EHT-10/EHT-10/2

Development Tool User's Guide

EPSON

Y24399100701
M0178

REMARKS

(1) All rights reserved.Reproduction of any part of this manual in any form whatsoever without SEIKO EPSON's expressed written permission is forbidden.

(2) The contents of this manual are subject to change without notice.

(3) All efforts have been made to ensure the accuracy of the contents of this manual. However, should any errors be detected, SEIKO EPSON would greatly appreciate being informed of them.

(4) The above notwithstanding, SEIKO EPSON can assume no responsibility for any errors in this manual or their consequences.

CONTENTS

Notes

Several marks used in this manual are different from the actual EHT-10/EHT-10/2's key marks.
The differences are as follows.

Marks in this manual	Actual Key
ENTER	Return key
<	Left arrow
>	Right arrow
^	Up arrow
v	Down arrow

PREFACE

It is very difficult to create software by EHT-10 series computers because of the hardware limitations such as screen size, and keyboard functions. As a result, the EHT-10 software development tool is created to provide development environment in real time by connecting the tool to a personal computer (CP/M or MS-DOS computer) used as the host computer.

Further, a utility is provided to convert created programs into the format required when the programs are stored in ROM.

This manual explains the EHT-10 software development tool and how to use the tool. Refer to the following manuals for details on handling a EHT-10 main frame and on BASIC language:

- EHT-10/EHT-10/2 Operation Manual
  This manual explains the EHT-10 basic functions and how to handle the EHT-10.

- EHT-10/EHT-10/2 BASIC Reference Manual
  This manual explains the BASIC statements and their functions in detail.

- EHT-10/EHT-10/2 System Development Guide
  This manual explains OS and the system in detail.

In this manual, EHT-10 series (EHT-10, EHT-10/2, and EHT-10/2B) is simply called EHT-10.

Chapter 1 SYSTEM INTRODUCTION

1.1 Overview

It is very difficult to create programs by using EHT-10 because software such as editor, assembler, and linker required for program development are not provided in a EHT-10 computer and because of hardware limitations such as the screen size and keyboard functions. The EHT-10 software development tool is provided to create programs for EHT-10 by using a general personal computer. The following items are required to create programs for EHT-10:

- EHT-10 main frame (See Note.)
- Development cartridge
- Host computer (CP/M or MS-DOS computer)
- Connection cable (RS-232C null model cable)
- Software development utility programs

Debugging using the development tool is operated in EHT-10. In other words, EHT-10 specifications for CPU, memory, OS, I/O ports are used. During debug operations, the host computer operates as a EHT-10 terminal. All I/O devices of EHT-10 are freed to be used for the program to be created so that debugging can be done by using all I/O devices that are in actual operation status.
Further, the debugger for creating machine-language programs has the commands compatible with CP/M-80 DDT.

(Note) Extended RAM may be required depending on the size and execution mode of the program to be created. See "Section 3.4 Development Utilities and RAM Configuration" for details.

This section explains programs that can be created by the EHT-10 software development tool. Programs can be created in the following two languages:

i) BASIC language

The BASIC language is used if a program is to be operated by the BASIC interpreter internally built in EHT-10 in the ROM base.

ii) Machine language

The machine language is used when program in Z-80 machine language are created.

Further, these languages can be combined to create programs. BASIC provided in a EHT-10 does not have command entry wait status. In other words, inputting, editing, and listing a program cannot be executed. By using the development tool, however, BASIC commands can be input from the host computer and listing can be output by the host computer. Since the created programs operate under CP/M environment, the programs can freely use BIOS and BDOS belonging to CP/M and the extended BIOS provided exclusively to the EHT-10.

The execution mode of a program to be created can be selected from the following two:

i) Load and execute mode

A program is loaded in the CP/M TPA or the BASIC program area before execution. Programs can be loaded from a media such as RAM disk, application ROM, IC card, or floppy disk, or from a communication port by using the DLL function.

ii) ROM-based mode

A program in an application ROM is directly executed in the ROM without being loaded in TPA. However, a BASIC program cannot be executed in this mode.

Table 1-2-1 shows the relationship between programs to be created and the above modes.

Table 1-2-1 Modes and Programs to be Created

	Load and execute mode	ROM-based mode
BASIC language	Allowed	———
Machine language	Allowed	Allowed

Figure 1-1 shows the system configuration used to create programs.

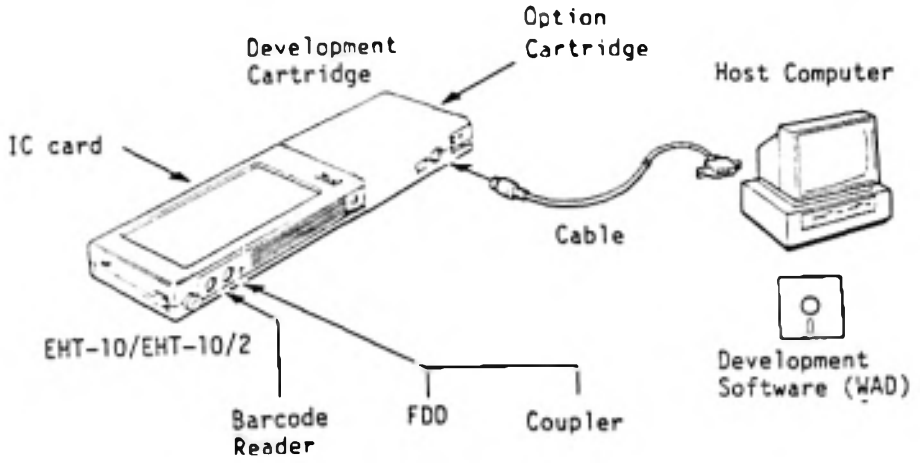


Fig. 1-1 Development Tool System Configuration

i) EHT-10 main frame

EHT-10, EHT-10/2, or EHT-10/2B can be used as the EHT-10 main frame. Programs, however, must be created according to the correct specifications because the specifications of LCD screen and keyboard differs depending on the main frame. All the EHT-10 functions are freed to be used for the program to be created during debug operation . Note that the user BIOS cannot be used and the size of usable RAM disk is limited, because the user BIOS and a part of extended RAM disk are used by the development utility programs. See "Section 3.4 Development Utilities and RAM Configuration" for details.

ii) Development cartridge

The development cartridge is connected to the cartridge interface used for an option cartridge such as a printer cartridge. However, the development cartridge also has its option cartridge interface so that an option cartridge can be used during debug operation. The development cartridge also has the RS-232C interface for the communication with the host computer.

iii) Host computer

One of the computers listed below can be used as the host computer. These are CP/M-80 or MS-DOS computers that have FDD.

- CP/M-80 computers
  EPSON QX-10
  PX-8
  PX-4/HX-40

- MS-DOS computers
  EPSON QX-11
  QX-16
  EPSON PC series (EQUITY series for U.S.A.)
  IBM PC, PC/XT, PC/AT
  IBM 5550

iv) Connection cable

The connection cable is used to connect the development unit to a host computer through the RS-232C interface. The connection cable differs depending on the host computer as shown in the table below.

Host computer	Cable type number
EPSON QX-10 QX-11 QX-16 IBM 5550	#725
EPSON PX-8 PX-4,HX-40	#726
EPSON PCs,EQUITYs IBM PC,PC/XT,PC/AT	#762

v) Development utilities

Various utilities required for software development are provided.

1. Application software debugger
   After initiation, the application software debugger sends, receives, analyzes, or executes commands stored at the host computer and at EHT-10. The debug commands are compatible with that of CP/M-80 DDT. Further, the application software debugger can use the host computer as the BASIC screen editor.

2. PROM formatter
   The PROM formatter converts the format of a program created for EHT-10 to the format required to store the program in an application ROM. This operation is required before the program is stored in a ROM by a PROM writer.

Chapter 2 DEVELOPMENT CARTRIDGE

The development cartridge provides two functions when connected to a EHT-10 main frame. The first function is that development cartridge becomes the hardware interface for communication with the host computer. The second function is that the development cartridge is used to connect an option cartridge.

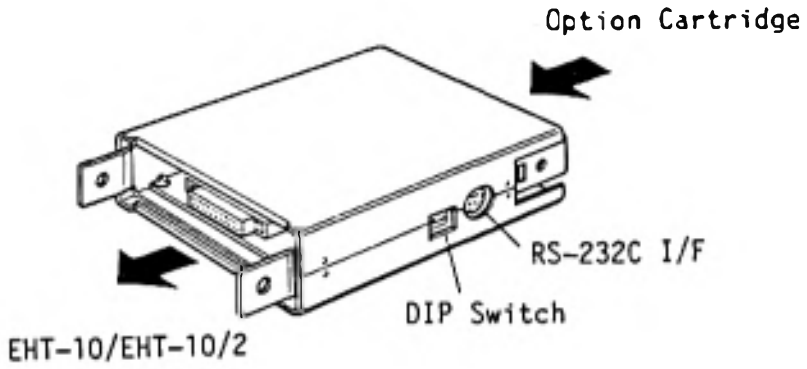


Fig. 2-1 Development Cartridge

2.1 Setting the DIP Switches

The development cartridge has DIP switches used to set conditions for communication with the host computer.

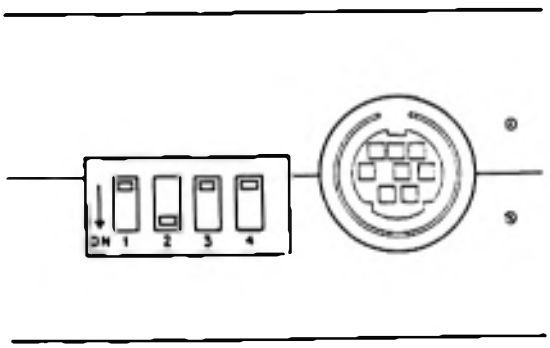


Fig. 2-2 Development Cartridge DIP Switches

The DIP switches must be set as shown below when the EHT-10 development tool is used.

SW1	2	3	4
OFF	ON	OFF	OFF

(Setting is the same as at shipping.)

The RS-232C interface is used for transmitting data to or from the host computer during debug operations. Therefore, the RS-232C interface cannot be used as a communication port by an application program.
The following figure and table show the connector pin assignment of RS-232C interface on the development cartridge.

Pin number	Direction	Signalname
1	-	GND
2	OUT	TXD
3	IN	RXD
4	OUT	RTS
5	IN	CTS
6	IN	DSR
7	OUT	DTR
8	IN	CD
E	-	CG

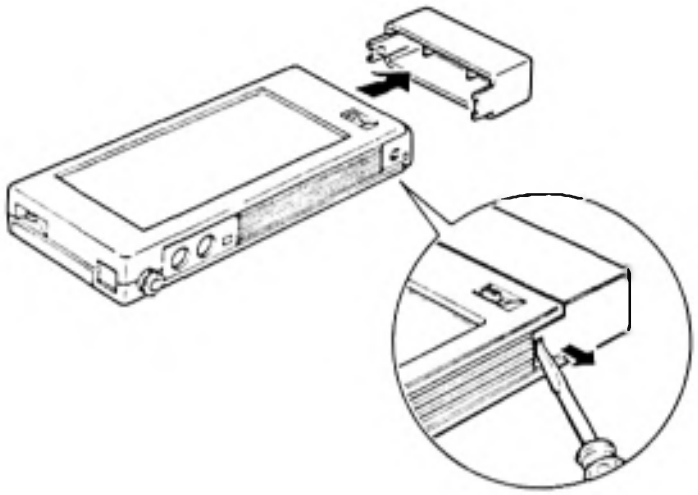
* The direction of a signal is indicated in the view from the development cartridge.

Thin standard and Phillips screwdrivers are required.

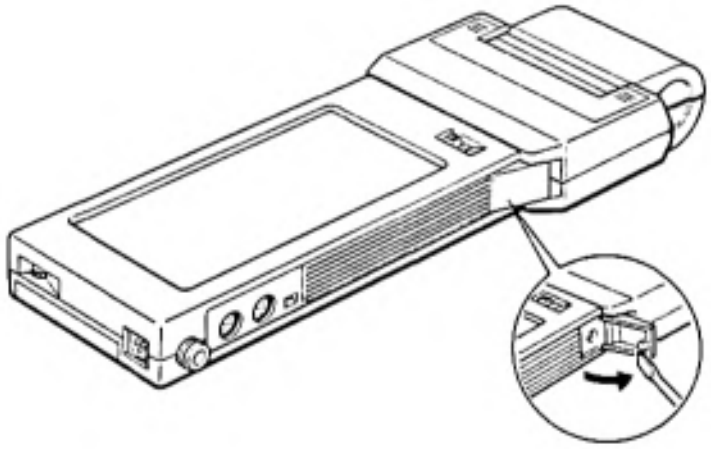
Step 1: Turn off the EHT-10 power.

Step 2: Proceed to step 4 if an option cartridge is required. Other wise, proceed to step 3.

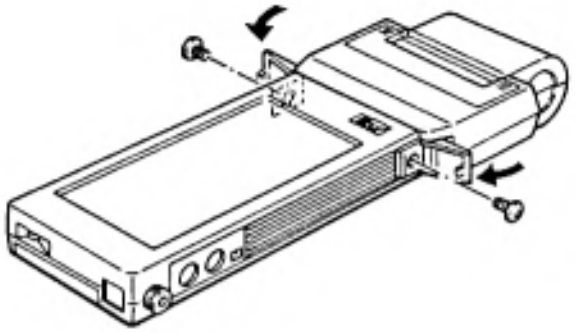
Step 3: Insert the tip of a thin standard screwdriver to a gap at both sides of the connector cover to remove the cover as shown below.



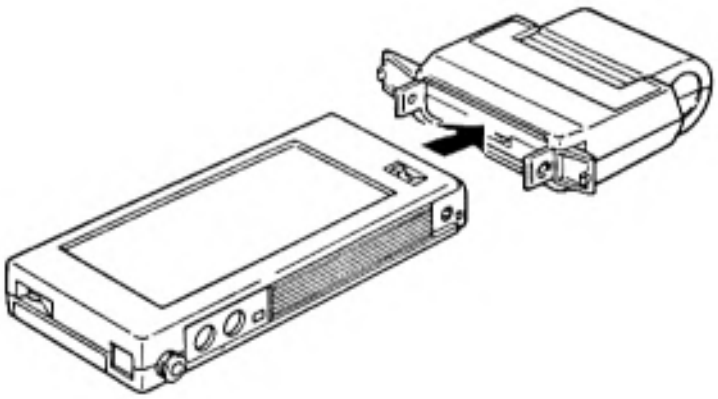
Step 4: Insert the tip of a thin standard screwdriver to a slit between a screw lid and the option cartridge at the both sides of the option cartridge to open the screw lids as shown below.



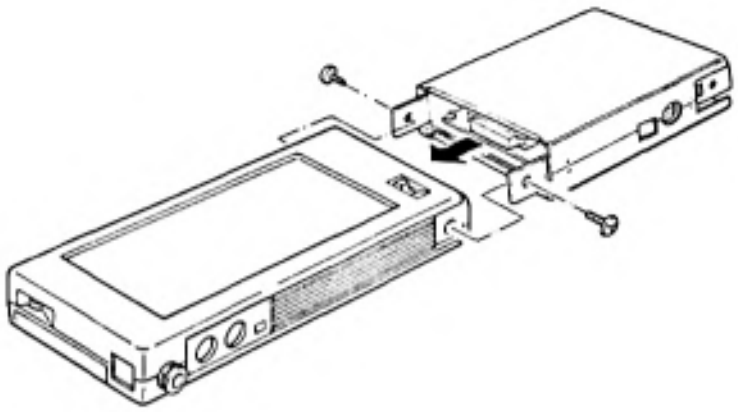
Step 5: Using a Phillips screwdriver, remove the two screws used to connect the option cartridge and the EHT-10 main frame.



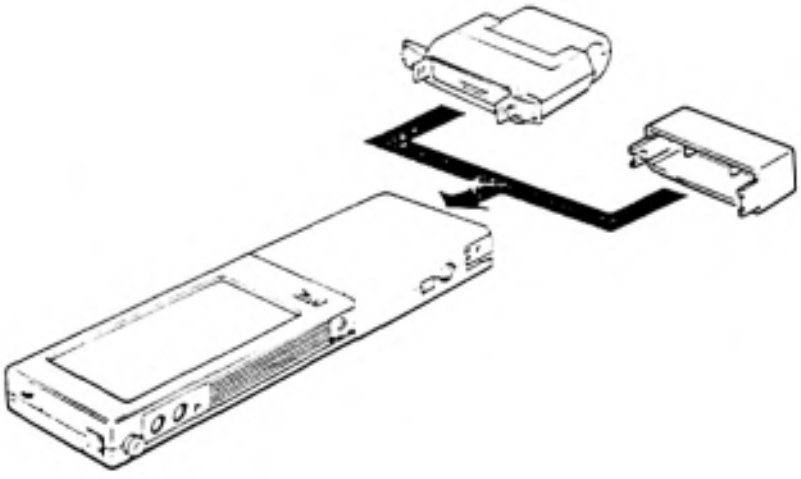
Step 6: Remove the option cartridge slowly from the EHT-10 main frame.



Step 7: Connect the development cartridge to the EHT-10 main frame in the direction indicated by the arrow shown in the figure below. Fasten the development cartridge to the main frame by two screws attached to the cartridge.



Step 8: If an option cartridge is required, insert the option cartridge to the head of the development cartridge, fasten the option cartridge with two screws, and close the screw lids. If no option cartridge is required, attach the connector cover removed from the EHT-10 main frame to the head of the development cartridge.



Chapter 3 DEVELOPMENT UTILITIES

3.1 Configuration of Development Utilities

The application software debugger and the PROM generation formatter are provided as the development utilities.

3.1.1 Application software debugger

The application software debugger functionally operates as machine-language and BASIC-language debuggers. The application software debugger consists of the following four program files (however, iv) WADINST.COM is used for MS-DOS computers only):

i) WAD.COM

WAD.COM is used at the host computer side to control the host computer. WAD.COM outputs debug commands from the host computer and outputs messages to the CRT screen from EHT-10. WAD.COM also operates as the screen editor for BASIC debug operations.
Further, WAD.COM automatically sets transmission speed: 9600 bps, data length: 8 bits, none parity, and stop bit: 2 bits, as the conditions for communication between the host computer and the EHT-10.

ii) WADL.COM

WADL.COM loads WADW.COM. WADL.COM is loaded to the EHT-10 TPA by the EHT-10 DLL function and is executed.

iii) WADW.COM

WADW.COM is loaded to a subbank (32 Kbytes) on the EHT-10 extended RAM if RAM is extended by WADL.COM. Other wise, WADW.COM is loaded to the RAM in the main frame. WADW.COM analyzes and executes the commands sent from the host computer and outputs the results to the host computer.

iv) WADINST.COM

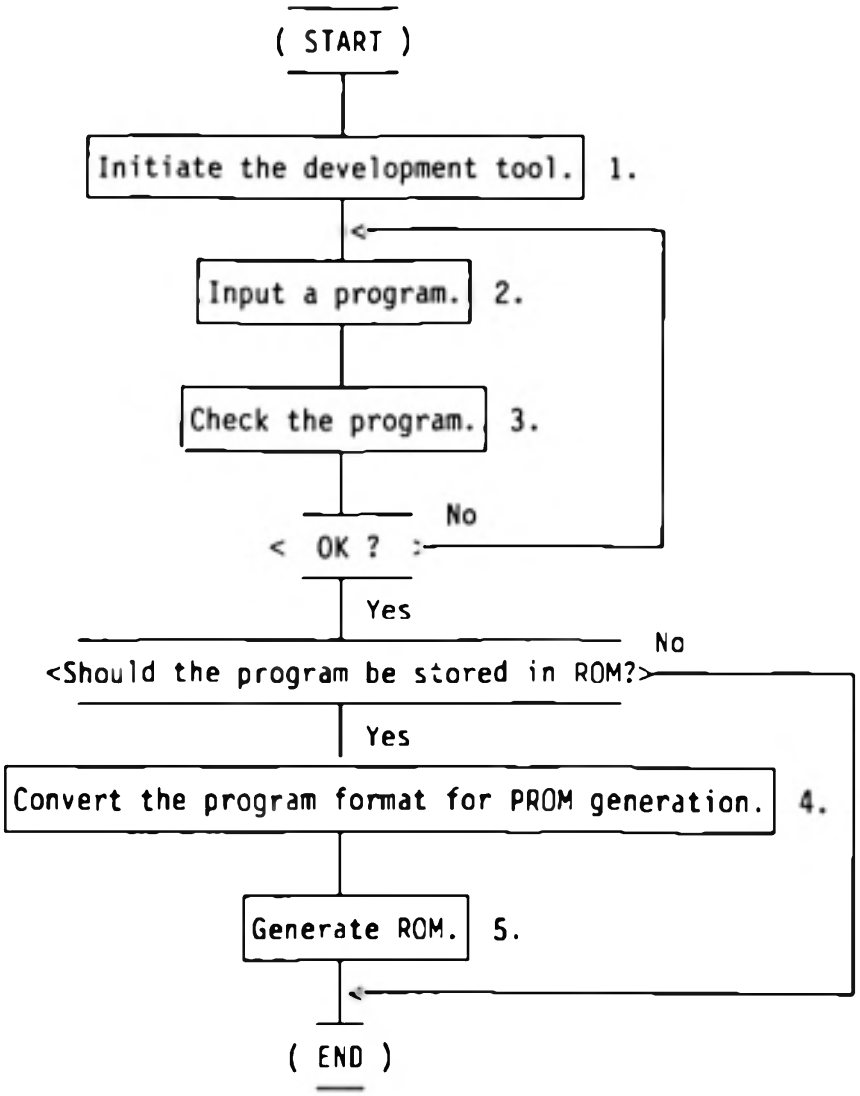
WADINST.COM is executed before WAD.COM is initiated if an MS-DOS computer is used as the host computer. WADINST.COM installs WAD.COM for the host computer. Once WAD.COM is installed, WADINST.COM is no longer required unless the type of host computer is changed.

3.1.2 PROM generation formatter

The PROM generation formatter is required when a created program is stored in a ROM. The PROM generations formatter is a utility used to convert a program into the file format exclusively used for application ROM.

The PROM generation formatter is stored as a file called WPROMFRM.EXE This formatter can converts programs into either the load and execute mode format or the ROM-based mode format.
Programs are sent to the PROM writer in the Intel HEX format.

This section explains how to create a program in BASIC language by using the EHT-10 development tool. The procedure to create a program is explained according to the flowchart shown below.



1 Initiating the development tool

See "Section 2.3 Attaching the Cartridge to EHT-10" for connecting the development tool. See "Section 5.1 Initiation and Termination" for initiating the development tool.

2 Inputting a BASIC source program

Select "BASIC program" of the development tool, initiate the EHT-10 BASIC interpreter, and then input a program. See "Chapter 5. BASIC-LANGUAGE DEBUGGER" for details.

3 Checking the input BASIC program

Select "BASIC program" as operation 2 and execute debug operation. See "Chapter 5. BASIC-LANGUAGE DEBUGGER" for details.

