MF568-02

CMOS 4-BIT SINGLE CHIP MICROCOMPUTER

E0C6262 DEVELOPMENT TOOL MANUAL

SEIKO EPSON CORPORATION

Published in English February, 1991
Printed in Japan

Seiko Epson Corporation products are sold by description only. Seiko Epson reserves the right to make changes in design, specifications and other information at any time without prior notice. Accordingly, the reader is cautioned to verify that the data and other information in this publication are current before placing orders. Application information is intended solely for general guidance; use of the information for the user's specific application is at the user's risk. Reference to products of other manufacturers is solely for convenience of comparison and does not imply total equivalency of design, performance or otherwise.

This document contains technology relating to strategic products controlled under the Foreign Exchange and Foreign Trade Control Law of Japan. This document or a portion thereof should not be exported without obtaining an export license from the Ministry of International Trade and Industry in accordance with the above law.

Trademarks

IBM, PC/AT, PC/XT, PC-DOS	International Business Machines Corporation
MS-DOS, EDLIN	Microsoft Corporation
NEC, PC-9801 Series	Nippon Electric Co., Ltd.

PREFACE

This manual is individualy described about the development tools such as the following for developing the 4-bit Single Chip Microcomputer E0C6262.

I. E0C6262 Development Tool User's Manual

This manual mainly explains the outline of the development support tool for the E0C6262 and starting procedures through the DEV6262 menu.

II. E0C6262 Cross Assembler Manual *

This manual mainly explains how to operate the ASM6262 Cross Assembler for the E0C6262, and how to generate source files.

III. E0C6262 Function Option Generator Manual

This manual mainly explains how to operate the FOG6262 Function Option Generator for setting the hardware options of the E0C6262 and details the specifications of their options.

IV. EVA6262 Manual

This manual explains the function of the EVA6262 Evaluetion Board, a debugging tool for the E0C6262, and the operation of the EVA6262.

V. E0C6262 ICE Operation Manual *

This manual explains the function of the ICE6200 In-circuit Emulator, a debugging tool for the E0C6262, and the operation of the ICS6262, its ICE control software.

VI. E0C6262 Mask Data Checker Manual

This manual explains how to operate the MDC6262 Mask Data Checker for the E0C6262.

USER'S
ASM6262
FOG6262
EVA6262
ICS6262
MDC6262

* In the "E0C6262 Cross Assembler Manual" and the "E0C6262 ICE Operation Manual", Chapter 2 and subsequent chapters provide information common to all E0C62 Family models, the model name being denoted "XX". Read these manuals, replacing "XX" with "62".

$$62\underline{XX} \rightarrow 62\underline{62}$$

$$C2\underline{XX} \rightarrow C2\underline{62}$$

For details on the E0C6262, refer to the "E0C6262 Thechnical Manual". For such items as development procedure, refer to the "E0C62 Family Technical Guide".

I. E0C6262 Development Tool User's Manual

III. E0C6262 Function Option Generator Manual

IV. EVA6262 Manual

V. E0C6262 ICE Operation Manual

VI. E0C6262 Mask Data Checker Manual

I. E0C6262 Development Tool User's Manual

CONTENTS

CHAPTER 1 OUTLINE OF THE E0C6262 DEVELOPMENT SUPPORT TOOL

1.1 Developmental Environment

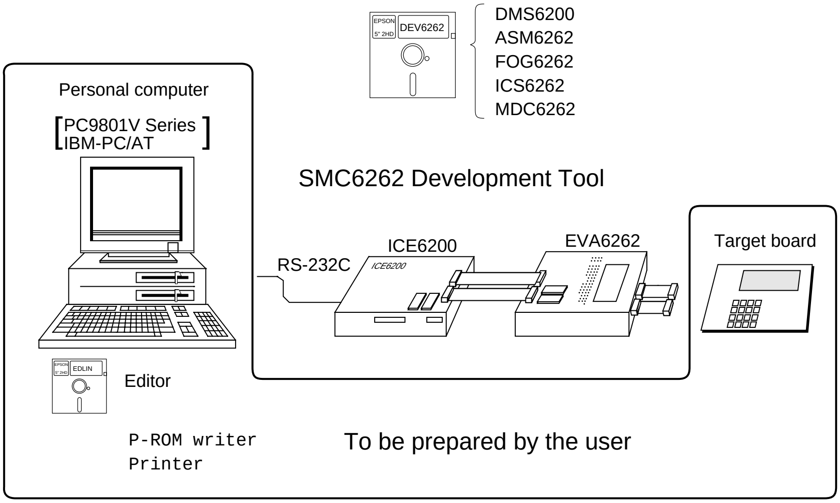
The software product of the E0C6262 development support tool (DEV6262) operates on the following host systems:

- IBM PC-XT/AT (at least PC-DOS Ver. 2.0)
- NEC PC-9801V Series (at least MS-DOS Ver. 3.1)

In order for the MDC6262 to handle numerous files, set the number of files described in the CONFIG.SYS to 10 or more (e.g., FILES = 20).

Since the ICE6200 is connected to the host computer with a RS-232C serial interface, adapter board for asynchronous communication will be required on IBM PC-XT. Moreover, install RS-232C driver with the CONFIG.SYS.

When developing the E0C6262, the above-mentioned host computer, editor, P-ROM writer, printer, etc. must be prepared by the user in addition to the development tool which is normally supported by Seiko Epson.



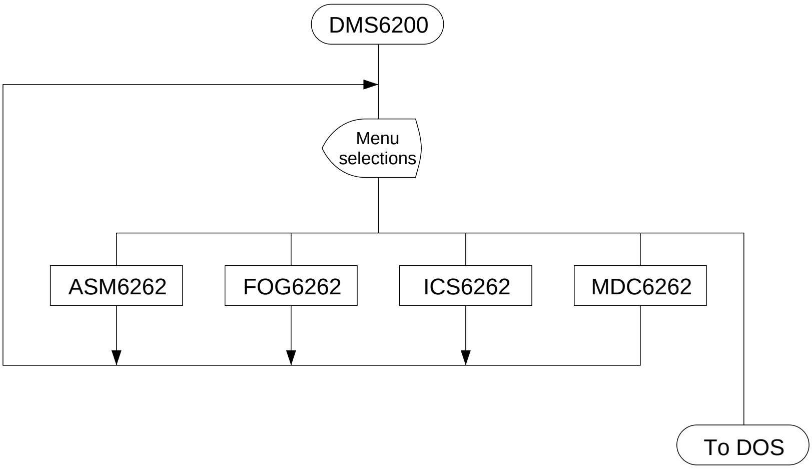
System Configuration

1.2 Development Tool Management System (DMS6200)

Outline: This is a software which selects the DEV6262 software development support tool in menu form and starts it.

Features:

- Simple and easy software development tool starting procedure in menu form
- By copying the external commands such as those of the editor to the execution disk, starting procedure in menu form can be possible



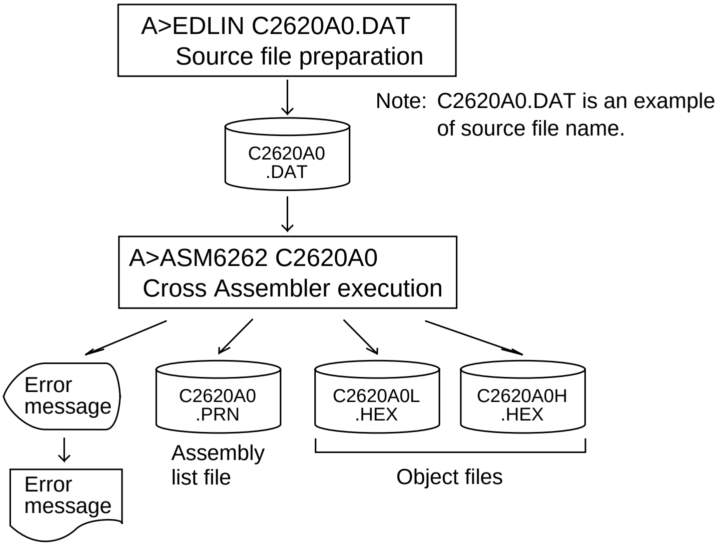
Development Tool Management System (DMS6200) Execution Flow

1.3 Cross Assembler (ASM6262)

Outline: The Cross Assembler ASM6262 will assemble the program source files which have been input by the user's editor and will generate an object file in Intel-Hex format and assembly list file.

Features:

- The macro definition function makes program modularization possible
- The automatic page setting function makes programming unconscious of ROM page structure possible
- Converts the source program to object codes in Intel-Hex format
- Attaches label table and cross-reference table to the assemble list file
- Checks program capacity (ROM capacity) overflows
- Checks undefined codes for errors



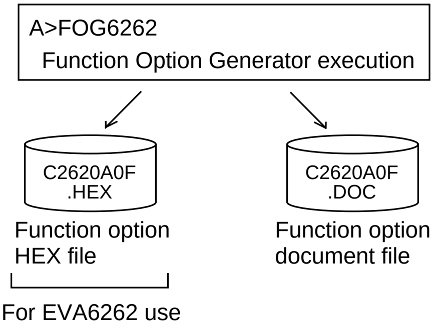
Cross Assembler ASM6262 Execution Flow

1.4 Function Option Generator (FOG6262)

Outline: In the E0C6262, I/O port specifications may be selected with the hardware option and the mask pattern according to the setting is generated on the general-purpose computer.

The Function Option Generator FOG6262 is a software that performs this hardware option selection on the personal computer and creates data files for mask pattern generation.

Features:
- Interactively selects mask option settings
- Creates data in Intel-Hex form for the hardware option ROM to be mounted on the EVA6262

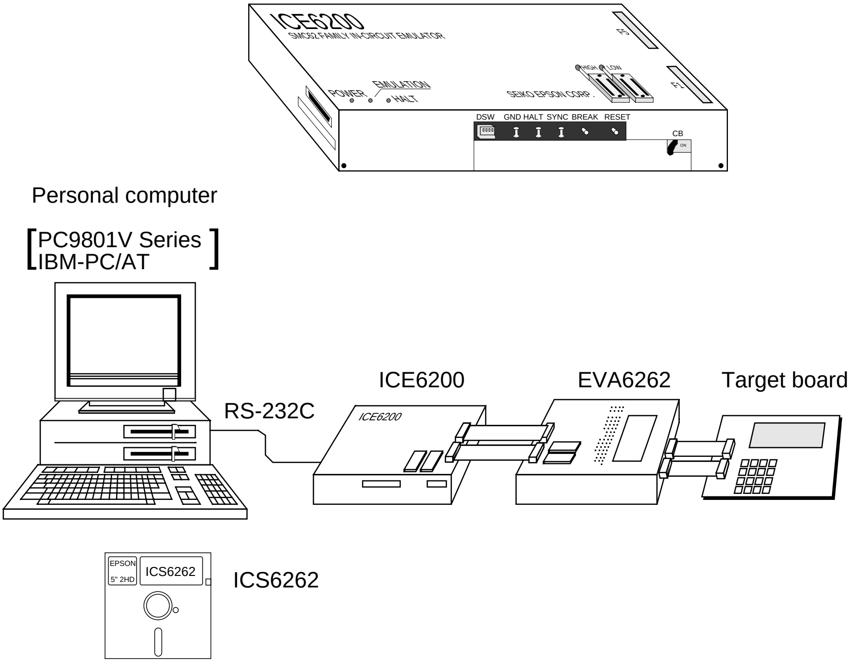


Function Option Generator
FOG6262 Execution Flow

1.5 In-Circuit Emulator (ICE6200) & ICE Control Software (ICS6262)

Outline: The In-circuit Emulator ICE6200 connects the target board produced by the user via the EVA6262 and performs real time target system evaluation and debugging by passing through the RS-232C from the host computer and controlling it. The operation on the host computer side and ICE6200 control is done through the ICE Control Software ICS6262.

Features:
- Establishes high-level debugging environment by utilizing the user's personal computer as host computer
- Has a set of numerous and highly functional emulation commands which provide sophisticated break function, on-the-fly data display, history display, etc.
- Power supply exclusively for ICE6200 is built-in (can supply power to EVA6262)
- Analysis of hardware is possible



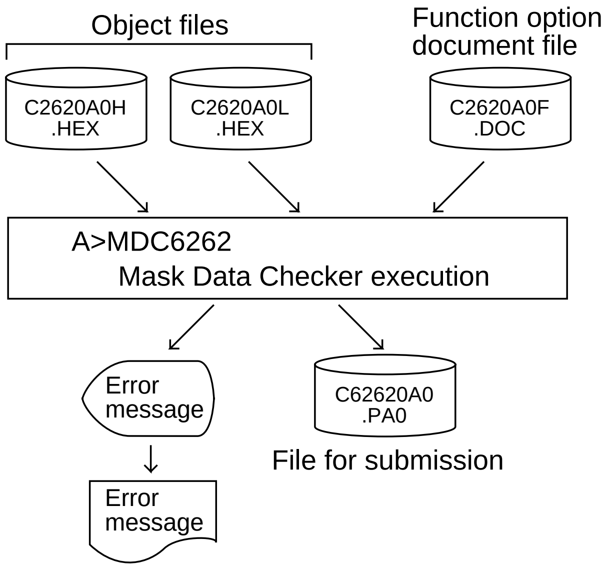
Debugging System
Using ICE6200

1.6 Mask Data Checker (MDC6262)

Outline: This is a software for checking the format of the debugged mask creation data (program data and option data) and creating the file for submission.

Features:

- Checks the mask creation data for submission (program data/option data)
- Performs packing and unpacking of program data and option data



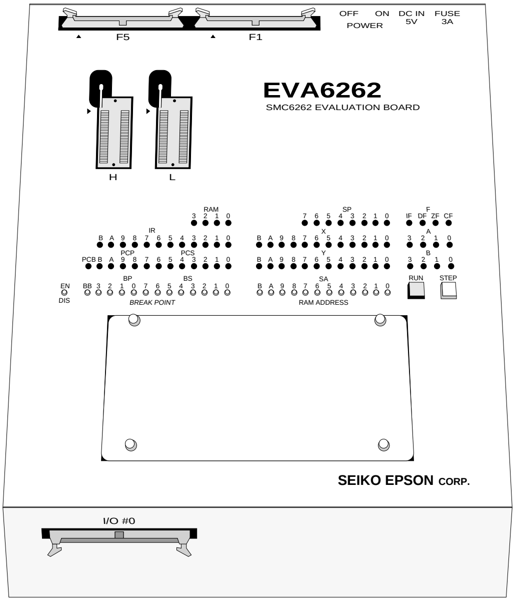
Mask Data Checker MDC6262 Execution Flow

1.7 Evaluation Board (EVA6262)

Outline: The Evaluation Board EVA6262 will implement almost the same functions as the actual CPU by creating ROM from the object files and function option data file generated through ASM6262 and FOG6262 and mounting it.

Features:

- May operate as a stand-alone board by installing a program ROM
- Makes option data setting possible by installing an option ROM
- Has a simple and easy debugging function for PC Break, Step operation and monitor display by LED
- May be connected to ICE6200 through a special cable



EVA6262

CHAPTER 2 CREATION OF DISK FOR DEV6262 EXECUTION

The DEV6262 software product is of two types: the PC-DOS version and the MS-DOS version, supplied in 5-inch 2D and 5-inch 2HD floppy disks, respectively. Note, however, that the DOS is not implemented. Copy the floppy disk and create a disk for execution. Keep the original floppy disk in a safe place as your master copy. When copying to a hard disk, create a sub-directory first and then make the copy to that sub-directory.

• Disk Contents: <PC-DOS Version>

ASM6262.EXE	Cross Assembler execution file
DMS6200.EXE	Development tool Management System execution file
FOG6262.EXE	Function Option Generator execution file
ICS6262B.BAT	ICE Control Software batch file
ICS6262P.PAR	ICE Control Software parameter file
ICS6262W.EXE	ICE Control Software execution file
MDC6262.EXE	Mask Data Checker execution file

<MS-DOS Version>

ASM6262.EXE	Cross Assembler execution file
DMS6200.EXE	Development tool Management System execution file
FOG6262.EXE	Function Option Generator execution file
ICS6262.BAT	ICE Control Software batch file
ICS6262J.EXE	ICE Control Software execution file
ICS6262P.PAR	ICE Control Software parameter file
MDC6262.EXE	Mask Data Checker execution file

CHAPTER 3 DEV6262 STARTING PROCEDURES IN MENU FORM

DMS6200 (Development tool Management System) can start the DEV6262 development support tools in menu form. Since the development support tools each require input files (e.g., source file), first create the input files according to the support tool manuals and then perform the following operations:

(1) The following is entered on the execution disk:

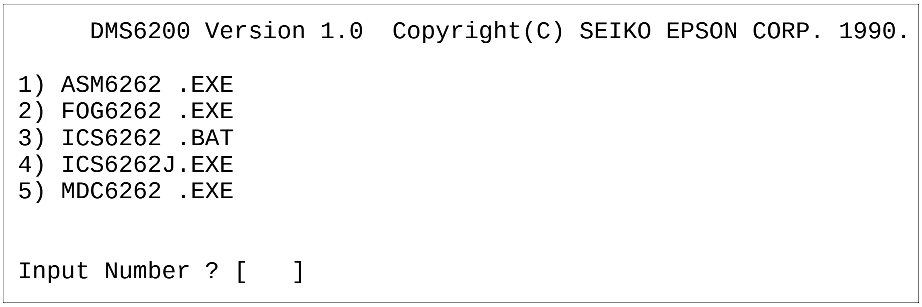
```
DMS6200↵
```

The title is then displayed. To return to DOS at this point, press ^C (CTRL + C).

```
   *** E0C6200 Development tool Management System. --- Ver 1.0 ***

EEEEEEEEEE   PPPPPPPP        SSSSSSS       OOOOOOOO     NNN     NNN
EEEEEEEEEE   PPPPPPPPPP     SSS   SSSS    OOO    OOO    NNNN    NNN
EEE          PPP     PPP   SSS     SSS   OOO      OOO   NNNNN   NNN
EEE          PPP     PPP    SSS          OOO      OOO   NNNNNN  NNN
EEEEEEEEEE   PPPPPPPPPP      SSSSSS      OOO      OOO   NNN NNN NNN
EEEEEEEEEE   PPPPPPPP           SSSS     OOO      OOO   NNN  NNNNNN
EEE          PPP                  SSS    OOO      OOO   NNN   NNNNN
EEE          PPP           SSS     SSS   OOO      OOO   NNN    NNNN
EEEEEEEEEE   PPP           SSSS   SSS     OOO    OOO    NNN     NNN
EEEEEEEEEE   PPP             SSSSSSS       OOOOOOOO     NNN      NN

                (C) Copyright 1990 SEIKO EPSON CORP.


                           STRIKE ANY KEY.
```

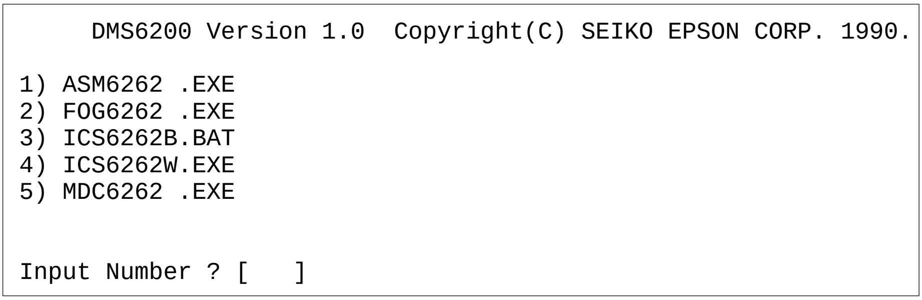
(2) Press any key and the following menu screen will be displayed. A list of all executable files having "EXE", "COM" and "BAT" extensions will appear on this menu screen; if any execution file other than DEV6262 were copied to the disk for execution, it will differ from the displays shown below.
To return to DOS at this point, press the <ESC> key.

<MS-DOS Version>



```
     DMS6200 Version 1.0  Copyright(C) SEIKO EPSON CORP. 1990.

1) ASM6262 .EXE
2) FOG6262 .EXE
3) ICS6262 .BAT
4) ICS6262J.EXE
5) MDC6262 .EXE


Input Number ? [   ]
```

<PC-DOS Version>



```
     DMS6200 Version 1.0  Copyright(C) SEIKO EPSON CORP. 1990.

1) ASM6262 .EXE
2) FOG6262 .EXE
3) ICS6262B.BAT
4) ICS6262W.EXE
5) MDC6262 .EXE


Input Number ? [   ]
```

(3) Input the number of the development support tool you wish to start and then press the RETURN key.

<Conditions for Starting>

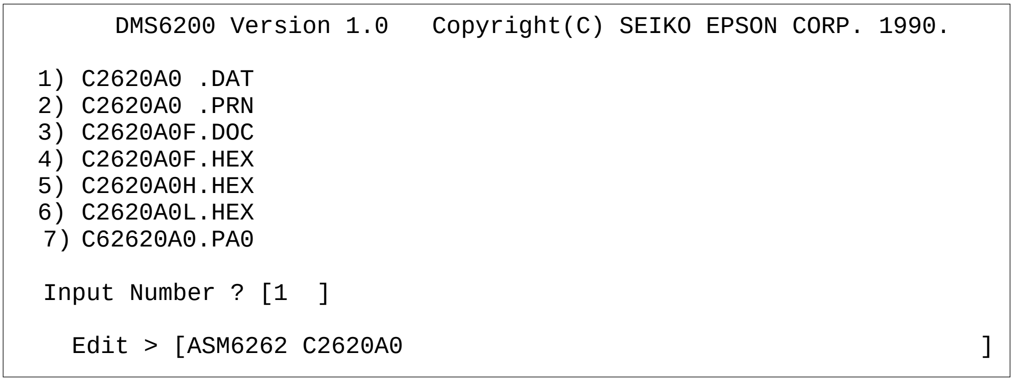
- ICS6262W.EXE, ICS6262J.EXE
  To start ICS6262W.EXE or ICS6262J.EXE, there is need to set the RS-232C beforehand. Set the RS-232C by using the RS-232C driver installed through the CONFIG.SYS and any of the following commands:
  MS-DOS: SPEED command
  PC-DOS: MODE command
  At least 140K bytes are required for the RAM.

- ICS6262.BAT, ICS6262B.BAT
  Since batch processing is programmed in the ICS6262.BAT and ICS6262B.BAT such that it will start with SPEED command or MODE command, ICS6262.BAT must be started after "PATH" of the disk containing SPEED command or MODE command and sub-directory has been specified.
  Likewise, the ICS6262W.EXE requires the installation of RS-232C driver through the CONFIG.SYS.
  At least 140K bytes are required for the RAM.

- MDC6262.EXE
  Because the MDC6262.EXE handles numerous files, set the number of files in the CONFIG.SYS to at least 10 files.

(4) