS (063AH)	Displays one space only at the cursor position on the display screen.	All registers except AF
CALL PRINT (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
CALL MSGX (06AFH)	Almost the same as MSG, except that cursor control codes are for reverse character display.	All registers except AF
CALL MSG (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
CALL ?DPCT (0714H)	Controls the display on the display screen. The relationship between A register at the time of the call and control is as follows. A register 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

Table 14-1 Subroutines of monitor SB-1511 (Continued)

Subroutine name (hexadecimal address)	Function	Registers preserved
CALL NL (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
CALL LETNL (0764H)	To change the line and set the cursor to the beginning of the next line.	All registers except AF

CALL ?PONT (0904H)	Sets the current position of the cursor on the display screen in register pair HL. The return conditions are as follows.
	CALL ?PONT
	HL ← cursor position on the display screen (V-RAM address)
	(Note) The X–Y coordinates of the cursor are contained in DSPXY (10D1H). The current position of the cursor is loaded as follows.
	LD HL, (DSPXY) ; H ← Y coordinate on the screen.
	L ← X coordinate on the screen.
	The cursor position is set as follows.
	LD (DSPXY), HL
		All registers except AF and HL

CALL CHR80 (0958H)	Sets the number of characters per line on the CRT screen to 80.	All registers except AF, BC, DE and HL
CALL CHR40 (098FH)	Sets the number of characters per line on the CRT screen to 40.	All registers except AF, BC, DE and HL
CALL XTEMP (09BEH)	Sets the musical tempo. The tempo data (1 to 7) is set in and called from A register. A ← 01H Slowest A ← 04H Medium speed A ← 07H Fastest 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
CALL TIMST (09CAH)	Sets the built-in clock. (The clock is activated by this call.) The call conditions are. A ← 0 (AM), A ← 1 (PM) DE ← the time in seconds (2 bytes)	All registers except AF
CALL TIMRD (0A16H)	Reads the value of the built-in clock. The conditions upon return are: A ← 0 (AM), A ← 1 (PM) DE ← the time in seconds (2 bytes)	All registers except AF and DE
CALL BELL (0A80H)	Sounds a momentary tone (approximately 880 Hz)	All registers except AF
CALL MELDY (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 BREAK was pressed.	All registers except AF

Table 14-1 Subroutines of monitor SB-1511 (Continued)

Subroutine name (hexadecimal address)	Function	Registers preserved
CALL GETL (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 [CR] (or [ENT]), 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 [BREAK] 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 BUFER (1100H) and location KNUMBS loaded with 39 (27H).	All registers
CALL GETCRT (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
CALL ??KEY (0D77H)	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.	All registers except AF
CALL PUSHR (0DF1H)	Pushes registers IX, HL, DE and BC. The RET instruction at the end of this subroutine then automatically POPs these registers. SUBR : CALL PUSHR RET Z ; POP and RET if Z flag = 1 RET ; POP and RET	All registers except IX
CALL PUSHR2 (0DFDH)	Pushes registers IX, HL and BC. The RET instruction at the end of this subroutine then automatically POPs these registers. SUBR2 : CALL PUSHR2 RET Z ; POP and RET if Z flag = 1 RET ; POP and RET	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.

Table 14-2 Header buffer of monitor SB-1511

Address	Contents
IBUFE (1180H)	This byte indicates one of the following file modes. 01 Object file (machine language program) 02 BASIC text file 03 BASIC data file PASCAL interpreter data file Source file (ASCII file) 05 Relocatable file (relocatable binary file) A0 PASCAL interpreter text file
IBU1 (1181H)	These 17 bytes indicate the filename. However, since 0DH is used as the end mark, in actuality the filename is limited to 16 bytes. Example: [S][A][M][P][L][E][0D]
IBU18 (1192H)	These two bytes indicate the byte size of the data block which is to follow.
IBU20 (1194H)	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 ".
IBU22 (1196H)	These two bytes indicate the execution address of the data block which is to follow.
IBU24 (1198H)	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 filename is 'SAMPLE'.
1182	'A'	
1183	'M'	
1184	'P'	
1185	'L'	
1186	'E'	
1187	0D	
1188 – 1191	Variable	
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	

14.3 Monitor SB-1511 assembly list

```
   **  Z80 ASSEMBLER SB-7201  <SB-1511>  PAGE 01          02.23.82

01 0000                    ;  ------------------------------------
02 0000                    ;  MZ-80B Monitor SB-1511
03 0000                    ;  Copyright 1981 by SHARP Corp.
04 0000                    ;  ------------------------------------
05 0000                    ;
06 0000                          ORG    0
07 0000                    ;------------------------------ RST0
08 0000 C33B00             MONIT: JP     START
09 0003                    ;
10 0003                          DEFS   5
11 0008                    ;------------------------------ RST1
12 0008 C3A509                   JP     REGIST
13 000B                    ;
14 000B                    SCROST: ENT                    ;SCROL START LINE +1
15 000B 01                       DEFB   01H
16 000C                    SCREND: ENT                    ;SCROL END LINE
17 000C 18                       DEFB   18H
18 000D                    REPTCT: ENT                    ;AUTO REPEAT SPEED
19 000D 400C                     DEFW   0C40H
20 000F                    SLOW:   ENT                    ;SLOW SCROL
21 000F 80                       DEFB   80H
22 0010                    ;------------------------------ RST2
23 0010 C3A509                   JP     REGIST
24 0013                    ;
25 0013                    KDATW:  DEFS   2
26 0015 FF                 SWRK:   DEFB   FFH
27 0016                    TEMPW:  DEFS   1
28 0017                    GATESI: ENT
29 0017 00                       DEFB   0
30 0018                    ;------------------------------ RST3
31 0018 C3A509                   JP     REGIST
32 001B                    ;
33 001B                          DEFS   5
34 0020                    ;------------------------------ RST4
35 0020  P                 .PUSHR: EQU    4
36 0020 C3F10D                   JP     PUSHR
37 0023                    ;
38 0023                          DEFS   5
39 0028                    ;------------------------------ RST5
40 0028  P                 .DI:    EQU    5
41 0028 C30E0E                   JP     DI
42 002B                    ;
43 002B 00                       NOP
44 002C 00                       NOP
45 002D                    SEL00:  ENT
46 002D 00                       DEFB   0
47 002E 00                       NOP
48 002F 00                       NOP
49 0030                    ;------------------------------ RST6
50 0030 C3A509                   JP     REGIST
51 0033                    ;
52 0033                          DEFS   5
53 0038                    ;------------------------------ RST7
54 0038 C3A509                   JP     REGIST
55 003B                    ;
56 003B                          SKP    H
```

```
01 003B 214900          START:  LD      HL,IOTBL        ;MONITOR COLD START
02 003E 7E                      LD      A,M
03 003F 23                      INC     HL
04 0040 4E                      LD      C,M
05 0041 23                      INC     HL
06 0042 ED79                    OUT     (C),A
07 0044 3C                      INC     A
08 0045 2822                    JR      Z,START2
09 0047 18F5                    JR      START+3
10 0049                 ;
11 0049 02E3            IOTBL:  DEFW    E302H           ;8255 CONTROL
12 004B 34E7                    DEFW    E734H           ;8253 C0 MODE2
13 004D 74E7                    DEFW    E774H           ;8253 C1 MODE2
14 004F B4E7                    DEFW    E7B4H           ;8253 C2 MODE2
15 0051 00E6                    DEFW    E600H           ;8253 C2=0
16 0053 00E6                    DEFW    E600H
17 0055 02E5                    DEFW    E502H           ;8253 C1=2
18 0057 00E5                    DEFW    E500H
19 0059 02E4                    DEFW    E402H           ;8253 C0=2
20 005B 00E4                    DEFW    E400H
21 005D CFE9                    DEFW    E9CFH           ;PIO A MODE3
22 005F 00E9                    DEFW    E900H           ;       ALL OUTPUT
23 0061 CFEB                    DEFW    EBCFH           ;PIO B MODE3
24 0063 FFEB                    DEFW    EBFFH           ;       ALL INPUT
25 0065                 ;
26 0065                 ;
27 0065 00                      NOP
28 0066 C3A509                  JP      REGIST          ;NMI
29 0069                 ;
30 0069 210118          START2: LD      HL,1801H
31 006C 220B00                  LD      (SCROST),HL
32 006F 318011                  LD      SP,IBUFE
33 0072 21D010                  LD      HL,MODE
34 0075 3E12                    LD      A,12H
35 0077 77                      LD      M,A
36 0078 D3E0                    OUT     (E0H),A         ; INIT CMT
37 007A 23                      INC     HL
38 007B 061E                    LD      B,1EH
39 007D CDD305                  CALL    ?CLER
40 0080 AF                      XOR     A
41 0081 32FD10                  LD      (KINTF),A
42 0084 CD8F09                  CALL    CHR40
43 0087 3E0D                    LD      A,0DH
44 0089 D3E3                    OUT     (E3H),A
45 008B 3E04                    LD      A,4
46 008D 321600                  LD      (TEMPW),A
47 0090 3C                      INC     A
48 0091 321A0B                  LD      (.ONTYO+1),A
49 0094 114F0E                  LD      DE,TITMES
50 0097 CDAC06                  CALL    NLMSG
51 009A ED56                    IM      1
52 009C AF                      XOR     A
53 009D 57                      LD      D,A
54 009E 5F                      LD      E,A
55 009F CDCA09                  CALL    TIMST
56 00A2 180A                    JR      GOOUT
57 00A4                         DEFS    10
58 00AE                 ;
59 00AE                 GOOUT:  ENT
60 00AE C3B100                  JP      ST              ;EXTI MONITOR
```

```
00B1                    ;
00B1                    ST:     ENT             ;MONITOR HOT START
00B1 F3                         DI
00B2 3E0D                       LD      A,0DH           ; READ MODE
00B4 D3E3                       OUT     (E3H),A
00B6 318011                     LD      SP,IBUFE
00B9 CD5707                     CALL    NL
00BC 3E2A                       LD      A,'*'
00BE CD3C06                     CALL    PRNT            ;PROMPT
00C1 3E27                       LD      A,39
00C3 32E30B                     LD      (KNUMBS),A
00C6 CDED00                     CALL    GETL0
00C9 FE2A                       CP      '*'
00CB 20E4                       JR      NZ,ST
00CD 13                         INC     DE
00CE 1A                         LD      A,(DE)
00CF 21B100                     LD      HL,ST
00D2 E5                         PUSH    HL
00D3 FE4A                       CP      'J'
00D5 CA1702                     JP      Z,JUMP
00D8 FE4D                       CP      'M'
00DA 281D                       JR      Z,MCLECT
00DC FE44                       CP      'D'
00DE 283D                       JR      Z,DUMP
00E0 FE4C                       CP      'L'
00E2 2865                       JR      Z,MLOAD
00E4 FE53                       CP      'S'
00E6 2864                       JR      Z,MSAVE
00E8 FE56                       CP      'V'
00EA 285B                       JR      Z,MVRFY
00EC C9                         RET
00ED                    ;
00ED                    ;
00ED                    ;
00ED 110011             GETL0:  LD      DE,BUFER
00F0 CDE50B             GETLBR: CALL    GETL
00F3 1A                         LD      A,(DE)
00F4 FE0B                       CP      0BH
00F6 28B9                       JR      Z,ST
00F8 C9                         RET
00F9                    ;
00F9                    ;
00F9                    ;
00F9 3E4D               MCLECT: LD      A,'M'
00FB CD4405                     CALL    KIN
00FE 2B                         DEC     HL
00FF 23                 MR:     INC     HL
0100 CD3406                     CALL    NLPHLS
0103 7E                         LD      A,M
0104 CD6D05                     CALL    PRTHX
0107 CD3A06                     CALL    PRNTS
010A CDED00                     CALL    GETL0
010D 110811                     LD      DE,BUFER+8
0110 1A                         LD      A,(DE)
0111 FE0D                       CP      0DH             ; CR
0113 28EA                       JR      Z,MR
0115 CDB105                     CALL    2HEX
0118 38DF                       JR      C,MCLECT
011A 77                         LD      M,A
011B 18E2                       JR      MR
```

```
011D                    ;
011D                    ;
011D                    ;
011D CD3F05             DUMP:   CALL    SSET
0120 E5                         PUSH    HL
0121 CD4205                     CALL    ESET
0124 EB                         EX      DE,HL
0125 E1                         POP     HL
0126 CD3406             DUMP0:  CALL    NLPHLS
0129 0610                       LD      B,16            ; XCHG
012B CD3A06             DUMP1:  CALL    PRNTS
012E 7E                         LD      A,M
012F CD6D05                     CALL    PRTHX
0132 E5                         PUSH    HL
0133 AF                         XOR     A
0134 ED52                       SBC     HL,DE
0136 E1                         POP     HL
0137 C8                         RET     Z
0138 23                         INC     HL
0139 10F0                       DJNZ    DUMP1
013B CD2705                     CALL    BRKEY
013E C8                         RET     Z
013F DBEA                       IN      A,(EAH)
0141 FEFD                       CP      FDH             ; SPKEY
0143 28FA                       JR      Z,-4
0145 18DF                       JR      DUMP0
0147                    ;
0147                    ;
0147                    ;
0147 AF                 MVRFY:  XOR     A
0148 01                         DEFB    1
0149                    ;
0149 3E01               MLOAD:  LD      A,1
014B 01                         DEFB    1
014C                    ;
014C 3E02               MSAVE:  LD      A,2
014E 325B01                     LD      (.MWARK+1),A
0151                    ;
0151 11A401             MENAME: LD      DE,FNCOM
0154 CDAC06                     CALL    NLMSG
0157 CDED00                     CALL    GETL0
015A 3E00               .MWARK: LD      A,0
015C FE02                       CP      2
015E 204F                       JR      NZ,MLOVE
0160 118011                     LD      DE,IBUFE
0163 3E01                       LD      A,1
0165 12                         LD      (DE),A
0166 13                         INC     DE
0167 210A11                     LD      HL,BUFER+10
016A 011000                     LD      BC,16
016D EDB0                       LDIR
016F 3E0D                       LD      A,0DH
0171 12                         LD      (DE),A
0172 CD3F05             MNAM1:  CALL    SSET
0175 229411                     LD      (IBU20),HL
0178 E5                         PUSH    HL
0179 CD4205                     CALL    ESET
017C D1                         POP     DE
017D AF                         XOR     A
017E ED52                       SBC     HL,DE
```

```
01 0180 38F0                    JR      C,MNAM1
02 0182 229211                  LD      (IBU18),HL
03 0185 21B100                  LD      HL,ST
04 0188 229611                  LD      (IBU22),HL
05 018B 3E4A                    LD      A,'J'
06 018D 325405                  LD      (KINP+4),A
07 0190 CD5005          KIN2:   CALL    KINP
08 0193 2808                    JR      Z,SAVEGO
09 0195 CDA205                  CALL    HLHEX
10 0198 38F6                    JR      C,KIN2
11 019A 229611                  LD      (IBU22),HL
12 019D CD1D02          SAVEGO: CALL    WRINF
13 01A0 D44E02                  CALL    NC,WRDAT
14 01A3 C9                      RET
15 01A4                 ;
16 01A4 46696C65        FNCOM:  DEFM    'File name:'
17 01A8 206E616D
18 01AC 653A
19 01AE 0D                      DEFB    0DH
20 01AF                 ;
21 01AF                 ;
22 01AF                 ;
23 01AF CD5F02          MLOVE:  CALL    RDINF
24 01B2 D8                      RET     C
25 01B3 3A0A11                  LD      A,(BUFER+10)
26 01B6 FE0D                    CP      0DH
27 01B8 C4D101                  CALL    NZ,NAMECK
28 01BB 20F2                    JR      NZ,MLOVE
29 01BD 3A5B01                  LD      A,(.MWARK+1)
30 01C0 3D                      DEC     A
31 01C1 2026                    JR      NZ,MVERY
32 01C3 11F901                  LD      DE,LOAMES
33 01C6 CDA406                  CALL    DSPNAM
34 01C9 CD7D02                  CALL    RDDAT
35 01CC D8                      RET     C
36 01CD 2A9611                  LD      HL,(IBU22)
37 01D0                 .HL:    ENT
38 01D0 E9                      JP      (HL)
39 01D1                 ;
40 01D1 110D02          NAMECK: LD      DE,FOUMES
41 01D4 CDA406                  CALL    DSPNAM
42 01D7 110A11                  LD      DE,BUFER+10
43 01DA 218111                  LD      HL,IBU1
44 01DD 0610                    LD      B,16
45 01DF CDC805                  CALL    SAME
46 01E2 C8                      RET     Z
47 01E3 CD6A04                  CALL    SERSP
48 01E6 F6FF                    OR      FFH
49 01E8 C9                      RET
50 01E9                 ;
51 01E9                 ;
52 01E9 110202          MVERY:  LD      DE,VERMES
53 01EC CDA406                  CALL    DSPNAM
54 01EF CD8602                  CALL    VERFY
55 01F2 D8                      RET     C
56 01F3 111402                  LD      DE,OKMES
57 01F6 C3AC06                  JP      NLMSG
58 01F9                 ;
59 01F9                 LOAMES: ENT
60 01F9 4C6F6164                DEFM    'Loading '
```

```
01 01FD 696E6720
02 0201 0D                      DEFB    0DH
03 0202 56657269        VERMES: DEFM    'Verifying '
04 0206 6679696E
05 020A 6720
06 020C 0D                      DEFB    0DH
07 020D                 FOUMES: ENT
08 020D 466F756E                DEFM    'Found '
09 0211 6420
10 0213 0D                      DEFB    0DH
11 0214 4F4B            OKMES:  DEFM    'OK'
12 0216 0D                      DEFB    0DH
13 0217                 ;
14 0217                 ;
15 0217                 ;
16 0217 3E4A            JUMP:   LD      A,'J'
17 0219 CD4405                  CALL    KIN
18 021C E9                      JP      (HL)
19 021D                 ;
20 021D                 ;
21 021D                 ;
22 021D                 WRINF:  ENT
23 021D E7                      RST     .PUSHR
24 021E EF                      RST     .DI
25 021F 1601                    LD      D,1             ; 0001:WI
26 0221 218011                  LD      HL,IBUFE
27 0224 018000                  LD      BC,0080H
28 0227 CDE203          WRI1:   CALL    CKSUM
29 022A CD1604                  CALL    MOTOR
30 022D 384B                    JR      C,STPRET
31 022F CB42                    BIT     0,D
32 0231 280B                    JR      Z,WRI2          ; WD
33 0233 D5                      PUSH    DE
34 0234 11E905                  LD      DE,WRIMES
35 0237 CDA406                  CALL    DSPNAM
36 023A D1                      POP     DE
37 023B CDC204                  CALL    TSPE
38 023E CD8603          WRI2:   CALL    GAP
39 0241 CD9B02                  CALL    WTAPE
40 0244 3834                    JR      C,STPRET
41 0246 CB4A                    BIT     1,D
42 0248 C4C204                  CALL    NZ,TSPE
43 024B 202D                    JR      NZ,STPRET
44 024D C9                      RET
45 024E                 ;
46 024E                 WRDAT:  ENT
47 024E E7                      RST     .PUSHR
48 024F EF                      RST     .DI
49 0250 1602                    LD      D,2             ; 0010:WD
50 0252 CD5702                  CALL    LDINF
51 0255 18D0                    JR      WRI1
52 0257                 ;
53 0257 ED4B9211        LDINF:  LD      BC,(IBU18)
54 025B 2A9411                  LD      HL,(IBU20)
55 025E C9                      RET
56 025F                 ;
57 025F                 ;
58 025F                 ;
59 025F                 RDINF:  ENT
60 025F E7                      RST     .PUSHR
```

```
01 0260 EF                      RST     .DI
02 0261 1604                    LD      D,4             ; 0100:RI
03 0263 218011                  LD      HL,IBUFE
04 0266 018000                  LD      BC,0080H
05 0269 CD1604          RD1:    CALL    MOTOR
06 026C D45910                  CALL    NC,TMARK'
07 026F D4CC02                  CALL    NC,RTAPE
08 0272 3806                    JR      C,STPRET
09 0274 CB5A            RD2:    BIT     3,D
10 0276 C8                      RET     Z
11 0277 CD6A04                  CALL    SERSP
12 027A C35310          STPRET: JP      MSTOP'
13 027D                 ;
14 027D                 RDDAT:  ENT
15 027D E7                      RST     .PUSHR
16 027E EF                      RST     .DI
17 027F 1608                    LD      D,8             ; 1000:RD
18 0281 CD5702                  CALL    LDINF
19 0284 18E3                    JR      RD1
20 0286                 ;
21 0286                 ;
22 0286                 ;
23 0286                 VERFY:  ENT
24 0286 E7                      RST     .PUSHR
25 0287 EF                      RST     .DI
26 0288 1608                    LD      D,8             ; RD
27 028A CD5702                  CALL    LDINF
28 028D CDE203                  CALL    CKSUM
29 0290 CD1604                  CALL    MOTOR
30 0293 D45910                  CALL    NC,TMARK'
31 0296 D41903                  CALL    NC,TVRFY
32 0299 18D7                    JR      RD2-2
33 029B                 ;
34 029B                 ;
35 029B                 ;
36 029B 1E02            WTAPE:  LD      E,2
37 029D C5                      PUSH    BC
38 029E E5                      PUSH    HL
39 029F 7E              WTAP1:  LD      A,M
40 02A0 CD4E03                  CALL    WBYTE
41 02A3 CD3105                  CALL    BRK
42 02A6 3818                    JR      C,RETHB
43 02A8 23              WTAP2:  INC     HL
44 02A9 0B                      DEC     BC
45 02AA 78                      LD      A,B
46 02AB B1                      OR      C
47 02AC 20F1                    JR      NZ,WTAP1
48 02AE 210000          .SUMDT: LD      HL,0
49 02B1 7C                      LD      A,H
50 02B2 CD4E03                  CALL    WBYTE
51 02B5 7D                      LD      A,L
52 02B6 CD4E03                  CALL    WBYTE
53 02B9 CDFE04                  CALL    LONG
54 02BC 1D                      DEC     E
55 02BD 2004                    JR      NZ,WTAP4
56 02BF AF                      XOR     A
57 02C0 E1              RETHB:  POP     HL
58 02C1 C1                      POP     BC
59 02C2 C9                      RET
60 02C3 CDE204          WTAP4:  CALL    SHORT
```

```
01 02C6 10FB                    DJNZ    -3
02 02C8 E1                      POP     HL
03 02C9 C1                      POP     BC
04 02CA 18D1                    JR      WTAPE+2
05 02CC                 ;
06 02CC                 ;
07 02CC                 ;
08 02CC 1E02            RTAPE:  LD      E,2
09 02CE C5                      PUSH    BC
10 02CF E5                      PUSH    HL
11 02D0 CD0504          RTAP1:  CALL    EDGE
12 02D3 38EB                    JR      C,RETHB
13 02D5 CD1905                  CALL    DLYR
14 02D8 DBE1                    IN      A,(E1H)
15 02DA E640                    AND     40H
16 02DC 28F2                    JR      Z,RTAP1
17 02DE 210000                  LD      HL,0000H
18 02E1 22AF02                  LD      (.SUMDT+1),HL
19 02E4 E1                      POP     HL
20 02E5 C1                      POP     BC
21 02E6 C5                      PUSH    BC
22 02E7 E5                      PUSH    HL
23 02E8 CD5F03          RTAP2:  CALL    RBYTE
24 02EB 38D3                    JR      C,RETHB
25 02ED 77                      LD      M,A
26 02EE 23                      INC     HL
27 02EF 0B                      DEC     BC
28 02F0 78                      LD      A,B
29 02F1 B1                      OR      C
30 02F2 20F4                    JR      NZ,RTAP2
31 02F4 2AAF02                  LD      HL,(.SUMDT+1)
32 02F7 CD5F03                  CALL    RBYTE
33 02FA 38C4                    JR      C,RETHB
34 02FC 4F                      LD      C,A
35 02FD CD5F03                  CALL    RBYTE
36 0300 38BE                    JR      C,RETHB
37 0302 BD                      CP      L
38 0303 2006                    JR      NZ,RTAP3
39 0305 79                      LD      A,C
40 0306 BC                      CP      H
41 0307 3E00                    LD      A,0
42 0309 28B5                    JR      Z,RETHB
43 030B 1D              RTAP3:  DEC     E
44 030C 20C2                    JR      NZ,RTAP1
45 030E 110006          TAPER:  LD      DE,SUMMES
46 0311 CDAC06                  CALL    NLMSG
47 0314 3EFF                    LD      A,FFH
48 0316 37                      SCF
49 0317 18A7                    JR      RETHB
50 0319                 ;
51 0319                 ;
52 0319                 ;
53 0319 1E02            TVRFY:  LD      E,2
54 031B C5                      PUSH    BC
55 031C E5                      PUSH    HL
56 031D CD0504          TVF1:   CALL    EDGE
57 0320 389E                    JR      C,RETHB
58 0322 CD1905                  CALL    DLYR
59 0325 DBE1                    IN      A,(E1H)
60 0327 E640                    AND     40H
```

```
0329 28F2                    JR     Z,TVF1
032B CD5F03          TVF2:   CALL   RBYTE
032E 3890                    JR     C,RETHB
0330 BE                      CP     M
0331 20DB                    JR     NZ,TAPER
0333 23                      INC    HL
0334 0B                      DEC    BC
0335 78                      LD     A,B
0336 B1                      OR     C
0337 20F2                    JR     NZ,TVF2
0339 210000          .CSMDT: LD     HL,0
033C CD5F03                  CALL   RBYTE
033F BC                      CP     H
0340 20CC                    JR     NZ,TAPER
0342 CD5F03                  CALL   RBYTE
0345 BD                      CP     L
0346 20C6                    JR     NZ,TAPER
0348 1D                      DEC    E
0349 E1                      POP    HL
034A C1                      POP    BC
034B C8                      RET    Z
034C 18CD                    JR     TVRFY+2
034E                 ;
034E                 ;
034E                 ;
034E C5              WBYTE:  PUSH   BC
034F 0608                    LD     B,8
0351 CDFE04                  CALL   LONG
0354 07              WBY1:   RLCA
0355 DCFE04                  CALL   C,LONG
0358 D4E204                  CALL   NC,SHORT
035B 10F7                    DJNZ   WBY1
035D C1                      POP    BC
035E C9                      RET
035F                 ;
035F                 ;
035F                 ;
035F E5              RBYTE:  PUSH   HL
0360 210008                  LD     HL,0800H
0363 CD0504          RBY1:   CALL   EDGE
0366 381C                    JR     C,RBY3
0368 CD1905                  CALL   DLYR
036B DBE1                    IN     A,(E1H)
036D E640                    AND    40H
036F 280A                    JR     Z,RBY2
0371 E5                      PUSH   HL
0372 2AAF02                  LD     HL,(.SUMDT+1)
0375 23                      INC    HL
0376 22AF02                  LD     (.SUMDT+1),HL
0379 E1                      POP    HL
037A 37                      SCF
037B CB15            RBY2:   RL     L
037D 25                      DEC    H
037E 20E3                    JR     NZ,RBY1
0380 CD0504                  CALL   EDGE
0383 7D                      LD     A,L
0384 E1              RBY3:   POP    HL
0385 C9                      RET
0386                 ;
0386                 ;
```

```
01 0386                  ;
02 0386 C5               GAP:    PUSH    BC
03 0387 E5                       PUSH    HL
04 0388 01F82A                   LD      BC,2AF8H
05 038B 211414                   LD      HL,1414H
06 038E CB4A                     BIT     1,D
07 0390 2004                     JR      NZ,GAP1         ; WD
08 0392 011027                   LD      BC,2710H        ; 55F0H(K)
09 0395 29                       ADD     HL,HL
10 0396 CDE204           GAP1:   CALL    SHORT
11 0399 0B                       DEC     BC
12 039A 78                       LD      A,B
13 039B B1                       OR      C
14 039C 20F8                     JR      NZ,GAP1
15 039E CDFE04           GAP2:   CALL    LONG
16 03A1 25                       DEC     H
17 03A2 20FA                     JR      NZ,GAP2
18 03A4 CDE204           GAP3:   CALL    SHORT
19 03A7 2D                       DEC     L
20 03A8 20FA                     JR      NZ,GAP3
21 03AA CDFE04                   CALL    LONG
22 03AD E1               RETHB1: POP     HL
23 03AE C1                       POP     BC
24 03AF C9                       RET
25 03B0                  ;
26 03B0                  ;
27 03B0                  ;
28 03B0 E5               TMARK:  PUSH    HL
29 03B1 2E14                     LD      L,14H
30 03B3 CB5A                     BIT     3,D
31 03B5 2002                     JR      NZ,TM1
32 03B7 CB05                     RLC     L
33 03B9 65               TM1:    LD      H,L
34 03BA CD0504           TM2:    CALL    EDGE
35 03BD 3821                     JR      C,TM4
36 03BF CD1905                   CALL    DLYR
37 03C2 DBE1                     IN      A,(E1H)
38 03C4 E640                     AND     40H
39 03C6 28F1                     JR      Z,TM1
40 03C8 25                       DEC     H
41 03C9 20EF                     JR      NZ,TM2
42 03CB 65                       LD      H,L
43 03CC CD0504           TM3:    CALL    EDGE
44 03CF 380F                     JR      C,TM4
45 03D1 CD1905                   CALL    DLYR
46 03D4 DBE1                     IN      A,(E1H)
47 03D6 E640                     AND     40H
48 03D8 20DF                     JR      NZ,TM1
49 03DA 25                       DEC     H
50 03DB 20EF                     JR      NZ,TM3
51 03DD CD0504                   CALL    EDGE
52 03E0 E1               TM4:    POP     HL
53 03E1 C9                       RET
54 03E2                  ;
55 03E2                  ;
56 03E2                  ;
57 03E2 C5               CKSUM:  PUSH    BC
58 03E3 E5                       PUSH    HL
59 03E4 D5                       PUSH    DE
60 03E5 110000                   LD      DE,0000H
```

```
01 03E8 78              CKS1:   LD      A,B
02 03E9 B1                      OR      C
03 03EA 200A                    JR      NZ,CKS2
04 03EC EB                      EX      DE,HL
05 03ED 22AF02                  LD      (.SUMDT+1),HL
06 03F0 223A03                  LD      (.CSMDT+1),HL
07 03F3 D1                      POP     DE
08 03F4 18B7                    JR      RETHB1
09 03F6 7E              CKS2:   LD      A,M
10 03F7 C5                      PUSH    BC
11 03F8 0608                    LD      B,8
12 03FA 07              CKS3:   RLCA
13 03FB 3001                    JR      NC,+3
14 03FD 13                      INC     DE
15 03FE 10FA                    DJNZ    CKS3
16 0400 C1                      POP     BC
17 0401 23                      INC     HL
18 0402 0B                      DEC     BC
19 0403 18E3                    JR      CKS1
20 0405                 ;
21 0405                 ;
22 0405                 ;
23 0405 DBE1            EDGE:   IN      A,(E1H)
24 0407 2F                      CPL
25 0408 07                      RLCA
26 0409 D8                      RET     C
27 040A 07                      RLCA
28 040B 30F8                    JR      NC,EDGE
29 040D DBE1            EDGE1:  IN      A,(E1H)
30 040F 2F                      CPL
31 0410 07                      RLCA
32 0411 D8                      RET     C
33 0412 07                      RLCA
34 0413 38F8                    JR      C,EDGE1
35 0415 C9                      RET
36 0416                 ;
37 0416                 ;
38 0416                 ;
39 0416 CD3605          MOTOR:  CALL    KBSET
40 0419 DBE1                    IN      A,(E1H)
41 041B E620                    AND     20H
42 041D 2818                    JR      Z,MOT2
43 041F D5                      PUSH    DE
44 0420 11E005                  LD      DE,SETMES
45 0423 CDAC06                  CALL    NLMSG
46 0426 D1                      POP     DE
47 0427 CD4B04                  CALL    OPEN
48 042A CD3105          MOT1:   CALL    BRK
49 042D D8                      RET     C
50 042E DBE1                    IN      A,(E1H)
51 0430 E620                    AND     20H
52 0432 20F6                    JR      NZ,MOT1
53 0434 CDDC04                  CALL    DEL1M
54 0437 3E03            MOT2:   LD      A,3             ; 0011:WRITE  1100:READ
55 0439 A2                      AND     D
56 043A 281E                    JR      Z,PLAY
57 043C DBE1            MOTW:   IN      A,(E1H)
58 043E E610                    AND     10H
59 0440 2814                    JR      Z,MOTWG
60 0442 D5                      PUSH    DE
```

```
01 0443 11F205                  LD      DE,WPRMES
02 0446 CDAC06                  CALL    NLMSG
03 0449 D1                      POP     DE
04 044A 37                      SCF
05 044B                 ;
06 044B                 OPEN:   ENT
07 044B 3E08                    LD      A,08H
08 044D D3E3                    OUT     (E3H),A
09 044F CDDC04                  CALL    DEL1M
10 0452 3C                      INC     A
11 0453 D3E3                    OUT     (E3H),A
12 0455 C9                      RET
13 0456                 ;
14 0456 3E0C            MOTWG:  LD      A,0CH           ; WRITE MODE
15 0458 D3E3                    OUT     (E3H),A
16 045A 7A              PLAY:   LD      A,D
17 045B E605                    AND     05H
18 045D C49904                  CALL    NZ,MPLAY
19 0460 CDA804                  CALL    FR
20 0463 3AD010                  LD      A,(MODE)
21 0466 CBD7                    SET     2,A
22 0468 1834                    JR      BLK4
23 046A                 ;
24 046A                 ;
25 046A                 ;
26 046A                 SERSP:  ENT
27 046A E7                      RST     .PUSHR
28 046B CD9504                  CALL    MSTOP
29 046E CDB204                  CALL    FFWD
30 0471 01004C          SSP1:   LD      BC,4C00H
31 0474 DBE1                    IN      A,(E1H)
32 0476 07                      RLCA
33 0477 3F                      CCF
34 0478 D8                      RET     C
35 0479 07                      RLCA
36 047A CB12                    RL      D
37 047C 7A                      LD      A,D
38 047D E63F                    AND     3FH
39 047F FE2A                    CP      2AH
40 0481 280A                    JR      Z,SSP2
41 0483 FE15                    CP      15H
42 0485 2806                    JR      Z,SSP2
43 0487 0B                      DEC     BC
44 0488 78                      LD      A,B
45 0489 B1                      OR      C
46 048A 20E8                    JR      NZ,SSP1+3
47 048C C9                      RET
48 048D 04              SSP2:   INC     B
49 048E 78                      LD      A,B
50 048F FE4C                    CP      4CH
51 0491 30DE                    JR      NC,SSP1
52 0493 18DF                    JR      SSP1+3
53 0495                 ;
54 0495                 ;
55 0495                 ;
56 0495                 MSTOP:  ENT
57 0495 3E0D                    LD      A,0DH           ; READ MODE
58 0497 D3E3                    OUT     (E3H),A
59 0499 3AD010          MPLAY:  LD      A,(MODE)
60 049C CBDF                    SET     3,A
```

```
01 049E CDA404          BLK4:   CALL    BLK1
02 04A1 3AD010          BLK3:   LD      A,(MODE)
03 04A4                 BLK1:   ENT
04 04A4 D3E0                    OUT     (E0H),A
05 04A6 182E                    JR      DEL6
06 04A8                 ;
07 04A8 3E0B            FR:     LD      A,0BH           ;FF/REW LATCH
08 04AA CDAE04                  CALL    +4
09 04AD 3D                      DEC     A
10 04AE D3E3                    OUT     (E3H),A
11 04B0 1824                    JR      DEL6
12 04B2                 ;
13 04B2                 ;
14 04B2                 ;
15 04B2                 FFWD:   ENT
16 04B2 CD3605                  CALL    KBSET
17 04B5 CDA104                  CALL    BLK3
18 04B8                 HIGHSC: ENT
19 04B8 CDA804                  CALL    FR
20 04BB CDA104                  CALL    BLK3
21 04BE CBC7                    SET     0,A
22 04C0 18DC                    JR      BLK4
23 04C2                 ;
24 04C2                 ;
25 04C2                 ;
26 04C2                 TSPE:   ENT
27 04C2 E7                      RST     .PUSHR
28 04C3 3E0E                    LD      A,0EH
29 04C5 D3E3                    OUT     (E3H),A
30 04C7 01561E                  LD      BC,7766         ;8S
31 04CA                 ;
32 04CA                 ;
33 04CA                 ;
34 04CA F5              D1M:    PUSH    AF
35 04CB AF                      XOR     A
36 04CC 3D                      DEC     A
37 04CD 20FD                    JR      NZ,-1
38 04CF 0B                      DEC     BC
39 04D0 78                      LD      A,B
40 04D1 B1                      OR      C
41 04D2 20F7                    JR      NZ,D1M+1
42 04D4 F1                      POP     AF
43 04D5 C9                      RET
44 04D6                 ;
45 04D6 E7              DEL6:   RST     .PUSHR
46 04D7 010100                  LD      BC,1
47 04DA 18EE                    JR      D1M
48 04DC                 ;
49 04DC                 DEL1M:  ENT
50 04DC E7                      RST     .PUSHR
51 04DD 019607                  LD      BC,1942         ; 2S
52 04E0 18E8                    JR      D1M
53 04E2                 ;
54 04E2                 ;
55 04E2                 ;
56 04E2 F5              SHORT:  PUSH    AF
57 04E3 3E0F                    LD      A,0FH
58 04E5 D3E3                    OUT     (E3H),A
59 04E7 0A                      LD      A,(BC)
60 04E8 3E2A                    LD      A,2AH           ; 2AH(H):166.75US
```

```
01 04EA 322105                  LD      (DLY+1),A       ; 3FH(L):240.25US
02 04ED CD2005                  CALL    DLY
03 04F0 3E0E                    LD      A,0EH
04 04F2 D3E3                    OUT     (E3H),A
05 04F4 3E25                    LD      A,25H           ; 25H(H):166US
06 04F6 322105                  LD      (DLY+1),A       ; 3AH(L):221.5US
07 04F9 CD2005                  CALL    DLY
08 04FC F1                      POP     AF
09 04FD C9                      RET
10 04FE                 ;
11 04FE F5              LONG:   PUSH    AF
12 04FF 3E0F                    LD      A,0FH
13 0501 D3E3                    OUT     (E3H),A
14 0503 3E5A                    LD      A,5AH           ; 5AH(H):333US
15 0505 322105                  LD      (DLY+1),A       ; 81H(L):469.5US
16 0508 CD2005                  CALL    DLY
17 050B 3E0E                    LD      A,0EH
18 050D D3E3                    OUT     (E3H),A
19 050F 3E55                    LD      A,55H           ; 55H(H):334US
20 0511 322105                  LD      (DLY+1),A       ; 7CH(L):452.5US
21 0514 CD2005                  CALL    DLY
22 0517 F1                      POP     AF
23 0518 C9                      RET
24 0519                 ;
25 0519 7C              DLYR:   LD      A,H
26 051A 7D                      LD      A,L
27 051B 3E                      DEFB    3EH             ;LD A,N
28 051C                 Z80KT:  ENT
29 051C 41                      DEFB    41H             ;66H(K)
30 051D 322105                  LD      (DLY+1),A       ;
31 0520                 ;
32 0520 3EFF            DLY:    LD      A,FFH
33 0522 3D                      DEC     A
34 0523 C22205                  JP      NZ,-1
35 0526 C9                      RET
36 0527                 ;
37 0527                 ;
38 0527                 BRKEY:  ENT
39 0527 CD3605                  CALL    KBSET
40 052A DBEA                    IN      A,(EAH)
41 052C E680                    AND     80H
42 052E C31F0E                  JP      KINT
43 0531                 ;
44 0531                 ;
45 0531                 ;
46 0531                 BRK:    ENT
47 0531 DBEA                    IN      A,(EAH)
48 0533 2F                      CPL
49 0534 07                      RLCA
50 0535 C9                      RET
51 0536                 ;
52 0536                 ;
53 0536                 ;
54 0536                 KBSET:  ENT
55 0536 DBE8                    IN      A,(E8H)
56 0538 E6E0                    AND     E0H
57 053A F613                    OR      13H
58 053C D3E8                    OUT     (E8H),A
59 053E C9                      RET
60 053F                 ;
```

```
01 053F                ;
02 053F                ;
03 053F 3E53           SSET:   LD      A,'S'
04 0541 21                     DEFB    21H
05 0542                ;
06 0542 3E45           ESET:   LD      A,'E'
07 0544                ;
08 0544                ;
09 0544 325405         KIN:    LD      (KINP+4),A
10 0547 CD5005         KIN1:   CALL    KINP
11 054A CDA205                 CALL    HLHEX
12 054D 38F8                   JR      C,KIN1
13 054F C9                     RET
14 0550                ;
15 0550                ;
16 0550                ;
17 0550 CD5707         KINP:   CALL    NL
18 0553 3EFF                   LD      A,FFH
19 0555 CD3C06                 CALL    PRNT
20 0558 11D905                 LD      DE,COMES
21 055B CDB506                 CALL    MSG
22 055E CDED00                 CALL    GETL0
23 0561 110711                 LD      DE,BUFER+7
24 0564 1A                     LD      A,(DE)
25 0565 FE0D                   CP      0DH
26 0567 C9                     RET
27 0568                ;
28 0568                ;
29 0568                ;
30 0568                PRTHL:  ENT
31 0568 7C                     LD      A,H
32 0569 CD6D05                 CALL    PRTHX
33 056C 7D                     LD      A,L
34 056D                ;
35 056D                PRTHX:  ENT
36 056D F5                     PUSH    AF
37 056E E6F0                   AND     F0H
38 0570 0F                     RRCA
39 0571 0F                     RRCA
40 0572 0F                     RRCA
41 0573 0F                     RRCA
42 0574 CD8305                 CALL    ASC
43 0577 CD3C06                 CALL    PRNT
44 057A F1                     POP     AF
45 057B E60F                   AND     0F
46 057D CD8305                 CALL    ASC
47 0580 C33C06                 JP      PRNT
48 0583                ;
49 0583                ;
50 0583                ;
51 0583                ASCI:   ENT
52 0583                ASC:    ENT
53 0583 E60F                   AND     0FH
54 0585 C630                   ADD     A,30H
55 0587 FE3A                   CP      3AH
56 0589 D8                     RET     C
57 058A C607                   ADD     A,07H
58 058C C9                     RET
59 058D                ;
60 058D                ;
```

```
01 058D                 ;
02 058D                 HEX:    ENT
03 058D FE47                    CP      47H
04 058F 3F                      CCF
05 0590 D8                      RET     C
06 0591 FE41                    CP      41H
07 0593 300A                    JR      NC,HEX1
08 0595 FE3A                    CP      3AH
09 0597 3F                      CCF
10 0598 D8                      RET     C
11 0599 FE30                    CP      30H
12 059B D8                      RET     C
13 059C D630                    SUB     30H
14 059E C9                      RET
15 059F D637            HEX1:   SUB     37H
16 05A1 C9                      RET
17 05A2                 ;
18 05A2                 ;
19 05A2                 ;
20 05A2                 HLHEX:  ENT
21 05A2 D5                      PUSH    DE
22 05A3 CDB105                  CALL    2HEX
23 05A6 3807                    JR      C,HL1
24 05A8 67                      LD      H,A
25 05A9 CDB105                  CALL    2HEX
26 05AC 3801                    JR      C,HL1
27 05AE 6F                      LD      L,A
28 05AF D1              HL1:    POP     DE
29 05B0 C9                      RET
30 05B1                 ;
31 05B1                 ;
32 05B1                 ;
33 05B1                 2HEX:   ENT
34 05B1 C5                      PUSH    BC
35 05B2 1A                      LD      A,(DE)
36 05B3 13                      INC     DE
37 05B4 CD8D05                  CALL    HEX
38 05B7 380D                    JR      C,2HEX1
39 05B9 07                      RLCA
40 05BA 07                      RLCA
41 05BB 07                      RLCA
42 05BC 07                      RLCA
43 05BD 4F                      LD      C,A
44 05BE 1A                      LD      A,(DE)
45 05BF 13                      INC     DE
46 05C0 CD8D05                  CALL    HEX
47 05C3 3801                    JR      C,2HEX1
48 05C5 B1                      OR      C
49 05C6 C1              2HEX1:  POP     BC
50 05C7 C9                      RET
51 05C8                 ;
52 05C8                 ;
53 05C8                 ;
54 05C8 E7              SAME:   RST      PUSHR
55 05C9 1A                      LD      A,(DE)
56 05CA BE                      CP      M
57 05CB C0                      RET     NZ
58 05CC FE0D                    CP      0DH
59 05CE C8                      RET     Z
60 05CF 23                      INC     HL
```

```
01 05D0 13                      INC     DE
02 05D1 18F6                    JR      SAME+1
03 05D3                 ;
04 05D3                 ;
05 05D3                 ;
06 05D3                 ?CLER:  ENT
07 05D3 AF                      XOR     A
08 05D4                 ;
09 05D4                 ?DINT:  ENT
10 05D4 77                      LD      M,A
11 05D5 23                      INC     HL
12 05D6 10FC                    DJNZ    -2
13 05D8 C9                      RET
14 05D9                 ;
15 05D9                 ;
16 05D9                 ;
17 05D9 2D616472        COMES:  DEFM    '-adr.$'
18 05DD 2E24
19 05DF 0D                      DEFB    0DH
20 05E0                 SETMES: ENT
21 05E0 53657420                DEFM    'Set tape'
22 05E4 74617065
23 05E8 0D                      DEFB    0DH
24 05E9 57726974        WRIMES: DEFM    'Writing '
25 05ED 696E6720
26 05F1 0D                      DEFB    0DH
27 05F2 57726974        WPRMES: DEFM    'Write protect'
28 05F6 65207072
29 05FA 6F746563
30 05FE 74
31 05FF 0D                      DEFB    0DH
32 0600 43686563        SUMMES: DEFM    'Check sum error'
33 0604 6B207375
34 0608 6D206572
35 060C 726F72
36 060F 0D                      DEFB    0DH
37 0610                 ;
38 0610                 ;
39 0610                 ;
40 0610                 GETKY:  ENT
41 0610 E7                      RST     .PUSHR
42 0611 CD370E                  CALL    PUSHKI
43 0614 CD710E                  CALL    KEY
44 0617 FE1E                    CP      1EH
45 0619 C0                      RET     NZ
46 061A AF                      XOR     A
47 061B C9                      RET
48 061C                 ;
49 061C                 ;
50 061C                 ;
51 061C 2AD110          CKRNGL: LD      HL,(DSPXY)
52 061F 7C                      LD      A,H
53 0620                 ;
54 0620 E7              CKRNG:  RST     .PUSHR
55 0621 210B00                  LD      HL,SCROST
56 0624 46                      LD      B,M
57 0625 05                      DEC     B
58 0626 B8                      CP      B
59 0627 D8                      RET     C
60 0628 23                      INC     HL
```

```
01 0629 BE                      CP      M
02 062A C8                      RET     Z
03 062B 3F                      CCF
04 062C C9                      RET
05 062D                 ;
06 062D                 ;
07 062D 2AD110          ?DSP79: LD      HL,(DSPXY)
08 0630 3E                      DEFB    3EH             ;LD A,N
09 0631                 CH3979: ENT
10 0631 4F                      DEFB    79
11 0632 BD                      CP      L
12 0633 C9                      RET
13 0634                 ;
14 0634                 ;
15 0634                 ;
16 0634 CD5707          NLPHLS: CALL    NL
17 0637 CD6805          PRTHLS: CALL    PRTHL
18 063A                 ;
19 063A                 PRNTS:  ENT
20 063A 3E20                    LD      A,' '
21 063C                 ;
22 063C                 PRNT:   ENT
23 063C FE10                    CP      10H
24 063E DA1407                  JP      C,?DPCT
25 0641                 ;
26 0641                 ?DSP:   ENT
27 0641 E7                      RST     .PUSHR
28 0642 FE1B                    CP      1BH
29 0644 2838                    JR      Z,DSPTAB
30 0646 FE1A                    CP      1AH
31 0648 2007                    JR      NZ,DSP0
32 064A 3E30                    LD      A,'0'
33 064C CD4206                  CALL    ?DSP+1
34 064F 3E30                    LD      A,'0'
35 0651 215E0D          DSP0:   LD      HL,BLINK2+1
36 0654 CB66                    BIT     4,M
37 0656 C2CA07                  JP      NZ,?INST
38 0659 CD0409                  CALL    ?PONT
39 065C CD2D09                  CALL    DSPW
40 065F CD2D06                  CALL    ?DSP79
41 0662 2008                    JR      NZ,DSP2
42 0664 7C                      LD      A,H
43 0665 CD2006                  CALL    CKRNG
44 0668 D8                      RET     C
45 0669 CDF608                  CALL    DSMAG
46 066C C2CD07          DSP2:   JP      NZ,CURSR
47 066F 2B                      DEC     HL
48 0670 7E                      LD      A,M
49 0671 FE                      DEFB    FEH             ;CP N
50 0672 03              #LINE:  DEFB    3
51 0673 3003                    JR      NC,DSP3
52 0675 23                      INC     HL
53 0676 3C                      INC     A
54 0677 77                      LD      M,A
55 0678 CDCD07          DSP3:   CALL    CURSR
56 067B C31108                  JP      DELLN
57 067E ED5BD110        DSPTAB: LD      DE,(DSPXY)
58 0682 1C              TAB1:   INC     E
59 0683 3AA207                  LD      A,(CH4080)
60 0686 BB                      CP      E
```

```
0687 C8                     RET     Z
0688 7B                     LD      A,E
0689 CDD20D                 CALL    STAB
068C 20F4                   JR      NZ,TAB1
068E EB                     EX      DE,HL
068F CD6708                 CALL    INSOFF
0692 C3E107                 JP      SAVEXY
0695                    ;
0695                    ;
0695                    ;
0695                    PRNTT:  ENT
0695 CD3A06                 CALL    PRNTS
0698 3AD110                 LD      A,(DSPXY)
069B B7                     OR      A
069C C8                     RET     Z
069D D60A                   SUB     10
069F 38F4                   JR      C,PRNTT
06A1 20FA                   JR      NZ,-4
06A3 C9                     RET
06A4                    ;
06A4 CDAC06             DSPNAM: CALL    NLMSG
06A7 118111                 LD      DE,IBU1
06AA 1803                   JR      MSGX
06AC                    ;
06AC                    ;
06AC                    NLMSG:  ENT
06AC CD5707                 CALL    NL
06AF                    ;
06AF                    MSGX:   ENT
06AF E7                     RST     .PUSHR
06B0 214106                 LD      HL,?DSP
06B3 1804                   JR      +6
06B5                    MSG:    ENT
06B5 E7                     RST     .PUSHR
06B6 213C06                 LD      HL,PRNT
06B9 1A                 MSGX1:  LD      A,(DE)
06BA FE0D                   CP      0DH
06BC C8                     RET     Z
06BD CDD001                 CALL    .HL
06C0 13                     INC     DE
06C1 18F6                   JR      MSGX1
06C3                    ;
06C3                    ;
06C3                    ; SCROL DATA IN
06C3                    ;   (SCROST)=START LINE+1
06C3                    ;   (SCREND)=END LINE
06C3                    ;  INITIAL:1 , 24
06C3                    SCRSET: ENT
06C3 E7                     RST     .PUSHR
06C4 EF                     RST     .DI
06C5 21D310                 LD      HL,MANG
06C8 061C                   LD      B,28
06CA CDD305                 CALL    ?CLER
06CD 210B00                 LD      HL,SCROST
06D0 E5                     PUSH    HL
06D1 66                     LD      H,M
06D2 25                     DEC     H
06D3 2E00                   LD      L,0
06D5 CD0709                 CALL    ?PNT1
06D8 229C07                 LD      (.SCRAD+1),HL
```

```
06DB E3                         EX      (SP),HL
06DC 23                         INC     HL
06DD 66                         LD      H,M
06DE 2E00                       LD      L,0
06E0 CD0709                     CALL    ?PNT1
06E3 D1                         POP     DE
06E4 B7                         OR      A
06E5 ED52                       SBC     HL,DE
06E7 229F07                     LD      (.SCRSZ+1),HL
06EA C3FD07                     JP      CLRS
06ED                    ;
06ED                    KEYKEY: ENT
06ED DBE8                       IN      A,(E8H)
06EF CBA7                       RES     4,A
06F1 D3E8                       OUT     (E8H),A
06F3 DBEA                       IN      A,(EAH)
06F5 DBEA                       IN      A,(EAH)
06F7 3C                         INC     A
06F8 F5                         PUSH    AF
06F9 0605                       LD      B,5
06FB AF                         XOR     A
06FC 3D                         DEC     A
06FD 20FD                       JR      NZ,-1
06FF 10FA                       DJNZ    -4
0701 F1                         POP     AF
0702 C2710E                     JP      NZ,KEY
0705                    ;
0705                    NOKKEY: ENT
0705 AF                         XOR     A
0706 21F110                     LD      HL,KYBDA
0709 77                         LD      M,A
070A 3D                         DEC     A
070B 060B                       LD      B,11
070D 23                         INC     HL
070E 77                         LD      M,A
070F 10FC                       DJNZ    -2
0711 3E1E                       LD      A,1EH
0713 C9                         RET
0714                    ;
0714                    ;
0714                    ;
0714                    ?DPCT:  ENT
0714 E7                         RST     .PUSHR
0715 212707                     LD      HL,TDPCT
0718 07                         RLCA
0719 4F                         LD      C,A
071A 0600                       LD      B,0
071C FE0E                       CP      14
071E DC6708                     CALL    C,INSOFF
0721 09                         ADD     HL,BC
0722 5E                         LD      E,M
0723 23                         INC     HL
0724 56                         LD      D,M
0725 EB                         EX      DE,HL
0726 E9                         JP      (HL)
0727                    ;
0727 1108               TDPCT:  DEFW    DELLN
0729 DA07                       DEFW    CURSD
072B F007                       DEFW    CURSU
072D CD07                       DEFW    CURSR
```

```
01 072F E507                    DEFW   CURSL
02 0731 0808                    DEFW   HOME
03 0733 FD07                    DEFW   CLRS
04 0735 2A08                    DEFW   DEL
05 0737 6408                    DEFW   INST
06 0739 CC0F                    DEFW   GRAPH
07 073B C30F                    DEFW   SMALL
08 073D 5607                    DEFW   .RET              ;BREAK
09 073F BA0F                    DEFW   RVS
10 0741 6407                    DEFW   LETNL
11 0743 B10F                    DEFW   LAMODE
12 0745 A80F                    DEFW   CANRVS
13 0747                 ;
14 0747                 ;
15 0747                 ;
16 0747 EF              ?SHIFT: RST    .DI
17 0748 DBE8                    IN     A,(E8H)
18 074A E6E0                    AND    E0H
19 074C F61B                    OR     1BH
20 074E D3E8                    OUT    (E8H),A
21 0750 DBEA                    IN     A,(EAH)
22 0752 DBEA                    IN     A,(EAH)
23 0754 E604                    AND    4
24 0756 C9              .RET:   RET
25 0757                 ;
26 0757                 NL:     ENT
27 0757 E7                      RST    .PUSHR
28 0758 CD0409                  CALL   ?PONT
29 075B E5                      PUSH   HL
30 075C CDC108                  CALL   ??TOL
31 075F D1                      POP    DE
32 0760 B7                      OR     A
33 0761 ED52                    SBC    HL,DE
34 0763 C8                      RET    Z
35 0764                 ;
36 0764                 LETNL:  ENT
37 0764 E7                      RST    .PUSHR
38 0765 CDD208                  CALL   ??EOL
39 0768 CD6708          LETNL2: CALL   INSOFF
40 076B 3A0C00                  LD     A,(SCREND)
41 076E 21D210                  LD     HL,DSPXY1
42 0771 BE                      CP     M
43 0772 D0                      RET    NC
44 0773 35                      DEC    M
45 0774 BE                      CP     M
46 0775 2807                    JR     Z,SCROL+1
47 0777 7E                      LD     A,M
48 0778 FE18                    CP     24
49 077A C8                      RET    Z
50 077B 34                      INC    M
51 077C C9                      RET
52 077D                 ;
53 077D E7              SCROL:  RST    .PUSHR
54 077E CD4707                  CALL   ?SHIFT
55 0781 3A0F00                  LD     A,(SLOW)
56 0784 4F                      LD     C,A
57 0785 0600                    LD     B,0
58 0787 CCCA04                  CALL   Z,D1M
59 078A 2AD110                  LD     HL,(DSPXY)
60 078D 3A0B00                  LD     A,(SCROST)
```

```
0790 3D                     DEC     A
0791 57                     LD      D,A
0792 1E00                   LD      E,0
0794 ED53D110               LD      (DSPXY),DE
0798 CD1408                 CALL    DELLN+3
079B 1100D0         .SCRAD: LD      DE,SCRN         ;SCROL START ADRS
079E 018007         .SCRSZ: LD      BC,1920         ;SCROL SIZE
07A1 21                     DEFB    21H             ;LD HL,NN
07A2                CH4080: ENT
07A2 5000                   DEFW    80
07A4 19                     ADD     HL,DE
07A5 CD3A09                 CALL    DWLDIR
07A8 EB                     EX      DE,HL
07A9 3AA207                 LD      A,(CH4080)
07AC 47                     LD      B,A
07AD CD4809                 CALL    DSCL
07B0 210C00                 LD      HL,SCREND
07B3 7E                     LD      A,M
07B4 2B                     DEC     HL
07B5 96                     SUB     M
07B6 C603                   ADD     A,3
07B8 4F                     LD      C,A
07B9 0600                   LD      B,0
07BB 11D310                 LD      DE,MANG
07BE 6E                     LD      L,M
07BF 2600                   LD      H,0
07C1 19                     ADD     HL,DE
07C2 54                     LD      D,H
07C3 5D                     LD      E,L
07C4 1B                     DEC     DE
07C5 EDB0                   LDIR
07C7 3600                   LD      M,0
07C9 C9                     RET
07CA                ;
07CA CD6D08         ?INST:  CALL    ??INST
07CD                ;
07CD CD2D06         CURSR:  CALL    ?DSP79
07D0 2803                   JR      Z,CURSR2
07D2 2C                     INC     L
07D3 180C                   JR      SAVEXY
07D5 2E00           CURSR2: LD      L,0
07D7 22D110                 LD      (DSPXY),HL
07DA                ;
07DA CD1C06         CURSD:  CALL    CKRNGL
07DD D8                     RET     C
07DE 289E                   JR      Z,SCROL+1
07E0 24                     INC     H
07E1 22D110         SAVEXY: LD      (DSPXY),HL
07E4 C9                     RET
07E5                ;
07E5 CD2D06         CURSL:  CALL    ?DSP79
07E8 2D                     DEC     L
07E9 F2E107                 JP      P,SAVEXY
07EC 6F                     LD      L,A
07ED 22D110                 LD      (DSPXY),HL
07F0                ;
07F0 CD1C06         CURSU:  CALL    CKRNGL
07F3 D8                     RET     C
07F4 3A0B00                 LD      A,(SCROST)
07F7 3D                     DEC     A
```

```
01 07F8 BC                      CP      H
02 07F9 C8                      RET     Z
03 07FA 25                      DEC     H
04 07FB 18E4                    JR      SAVEXY
05 07FD                 ;
06 07FD                 ;
07 07FD                 ;
08 07FD CD0808          CLRS:   CALL    HOME
09 0800 CD1B08                  CALL    DELSUB
10 0803 CD1C06                  CALL    CKRNGL
11 0806 30F8                    JR      NC,-6
12 0808                 ;
13 0808 3A0B00          HOME:   LD      A,(SCROST)
14 080B 3D                      DEC     A
15 080C 67                      LD      H,A
16 080D 2E00                    LD      L,0
17 080F 18D0                    JR      SAVEXY
18 0811                 ;
19 0811                 DELLN:  ENT
20 0811 2AD110                  LD      HL,(DSPXY)
21 0814 E5                      PUSH    HL
22 0815 CD1B08          DELLN2: CALL    DELSUB
23 0818 E1              POPXY:  POP     HL
24 0819 18C6                    JR      SAVEXY
25 081B 3E20            DELSUB: LD      A,' '
26 081D 32E30C                  LD      (.FLSDT+1),A
27 0820 0EFF                    LD      C,FFH
28 0822 CDEA08                  CALL    LENG+2
29 0825 45                      LD      B,L
30 0826 EB                      EX      DE,HL
31 0827 C34809                  JP      DSCL
32 082A                 ;
33 082A CD1C06          DEL:    CALL    CKRNGL
34 082D 380A                    JR      C,DEL0
35 082F 3A0B00                  LD      A,(SCROST)
36 0832 3D                      DEC     A
37 0833 BC                      CP      H
38 0834 2003                    JR      NZ,DEL0
39 0836 AF                      XOR     A
40 0837 B5                      OR      L
41 0838 C8                      RET     Z
42 0839 7D              DEL0:   LD      A,L
43 083A B7                      OR      A
44 083B 200F                    JR      NZ,DEL1
45 083D 5C                      LD      E,H
46 083E CDFB08                  CALL    MAGA
47 0841 2009                    JR      NZ,DEL1
48 0843 CD0409                  CALL    ?PONT
49 0846 2B                      DEC     HL
50 0847 CD2B09                  CALL    DSCL1
51 084A 1899                    JR      CURSL
52 084C 2AD110          DEL1:   LD      HL,(DSPXY)
53 084F E5                      PUSH    HL
54 0850 CDE808                  CALL    LENG
55 0853 D5                      PUSH    DE
56 0854 E3                      EX      (SP),HL
57 0855 C1                      POP     BC
58 0856 1B                      DEC     DE
59 0857 CD3A09                  CALL    DWLDIR
60 085A 2B                      DEC     HL
```

```
01 085B CD2B09                  CALL    DSCL1
02 085E E1                      POP     HL
03 085F 22D110                  LD      (DSPXY),HL
04 0862 1881                    JR      CURSL
05 0864                 ;
06 0864 3E10            INST:   LD      A,10H
07 0866 21                      DEFB    21H
08 0867                 ;
09 0867 3E40            INSOFF: LD      A,40H
10 0869 325E0D                  LD      (BLINK2+1),A
11 086C C9                      RET
12 086D                 ;
13 086D EF              ??INST: RST     .DI
14 086E 47                      LD      B,A
15 086F 2AD110                  LD      HL,(DSPXY)
16 0872 E5                      PUSH    HL
17 0873 CDE808                  CALL    LENG
18 0876 4D                      LD      C,L
19 0877 EB                      EX      DE,HL
20 0878 78                      LD      A,B
21 0879 41                      LD      B,C
22 087A 0EE8                    LD      C,E8H
23 087C ED50                    IN      D,(C)
24 087E CBFA                    SET     7,D
25 0880 ED51                    OUT     (C),D
26 0882 5E              INST1:  LD      E,M
27 0883 77                      LD      M,A
28 0884 7B                      LD      A,E
29 0885 23                      INC     HL
30 0886 10FA                    DJNZ    INST1
31 0888 CBBA                    RES     7,D
32 088A ED51                    OUT     (C),D
33 088C FE20                    CP      ' '
34 088E 2888                    JR      Z,POPXY
35 0890 4F                      LD      C,A
36 0891 3AD210                  LD      A,(DSPXY1)
37 0894 3D                      DEC     A
38 0895 5F                      LD      E,A
39 0896 CD2006                  CALL    CKRNG
40 0899 DA1808                  JP      C,POPXY
41 089C C2AC08                  JP      NZ,INST2
42 089F CD7D07                  CALL    SCROL
43 08A2 1D                      DEC     E
44 08A3 E1                      POP     HL
45 08A4 25                      DEC     H
46 08A5 E5                      PUSH    HL
47 08A6 24                      INC     H
48 08A7 2E00                    LD      L,0
49 08A9 22D110                  LD      (DSPXY),HL
50 08AC 3A7206          INST2:  LD      A,(#LINE)
51 08AF 47                      LD      B,A
52 08B0 CDFB08                  CALL    MAGA
53 08B3 B8                      CP      B
54 08B4 D21808                  JP      NC,POPXY
55 08B7 23                      INC     HL
56 08B8 3C                      INC     A
57 08B9 77                      LD      M,A
58 08BA 79                      LD      A,C
59 08BB CD4206                  CALL    ?DSP+1
60 08BE C31508                  JP      DELLN2
```

```
01 08C1                 ;
02 08C1                 ;
03 08C1                 ??TOL:  ENT
04 08C1 2AD110                  LD      HL,(DSPXY)
05 08C4 E5                      PUSH    HL
06 08C5 5C                      LD      E,H
07 08C6 CDFB08                  CALL    MAGA
08 08C9 E1                      POP     HL
09 08CA ED44                    NEG
10 08CC 84                      ADD     A,H
11 08CD 67                      LD      H,A
12 08CE AF                      XOR     A
13 08CF 6F                      LD      L,A
14 08D0 180F                    JR      ??EOL2
15 08D2                 ;
16 08D2                 ??EOL:  ENT
17 08D2 0E00                    LD      C,0
18 08D4 CDF608                  CALL    DSMAG
19 08D7 53                      LD      D,E
20 08D8 EB                      EX      DE,HL
21 08D9 6F                      LD      L,A
22 08DA AF                      XOR     A
23 08DB CB41                    BIT     0,C
24 08DD 2801                    JR      Z,+3
25 08DF 12                      LD      (DE),A
26 08E0 B5                      OR      L
27 08E1 22D110          ??EOL2: LD      (DSPXY),HL
28 08E4 20EE                    JR      NZ,??EOL+2
29 08E6 181C                    JR      ?PONT
30 08E8                 ;
31 08E8 0E00            LENG:   LD      C,0
32 08EA CD0409                  CALL    ?PONT
33 08ED E5                      PUSH    HL
34 08EE CDD408                  CALL    ??EOL+2
35 08F1 D1                      POP     DE
36 08F2 B7                      OR      A
37 08F3 ED52                    SBC     HL,DE
38 08F5 C9                      RET
39 08F6                 ;
40 08F6 2AD110          DSMAG:  LD      HL,(DSPXY)
41 08F9 5C                      LD      E,H
42 08FA 1C                      INC     E
43 08FB                 MAGA:   ENT
44 08FB 1600                    LD      D,0
45 08FD 21D310                  LD      HL,MANG
46 0900 19                      ADD     HL,DE
47 0901 7E                      LD      A,M
48 0902 B7                      OR      A
49 0903 C9                      RET
50 0904                 ;
51 0904                 ?PONT:  ENT
52 0904 2AD110                  LD      HL,(DSPXY)
53 0907 C5              ?PNT1:  PUSH    BC
54 0908 D5                      PUSH    DE
55 0909 44                      LD      B,H
56 090A 4D                      LD      C,L
57 090B 04                      INC     B
58 090C ED5BA207                LD      DE,(CH4080)
59 0910 21D8CF          .PON1:  LD      HL,SCRN-40      ; XCHG
60 0913 19                      ADD     HL,DE
```

```
01 0914 10FD                    DJNZ    -1
02 0916 09                      ADD     HL,BC
03 0917 D1                      POP     DE
04 0918 C1                      POP     BC
05 0919 C9                      RET
06 091A                 ;
07 091A                 DSPRED: ENT
08 091A EF                      RST     .DI
09 091B C5                      PUSH    BC
10 091C 0EE8                    LD      C,E8H
11 091E ED40                    IN      B,(C)
12 0920 CBF8                    SET     7,B
13 0922 ED41                    OUT     (C),B
14 0924 7E                      LD      A,M
15 0925 CBB8            DSPWRR: RES     7,B
16 0927 ED41                    OUT     (C),B
17 0929 C1                      POP     BC
18 092A C9                      RET
19 092B                 ;
20 092B 3E20            DSCL1:  LD      A,' '
21 092D                 ;
22 092D                 DSPW:   ENT
23 092D EF                      RST     .DI
24 092E C5                      PUSH    BC
25 092F 0EE8                    LD      C,E8H
26 0931 ED40                    IN      B,(C)
27 0933 CBF8                    SET     7,B
28 0935 ED41                    OUT     (C),B
29 0937 77                      LD      M,A
30 0938 18EB                    JR      DSPWRR
31 093A                 ;
32 093A DBE8            DWLDIR: IN      A,(E8H)
33 093C CBFF                    SET     7,A
34 093E EF              DWLDI': RST     .DI
35 093F D3E8                    OUT     (E8H),A
36 0941 EDB0                    LDIR
37 0943 E63F            DWLDRN: AND     3FH
38 0945 D3E8                    OUT     (E8H),A
39 0947 C9                      RET
40 0948                 ;
41 0948 EF              DSCL:   RST     .DI
42 0949 DBE8                    IN      A,(E8H)
43 094B CBFF                    SET     7,A
44 094D D3E8                    OUT     (E8H),A
45 094F 3620                    LD      M,' '
46 0951 23                      INC     HL
47 0952 10FB                    DJNZ    -3
48 0954 DBE8                    IN      A,(E8H)
49 0956 18EB                    JR      DWLDRN
50 0958                 ;
51 0958                 ;
52 0958                 ;
53 0958                 CHR80:  ENT
54 0958 EF                      RST     .DI
55 0959 3E01                    LD      A,1
56 095B 327206                  LD      (#LINE),A
57 095E 3E10                    LD      A,10H
58 0960 322A01                  LD      (DUMP0+4),A
59 0963 3EEF                    LD      A,EFH
60 0965 327A09                  LD      (CHX2+1),A
```

```
0968 3E4F                       LD      A,79
096A 21B0CF                     LD      HL,SCRN-80
096D 323106             CHX0:   LD      (CH3979),A
0970 3C                         INC     A
0971 32A207                     LD      (CH4080),A
0974 221109                     LD      (.PON1+1),HL
0977 DBE8                       IN      A,(E8H)
0979 CBEF               CHX2:   SET     5,A
097B D3E8                       OUT     (E8H),A
097D 2100D0                     LD      HL,SCRN
0980 010800                     LD      BC,8
0983 CD4809                     CALL    DSCL
0986 0D                         DEC     C
0987 20FA                       JR      NZ,-4
0989 CDC306                     CALL    SCRSET
098C C30808                     JP      HOME
098F                    ;
098F                    CHR40:  ENT
098F 3E03                       LD      A,3
0991 327206                     LD      (#LINE),A
0994 3E08                       LD      A,08H
0996 322A01                     LD      (DUMP0+4),A
0999 3EAF                       LD      A,AFH
099B 327A09                     LD      (CHX2+1),A
099E 3E27                       LD      A,39
09A0 21D8CF                     LD      HL,SCRN-40
09A3 18C8                       JR      CHX0
09A5                    ;
09A5                    ;
09A5                    ;
09A5 E3                 REGIST: EX      (SP),HL
09A6 2B                         DEC     HL
09A7 E3                         EX      (SP),HL
09A8 E5                         PUSH    HL
09A9 D5                         PUSH    DE
09AA C5                         PUSH    BC
09AB F5                         PUSH    AF
09AC 0605                       LD      B,5
09AE E1                         POP     HL
09AF CD3706                     CALL    PRTHLS
09B2 10FA                       DJNZ    -4
09B4 210000                     LD      HL,0
09B7 39                         ADD     HL,SP
09B8 CD6805                     CALL    PRTHL
09BB C3B100                     JP      ST
09BE                    ;
09BE                    ;
09BE                    ;TEMPO
09BE                    ;       A=VALUE
09BE                    XTEMP:  ENT
09BE F5                         PUSH    AF
09BF E60F                       AND     0FH
09C1 ED44                       NEG
09C3 C608                       ADD     A,8
09C5 321600                     LD      (TEMPW),A
09C8 F1                         POP     AF
09C9 C9                         RET
09CA                    ;
09CA                    ;TIME SET
09CA                    ;       BC=C2
```

```
09CA                    ;       DE=SECOND
09CA                    ;       C2=0-FFFF 12H
09CA                    ;       C1=A8C0H=12HSEC
09CA                    ;       C0=7A12H=31.25KHZ
09CA                    TIMST:  ENT
09CA EF                         RST     .DI
09CB C5                         PUSH    BC
09CC 32560A                     LD      (.AMPM+1),A
09CF ED53460A                   LD      (.INIC1+1),DE
09D3 3EC1                       LD      A,C1H           ;C1=A8C1 SET
09D5 D3E5                       OUT     (E5H),A
09D7 3EA8                       LD      A,A8H
09D9 D3E5                       OUT     (E5H),A
09DB 3E02                       LD      A,02H           ;C0=0002 SET
09DD D3E4                       OUT     (E4H),A
09DF AF                         XOR     A
09E0 D3E4                       OUT     (E4H),A
09E2                    ;
09E2 D3F0                       OUT     (F0H),A         ;C0 C1 RESET
09E4                    ;
09E4 3E44               TMS1:   LD      A,44H           ;C1 LATCH
09E6 D3E7                       OUT     (E7H),A
09E8 DBE5                       IN      A,(E5H)         ;C1 READ
09EA 4F                         LD      C,A
09EB DBE5                       IN      A,(E5H)
09ED FEA8                       CP      A8H
09EF 20F3                       JR      NZ,TMS1
09F1 3EC0                       LD      A,C0H
09F3 B9                         CP      C
09F4 20EE                       JR      NZ,TMS1
09F6 3EC0                       LD      A,C0H           ;C1=A8C0 SET
09F8 D3E5                       OUT     (E5H),A
09FA 3EA8                       LD      A,A8H
09FC D3E5                       OUT     (E5H),A
09FE 3E12                       LD      A,12H           ;C0=7A12 SET
0A00 D3E4                       OUT     (E4H),A
0A02 3E7A                       LD      A,7AH
0A04 D3E4                       OUT     (E4H),A
0A06 3E84                       LD      A,84H           ;C2 LATCH
0A08 D3E7                       OUT     (E7H),A
0A0A DBE6                       IN      A,(E6H)         ;C2 READ
0A0C 4F                         LD      C,A
0A0D DBE6                       IN      A,(E6H)
0A0F 47                         LD      B,A
0A10 ED432E0A                   LD      (.C2DAT+1),BC
0A14 C1                         POP     BC
0A15 C9                         RET
0A16                    ;
0A16                    ;TIME READ
0A16                    ;       BC=C2 12H
0A16                    ;       DE=SECOND
0A16                    TIMRD:  ENT
0A16 EF                         RST     .DI
0A17 C5                         PUSH    BC
0A18 E5                         PUSH    HL
0A19 3E84                       LD      A,84H           ;C2 LATCH
0A1B D3E7                       OUT     (E7H),A
0A1D 3E44                       LD      A,44H           ;C1 LATCH
0A1F D3E7                       OUT     (E7H),A
0A21 DBE6                       IN      A,(E6H)         ;C2 READ
```

```
01 0A23 4F                      LD      C,A
02 0A24 DBE6                    IN      A,(E6H)
03 0A26 47                      LD      B,A
04 0A27 DBE5                    IN      A,(E5H)         ;C1 READ
05 0A29 5F                      LD      E,A
06 0A2A DBE5                    IN      A,(E5H)
07 0A2C 57                      LD      D,A
08 0A2D 210000          .C2DAT: LD      HL,0
09 0A30 AF                      XOR     A
10 0A31 ED42                    SBC     HL,BC
11 0A33 7D                      LD      A,L
12 0A34 0F                      RRCA
13 0A35 DC730A                  CALL    C,TMUP
14 0A38 D5                      PUSH    DE
15 0A39 7A                      LD      A,D
16 0A3A B3                      OR      E
17 0A3B 2003                    JR      NZ,TMX
18 0A3D 11C0A8                  LD      DE,A8C0H
19 0A40 21C0A8          TMX:    LD      HL,A8C0H        ; HL=A8C0-C1
20 0A43 ED52                    SBC     HL,DE
21 0A45 110000          .INIC1: LD      DE,0            ; HL=HL+INISET
22 0A48 19                      ADD     HL,DE
23 0A49 3822                    JR      C,TMX1
24 0A4B E5                      PUSH    HL
25 0A4C 11C0A8                  LD      DE,A8C0H
26 0A4F ED52                    SBC     HL,DE
27 0A51 3813                    JR      C,TMR1
28 0A53 F1                      POP     AF              ; ADJ
29 0A54 EB              TMX2:   EX      DE,HL
30 0A55 3E00            .AMPM:  LD      A,0
31 0A57 EE01                    XOR     01H
32 0A59 E1                      POP     HL
33 0A5A 010100                  LD      BC,0001H
34 0A5D ED42                    SBC     HL,BC
35 0A5F 2002                    JR      NZ,+4
36 0A61 EE01                    XOR     01H
37 0A63 E1                      POP     HL
38 0A64 C1                      POP     BC
39 0A65 C9                      RET
40 0A66 D1              TMR1:   POP     DE
41 0A67 E1                      POP     HL
42 0A68 3A560A                  LD      A,(.AMPM+1)
43 0A6B 18F6                    JR      -8
44 0A6D 114057          TMX1:   LD      DE,5740H
45 0A70 19                      ADD     HL,DE
46 0A71 18E1                    JR      TMX2
47 0A73 ED432E0A        TMUP:   LD      (.C2DAT+1),BC
48 0A77 3A560A                  LD      A,(.AMPM+1)
49 0A7A EE01                    XOR     01H
50 0A7C 32560A                  LD      (.AMPM+1),A
51 0A7F C9                      RET
52 0A80                 ;
53 0A80                 ;  BELL
54 0A80                 ;
55 0A80                 BELL:   ENT
56 0A80 E7                      RST     .PUSHR
57 0A81 013000                  LD      BC,0030H
58 0A84 216000                  LD      HL,0060H
59 0A87                 ;
60 0A87                 ; SOUND OUT
```

```
0A87                    ;       BC=ONCHOO
0A87                    ;       HL=ONTEI
0A87 E7                 SOUT:   RST     .PUSHR
0A88 EF                         RST     .DI
0A89 3E05               SOUT1:  LD      A,05H
0A8B CD990A                     CALL    SOUT2
0A8E 3E04                       LD      A,04H
0A90 CD990A                     CALL    SOUT2
0A93 0B                         DEC     BC
0A94 79                         LD      A,C
0A95 B0                         OR      B
0A96 20F1                       JR      NZ,SOUT1
0A98 C9                         RET
0A99                    ;
0A99 D3E3               SOUT2:  OUT     (E3H),A
0A9B 54                         LD      D,H
0A9C 5D                         LD      E,L
0A9D 1B                         DEC     DE
0A9E 7A                         LD      A,D
0A9F B3                         OR      E
0AA0 20FB                       JR      NZ,-3
0AA2 C9                         RET
0AA3                    ;
0AA3                    ; MELODY
0AA3                    ;       DE=DATA LOW ADDRESS
0AA3                    MELDY:  ENT
0AA3 E7                         RST     .PUSHR
0AA4 3E02                       LD      A,2
0AA6 32040B                     LD      (.OCTV+1),A
0AA9 1A                 MLD1:   LD      A,(DE)
0AAA FE0D                       CP      0DH
0AAC C8                         RET     Z
0AAD FE2A                       CP      '*'             ; END MARK
0AAF C8                         RET     Z
0AB0 FE2D                       CP      '-'             ; UNDER OCTAVE
0AB2 2825                       JR      Z,MLD2
0AB4 FE2B                       CP      '+'             ; UPPER OCTAVE
0AB6 2829                       JR      Z,MLD3
0AB8 21430B                     LD      HL,MTBL
0ABB FE23                       CP      '#'
0ABD 3E00                       LD      A,00
0ABF 2005                       JR      NZ,+7
0AC1 215B0B                     LD      HL,M#TBL
0AC4 3C                         INC     A
0AC5 13                         INC     DE
0AC6 32990B                     LD      (.CH#+1),A
0AC9 CDE50A                     CALL    ONPU
0ACC 38DB                       JR      C,MLD1
0ACE CD8C0B                     CALL    RYTHM
0AD1 3E02                       LD      A,2
0AD3 32040B                     LD      (.OCTV+1),A
0AD6 D8                         RET     C
0AD7 18D0                       JR      MLD1
0AD9 3E03               MLD2:   LD      A,3
0ADB 32040B                     LD      (.OCTV+1),A
0ADE 13                         INC     DE
0ADF 18C8                       JR      MLD1
0AE1 3E01               MLD3:   LD      A,1
0AE3 18F6                       JR      MLD2+2
0AE5                    ;
```

```
0AE5                    ; ONPU TO RATIO CONV
0AE5                    ;       (RATIO)=ONTEI
0AE5                    ;       C=ONCHOO*TEMPO
0AE5 C5                 ONPU:   PUSH    BC
0AE6 0608                       LD      B,8
0AE8 1A                 ONP1:   LD      A,(DE)
0AE9 BE                         CP      M
0AEA 2809                       JR      Z,ONP2
0AEC 23                         INC     HL
0AED 23                         INC     HL
0AEE 23                         INC     HL
0AEF 10F8                       DJNZ    ONP1+1
0AF1 37                         SCF
0AF2 13                         INC     DE
0AF3 C1                         POP     BC
0AF4 C9                         RET
0AF5 78                 ONP2:   LD      A,B
0AF6 32A20B                     LD      (RYTHM1+1),A
0AF9 23                         INC     HL
0AFA D5                         PUSH    DE
0AFB 5E                         LD      E,M
0AFC 23                         INC     HL
0AFD 56                         LD      D,M
0AFE EB                         EX      DE,HL
0AFF 7C                         LD      A,H
0B00 B5                         OR      L
0B01 2809                       JR      Z,ONP3
0B03 3E00               .OCTV:  LD      A,0
0B05 3D                         DEC     A
0B06 2835                       JR      Z,HOCT
0B08 3D                         DEC     A
0B09 2801                       JR      Z,ONP3
0B0B 29                         ADD     HL,HL
0B0C 22C90B             ONP3:   LD      (.RATIO+1),HL
0B0F D1                         POP     DE
0B10 13                         INC     DE
0B11 1A                         LD      A,(DE)
0B12 47                         LD      B,A
0B13 E6F0                       AND     F0H
0B15 FE30                       CP      30H
0B17 2804                       JR      Z,+6
0B19 3E00               .ONTYO: LD      A,0
0B1B 1807                       JR      +9
0B1D 13                         INC     DE
0B1E 78                         LD      A,B
0B1F E60F                       AND     0FH
0B21 321A0B                     LD      (.ONTYO+1),A
0B24 4F                         LD      C,A
0B25 0600                       LD      B,0
0B27 21730B                     LD      HL,OPTBL
0B2A 09                         ADD     HL,BC
0B2B D5                         PUSH    DE
0B2C 5E                         LD      E,M
0B2D 50                         LD      D,B
0B2E 3A1600                     LD      A,(TEMPW)
0B31 47                         LD      B,A
0B32 62                         LD      H,D
0B33 6A                         LD      L,D
0B34 19                         ADD     HL,DE
0B35 10FD                       DJNZ    -1
```

```
0B37 D1                     POP     DE
0B38 C1                     POP     BC
0B39 44                     LD      B,H
0B3A 4D                     LD      C,L
0B3B AF                     XOR     A
0B3C C9                     RET
0B3D CB3C           HOCT:   SRL     H
0B3F CB1D                   RR      L
0B41 18C9                   JR      ONP3
0B43                ;
0B43 43             MTBL:   DEFB    'C'
0B44 2501                   DEFW    0125H
0B46 44                     DEFB    'D'
0B47 0501                   DEFW    0105H
0B49 45                     DEFB    'E'
0B4A E900                   DEFW    00E9H
0B4C 46                     DEFB    'F'
0B4D DC00                   DEFW    00DCH
0B4F 47                     DEFB    'G'
0B50 C300                   DEFW    00C3H
0B52 41                     DEFB    'A'
0B53 AE00                   DEFW    00AEH
0B55 42                     DEFB    'B'
0B56 9B00                   DEFW    009BH
0B58 52                     DEFB    'R'
0B59 0000                   DEFW    0000H
0B5B 43             M#TBL:  DEFB    'C'
0B5C 1501                   DEFW    0115H
0B5E 44                     DEFB    'D'
0B5F F600                   DEFW    00F6H
0B61 45                     DEFB    'E'
0B62 DC00                   DEFW    00DCH
0B64 46                     DEFB    'F'
0B65 CF00                   DEFW    00CFH
0B67 47                     DEFB    'G'
0B68 B800                   DEFW    00B8H
0B6A 41                     DEFB    'A'
0B6B A400                   DEFW    00A4H
0B6D 42                     DEFB    'B'
0B6E 9200                   DEFW    0092H
0B70 52                     DEFB    'R'
0B71 0000                   DEFW    0000H
0B73 01             OPTBL:  DEFB    01H
0B74 02                     DEFB    02H
0B75 03                     DEFB    03H
0B76 04                     DEFB    04H
0B77 06                     DEFB    06H
0B78 08                     DEFB    08H
0B79 0C                     DEFB    0CH
0B7A 10                     DEFB    10H
0B7B 18                     DEFB    18H
0B7C 20                     DEFB    20H
0B7D                ;
0B7D 08             TABLE1: DEFB    8
0B7E 0F                     DEFB    15
0B7F 0D                     DEFB    13
0B80 0C                     DEFB    12
0B81 0B                     DEFB    11
0B82 0A                     DEFB    10
0B83 09                     DEFB    9
```

```
0B84 08                 DEFB   8
0B85 10                 DEFB   16
0B86 0E                 DEFB   14
0B87 0D                 DEFB   13
0B88 0B                 DEFB   11
0B89 0B                 DEFB   11
0B8A 0A                 DEFB   10
0B8B 08                 DEFB   8
0B8C                ;
0B8C                ; RHYTHM
0B8C                ;
0B8C                RYTHM:  ENT
0B8C CD3605             CALL   KBSET
0B8F CD3105             CALL   BRK
0B92                RYTHMB: ENT
0B92 D8                 RET    C
0B93 E7                 RST    .PUSHR
0B94 C5                 PUSH   BC
0B95 217C0B             LD     HL,TABLE1-1
0B98 3E00           .CH#:   LD     A,0
0B9A B7                 OR     A
0B9B 2804               JR     Z,RYTHM1
0B9D 010700             LD     BC,7
0BA0 09                 ADD    HL,BC
0BA1 3E00           RYTHM1: LD     A,0
0BA3 4F                 LD     C,A
0BA4 FE01               CP     1                ;"R" ?
0BA6 2005               JR     NZ,RYTHM3
0BA8 3E02               LD     A,2
0BAA 32040B             LD     (.OCTV+1),A
0BAD 09             RYTHM3: ADD    HL,BC
0BAE 46                 LD     B,M
0BAF 3A040B             LD     A,(.OCTV+1)
0BB2 3D                 DEC    A
0BB3 2807               JR     Z,RYTHM2
0BB5 3D                 DEC    A
0BB6 2806               JR     Z,*N
0BB8 CB38               SRL    B
0BBA 1802               JR     *N
0BBC CB20           RYTHM2: SLA    B
0BBE D1             *N:     POP    DE
0BBF 210000             LD     HL,0000H
0BC2 19                 ADD    HL,DE
0BC3 10FD               DJNZ   -1
0BC5 44                 LD     B,H
0BC6 4D                 LD     C,L
0BC7 EF                 RST    .DI
0BC8 210000         .RATIO: LD     HL,0
0BCB 7C                 LD     A,H
0BCC B5                 OR     L
0BCD C2890A             JP     NZ,SOUT1
0BD0 E5                 PUSH   HL
0BD1 3E04               LD     A,4
0BD3 328A0A             LD     (SOUT1+1),A
0BD6 212501             LD     HL,0125H
0BD9 CD870A             CALL   SOUT
0BDC 3E05               LD     A,5
0BDE 328A0A             LD     (SOUT1+1),A
0BE1 E1                 POP    HL
0BE2 C9                 RET
```

```
0BE3                    ;
0BE3                    ;
0BE3                    ; GETL KEY
0BE3                    ;
0BE3                    ;
0BE3                    KNUMBS: ENT
0BE3 50                         DEFB    80
0BE4 1E                 LASTD:  DEFB    1EH
0BE5                    ;
0BE5                    GETL:   ENT
0BE5 E7                         RST     .PUSHR
0BE6 F5                         PUSH    AF
0BE7 3AA207                     LD      A,(CH4080)
0BEA 4F                         LD      C,A
0BEB 3AE30B                     LD      A,(KNUMBS)
0BEE 3D                         DEC     A
0BEF 06FF                       LD      B,FFH
0BF1 04                         INC     B
0BF2 91                         SUB     C
0BF3 30FC                       JR      NC,-2
0BF5 78                         LD      A,B
0BF6 327206                     LD      (#LINE),A
0BF9 D9                         EXX
0BFA CD000C                     CALL    GETL00
0BFD D9                         EXX
0BFE F1                         POP     AF
0BFF C9                         RET
0C00 E7                 GETL00: RST     .PUSHR
0C01 D9                         EXX
0C02 D5                         PUSH    DE
0C03 D9                         EXX
0C04 CD370E                     CALL    PUSHKI
0C07 CD1A0D                     CALL    ?SAVE
0C0A CD270D             GETL01: CALL    KEYREP
0C0D FE0B                       CP      0BH
0C0F 28F9                       JR      Z,GETL01
0C11 FE0D                       CP      0DH
0C13 28F5                       JR      Z,GETL01
0C15 32E40B                     LD      (LASTD),A
0C18 FE1E                       CP      1EH
0C1A C2E90C                     JP      NZ,AUTRT3
0C1D CD580D             GETL02: CALL    BLINK
0C20 CDED06                     CALL    KEYKEY
0C23 FE1E               GETL03: CP      1EH
0C25 28F6                       JR      Z,GETL02
0C27 32E40B                     LD      (LASTD),A
0C2A D9                         EXX
0C2B 0E00                       LD      C,0
0C2D D9                         EXX
0C2E 4F                 GETL04: LD      C,A
0C2F 3A1500                     LD      A,(SWRK)
0C32 B7                         OR      A
0C33 CC800A                     CALL    Z,BELL
0C36 CDDE0C                     CALL    FLASW
0C39 79                         LD      A,C
0C3A FE01                       CP      01H
0C3C 380B                       JR      C,DISPM
0C3E FE07                       CP      07H
0C40 3007                       JR      NC,DISPM
0C42 21A70F                     LD      HL,MODE'
```

```
01 0C45 CB76                    BIT     6,M
02 0C47 2005                    JR      NZ,GT2
03 0C49 4F              DISPM:  LD      C,A
04 0C4A FE1A                    CP      1AH
05 0C4C 3814                    JR      C,FUNC
06 0C4E CD4106          GT2:    CALL    ?DSP
07 0C51 CD1A0D          KFIN0:  CALL    ?SAVE
08 0C54 D9                      EXX
09 0C55 79                      LD      A,C
10 0C56 D9                      EXX
11 0C57 B7                      OR      A
12 0C58 CAE90C                  JP      Z,AUTRT3
13 0C5B CDFD0C                  CALL    AUTRT4
14 0C5E 30C3                    JR      NC,GETL03
15 0C60 18CC                    JR      GETL04
16 0C62                 ;
17 0C62 CB67            FUNC:   BIT     4,A
18 0C64 204E                    JR      NZ,FUNC2
19 0C66 FE0D                    CP      0DH
20 0C68 2814                    JR      Z,GTCR
21 0C6A FE0B                    CP      0BH
22 0C6C 2805                    JR      Z,GTBRK
23 0C6E CD1407          GT5:    CALL    ?DPCT
24 0C71 18DE                    JR      KFIN0
25 0C73 E1              GTBRK:  POP     HL
26 0C74 360B                    LD      M,0BH           ;BREAK
27 0C76 23                      INC     HL
28 0C77 360D                    LD      M,0DH           ;CR
29 0C79 C36407          GETLR:  JP      LETNL
30 0C7C                 ;
31 0C7C                 GETCRT: ENT
32 0C7C E7                      RST     .PUSHR
33 0C7D D5                      PUSH    DE
34 0C7E CDC108          GTCR:   CALL    ??TOL
35 0C81 CDE808                  CALL    LENG
36 0C84 3AE30B                  LD      A,(KNUMBS)
37 0C87 3D                      DEC     A
38 0C88 BD                      CP      L
39 0C89 3001                    JR      NC,+3
40 0C8B 6F                      LD      L,A
41 0C8C D5                      PUSH    DE
42 0C8D E3                      EX      (SP),HL
43 0C8E C1                      POP     BC
44 0C8F D1                      POP     DE
45 0C90 C5                      PUSH    BC
46 0C91 CB7A                    BIT     7,D
47 0C93 280B                    JR      Z,GTCR2
48 0C95 CBBC                    RES     7,H
49 0C97 DBE8                    IN      A,(E8H)
50 0C99 F6C0                    OR      C0H
51 0C9B CD3E09                  CALL    DWLDI'
52 0C9E 1803                    JR      +5
53 0CA0 CD3A09          GTCR2:  CALL    DWLDIR
54 0CA3 C1                      POP     BC
55 0CA4 41                      LD      B,C
56 0CA5 EB                      EX      DE,HL
57 0CA6 04                      INC     B
58 0CA7 360D            GLOP2:  LD      M,0DH
59 0CA9 2B                      DEC     HL
60 0CAA 7E                      LD      A,M
```

```
01 0CAB FE20                    CP      ' '
02 0CAD 2002                    JR      NZ,+4
03 0CAF 10F6                    DJNZ    GLOP2
04 0CB1 C36807                  JP      LETNL2
05 0CB4 E60F            FUNC2:  AND     0FH             ;00-09  F1-F10
06 0CB6 3C                      INC     A
07 0CB7 21F110                  LD      HL,KYBDA
08 0CBA CB56                    BIT     2,M
09 0CBC 2802                    JR      Z,+4
10 0CBE C60A                    ADD     A,10            ;F11-F20
11 0CC0 47                      LD      B,A
12 0CC1 210012                  LD      HL,FARE
13 0CC4 54                      LD      D,H
14 0CC5 5D                      LD      E,L
15 0CC6 7E                      LD      A,M
16 0CC7 23                      INC     HL
17 0CC8 FE0D                    CP      0DH
18 0CCA 20FA                    JR      NZ,-4
19 0CCC 10F7                    DJNZ    -7
20 0CCE 1A              MRUN:   LD      A,(DE)
21 0CCF FE7F                    CP      7FH             ;?CR
22 0CD1 28AB                    JR      Z,GTCR
23 0CD3 FE0D                    CP      0DH
24 0CD5 CA510C                  JP      Z,KFIN0
25 0CD8 CD3C06                  CALL    PRNT
26 0CDB 13                      INC     DE
27 0CDC 18F0                    JR      MRUN
28 0CDE                 ;
29 0CDE F5              FLASW:  PUSH    AF
30 0CDF CD0409                  CALL    ?PONT
31 0CE2 3E00            .FLSDT: LD      A,0
32 0CE4 CD2D09                  CALL    DSPW
33 0CE7 F1                      POP     AF
34 0CE8 C9                      RET
35 0CE9                 ;
36 0CE9 3A0E00          AUTRT3: LD      A,(REPTCT+1)
37 0CEC 5F                      LD      E,A
38 0CED CDFD0C                  CALL    AUTRT4
39 0CF0 D2230C                  JP      NC,GETL03
40 0CF3 1D                      DEC     E
41 0CF4 20F7                    JR      NZ,AUTRT3+4
42 0CF6 D9                      EXX
43 0CF7 0E01                    LD      C,1
44 0CF9 D9                      EXX
45 0CFA C32E0C                  JP      GETL04
46 0CFD                 ;
47 0CFD D5              AUTRT4: PUSH    DE
48 0CFE 3A0D00                  LD      A,(REPTCT)
49 0D01 5F                      LD      E,A
50 0D02 CD0C0D                  CALL    AUTRT6
51 0D05 3003                    JR      NC,+5
52 0D07 1D                      DEC     E
53 0D08 20F8                    JR      NZ,AUTRT4+5
54 0D0A D1                      POP     DE
55 0D0B C9                      RET
56 0D0C                 ;
57 0D0C CD270D          AUTRT6: CALL    KEYREP
58 0D0F F5                      PUSH    AF
59 0D10 3AE40B                  LD      A,(LASTD)
60 0D13 4F                      LD      C,A
```

```
01 0D14 F1                      POP     AF
02 0D15 B9                      CP      C
03 0D16 37                      SCF
04 0D17 C8                      RET     Z
05 0D18 B7                      OR      A
06 0D19 C9                      RET
07 0D1A                 ;
08 0D1A CD0409          ?SAVE:  CALL    ?PONT
09 0D1D CD1A09                  CALL    DSPRED
10 0D20 D9                      EXX
11 0D21 57                      LD      D,A
12 0D22 32E30C                  LD      (.FLSDT+1),A
13 0D25 1836                    JR      BLINK2
14 0D27                 ;
15 0D27 2A1300          KEYREP: LD      HL,(KDATW)
16 0D2A CD1006                  CALL    GETKY
17 0D2D C0                      RET     NZ
18 0D2E 221300                  LD      (KDATW),HL
19 0D31 7D                      LD      A,L
20 0D32 B7                      OR      A
21 0D33 C44B0D                  CALL    NZ,KEYRP2
22 0D36 A5                      AND     L
23 0D37 3E1E                    LD      A,1EH
24 0D39 C8                      RET     Z
25 0D3A 3AF110                  LD      A,(KYBDA)
26 0D3D 6F                      LD      L,A
27 0D3E 260B                    LD      H,0BH
28 0D40 CD4B0D                  CALL    KEYRP2
29 0D43 BD                      CP      L
30 0D44 3E1E                    LD      A,1EH
31 0D46 C0                      RET     NZ
32 0D47 3AE40B                  LD      A,(LASTD)
33 0D4A C9                      RET
34 0D4B                 ;
35 0D4B DBE8            KEYRP2: IN      A,(E8H)
36 0D4D E6F0                    AND     F0H
37 0D4F B4                      OR      H
38 0D50 D3E8                    OUT     (E8H),A
39 0D52 DBEA                    IN      A,(EAH)
40 0D54 DBEA                    IN      A,(EAH)
41 0D56 2F                      CPL
42 0D57 C9                      RET
43 0D58                 ;
44 0D58 D9              BLINK:  EXX
45 0D59 05                      DEC     B
46 0D5A D9                      EXX
47 0D5B C0                      RET     NZ
48 0D5C D9                      EXX
49 0D5D 0640            BLINK2: LD      B,40H
50 0D5F 3E1F                    LD      A,1FH
51 0D61 BA                      CP      D
52 0D62 200B                    JR      NZ,BLINK4
53 0D64 3AE30C                  LD      A,(.FLSDT+1)
54 0D67 FE1F                    CP      1FH
55 0D69 2004                    JR      NZ,BLINK4
56 0D6B 0610                    LD      B,10H
57 0D6D 3E20                    LD      A,' '
58 0D6F 57              BLINK4: LD      D,A
59 0D70 D9                      EXX
60 0D71 CD0409                  CALL    ?PONT
```

```
0D74 C32D09              JP    DSPW
0D77                ;
0D77                ;
0D77                ;
0D77                ??KEY:  ENT
0D77 E7                     RST   .PUSHR
0D78 D9                     EXX
0D79 CD7E0D                 CALL  +5
0D7C D9                     EXX
0D7D C9                     RET
0D7E E7                     RST   .PUSHR
0D7F CD370E                 CALL  PUSHKI
0D82 CD710E                 CALL  KEY
0D85 CD1A0D                 CALL  ?SAVE
0D88 CD580D                 CALL  BLINK
0D8B CDED06                 CALL  KEYKEY
0D8E FE1E                   CP    1EH
0D90 28F6                   JR    Z,-8
0D92 C3DE0C                 JP    FLASW
0D95                ;
0D95                ;
0D95                ;
0D95                TENTBL: ENT
0D95 5809                   DEFW  CHR80         ;8
0D97 5607                   DEFW  .RET          ;9
0D99 5607                   DEFW  .RET          ;00
0D9B 5607                   DEFW  .RET          ;.
0D9D 5607                   DEFW  .RET          ;+
0D9F 5607                   DEFW  .RET          ;-
0DA1 1108                   DEFW  DELLN         ;0
0DA3 B10D                   DEFW  SETTAB        ;1
0DA5 C00D                   DEFW  CLRTAB        ;2
0DA7 CA0D                   DEFW  CLATAB        ;3
0DA9 8F09                   DEFW  CHR40         ;4
0DAB DD0D                   DEFW  CHANGE        ;5
0DAD 5607                   DEFW  .RET          ;6
0DAF 5607                   DEFW  .RET          ;7
0DB1                ;
0DB1 3AD110         SETTAB: LD    A,(DSPXY)
0DB4 4F                     LD    C,A
0DB5 CDD20D                 CALL  STAB
0DB8 C8                     RET   Z
0DB9 AF                     XOR   A
0DBA CDD20D                 CALL  STAB
0DBD C0                     RET   NZ
0DBE 71                     LD    M,C
0DBF C9                     RET
0DC0 3AD110         CLRTAB: LD    A,(DSPXY)
0DC3 CDD20D                 CALL  STAB
0DC6 C0                     RET   NZ
0DC7 3600                   LD    M,0
0DC9 C9                     RET
0DCA 0610           CLATAB: LD    B,16
0DCC 21C010                 LD    HL,TABDAT
0DCF C3D305                 JP    ?CLER
0DD2 0610           STAB:   LD    B,16
0DD4 21C010                 LD    HL,TABDAT
0DD7 BE                     CP    M
0DD8 C8                     RET   Z
0DD9 23                     INC   HL
```

```
0DDA 10FB                       DJNZ    -3
0DDC C9                         RET
0DDD                    ;
0DDD                    CHANGE: ENT
0DDD EF                         RST     .DI
0DDE 21610F                     LD      HL,.CHG1
0DE1 11730F                     LD      DE,.CHG2
0DE4 0607                       LD      B,7
0DE6                    ;
0DE6                    ?SWAP:  ENT
0DE6 4E                         LD      C,(HL)
0DE7 1A                         LD      A,(DE)
0DE8 77                         LD      (HL),A
0DE9 79                         LD      A,C
0DEA 12                         LD      (DE),A
0DEB 23                         INC     HL
0DEC 13                         INC     DE
0DED 10F7                       DJNZ    ?SWAP
0DEF AF                         XOR     A
0DF0 C9                         RET
0DF1                    ;
0DF1                    ;
0DF1                    ; PUSHR : PUSH IX,HL,BC,DE
0DF1                    ; PUSHR2: PUSH IX,HL,BC
0DF1                    ;  DESTROY IX
0DF1                    ;
0DF1                    PUSHR:  ENT
0DF1 DDE3                       EX      (SP),IX
0DF3 E5                         PUSH    HL
0DF4 C5                         PUSH    BC
0DF5 D5                         PUSH    DE
0DF6 E5                         PUSH    HL
0DF7 21080E                     LD      HL,POPR
0DFA E3                         EX      (SP),HL
0DFB DDE9                       JP      (IX)
0DFD                    PUSHR2: ENT
0DFD DDE3                       EX      (SP),IX
0DFF E5                         PUSH    HL
0E00 C5                         PUSH    BC
0E01 E5                         PUSH    HL
0E02 21090E                     LD      HL,POPR2
0E05 E3                         EX      (SP),HL
0E06 DDE9                       JP      (IX)
0E08 D1                 POPR:   POP     DE
0E09 C1                 POPR2:  POP     BC
0E0A E1                         POP     HL
0E0B DDE1                       POP     IX
0E0D C9                         RET
0E0E                    ;
0E0E                    ; DI: PUSH IFF
0E0E                    ;
0E0E                    DI:     ENT
0E0E F5                         PUSH    AF
0E0F ED57                       LD      A,I
0E11 F3                         DI
0E12 E2350E                     JP      PO,DI4
0E15 F1                         POP     AF
0E16 E3                         EX      (SP),HL
0E17 221C0E                     LD      (DI2+1),HL
0E1A E1                         POP     HL
```

```
0E1B CD0000          DI2:    CALL    0
0E1E FB                      EI
0E1F                 KINT:   ENT
0E1F F5                      PUSH    AF
0E20 3AFD10                  LD      A,(KINTF)
0E23 B7                      OR      A
0E24 280F                    JR      Z,DI4
0E26 DBE8                    IN      A,(E8H)
0E28 E6E0                    AND     E0H
0E2A F612                    OR      12H
0E2C D3E8                    OUT     (E8H),A
0E2E 3E97                    LD      A,97H
0E30 D3EB                    OUT     (EBH),A
0E32 AF                      XOR     A
0E33 D3EB                    OUT     (EBH),A
0E35 F1              DI4:    POP     AF
0E36 C9                      RET
0E37                 ;
0E37                 ; PUSH KINT FLAG
0E37                 ;
0E37                 PUSHKI: ENT
0E37 21FD10                  LD      HL,KINTF
0E3A 7E                      LD      A,M
0E3B B7                      OR      A
0E3C C8                      RET     Z
0E3D 3600                    LD      M,0
0E3F 3E03                    LD      A,3
0E41 D3EB                    OUT     (EBH),A
0E43 E1                      POP     HL
0E44 CDD001                  CALL    .HL
0E47                 KINTON: ENT
0E47 F5                      PUSH    AF
0E48 3EFF                    LD      A,FFH
0E4A 32FD10                  LD      (KINTF),A
0E4D 18D1                    JR      KINT+1
0E4F                         SKP     H
```

```
0E4F 2A2A204D          TITMES: DEFM   '** Monitor SB-1511 **'
0E53 6F6E6974
0E57 6F722053
0E5B 422D3135
0E5F 3131202A
0E63 2A
0E64 0D                        DEFB   0DH
0E65                   ;
0E65                   ;
0E65 DBE8              SCAN:   IN     A,(E8H)
0E67 E6F0                      AND    F0H
0E69 B2                        OR     D
0E6A D3E8                      OUT    (E8H),A
0E6C DBEA                      IN     A,(EAH)
0E6E DBEA                      IN     A,(EAH)
0E70 C9                        RET
0E71                   ;
0E71 EF                KEY:    RST    .DI
0E72 210000                    LD     HL,0
0E75 221300                    LD     (KDATW),HL
0E78 21FC10                    LD     HL,KSTD+10
0E7B 161B                      LD     D,1BH
0E7D CD650E                    CALL   SCAN
0E80 2F                        CPL
0E81 47                        LD     B,A             ;B=BIT DATA
0E82 15                SWEP:   DEC    D
0E83 CD650E                    CALL   SCAN
0E86 5F                        LD     E,A
0E87 2F                        CPL
0E88 A6                        AND    M
0E89                   ..KEY:  ENT
0E89 73                        LD     M,E
0E8A 2B                        DEC    HL
0E8B 5F                        LD     E,A
0E8C B7                        OR     A
0E8D 2804                      JR     Z,+6
0E8F ED531300                  LD     (KDATW),DE
0E93 7A                        LD     A,D             ;STROB END?
0E94 E60F                      AND    0FH
0E96 20EA                      JR     NZ,SWEP
0E98 ED5B1300                  LD     DE,(KDATW)
0E9C 7B                        LD     A,E
0E9D B7                        OR     A
0E9E 2045                      JR     NZ,DATA         ;KSWEP END
0EA0 78                        LD     A,B
0EA1 BE                        CP     M
0EA2 282E                      JR     Z,KFINA
0EA4 77                        LD     M,A
0EA5 21D010                    LD     HL,MODE         ;KGSX XXXX
0EA8 FE01                      CP     01H
0EAA 281B                      JR     Z,GRPHO         ;G
0EAC FE02                      CP     02H
0EAE 281E                      JR     Z,SMALLO        ;SL
0EB0 FE03                      CP     03H             ;DISP CR
0EB2 2821                      JR     Z,CRDIS         ;G+SL
0EB4 FE05                      CP     05H
0EB6 2820                      JR     Z,GSHFO         ;G+S
0EB8 FE06                      CP     06H
0EBA 281F                      JR     Z,SMSHFO        ;SL+S
0EBC FE08                      CP     08H
```

```
01 0EBE 281E                    JR    Z,RVSO          ;R
02 0EC0 FE0C                    CP    0CH
03 0EC2 200E                    JR    NZ,KFINA
04 0EC4 3E0C            RSHFO:  LD    A,0CH
05 0EC6 C9                      RET
06 0EC7 CB76            GRPHO:  BIT   6,M
07 0EC9 2807                    JR    Z,KFINA
08 0ECB 3E0E            LMOD:   LD    A,0EH
09 0ECD C9                      RET
10 0ECE CB6E            SMALLO: BIT   5,M
11 0ED0 20F9                    JR    NZ,LMOD
12 0ED2 3E1E            KFINA:  LD    A,1EH           ;NOKEY DATA
13 0ED4 C9                      RET
14 0ED5 3E7F            CRDIS:  LD    A,7FH
15 0ED7 C9                      RET
16 0ED8 3E09            GSHFO:  LD    A,09H
17 0EDA C9                      RET
18 0EDB 3E0A            SMSHFO: LD    A,0AH
19 0EDD C9                      RET
20 0EDE CB7E            RVSO:   BIT   7,M
21 0EE0 28F0                    JR    Z,KFINA
22 0EE2 3E0F                    LD    A,0FH
23 0EE4 C9                      RET
24 0EE5                 ;
25 0EE5 CDF40E          DATA:   CALL  DATA2
26 0EE8 CB58                    BIT   3,B             ;R
27 0EEA C8                      RET   Z
28 0EEB FE7F                    CP    7FH
29 0EED D0                      RET   NC
30 0EEE FE20                    CP    20H
31 0EF0 D8                      RET   C
32 0EF1 F680                    OR    80H
33 0EF3 C9                      RET
34 0EF4 7B              DATA2:  LD    A,E
35 0EF5 1E00                    LD    E,00H
36 0EF7 B7                      OR    A
37 0EF8 1F              ROT:    RRA
38 0EF9 3803                    JR    C,ROTE
39 0EFB 1C                      INC   E
40 0EFC 18FA                    JR    ROT
41 0EFE B7              ROTE:   OR    A
42 0EFF 20D1                    JR    NZ,KFINA
43 0F01 3E0F            KDIN:   LD    A,0FH
44 0F03 A2                      AND   D               ;D=STROB DATA
45 0F04 07                      RLCA
46 0F05 07                      RLCA
47 0F06 07                      RLCA
48 0F07 B3                      OR    E
49 0F08 5F                      LD    E,A
50 0F09 1600                    LD    D,0
51 0F0B 21D010                  LD    HL,MODE
52 0F0E CB7E                    BIT   7,M             ;R
53 0F10 2802                    JR    Z,+4
54 0F12 CBD8                    SET   3,B
55 0F14 CB76                    BIT   6,M             ;G
56 0F16 2802                    JR    Z,+4
57 0F18 CBC0                    SET   0,B
58 0F1A 21A70F                  LD    HL,MODE'
59 0F1D 3600                    LD    M,0
60 0F1F CB40                    BIT   0,B             ;G
```

```
01 0F21 2802                    JR      Z,+4
02 0F23 CBF6                    SET     6,M
03 0F25 FE0A                    CP      0AH
04 0F27 3831                    JR      C,LMONLY
05 0F29 FE18                    CP      18H
06 0F2B 3850                    JR      C,TENKEY
07 0F2D 2870                    JR      Z,KYTAB
08 0F2F FE20                    CP      20H
09 0F31 3827                    JR      C,LMONLY
10 0F33 7B                      LD      A,E
11 0F34 FE52                    CP      52H
12 0F36 285D                    JR      Z,KYCTRL
13 0F38 FE53                    CP      53H
14 0F3A 2859                    JR      Z,KYCTRL
15 0F3C 21D010                  LD      HL,MODE
16 0F3F CB6E                    BIT     5,M             ;SR
17 0F41 2802                    JR      Z,+4
18 0F43 CBC8                    SET     1,B
19 0F45 CB48                    BIT     1,B
20 0F47 2802                    JR      Z,+4
21 0F49 CBD0                    SET     2,B
22 0F4B FE40                    CP      40H
23 0F4D 3007                    JR      NC,TWO
24 0F4F CB40                    BIT     0,B             ;G
25 0F51 211310                  LD      HL,KTBLG-32
26 0F54 2013                    JR      NZ,KADD
27 0F56 CB50            TWO:    BIT     2,B             ;S
28 0F58 2012                    JR      NZ,KSML
29 0F5A 21D70F          LMONLY: LD      HL,KTBL
30 0F5D FE21                    CP      21H
31 0F5F 3808                    JR      C,KADD
32 0F61 C620            .CHG1:  ADD     A,20H
33 0F63 FE5B                    CP      5BH
34 0F65 D8                      RET     C
35 0F66 D63A                    SUB     3AH
36 0F68 5F                      LD      E,A
37 0F69 19              KADD:   ADD     HL,DE
38 0F6A 7E                      LD      A,M
39 0F6B C9                      RET
40 0F6C 21F50F          KSML:   LD      HL,KTBLS-32
41 0F6F FE21                    CP      21H
42 0F71 38F6                    JR      C,KADD
43 0F73 C640            .CHG2:  ADD     A,40H
44 0F75 FE7B                    CP      7BH
45 0F77 D8                      RET     C
46 0F78 D65A                    SUB     5AH
47 0F7A 5F                      LD      E,A
48 0F7B 18EC                    JR      KADD
49 0F7D                 ;
50 0F7D CB50            TENKEY: BIT     2,B
51 0F7F 28D9                    JR      Z,LMONLY
52 0F81 D60A                    SUB     0AH
53 0F83 87                      ADD     A,A
54 0F84 21950D                  LD      HL,TENTBL
55 0F87 5F                      LD      E,A
56 0F88 1600                    LD      D,0
57 0F8A 19                      ADD     HL,DE
58 0F8B 5E                      LD      E,M
59 0F8C 23                      INC     HL
60 0F8D 56                      LD      D,M
```

```
0F8E EB                         EX      DE,HL
0F8F CDD001                     CALL    .HL
0F92 C3D20E             KFINB:  JP      KFINA
0F95                    ;
0F95 D652               KYCTRL: SUB     52H
0F97 87                         ADD     A,A
0F98 C605                       ADD     A,5
0F9A CB50                       BIT     2,B
0F9C C8                         RET     Z
0F9D 3C                         INC     A
0F9E C9                         RET
0F9F CB50               KYTAB:  BIT     2,B
0FA1 3E1B                       LD      A,1BH
0FA3 C8                         RET     Z
0FA4 3E1F                       LD      A,1FH
0FA6 C9                         RET
0FA7                    ;
0FA7 00                 MODE':  DEFB    0
0FA8                    ;
0FA8                    ;
0FA8                    ;
0FA8                    ;
0FA8 21D010             CANRVS: LD      HL,MODE
0FAB CBBE                       RES     7,M
0FAD 7E                 OUTRT:  LD      A,M
0FAE D3E0                       OUT     (E0H),A
0FB0 C9                         RET
0FB1 21D010             LAMODE: LD      HL,MODE
0FB4 CBB6                       RES     6,M
0FB6 CBAE                       RES     5,M
0FB8 18F3                       JR      OUTRT
0FBA 21D010             RVS:    LD      HL,MODE
0FBD CBFE                       SET     7,M
0FBF CBB6                       RES     6,M
0FC1 18EA                       JR      OUTRT
0FC3 21D010             SMALL:  LD      HL,MODE
0FC6 CBEE                       SET     5,M
0FC8 CBB6                       RES     6,M
0FCA 18E1                       JR      OUTRT
0FCC 21D010             GRAPH:  LD      HL,MODE
0FCF CBAE                       RES     5,M
0FD1 CBF6                       SET     6,M
0FD3 CBBE                       RES     7,M
0FD5 18D6                       JR      OUTRT
0FD7                    ;
0FD7                    ;
0FD7                    ;KEY TABL
0FD7                    ;
0FD7                    ;
0FD7 1011               KTBL:   DEFW    1110H           ;F1 F2
0FD9 1213                       DEFW    1312H           ;F3 F4
0FDB 1415                       DEFW    1514H           ;F5 F6
0FDD 1617                       DEFW    1716H           ;F7 F8
0FDF                    ;
0FDF 1819                       DEFW    1918H           ;F9 F10
0FE1 3839                       DEFM    "89"
0FE3 1A2E                       DEFW    2E1AH           ;00 .
0FE5 2B2D                       DEFM    "+-"
0FE7                    ;
0FE7 30313233                   DEFM    "01234567"
```

```
0FEB 34353637
0FEF                    ;
0FEF 1B20                       DEFW    201BH           ;TAB SP
0FF1 0D02                       DEFW    020DH           ;CR UP
0FF3 0104                       DEFW    0401H           ;DOWN LEFT
0FF5 030B                       DEFW    0B03H           ;RIGHT BREAK
0FF7                    ;
0FF7 2F                         DEFM    "/"
0FF8                    ;
0FF8 5E5C3F2E                   DEFM    "^\?.,"
0FFC 2C
0FFD                    ;
0FFD 30313233                   DEFM    "01234567"
1001 34353637
1005                    ;
1005 3839                       DEFM    "89"
1007 3A3B                       DEFW    3B3AH           ; : ;
1009 2D40                       DEFM    "-@"
100B 5B1E                       DEFW    1E5BH           ;[ NUL
100D                    ;
100D 5D1E                       DEFW    1E5DH           ;NUL ]
100F 0507                       DEFW    0705H           ;HOME DEL
1011 1E1E                       DEFW    1E1EH
1013 1E1E                       DEFW    1E1EH
1015 84                 KTBLS:  DEFB    84H
1016                    ;
1016 7E                         DEFB    7EH
1017 7C82                       DEFW    827CH
1019 3E3C                       DEFW    3C3EH
101B                    ;
101B 5F21                       DEFW    215FH
101D 2223                       DEFW    2322H
101F 2425                       DEFW    2524H
1021 2627                       DEFW    2726H
1023                    ;
1023 2829                       DEFW    2928H
1025 2A2B                       DEFW    2B2AH
1027 3D60                       DEFW    603DH
1029 7B1E                       DEFW    1E7BH
102B                    ;
102B 7D1E                       DEFW    1E7DH
102D 1E1E                       DEFW    1E1EH
102F 1E1E                       DEFW    1E1EH
1031 1E1E                       DEFW    1E1EH
1033 8398               KTBLG:  DEFW    9883H
1035 8886                       DEFW    8688H
1037 9A9E                       DEFW    9E9AH
1039 9B99                       DEFW    999BH
103B                    ;
103B 8089                       DEFW    8980H
103D 908D                       DEFW    8D90H
103F 8F92                       DEFW    928FH
1041 948C                       DEFW    8C94H
1043                    ;
1043 8B97                       DEFW    978BH
1045 9F96                       DEFW    969FH
1047 9C8A                       DEFW    8A9CH
1049 8795                       DEFW    9587H
104B                    ;
104B 859D                       DEFW    9D85H
```

```
01 104D 8E5E                        DEFW   5E8EH
02 104F 5C81                        DEFW   815CH
03 1051 FF91                        DEFW   91FFH
04 1053                     ;
05 1053 F5                  MSTOP':  PUSH   AF
06 1054 CD9504                       CALL   MSTOP
07 1057 F1                           POP    AF
08 1058 C9                           RET
09 1059 CDDC04              TMARK':  CALL   DEL1M
10 105C C3B003                       JP     TMARK
11 105F                              SKP    H
```

```
01 105F                 ;   DATA PART
02 105F                 ;
03 10C0                         ORG     10C0H
04 10C0                 ;
05 10C0                 TABDAT: ENT                     ;TAB DATA AREA (16 BYTE)
06 10C0 05                      DEFB    5
07 10C1 0A                      DEFB    10
08 10C2 0F                      DEFB    15
09 10C3 14                      DEFB    20
10 10C4 19                      DEFB    25
11 10C5 1E                      DEFB    30
12 10C6 23                      DEFB    35
13 10C7 28                      DEFB    40
14 10C8 2D                      DEFB    45
15 10C9 32                      DEFB    50
16 10CA 37                      DEFB    55
17 10CB 3C                      DEFB    60
18 10CC 41                      DEFB    65
19 10CD 46                      DEFB    70
20 10CE 4B                      DEFB    75
21 10CF 00                      DEFB    0
22 10D0                 ;
23 10D0                 MODE:   ENT
24 10D0 12                      DEFB    12H
25 10D1                 DSPXY:  ENT                     ;CURSOR X POSITION
26 10D1 00                      DEFB    0
27 10D2                 DSPXY1: ENT                     ;CURSOR Y POSITION
28 10D2 00                      DEFB    0
29 10D3                 MANG:   ENT                     ;CONT LINE? (28 BYTE)
30 10D3 0000                    DEFW    0
31 10D5 0000                    DEFW    0
32 10D7 0000                    DEFW    0
33 10D9 0000                    DEFW    0
34 10DB 0000                    DEFW    0
35 10DD 0000                    DEFW    0
36 10DF 0000                    DEFW    0
37 10E1 0000                    DEFW    0
38 10E3 0000                    DEFW    0
39 10E5 0000                    DEFW    0
40 10E7 0000                    DEFW    0
41 10E9 0000                    DEFW    0
42 10EB 0000                    DEFW    0
43 10ED 0000                    DEFW    0
44 10EF                 FKAE:   ENT                     ;FKEY AREA END ADRS
45 10EF 1412                    DEFW    FKEND
46 10F1                 KYBDA:  ENT
47 10F1 00                      DEFB    0
48 10F2                 KSTD:   ENT
49 10F2 00                      DEFB    0
50 10F3 0000                    DEFW    0
51 10F5 0000                    DEFW    0
52 10F7 0000                    DEFW    0
53 10F9 0000                    DEFW    0
54 10FB 0000                    DEFW    0
55 10FD                 ;
56 10FD                 KINTF:  ENT                     ;KEY INT FLAG
57 10FD 00                      DEFB    0
58 10FE                         SKP     H
```

```
01 1100                         ORG     1100H
02 1100                 BUFER:  ENT
03 1100                         DEFS    128             ;BUFER 40B, STACK 88B
04 1180                 IBUFE:  ENT
05 1180                         DEFS    1
06 1181                 IBU1:   ENT
07 1181                 NAME:   ENT
08 1181                         DEFS    17
09 1192                 IBU18:  ENT
10 1192                 SIZE:   ENT
11 1192                         DEFS    2
12 1194                 IBU20:  ENT
13 1194                 DTADR:  ENT
14 1194                         DEFS    2
15 1196                 IBU22:  ENT
16 1196                 EXADR:  ENT
17 1196                         DEFS    2
18 1198                 IBU24:  ENT
19 1198                 ;
20 1198  P              FARE:   EQU     1200H
21 1200                         ORG     FARE
22 1200 0D                      DEFB    0DH
23 1201 0D                      DEFB    0DH
24 1202 0D                      DEFB    0DH
25 1203 0D                      DEFB    0DH
26 1204 0D                      DEFB    0DH
27 1205 0D                      DEFB    0DH
28 1206 0D                      DEFB    0DH
29 1207 0D                      DEFB    0DH
30 1208 0D                      DEFB    0DH
31 1209 0D                      DEFB    0DH
32 120A 0D                      DEFB    0DH
33 120B 0D                      DEFB    0DH
34 120C 0D                      DEFB    0DH
35 120D 0D                      DEFB    0DH
36 120E 0D                      DEFB    0DH
37 120F 0D                      DEFB    0DH
38 1210 0D                      DEFB    0DH
39 1211 0D                      DEFB    0DH
40 1212 0D                      DEFB    0DH
41 1213 0D                      DEFB    0DH
42 1214                 FKEND:  DEFS    140
43 12A0                 ;
44 12A0  P              FAREN:  EQU     A0H             ;FKEY BYTE SIZE
45 12A0  P              FKEYN:  EQU     20              ;# OF FKEY
46 12A0                 ;
47 12A0                 FKMAX:  EQU     FARE+FAREN
48 12A0  P              SCRN:   EQU     D000H           ;VRAM ADRS
49 12A0                         END
```

```
#LINE  0672  *N     0BBE  ..KEY  0E89  .AMPM  0A55  .C2DAT 0A2D
.CH#   0B98  .CHG1  0F61  .CHG2  0F73  .CSMDT 0339  .DI    0005
.FLSDT 0CE2  .HL    01D0  .INIC1 0A45  .MWARK 015A  .OCTV  0B03
.ONTYO 0B19  .PON1  0910  .PUSHR 0004  .RATIO 0BC8  .RET   0756
.SCRAD 079B  .SCRSZ 079E  .SUMDT 02AE  2HEX   05B1  2HEX1 05C6
??EOL  08D2  ??EOL2 08E1  ??INST 086D  ??KEY  0D77  ??TOL  08C1
?CLER  05D3  ?DINT  05D4  ?DPCT  0714  ?DSP   0641  ?DSP79 062D
?INST  07CA  ?PNT1  0907  ?PONT  0904  ?SAVE  0D1A  ?SHIFT 0747
?SWAP  0DE6  ASC    0583  ASCI   0583  AUTRT3 0CE9  AUTRT4 0CFD
AUTRT6 0D0C  BELL   0A80  BLINK  0D58  BLINK2 0D5D  BLINK4 0D6F
BLK1   04A4  BLK3   04A1  BLK4   049E  BRK    0531  BRKEY  0527
BUFER  1100  CANRVS 0FA8  CH3979 0631  CH4080 07A2  CHANGE 0DDD
CHR40  098F  CHR80  0958  CHX0   096D  CHX2   0979  CKRNG  0620
CKRNGL 061C  CKS1   03E8  CKS2   03F6  CKS3   03FA  CKSUM  03E2
CLATAB 0DCA  CLRS   07FD  CLRTAB 0DC0  COMES  05D9  CRDIS  0ED5
CURSD  07DA  CURSL  07E5  CURSR  07CD  CURSR2 07D5  CURSU  07F0
D1M    04CA  DATA   0EE5  DATA2  0EF4  DEL    082A  DEL0   0839
DEL1   084C  DEL1M  04DC  DEL6   04D6  DELLN  0811  DELLN2 0815
DELSUB 081B  DI     0E0E  DI2    0E1B  DI4    0E35  DISPM  0C49
DLY    0520  DLYR   0519  DSCL   0948  DSCL1  092B  DSMAG  08F6
DSP0   0651  DSP2   066C  DSP3   0678  DSPNAM 06A4  DSPRED 091A
DSPTAB 067E  DSPW   092D  DSPWRR 0925  DSPXY  10D1  DSPXY1 10D2
DTADR  1194  DUMP   011D  DUMP0  0126  DUMP1  012B  DWLDI' 093E
DWLDIR 093A  DWLDRN 0943  EDGE   0405  EDGE1  040D  ESET   0542
EXADR  1196  FARE   1200  FAREN  00A0  FFWD   04B2  FKAE   10EF
FKEND  1214  FKEYN  0014  FKMAX  12A0  FLASW  0CDE  FNCOM  01A4
FOUMES 020D  FR     04A8  FUNC   0C62  FUNC2  0CB4  GAP    0386
GAP1   0396  GAP2   039E  GAP3   03A4  GATESI 0017  GETCRT 0C7C
GETKY  0610  GETL   0BE5  GETL0  00ED  GETL00 0C00  GETL01 0C0A
GETL02 0C1D  GETL03 0C23  GETL04 0C2E  GETLBR 00F0  GETLR  0C79
GLOP2  0CA7  GOOUT  00AE  GRAPH  0FCC  GRPHO  0EC7  GSHFO  0ED8
GT2    0C4E  GT5    0C6E  GTBRK  0C73  GTCR   0C7E  GTCR2  0CA0
HEX    058D  HEX1   059F  HIGHSC 04B8  HL1    05AF  HLHEX  05A2
HOCT   0B3D  HOME   0808  IBU1   1181  IBU18  1192  IBU20  1194
IBU22  1196  IBU24  1198  IBUFE  1180  INSOFF 0867  INST   0864
INST1  0882  INST2  08AC  IOTBL  0049  JUMP   0217  KADD   0F69
KBSET  0536  KDATW  0013  KDIN   0F01  KEY    0E71  KEYKEY 06ED
KEYREP 0D27  KEYRP2 0D4B  KFIN0  0C51  KFINA  0ED2  KFINB  0F92
KIN    0544  KIN1   0547  KIN2   0190  KINP   0550  KINT   0E1F
KINTF  10FD  KINTON 0E47  KNUMBS 0BE3  KSML   0F6C  KSTD   10F2
KTBL   0FD7  KTBLG  1033  KTBLS  1015  KYBDA  10F1  KYCTRL 0F95
KYTAB  0F9F  LAMODE 0FB1  LASTD  0BE4  LDINF  0257  LENG   08E8
LETNL  0764  LETNL2 0768  LMOD   0ECB  LMONLY 0F5A  LOAMES 01F9
LONG   04FE  M#TBL  0B5B  MAGA   08FB  MANG   10D3  MCLECT 00F9
MELDY  0AA3  MENAME 0151  MLD1   0AA9  MLD2   0AD9  MLD3   0AE1
MLOAD  0149  MLOVE  01AF  MNAM1  0172  MODE   10D0  MODE'  0FA7
MONIT  0000  MOT1   042A  MOT2   0437  MOTOR  0416  MOTW   043C
MOTWG  0456  MPLAY  0499  MR     00FF  MRUN   0CCE  MSAVE  014C
MSG    06B5  MSGX   06AF  MSGX1  06B9  MSTOP  0495  MSTOP' 1053
MTBL   0B43  MVERY  01E9  MVRFY  0147  NAME   1181  NAMECK 01D1
NL     0757  NLMSG  06AC  NLPHLS 0634  NOKKEY 0705  OKMES  0214
ONP1   0AE8  ONP2   0AF5  ONP3   0B0C  ONPU   0AE5  OPEN   044B
OPTBL  0B73  OUTRT  0FAD  PLAY   045A  POPR   0E08  POPR2  0E09
POPXY  0818  PRNT   063C  PRNTS  063A  PRNTT  0695  PRTHL  0568
PRTHLS 0637  PRTHX  056D  PUSHKI 0E37  PUSHR  0DF1  PUSHR2 0DFD
RBY1   0363  RBY2   037B  RBY3   0384  RBYTE  035F  RD1    0269
RD2    0274  RDDAT  027D  RDINF  025F  REGIST 09A5  REPTCT 000D
RETHB  02C0  RETHB1 03AD  ROT    0EF8  ROTE   0EFE  RSHFO  0EC4
RTAP1  02D0  RTAP2  02E8  RTAP3  030B  RTAPE  02CC  RVS    0FBA
RVSO   0EDE  RYTHM  0B8C  RYTHM1 0BA1  RYTHM2 0BBC  RYTHM3 0BAD
```

```
RYTHMB 0B92  SAME   05C8  SAVEGO 019D  SAVEXY 07E1  SCAN   0E65
SCREND 000C  SCRN   D000  SCROL  077D  SCROST 000B  SCRSET 06C3
SELO0  002D  SERSP  046A  SETMES 05E0  SETTAB 0DB1  SHORT  04E2
SIZE   1192  SLOW   000F  SMALL  0FC3  SMALLO 0ECE  SMSHFO 0EDB
SOUT   0A87  SOUT1  0A89  SOUT2  0A99  SSET   053F  SSP1   0471
SSP2   048D  ST     00B1  STAB   0DD2  START  003B  START2 0069
STPRET 027A  SUMMES 0600  SWEP   0E82  SWRK   0015  TAB1   0682
TABDAT 10C0  TABLE1 0B7D  TAPER  030E  TDPCT  0727  TEMPW  0016
TENKEY 0F7D  TENTBL 0D95  TIMRD  0A16  TIMST  09CA  TITMES 0E4F
TM1    03B9  TM2    03BA  TM3    03CC  TM4    03E0  TMARK  03B0
TMARK' 1059  TMR1   0A66  TMS1   09E4  TMUP   0A73  TMX    0A40
TMX1   0A6D  TMX2   0A54  TSPE   04C2  TVF1   031D  TVF2   032B
TVRFY  0319  TWO    0F56  VERFY  0286  VERMES 0202  WBY1   0354
WBYTE  034E  WPRMES 05F2  WRDAT  024E  WRI1   0227  WRI2   023E
WRIMES 05E9  WRINF  021D  WTAP1  029F  WTAP2  02A8  WTAP4  02C3
WTAPE  029B  XTEMP  09BE  Z80KT  051C
```

SHARP CORPORATION

TINSE0054PAZZ
Printed in Japan

SYSTEM PROGRAM MANUAL

SHARP