4 Converting the program format for PROM

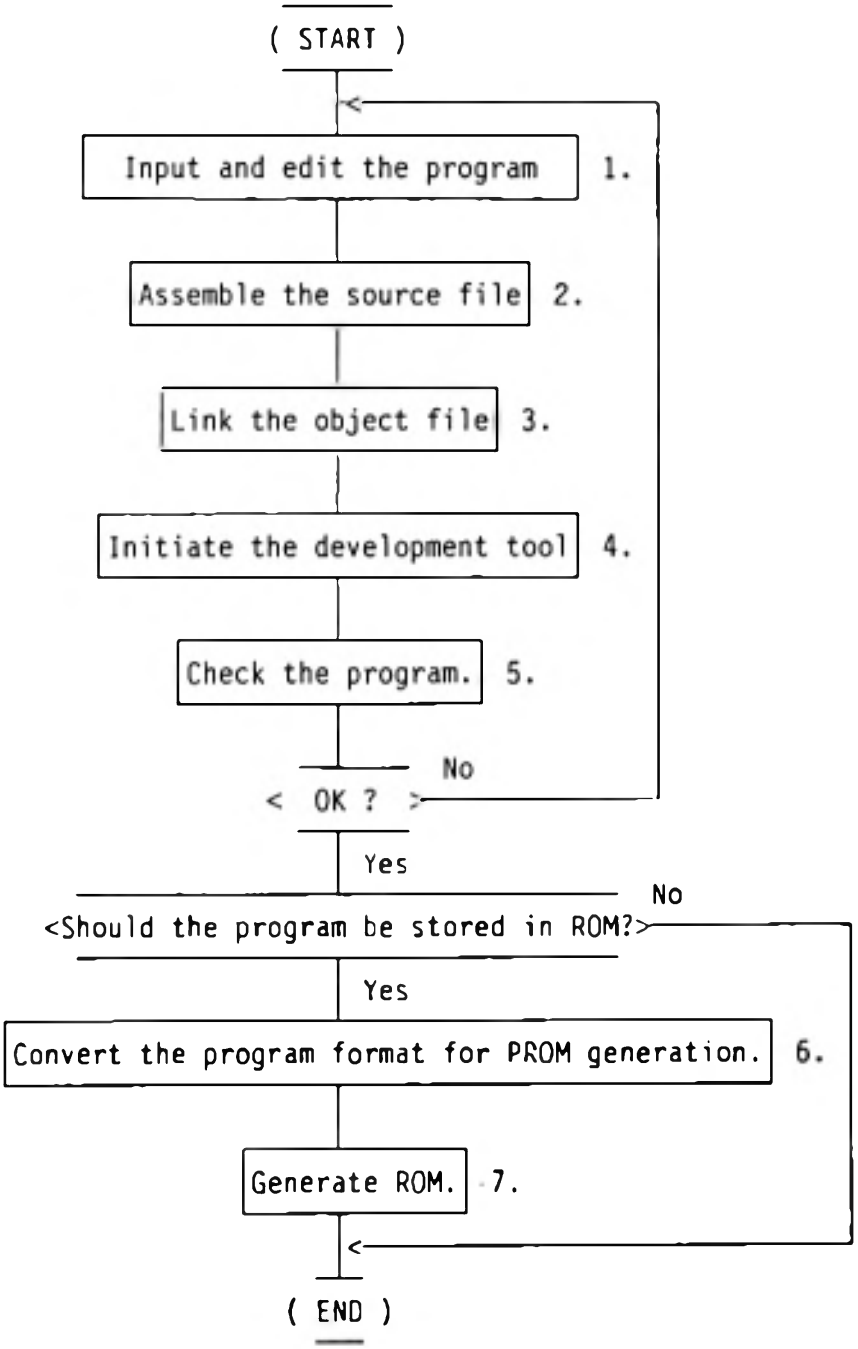
Use development utility program WPROMFRM to convert the format. In this case, format P cannot be selected for ROM-based mode. See "Chapter 6. PROM FORMAT UTILITY" for details.

5 Generating ROM

Send the created program file from the host computer to the PROM writer through a communication port and generate a ROM. Note that the Intel HEX format is used as the file format for send operation.

3.3 Machine - Language Program Development Procedure

This section explains how to create a program in the machine language by using the EHT-10 development tool. The procedure to create a program is explained in the flowchart shown below.



1 Inputting a source program written in machine language

Input a source program in mnemonic representation by using the text editor of the host computer.

<Example>
CP/M-80
- ED (built in a CP/M-80)
- WM (MicroPro)

MS/DOS
- EDLIN (built in a MS-DOS)
- WS (MicroPro)

2 Assembling the source file

Assemble the source file by using the Z-80 assembler or the cross assembler of the host computer.

<Example>
CP/M-80
- MACRO80 (Microsoft)

MS-DOS
- XMACRO80 (Nikkei)

3 Linking the object file

Use the linker corresponding to the assembler to link the object file.

<Example>
CP/M-80
- L80 (Microsoft)

MS-DOS
- XLINK80 (Nikkei)

4 Initiating the development tool

See "Section 2.3 Attaching the Cartridge to EHT-10" for connecting the development tool. See "Section 4.1 Initiation and Termination" for initiating the debugger.

5 Checking the machine-language program

The operation slightly differs depending on the mode of the target program. Further, if the target program is in ROM-based mode, note that the different banks are used for debug operation and after the program is stored in a ROM. See "Chapter 4. MACHINE-LANGUAGE DEBUGGER" for details.

6 Converting the program format for PROM

Convert the program format by the development utility PROM generation formatter (WPROMFRM.EXE). Select the ROM-based mode or the load and execute mode. See "Chapter 6. PROM FORMAT UTILITY" for details.

7 Generating a ROM

Send the program file from the host computer to the PROM writer through a communication port. Note that the Intel HEX format is used as the file format.

During debug operations, development utility WADW.COM is loaded and made resident in EHT-10 RAM. The WADW.COM location in the RAM differs depending on the program execution mode and whether an extended RAM is mounted.

In addition to WADW.COM, a part (256 x 5 bytes) of WAD.COM is loaded in the EHT-10 RAM. This part of WAD.COM is stored in the user BIOS in the EHT-10 main RAM regardless of the execution mode and of whether an extended RAM is mounted.

The rest of this section explains the RAM configuration. See "Appendix III RAM MEMORY MAP" for reference.

<For load and execute mode>

When the program to be debugged is in load and execute mode and an extended RAM is not mounted, WADW.COM is loaded in the user BIOS area (lower address of the part of WAD.COM 256 x 5 bytes) of the EHT-10 main RAM. In this case, the disk size of internal RAM in the main frame must be made smaller. The program to be debugged is stored in TPA of the main RAM.

When the program to be debugged is in load and execute mode and an extended RAM is mounted, WADW.COM is stored in the extended RAM and the program to be debugged is stored in TPA.

<For ROM-based mode>

An extended RAM must be mounted when the program to be debugged is in ROM-based mode. WADW.COM is loaded in the last 32 Kbytes of the extended RAM. The program to be debugged is loaded at the lower bank of WADW.COM in the extended RAM. In this case, the size of extended RAM must be determined according to the size of the program to be debugged.

If the size of a program to be debugged exceeds one bank size (32 Kbytes), split the program in the units of 32 Kbytes and store the split program in two or more banks. In this case, bank switch operation is required for moving data between the banks or for addressing.

Different banks are used for debug operation in an extended RAM and for expanding the created program in PROM. Because of this, the bank numbers and required addresses of a program that require bank switch operations must be modified before the program is stored in PROM so that the program can be executed in PROM.

<For a BASIC program>

When a BASIC application program is created or debugged, loading WADW.COM is not required because the EHT-10 BASIC interpreter is used. Therefore, the RAM configuration remains unchanged regardless of whether an extended RAM is mounted. The program to be debugged is stored in the EHT-10 BASIC program area.

As explained above, development utilities and a program to be created are loaded in an extended RAM, user BIOS, or TPA.

Table 3.1 shows the relationship between program mode, size, and EHT-10 RAM size conditions.

Table 3.1 RAM Size Conditions for Debug Initiation

Mode of a program to be created	Program size	Extended RAM	RAM size conditions
ROM-based mode	32 KB	Mounted	Extended RAM 64 Kbytes or more and internal RAM disk 38 Kbytes or less
	64 KB	Mounted	Extended RAM 96 Kbytes or more and internal RAM disk 38 Kbytes or less
	128 KB	Mounted	Extended RAM 160 Kbytes or more and internal RAM disk 38 Kbytes or less
Load and execute mode	Depends on the TPA size.	Mounted	Extended RAM 32 Kbytes or more and initial RAM disk 38 Kbytes or less
		Not mounted	Internal RAM disk 22 Kbytes or less WAD and WADW are loaded in a user BIOS area of 69 x 256 bytes.
BASIC	Depends on the BASIC program area size.	Whether an extended RAM is mounted does not make any differ-ence.	Internal RAM disk 38 Kbytes or less.

4.1 Initiation and Termination

Step 1 : Prepare the required devices and equipment

1 EHT-10
An EHT-10 or EHT-10/2/EHT-10/2B is required. Further, prepare other options if required.

2 Development cartridge

3 Host computer
A CP/M or MS-DOS computer explained in "Section 1.3 System Configuration" is required.

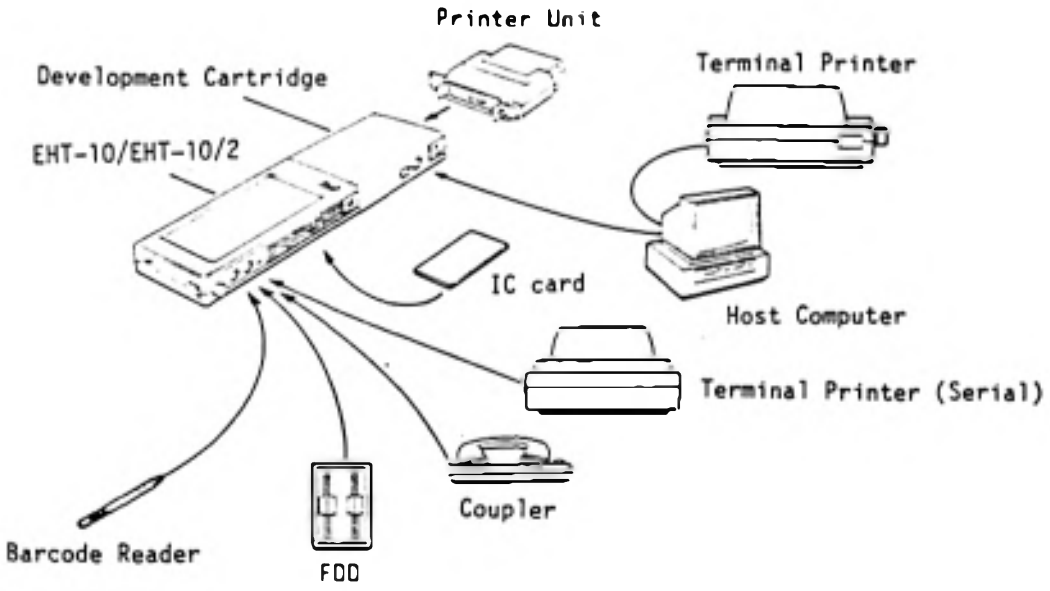
4 Connection cable
The required connection cable differs depending on the host computer. See "Section 1.3 System Configuration" for details.

5 Development utilities
Make a backup copy of the development utilities.

See "Section 1.3 System Configuration" and "Chapter3. DEVELOPMENT UTILITIES" for details.

Step 2 : Connecting each unit

Connect each unit as shown below. Keep the power of each unit off for safety cause. Options are connected if required.



Step 3 : Turning the power on

Turn on the power of each terminal, EHT-10, and host computer in this order.

Step 4 : Installing WAD
(This operation is required only when a MS-DOS computer is used as the host computer.)

Make the development utility disk enter writable status (free the write protect status). Store device driver ANSI.SYS in file CONFIG.SYS by specifying

```
DEVICE=ANSI.SYS
```

when the host computer is one of the following computers:

- QX-16
- EPSON PC, PC+
- IBM PC, PC/XT, PC/AT

Initiate MS-DOS, change the login drive to the drive containing the development utility disk, and then initiate WADINST.COM.

```
A>WADINST
```

Select a number corresponding to the host computer in the screen below.

```
WAD Install program Ver. X.XX
Current Machine type = None

0 - EPSON QX-11/QC-11
1 - EPSON QX-16
2 - IBM PC/EPSON PC
3 - IBM 5550
4 - NEC PC-9801 series
```

Step 5 : Initiating EHT-10 DLL

Initiate EHT-10 DLL and make EHT-10 enter receive wait status. Refer to EHT-10/EHT-10/2 Operations Manual for these operations.

Step 6 : Initiating WAD and loading WADL

Initiate CP/M of the host computer (MS-DOS is already initiated in step 4 if a MS-DOS computer is used as the host computer), change the login drive to the drive containing the development utility disk, and then initiate development utility WAD.COM.

```
A>WAD   (when WAD.COM is in drive A)
```

When WAD is initiated, the message shown below is displayed and WADL.COM is transferred.

```
EPSON Debbuger WAD       Ver. X.X
 WADL.COM Sending ................
```

When loading is ended, EHT-10 displays the DLL completion screen.

Step 7 : Development program mode selection screen

When the PRESS ( ) key is pressed in the DLL completion screen, the host computer displays the development program mode selection screen. At this point, message "User Bios is too small" is output to EHT-10 if the required user BIOS area cannot be reserved. See the section explaining error handling if this message is output.

```
Select execution type
  1 -- ROM-based program
  2 -- Load and execute program
  3 -- BASIC program

  Select (1/2/3)   ▌
```

Select a number corresponding to the program mode.

Step 8 : For a ROM-based mode program

(Proceed to step 9 for a load and execute mode program.) The ROM-size selection screen shown below is displayed when 1 corresponding to ROM-based mode is selected in the program mode selection screen.

```
  Select (1/2/3)   1

ROM-based program selected
Select ROM size
  1 -- 32 Kbytes
  2 -- 64 Kbytes
  3 -- 128 Kbytes

  Select (1/2/3)   ▌
```

Select a number corresponding to the ROM size to be specified.

Then, the message shown below is displayed and transferring WADW.COM is started.

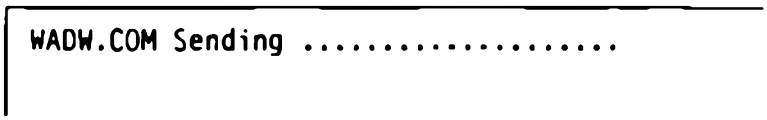
```
WADW.COM Sending ...................
```

When the transfer operation is ended, a prompt is displayed and processing enters debug command input wait status. If the RAM disk is not mounted as much as required, "extend RAM not mounted" is displayed at EHT-10 and operation must go back to step 8.

Use a R command to load the program to be created and to start debug operation. The debugger is terminated by a Z command.

Step 9 : For a load and execute mode program

When 2 corresponding to load and execute mode is selected, the message shown below is displayed and transferring WADW.COM is started.



When the transfer operation is ended, a prompt is displayed and processing enters debug command input wait status. "extend RAM not mounted" is displayed at the EHT-10 if the RAM disk is not mounted as much as required.

Use a R command to load the target program and to start debug operation. The debugger is terminated by a Z command.

* Effective function keys

CTRL + C

Processing branches out of WAD.COM when these keys are pressed during steps 6 to 9. However, these keys are not effective if a prompt is already being displayed.

ESC

The development program mode selection screen of step 7 is displayed when this key is pressed during steps 7 to 9. However, this key is not effective when a program is being loaded or a prompt is being displayed.

* Error handling

If an error occurs during steps 4 to 9, an error message is output and processing branches out of the debugger.

(1)Message: User Bios area is too small (displayed at the EHT-10.)

Cause: The user BIOS area could not be reserved as much as required.

Processing: The DLL initial screen is displayed at EHT-10 when the PRESS (<┘) key is pressed. At the host computer, processing branches out of the debugger when the CTRL + C keys are pressed.

Action: Set 38 Kbytes or less as the internal RAM disk size.

(2)Message: extend RAM not mounted (displayed at the EHT-10.)

Cause: An extended RAM was not mounted.

Processing: The DLL initial screen is displayed at EHT-10 when the PRESS (<┘) key is pressed. At the host computer, processing branches out of the debugger when the CTRL + C keys are pressed.

Action: Mount a RAM disk. If load and execute mode is selected, set 22 Kbytes or less as the internal RAM disk size.

The machine-language debugger provides powerful 10 commands that can specify subbanks.

Table 4-2-1 Machine-Language Debugger Commands

Command name	Function
A: Assemble	Translates the input mnemonic codes to the machine language.
C: Control print	Specifies to output the screen contents to a printer or clears the specification.
D: Dump	Displays the memory contents of the specified address range in hexadecimal or in ASCII code representation.
E: Search	Searches the specified address range for the specified data and displays all the addresses storing data equal to the specified data.
F: Fill	Fills the specified address range with the specified data.
G: Go	Executes the contents in the specified addresses.
K: Arithmetic operations and numeric conversion	Executes arithmetic operations (add and subtract) of the specified numerics and numeric conversion (from hexadecimal representation to decimal representation or vice versa).
L: List	Unassembles the machine language in the specified address range and displays the results in mnemonic codes.
M: Move	Moves the data blocks in the specified address range to a different specified address.
O: Port	Displays data from or outputs data to the specified I/O port.
P: Pass count set	Sets a break address and a pass count.
R: File read	Reads the specified file and sets the results in the debug bank from the specified address.
S: Memory set	Displays or modifies the contents of the specified address.
T: Trace	Executes the instructions starting from the address indicated by the program counter as much as the specified steps and displays the register contents every time a step is executed.

Command name	Function
U: Untrace	Executes the instructions starting from the address indicated by the program counter as much as the specified steps and displays the register contents before and after execution.
W: Write	Saves the memory contents of the specified address range with the specified file name.
X: Examine	Displays and modifies the contents of each register.
Z: End	Ends the debugger.

(1) Control keys

The following control keys can be used during command input operation:

BS (Back space key)
This key deletes a character preceding the cursor.

CTRL + X
These keys delete all the entered commands.

RETURN (Return key)
This key indicates the end of command input operation.

(2) Specifying a drive for file input operation

The drive specification for file input operation differs depending on the host computer or EHT-10.

(i) Drive specification at the host computer
Specify the drive name.

<Example> B:TEST.COM

(ii) Drive specification at the EHT-10
Specify W before the drive name.

<Example> WB:TEST.COM

(3) Specifying an address

An address specified is an address in the specified subbank of bank 0.

(4) Symbols

(i) [ ] : Contains a parameter that can be omitted.
(ii) <CR> : Press the RETURN key.
(iii) addr: Address (hexadecimal representation) in RAM
(iv) DATA : ASCII data or HEX data (Enclose ASCII data in a pair of double-quotation marks (").)

(5) Notes on command input operation

(i) Do not enter any blank between a command and its parameters.

(ii) "???" is displayed if an error is detected.

A: Assemble

Format

A[addr[,n]]

addr: Assemble start address (hexadecimal representation)

n: Subbank number

Function

The A command assembles the input Z-80 mnemonic codes in the units of lines and stores the results in the specified memory.

Explanation

The corresponding address is displayed when a command is input. Enter Z-80 mnemonic codes after the address.

(i) Specify an absolute address or * (own address) ± n for a relative instruction.

(ii) If an address is omitted, the address following an A, L, G, T, or U command is used as the default. However, the program start address is used immediately after a program is loaded by an R command.

(iii) An A command is ended under one of the following conditions:
- The return key is pressed without entering mnemonic codes.
- The return key is pressed after a period (.).

(iv) The same address is displayed again if there is an error in the entered mnemonic codes.

(v) If the subbank number is omitted, the subbank specified during R command execution is used.

(vi) Mnemonic code input conditions
- One or more blank codes must be placed between an operation code and parameters.
- Operands must not include any blank codes.

C: Control print

Format

C

Function

The C command outputs debug information displayed in the screen to the printer until the subsequent C command is input.

Explanation

The C command operates as a toggle switch for the printer output specification and the specification clearance. Processing enters wait status if the printer is busy when printer output is specified. When the CTRL + C keys are pressed in wait status, the printer output specification is cleared.

D: Dump

Format

D[addr1][,[addr2][,n]]

addr1: Dump start address (hexadecimal representation)
addr2: Dump end address (hexadecimal representation)
n: Subbank number

Function

The D command displays the contents of the specified address range in hexadecimal or ASCII code representation.

Explanation

(i) If the end address is omitted, the fixed number of bytes is used as data length starting from the start address. This number of bytes differs depending on the host computer as follows:
(1) PX-4/HX-40 or PX-8: 48 bytes
(2) Other host computer: 96 bytes

(ii) If the start address is omitted, the previous continuation address (the address next to the one displayed by the previous D command) is used as the start address.

(iii) If the subbank number is omitted, the subbank specified during R command execution is used.

(iv) Data 00 to 1F and A0 to FF are all indicated as periods(.) in the ASCII code representation.

(v) Dump operation can be temporarily interrupted by the CTRL + S keys or by the space key. Press the CTRL + S keys or the space key again to restart dump operation.

(vi) Dump operation can be ended by the RETURN key.

E: Search

Format

E[addr1],[addr2],data[,n]

addr1: Search start address (hexadecimal representation)

addr2: Search end address (hexadecimal representation)

data: Search data

Numerics...Up to 8 digits (hexadecimal representation)

Characters...Up to 4 characters (ASCII code)

n: Subbank number

Function

The E command searches the specified address range for the specified data and displays all the address containing data equal to the specified data.

Explanation

(i) An error occurs if addr1 is greater than addr2.

(ii) If the start address is omitted, search operation is done starting from the beginning of the specified bank.

(iii) If the end address is omitted, search operation is done until the end of the specified bank.

(iv) If the subbank number is omitted, the subbank specified during R command execution is used.

(v) Display operation can be temporarily interrupted by the CTRL + S keys or the space key. Press the CTRL + S keys or the space key again to restart display operation.

Example : Search with numerics

```
-E4000,4FFF,789ABC
  401F                -> Top address of the data equal to the specified
  4960                   data
-
```

F: Fill

Format

Faddr1,addr2,data[,n]

- addr1: Set start address (hexadecimal representation)
- addr2: Set end address (hexadecimal representation)
- data: Set data (1 byte, hexadecimal representation or ASCII 1 character)
- n: Subbank number

Function

The F command fills the specified address range with the specified data.

Explanation

(i) An error occurs if addr1 is greater than addr2.

(ii) If the subbank number is omitted, the subbank specified during R command execution is used.

Format

G[[addr1][,[addr2][,[n1][,n2]]]][/p]

addr1: Execution start address (hexadecimal representation)
n1: Execution start subbank number
addr2: Break address (hexadecimal representation)
n2: Break subbank number
/P: P=1. The break address is effective for a P command.
P=0. The break address is not effective for a P command.

Function

The G command executes the program from the specified address to the break address.

Explanation

The instruction at the break address is not executed and processing returns to the debugger.

(i) If the start address is omitted, the current PC value is used.

(ii) If the start subbank number is omitted, current subbank is used.

(iii) If the break subbank number is omitted, the subbank specified during R command execution is used.

(iv) If option P is omitted, /1 (effective) is used as the default value.

(v) A break address can be set by a P command also.

(vi) If a pass count is specified by a P command, the register information is displayed and the specified count is decreased by 1, every time processing passes through the break address. When the pass count becomes zero, program execution is terminated and the pass count is modified back to the specified value. However, if /0 is specified as option P, the above operation is not performed.

(vii) The address of the instruction placed next to the break address is displayed when processing returns to the debugger.

(viii) If a break address specified by a P command is reached before the break address specified by the G command, the break address specified by the G command is made ineffective.

(ix) Operation can be forcibly terminated. However, the PC and SP register contents are not guaranteed in this case. Because of this, load the program again by a R command and restart debug operation.

K: Arithmetic operations and numeric conversion

Format

Km[,n]

m and n: Arithmetic operations and numeric conversion (hexadecimal and decimal representation)

Function

The K command performs the arithmetic operations (add and subtract) of the specified numerics or converts numerics from hexadecimal representation to decimal representation or vice versa.

Explanation

(i) If m and n are both specified as the parameters, arithmetic operations m + n and m - n are executed and the results are displayed in hexadecimal representation in the order of addition and subtraction.

(ii) If only m is specified, numeric conversion is executed and the result is displayed.

(iii) A percent mark (%) is placed before the integer to input or display an integer.

Example

```
-K10              (A hexadecimal number is converted to an integer.)
 %16
-K10,20           (Arithmetic operations of two hexadecimal numbers)
 0030  FFF0       (Result of addition and result of subtraction)
-K%32             (An integer is converted to a hexadecimal number.)
 20
-K10,%10          (Arithmetic operations of a hexadecimal number and an
 001A,0006         integer)
-
```

L: List

Format

L[[addr1][,[addr2][,n]]]

addr1: Display start address (hexadecimal representation)
addr2: Display end address (hexadecimal representation)
n: Subbank number

Function

The L command unassembles data stored in the specified address range and displays the results in mnemonic codes.

Explanation

(i) If the start address is omitted, the continuation address of A, G, L, T, or U command is used.

(ii) If the end address is omitted, the fixed number of steps is used as the length starting from the start address. This number of steps differs depending on the host computer as follows:
(1) PX-4/HX-40 and PX-8: 6 steps
(2) Other host computer: 12 steps

(iii) If the subbank number is omitted, the subbank specified during R command execution is used.

(iv) Display operation can be temporarily interrupted by the CTRL + S keys or the space key. Press the CTRL + S keys or the space key again to restart display operation.

(v) Display operation can be terminated by the RETURN key.

M: Move

Format

Maddr1,addr2,addr3[,[B1][,B2]]

- addr1: Send start address (hexadecimal representation)
- addr2: Send end address (hexadecimal representation)
- addr3: Receive start address (hexadecimal representation)
- B1: Send subbank number
- B2: Receive subbank number

Function

The M command sends data blocks in the specified address range to the specified address.

Explanation

(i) An error occurs if addr1 is greater than addr2.

(ii) If send or receive subbank number is omitted, the subbank number specified during R command execution is used.

O: Port

Format

```
OI/O addr[,data]
   I/O addr:  I/O port address (hexadecimal representation)
       data:  1-byte output data (hexadecimal representation)
```

Function

The O command reads data from the specified input port and displays the results, or outputs specified data to the specified output port.

Explanation

(i) Data is output to the output port if output data is specified.

(ii) If output data is omitted, data is read from the specified port and the result is displayed.

P: Pass count set

Format

```
P[addr1[,[n][,n1]]]
   addr1:  Break address (hexadecimal representation)
     n1:  Subbank number
      n:  Pass count (hexadecimal representation)
```

Function

The P command sets a break address and a pass count.

Explanation

(i) Up to 8 break addresses can be set (up to 8 P commands must be used).

(ii) The maximum number that can be set as the pass count is 255. The pass count is decreased by 1 every time processing passes through the break address during G command execution. When the pass count becomes zero, program execution is terminated and the pass count is modified back to the specified value.

(iii) Specify 0 as n to clear a break address.

(iv) If addr1, n1, or n is omitted, the break address and pass count already set are displayed.

(v) If only n is omitted, 1 is set as the pass count.

(vi) If the subbank number is omitted, the subbank specified during R command execution is used.

Example

```
-P                          (Breakpoint display)
 1F   4273 01               (Count, address, and subbank number)
 05   4621 01
-P542F,3                    (Breakpoint setting)
-
```

R: File read

Format

```
RFilename[,addr[,n]]
   Filename:  Read file name
       addr:  Write address (hexadecimal representation)
          n:  Subbank number
```

Function

The R command reads the specified file (including the specified drive) and writes the read data in the specified address in the debug bank.

Explanation

This command loads the target program for debug operations.

(i) If the file type is HEX, the file assumed as in Intel HEX format file.

(ii) If one of the following files is specified, the write address can be omitted or is ignored even if specified:
   1 Intel HEX-format file
   2 Load and execute mode program

(iii) If the subbank number is omitted, the first subbank (with smaller number) in the area to which the target program has been loaded is used. See "Appendix III RAM MEMORY MAP" for details.

Example

```
-RA:TEST.COM
 HEAD END
 xxxx xxxx xx     (Start address, end address, and subbank number)
-
```

S: Memory set

Format

Saddr[,n]

addr: Modification address

n: Subbank number

Function

The S command displays or modifies the contents of the specified address.

Explanation

The specified address and the 1-byte contents are displayed after the command is input. Enter the new data for modification and press the RETURN key. The modification data must be 1-byte data in hexadecimal or ASCII code representation. If the modification data is in ASCII codes, enclose the data with a pair of single-quotation marks (').

(i) If only the RETURN key is pressed without inputting the modification data, the memory contents are not modified and the next address and its contents are displayed.

(ii) If numerics or codes other than ASCII codes are input, the same address and its contents are displayed again.

(iii) The S command is terminated when a period (.) is entered and the RETURN key is pressed.

(iv) If the subbank number is omitted, the subbank specified during R command execution is used.

Example

```
-S100
 0100  31  2A          (Address, data, and modification data)
 0101  FE    <-'
 0102  4A  .   <-'
-
```

Format

T[n[,S]][/P]

n: Number of execution steps (hexadecimal representation)
S: Subroutine trace skip indication (character S)
/P: P=1. Break address set by P command is effective.
P=0. Break address set by P command is not effective.

Function

The T command executes the instructions as much as the specified steps starting from the address indicated by the current PC and displays the register contents for each step.

Explanation

(i) If the number of execution steps is omitted, 1 is used as the default value. The maximum number of execution steps is FFFF(H).

(ii) If S is specified, tracing in the subroutine is skipped.

(iii) Display operation can be temporarily interrupted by the CTRL + S keys or by the space key. Press the CTRL + S keys or the space key again to restart display operation.

(iv) The T command is terminated when the RETURN key is pressed during display operation.

(v) At the end of execution, the address of the next instruction is displayed.

(vi) The T command is terminated when processing passes through the break point set by a P command as many times as the specified count. However, this operation is not performed if /0 is specified as option P.

(vii) If option P is omitted, /1 (effective) is used as the default value.

* An X command sets PC. A T command can be executed only in a target program. Operation is not guaranteed if a T command is executed outside a target program.

U: Untrace

Format

U[n[,s]][/P]

- n: Number of execution steps (hexadecimal representation)
- s: Subroutine trace skip indication (character S)
- /P: P=1. Break address set by P command is effective.
  P=0. Break address set by P command is not effective.

Function

The U command executes the instructions as much as the specified number of steps starting from the address indicated by the current PC and displays the register contents before and after execution.

Explanation

The U command can be used as a T command but the U command cannot display register contents for each step.

(i) If the number os execution steps is omitted, 1 is used as the default value.

(ii) If S is specified, tracing in the subroutine is skipped.

(iii) The U command is terminated when the RETURN key is pressed during display operation.

(iv) At the end of execution, the address of the next instruction is displayed.

(v) The U command is terminated when processing passes through the break point set by a P command as many times as the specified count. However, this operation is not performed if /0 is specified as option P.

(vii) If option P is omitted, /1 (effective) is used as the default value.

* An X command sets PC. A U command can be executed only in a target program. Operation is not guaranteed if a U command is executed outside a target program.

W: Write

Format

```
Wfilename,addr1,addr2[,n]
   filename:  File name for saving data in disk
      addr1:  Save start address
      addr2:  Save end address
          n:  Subbank number
```

Function

The W command saves the contents of the specified address range from memory to a disk with the specified file name.

Explanation

(i) If a file name is specified and the file type is HEX, data is saved in the Intel HEX format.

(ii) In any other cases, data is saved in binary format.

(iii) An error occurs if addr1 is greater than addr2.

(iv) If the subbank number is omitted, the subbank specified during R command execution is used.

X: Examine

Format

X[n]

n: Register or flag type

Function

The X command displays or modifies the contents of registers and flags.

Explanation

A register or a flag is specified as follows:

n	A	B	D	H	S	P	X	Y	A'	B'	D	H'
Register	A	BC	DE	HL	SP	PC	IX	IY	A'	BC'	DE'	HL'

n	C	E	I	Z	M
Register	Carry	Even parity	Half carry	Zero	Sign

(i) If n is omitted, the contents of all registers and flags are displayed.

(ii) If n is specified, the contents of register or flag corresponding to n is displayed. Input the modification data in hexadecimal representation to modify the contents. A flag is specified by 0 (reset) or 1 (set).

Example

```
-X <┘
 CEIZN
 00010   A =xx  B =xxxx  D =xxxx  H =xxxx  S =xxxx
 00000   A'=xx  B'=xxxx  D'=xxxx  H'=xxxx  P =xxxx
                x=xxxx   y=xxxx
         LD     SP,xxxx  (Mnemonic code in the current PC)

-XA <┘
 A =xx xx              (register contents and set value)

-XZ <┘
 Z=x                   (Flag value)  Input 1 or 0 to modify the value.
```

Z: End

Format

Z

Function

The Z command ends debug operations. When debug operations are ended, both the host computer and EHT-10 go out of the debugger.

(1) A target program cannot use the user BIOS area because WAD allocates the interrupt processing and communication buffers to the user BIOS area.

(2) Since WAD uses 5 x 256 bytes of the user BIOS area, the TPA and RAM disk sizes are decreased by 5 x 256 bytes (38 Kbytes or less must be set as the main-RAM disk size).

(3) A target program cannot use the OVF interrupt processing hook because WAD uses it.

(4) A target command cannot use RST 7 because G commands use it.

(5) A program in ROM-base mode is actually executed in a bank different from the one in ROM because the program is emulated in an extended RAM. Therefore, when a program in ROM-base mode exceeding 32 Kbytes is debugged, the program actually stored in ROM differs from the program used for debug operations.

(6) The following four modes are provided as the cartridge interface modes:

1 Handshake (HS) mode
This mode has a CPU-to-CPU interface used for a device that has CPU at the option side.

2 Input output port (IO) mode
This mode has an interface in the form of 4-bit input port and 4-bit output port.

3 Data bus (DB) mode
In this mode, an option operates as if it were a general I/O device in the view of the main frame. For the cartridge interface, the data bus of the main frame is simply connected to the data bus of the cartridge.

4 Out port (OT) mode
This mode has an interface in the form of 8-bit output port.

(Note) Refer to EHT-10/EHT-10/2 System Description for details on these modes.

The interface mode of the development cartridge must also be switched when a target program switches the current cartridge interface mode. For this operation, a routine that switches the cartridge interface mode is provided in the jump table stored in the resident area.

<Routine name>
JRDBGIOX (0FFB4H)

<Function>
If the development cartridge of which the cartridge interface mode is to be switched is mounted, this routine switches the mode of the development cartridge.

<Parameters>
B=09H
A=00H: HS mode
A=01H: IO mode
A=02H: DB mode
A=03H: OT mode

(Notes)
At initiation, the debugger checks the option cartridge and sets the appropriate mode for both the cartridge interface and the development cartridge interface.

However, since a target program does not operates under the debugger after it is created, the target program must set a mode appropriate to the option cartridge status to use an option cartridge.

(7) 005C to 00FF in the main RAM cannot be modified. If they are modified, the debugger operations are not guaranteed.

(8) If a ROM-based program to be created exceeds one bank (32 Kbytes), the program must be split in the units of 32 Kbytes and stored in two or more banks. If a program is split into two or more banks, addressing and bank switching at PROM expansion are made easier.

(9) ASCII data must be enclosed in a pair of single-quotation marks (') in a program.

5.1 Initiation and Termination

See "Section 4.1 Initiation and Termination" for steps 1 to 7 that are exactly same as steps 1 to 7 of the BASIC-language debugger.

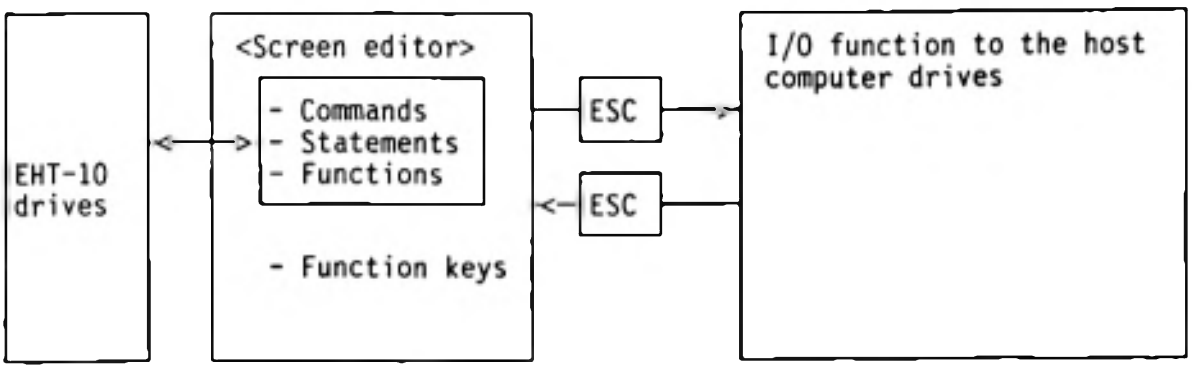
Step 8
Select 3 corresponding to BASIC program in the program mode selection screen of step 7. When 3 is selected, the screen editor screen is displayed.

* The screen size of the screen editor differs depending on the host computer as follows:
  PX-4 and HX-40: 40 columns x 25 lines (virtual screen)
  Other than above: 80 columns x 25 lines (virtual screen for PX-8)

The debugger is terminated by the SYSTEM command. EHT-10 returns back to the DLL screen and the host computer ends the debugger.

The BASIC-language screen editor screen is displayed at step 8 explained in "Section 5.1 Initiation and Termination". This editor enables debugging and creating a BASIC-language program. The editor can accept any instructions such as EHT-10 BASIC commands, statements, and functions. The EHT-10 drives are specified as the file input/output drives in the editor and in the BASIC-language program. To input or output a BASIC-language program from or to one of the host computer drive, press the ESC key and take the action indicated in the screen.

The following figure shows the relationship between the screen editor and the operations performed after the ESC key is pressed:



(1) Screen editor

Refer to the following for using the screen editor:
- BASIC commands, statements, and functions
  EHT-10/EHT-10/2 BASIC Reference Manual
- Function keys of the screen editor
  "Section 5.3 Screen Editor for BASIC-Language Debugger"

(2) I/O function to the host computer drives

a) Press the ESC key when the screen editor is in command input wait status.

b) In the following screen, select 1 to save the program being edited to a host computer drive or select 2 to load a program from a host computer drive.

```
Select command
<Press ESC to return to Screen editor>
  1 -- SAVE BASIC PROGRAM TO HOST
  2 -- LOAD BASIC PROGRAM FROM HOST
```

The screen changes to the one shown in c) when 1 is selected or the screen changes to the one shown in d) when 2 is selected.

c) Specify the file format in the following screen for save operation:

```
Select BASIC program Save-type
  1 -- ASCII Save
  2 -- BINARY Save
Select (1/2) ▌
```

d) Input the name of the file to be saved or loaded in the screen shown below. Save or load operation is executed after a file name is input.

```
Enter file name ▌
```

* Up to eight characters can be specified as a file name when CP/M computer is used as the host computer and up to 62 characters including path name can be specified as a file name when an MS-DOS computer is used as the host computer. A qualifier must not be included in a file name (it will be assumed as BAS even if it is input).

e) Processing automatically returns back to the screen editor (in the status held before the ESC key is pressed) when load or save operation ends.

- Processing returns back to the screen editor when the ESC key is pressed during operations a) to d). However, this key cannot be used during file save or load operation.

- Processing returns back to a) when the CTRL + C keys are pressed during operations a) to d). However, these keys cannot be used druign file save or load operation.

5.3 Screen Editor for BASIC - Language Debugger

The host computer supports the screen editor so that a BASIC-language program can be created or modified efficiently. As a general BASIC interpreter, the screen editor is supported in command input wait status. The screen editor provides the following functions:

CTRL + K This function moves the cursor to the top of screen.

CTRL + L This function deletes all the text in the screen and moves the cursor to the top of screen.

CTRL + R This function makes processing enter insert mode so that a character can be inserted at the cursor position. To make processing come out of this mode, press the CTRL + R keys again, cursor move key, or the return key.

DEL This function deletes the character at the cursor position and shifts the characters placed after the cursor position.

^
|,|,->,<- These keys move the cursor in the direction of the arrows.
v

CTRL + H or BS
These keys delete the character placed before the cursor position.

CTRL + I or TAB
These keys move the cursor to a tab position and fill the original cursor position to the tab position with blank codes.

CTRL + A This function moves to the beginning of the current line.

CTRL + B This function returns the cursor back to the beginning of the previous word. A word means a character string delimited by blanks.

CTRL + E This function deletes character strings placed from the cursor position to the end of the line.

CTRL + F This function moves the cursor to the beginning of the next word.

CTRL + X This function moves the cursor to the end of the line.

CTRL + Z This function deletes data beginning from the cursor position to the end of the text in the screen.

CTRL + C This function makes modifications in the line ineffective and moves the cursor to the beginning of the next line.

- Press the CTRL + C keys to terminate BASIC program execution.

- Press the CTRL + S keys to temporarily interrupt BASIC program execution. Press the CTRL + S keys again to restart processing.

(1) Processing does not enter command input wait status when an EHT-10 BASIC operates independently without the development tool.

(2) The following commands are effective only for the screen editor at the host computer:

AUTO, CONT, DELETE, EDIT, END, FILES, LIST, LLIST,MERGE, NEW, RENUM, SYSTEM, TROFF, TRON

(3) If an error occurs, an error message is displayed and control returns to the system.

(4) Character W is not required before the specified drive name as in the case of machine-language program debug operations, when the editor or a program accesses a drive.

* Refer to EHT-10/EHT-10/2 BASIC Reference Manual for details on each BASIC command.

Chapter 6 PROM FORMAT UTILITY

6.1 Overview

The PROM format utility converts files for EHT-10 application ROM.

This utility reads files from the host computer, converts the file image to the ROM format, and create a directory of each file. The results can be written in a disk in Intel HEX format or output from the RS-232C interface.

A file that has qualifier HEX after the file name is assumed as a file written in Intel HEX format. When the program is executed, this file is converted to a COM file and the qualifier is modified to COM and stored in the directory. Any other file name qualifiers are not modified.

PROM is in one of the following sizes:

(i) 256 Kbits
(ii) 512 Kbits
(iii) 1024 Kbits

The ROM format is format P or format M depending on the program execution mode. See "Appendix I ROM FORMAT" for details.

The PROM format utility is stored as WPROMFRM.EXE in the development utilities. If you use CP/M machine, the file name of the utility is WPROMFRM.COM.

To execute this utility, set the development utilities in a host computer drive, switch the login drive to the host computer drive, and then execute the following:

A > WPROMFRM <↵
(when the development utilities are set in drive A)

(1) The initial screen shown below is displayed. Select 1 for format M or select 2 for format P.

```
PROM format and write program version x.x (c) by EPSON
This program converts program and data files into a hex file with the ROM
Capsule format which may be written into PROM.

Press ESC to restart WPROMFRM, STOP(or BREAK) to exit from WPROMFRM.



      Formats of PROM

        1 = M format
        2 = P format

      Select a format
```

(2) In the screen shown below, select 1 to store the file converted in format M or P into a floppy disk or select 2 to send the converted file directly from the RS-232C interface to the PROM writer. To select 2, the parameters specifying the transfer rate, character length, parity bit, etc., must be set in advance by CONFIG so that the transfer conditions correspond to that of the PROM writer.

```
      Output devices

        1 = FDD
        2 = RS-232C

      Select a device
```

(3) Set the size of ROM.

```
PROM types

  1 = 256K bit
  2 = 512K bit
  3 =  1 M bit

Select PROM type
```

(4) Input the system name, ROM name, version number, and date structuring the ROM header. Up to three characters can be input as a system name, up to 14 characters can be input as a ROM name, a version number is a 2-digit number, and date is a 6-digit number indicating month, day, and year.

```
Enter system name (3 characters max.) xxx

Enter ROM name (14 characters max.)   xxxxxxxxxxxxxx

Enter version number (nn)             xx

Enter date (mmddyy)                   xxxxxx
```

(5) Input the file names of files to be converted.

```
Enter file names (31 directories max.), RETURN to end.

  1 TEST1.COM    200 records    25 K bytes     total    25 K bytes
  2 TEST2.COM     40 records     5 K bytes     total    30 K bytes
  3
```

When only the return key is pressed, file name input operation is ended and the ROM header and file directory are displayed. Confirm these information. Press the return key if the information is correct. Other wise, press the ESC key to input file names again.

(i) The following error message is displayed if a file with an input file name does not exist:

*** File not found ***

(ii) The following error message is displayed if an input HEX file is not in Intel HEX format:

*** File not INTEL HEX FORMAT ***

(iii) The following error message is displayed if the input HEX files are not stored consecutively in the address order:

```
*** Address data not sequential ***
```

(iv) The following error message is displayed if the PROM size is insufficient:

```
Enter file names (31 directories max.), RETURN to end.

  1 TEST1.COM    200 records    25 K bytes     total    25 K bytes
  2 TEST2.COM     40 records     5 K bytes     total    30 K bytes
  3 TEST3.COM    *** Out of memory ***

A : Abort WPROMFRM   D : Delete last file   I : Ignore
```

File information is displayed when file name input operation normally ends.

```
Header information is below.

System name           : xxx
ROM name              : xxxxxxxxxxxxx
Version No.           : xx
Date                  : xx/xx/xx


Directory information is below.

TEST1.COM    TEST2.COM


Directory size        : 3 entries and 128 bytes
Files size            : 2 files and 30 K bytes

Press RETURN to proceed
```

(6) (i) If FDD is selected in operation (2), input the file names of files to be saved in a disk as follows:

```
Enter output name xxxxxxxx.xxx
```

The following error message is displayed if one of the input file names already exist:

```
*** File already exist *** Overwrite (Y/N)?
```

(ii) If RS-232C is selected in operation (2), the message shown below is displayed. Press the return key to start send operation.

```
Ready to output ROM data on RS-232C ?
```

During execution, the file name of the current source file being converted is displayed.

The following message is displayed if the disk becomes full:

*** Disk Full ***

(7) When saving files in a disk or sending files to the RS-232C ends, the message shown below is displayed. Press the ESC key to return back to the beginning of program execution and then press the STOP key to end program execution.

```
Done
Press ESC to restart WPROMFRM, STOP(or BREAK) to exit from WPROMFRM.
```

(Execution time varies depending on the file sizes.)

APPENDIX

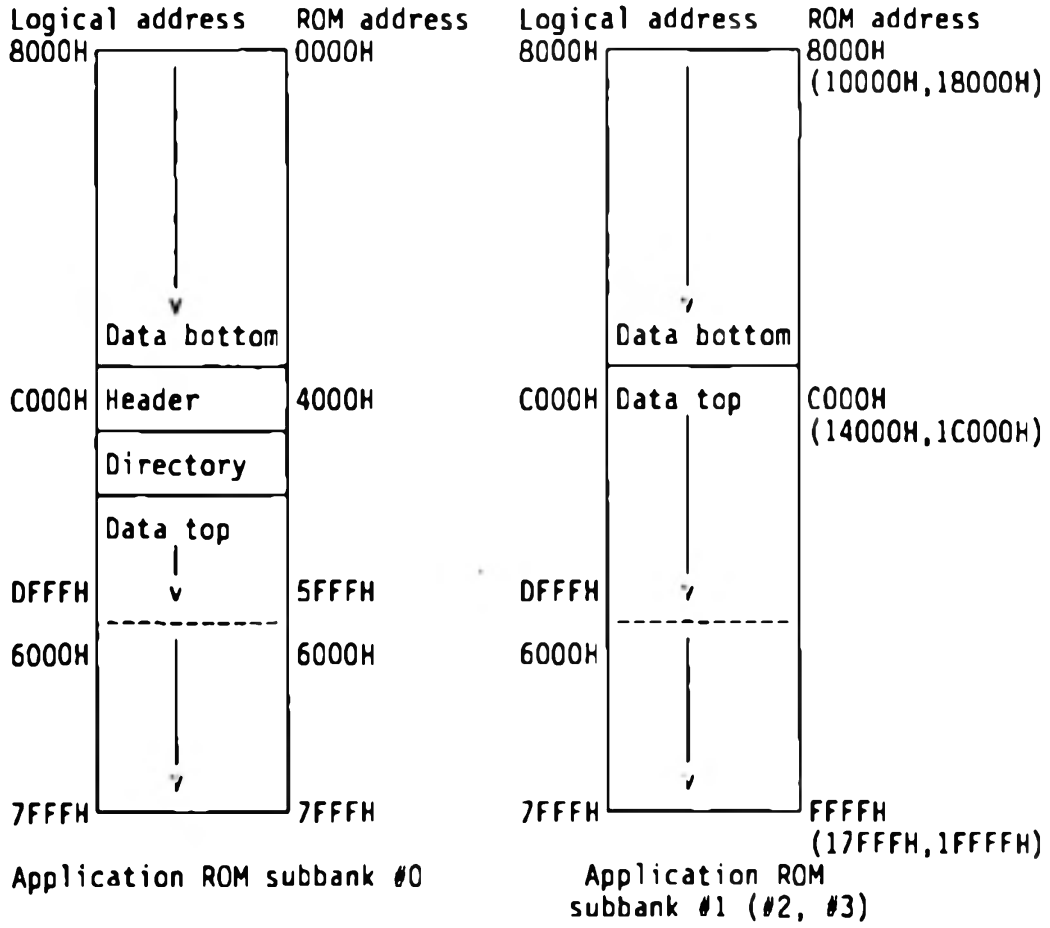
1. ROM FORMAT

(1) The following two formats are provided for ROMs mounted to EHT-10 ROM plug :

1 M Format
A program created in format M is loaded to CP/M TPA for execution. Therefore, the program is allocated starting from address 100H (program in load and execute mode).

2 P Format
A program created in P format is directly executed in an application ROM (program in ROM-based mode). Because of this, the program start address must be determined by taking into account of the area required for the header and directory area.
To execute a program stored in 512-Kbit or 1-Mbit ROM, addresses must be sufficiently managed and bank switching must be taken into account within the program.
Because of this, note that a program actually executed in ROM differs from a program emulated by the development tool for debug operations.

(2) M Format

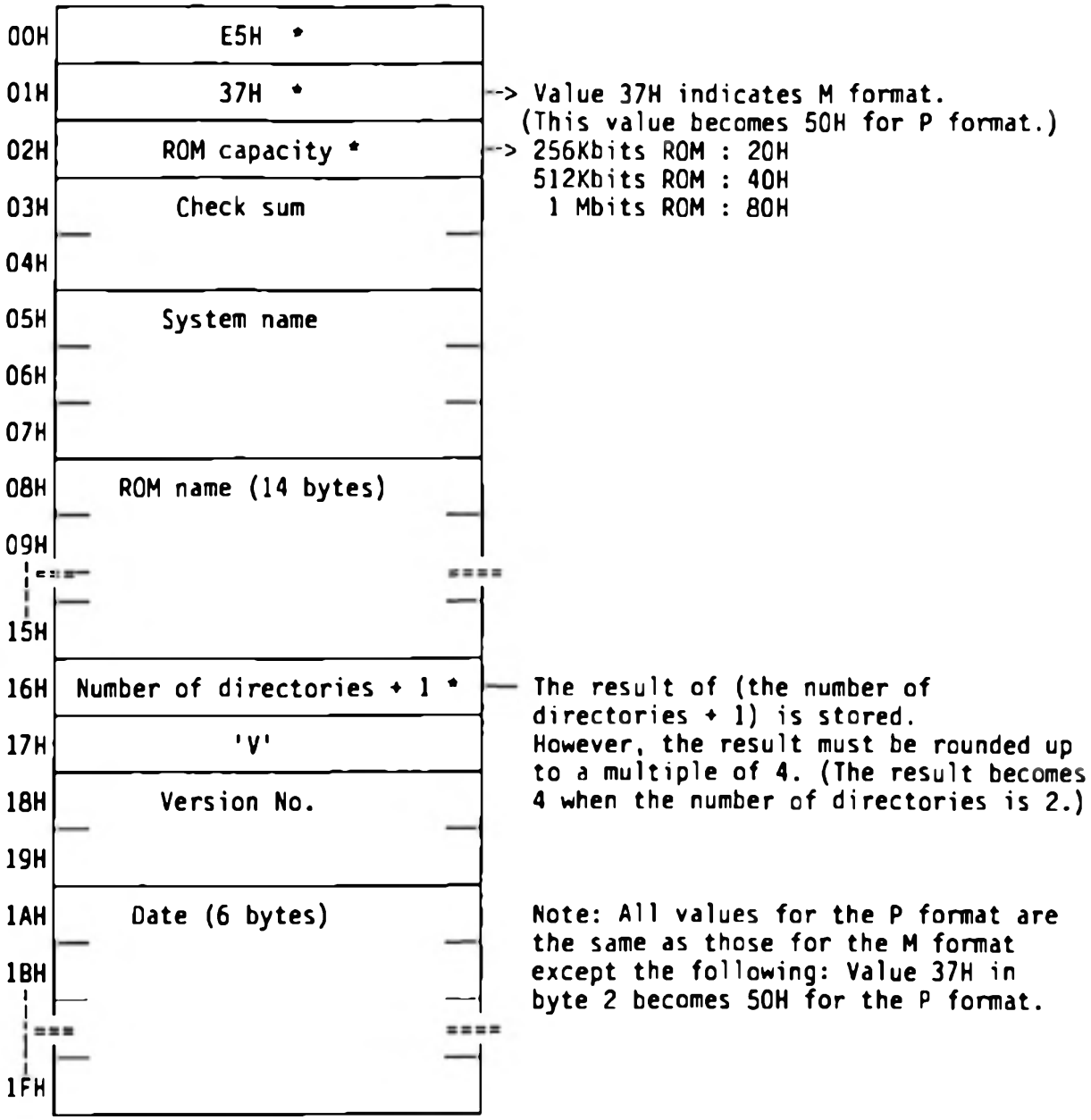


Appendix Fig.-1 Relationship between ROM and Data Addresses

(Note 1) A 256-Kbit ROM has only subbank #0, a 512-Kbit ROM has subbanks #0 and #1 consecutively, and a 1-Mbit ROM has subbanks #0, #1, #2, and #3 consecutively.

1 Header area (32 bytes)

A 32-byte header contains information of ROM format, size, number of directories, etc. Appendix Figure-2 shows the header structure.



The items with an asterisk (*) must be set.

Appendix Fig. -2 Header Structure

2 Directory area

The length of a directory is 32 bytes. Up to 31 directories can be registered. Refer to CP/M Manual for directory structure.

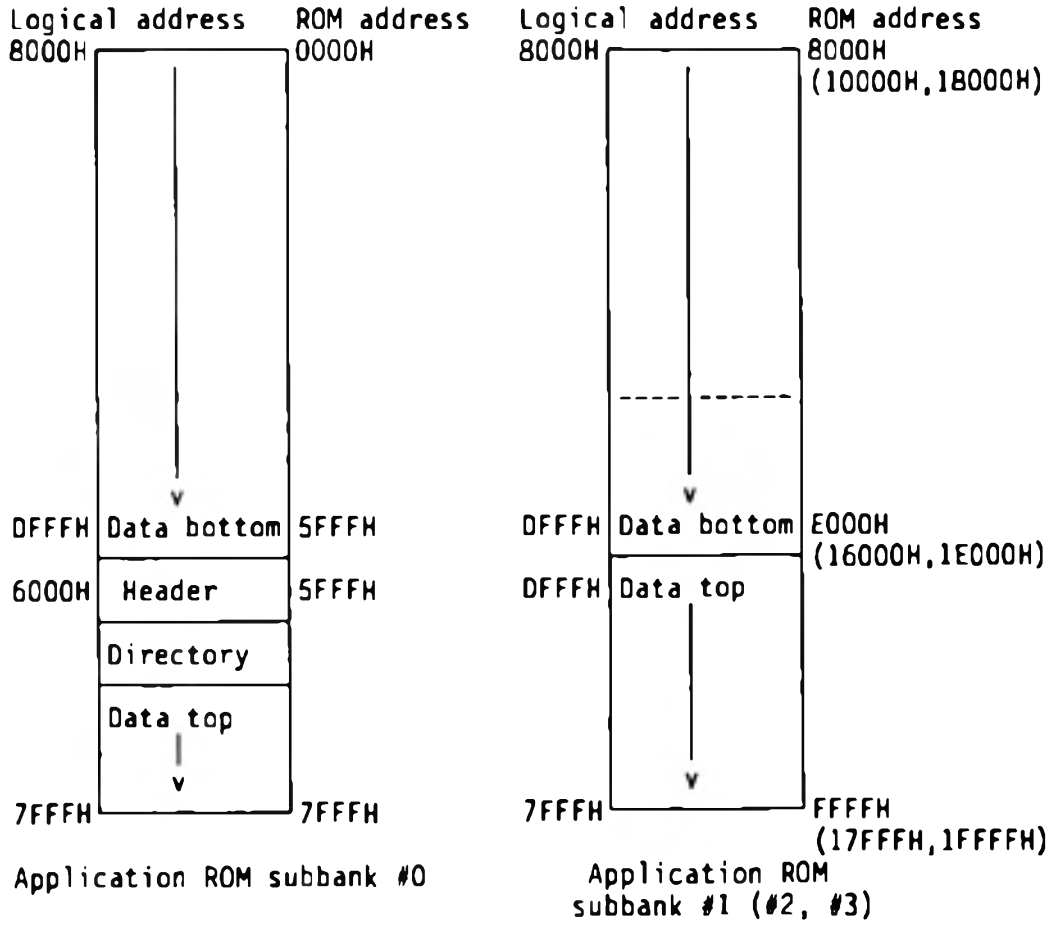
3 Data area

The start address of data area differs depending on the number of directories. The start address is determined as follows:
Where the number of directories is n,

- n is increased by 1,
- the result of addition is rounded up to a multiple (m) of 4,
- and the header start address is increased by 20H x m.

However, regardless of value m, data is started from track 0 sector 8 for BIOS.

(3) P Format



Appendix Fig. -3 Relationship between ROM and Data Addresses

(Note 1) A 256-Kbit ROM has only subbank #0, a 512-Kbit ROM has subbanks #0 and #1 consecutively, and a 1-Mbit ROM has subbanks #0, #1, #2, and #3 consecutively.

1 Header area (32 bytes)

Instead of 37H, 50H is placed in byte 2 to indicate P format . Any other items are same as the header used for M format.

2 Directory area

Same as the directory area used for M format .

3 Data area

The start address of data area is determined in the same way as M format.

For a program in ROM-based mode, the following 5-byte data must be added at the beginning of the application program:

DDH, DEH, 00H, 00H, 00H
(DDH and DEH are the ID indicating that the program is in ROM-based mode.)

* ID : ROM-based program identifier (DDH and DEH)

An application program is initiated from immediately after the above 5 bytes.

(Note 1) The above 5-byte data must not be added for a program other than programs in ROM-based mode.

(Note 2) Programs in format P are in load and execute mode (without ID) or in ROM-based mode (with ID). One ROM can store programs in both modes.

<Calculating the sizes of header and directory area>

Suppose that there are the following three programs:

File name	Number of records	File size	Required number of directory entries
FILE1.COM	30	4 Kbytes	1
FILE2.COM	150	19 Kbytes	2
FILE3.COM	50	7 Kbytes	1

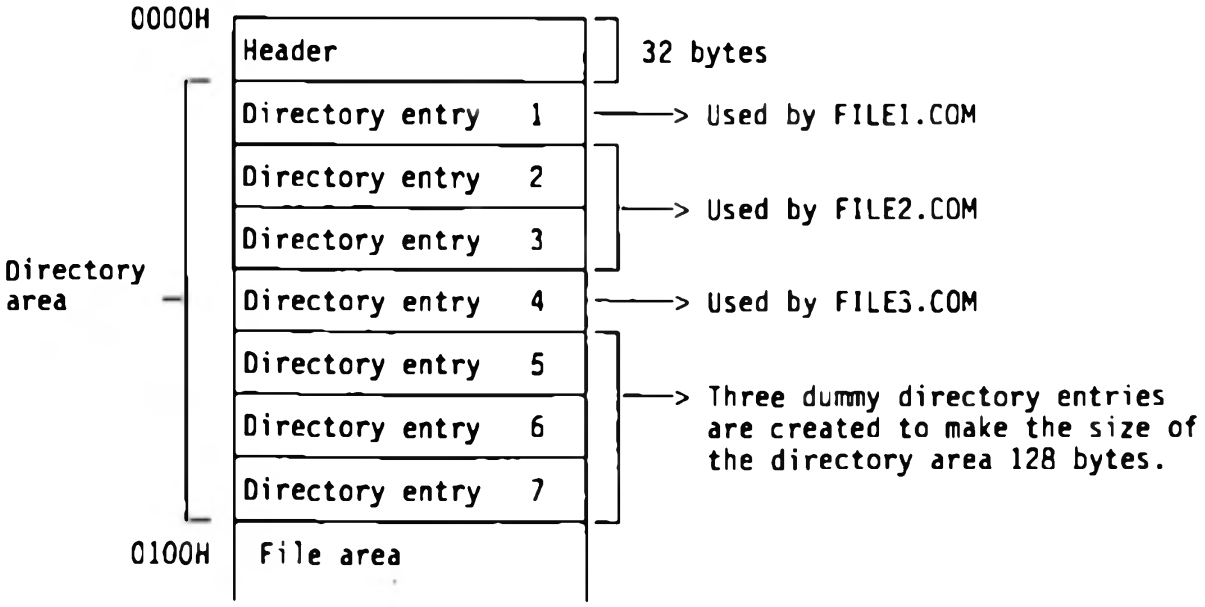
Number of records: 1 record = 128 bytes
((Program size - 1) / 128) + 1

File size: The size of a file to be stored in the ROM.
((Number of records - 1) / 8) + 1

Required number of directory entries:
Number of directory entries required by the file.
((File size - 1) / 16) + 1

One directory is used to manage a file of which the size is up to 16-Kbyte. Any file of which the size is lager than 16 Kbytes is managed using directory entries. The size of one directory is 32 bytes and the directory areas are created in the units of 128 bytes (a header is assumed as one directory).

In other words, the total size of the header and directory area for the three programs is 256 bytes.



Detailed directory area

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E5H	37H	20H	?	?	?	?	?	"S"	"A"	"M"	"P"	"L"	"E"	_	"R"
"O"	"M"	_	_	_	_	08H	"V"	"1"	"0"	"0"	"2"	"1"	"7"	"8"	"7"
00H	F	I	L	E	1	_	_	_	C	O	M	00H	00H	00H	1EH
Block 1 to block 4 of the file area in total of 4 Kbytes are used.															
01H	02H	03H	04H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H
00H	F	I	L	E	2	_	_	_	C	O	M	00H	00H	00H	80H
Block 5H to block 14H of the file area in total of 16 Kbytes are used.															
05H	06H	07H	08H	09H	0AH	0BH	0CH	0DH	0EH	0FH	10H	11H	12H	13H	14H
00H	F	I	L	E	2	_	_	_	C	O	M	01H	00H	00H	16H
Block 15H to block 17H of the file area in total of 3 Kbytes are used.															
15H	16H	17H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H
00H	F	I	L	E	3	_	_	_	C	O	M	00H	00H	00H	32H
Block 18H to block 1EH of the file area in total of 7 Kbytes are used.															
18H	19H	1AH	1BH	1CH	1DH	1EH	00H	00H	00H	00H	00H	00H	00H	00H	00H
E5H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H
00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H
E5H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H
00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H
E5H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H
00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H

* _ indicates a blank.

In a ROM, files are placed after the directory area and are managed in the units of 1 Kbytes. Because of this, the program start address of the second or later program is not the address next to the end address of the previous program. The program start address of the second or later program is the logical start address increased by n determined as follows:

n = (Total file size of the first to the previous files) x 400H
+ (directory area size)

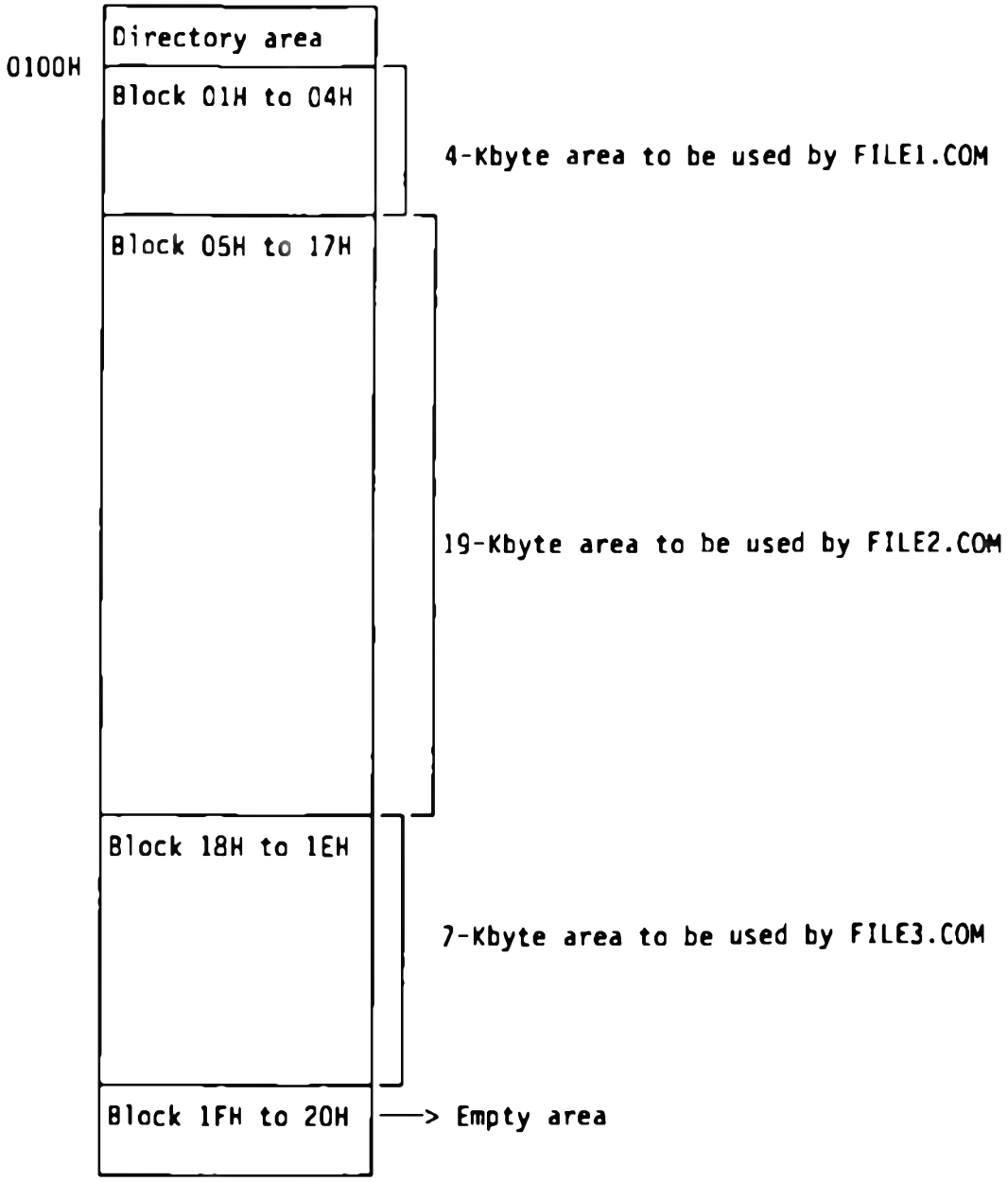
(Note) The directory area size is 100H in this example.

The next page shows the detailed file area.

The start addresses (logical addresses) of the three programs in this example are as follows when 256-Kbit PROM is used:

FILE1.COM 6100H
FILE2.COM 7100H
FILE3.COM B000H

Detailed file area



These block numbers are set in the disk allocation map in the directory entries of each file.

2. LIST OF COMPUTERS THAT CAN BE USED AS THE HOST COMPUTER

Machine name		OS
EPSON	QX-10	CP/M
	PX-4/HX-40	CP/M
	PX-8	CP/M
	QX-11	MS-DOS
	QX-16	CP/M MS-DOS
	EPSON PC series	MS-DOS
IBM	PC/PC-XT	PC-DOS (MS-DOS)
	PC AT	PC-DOS (MS-DOS)
	5550	MS-DOS

3. RAM MEMORY MAP

This section shows mapping of the main and extended RAMs used during debug operations. An address enclosed in a pair of parentheses ( ) is an EHT-10/2 or EHT-10/28 address.

<Load and execute mode>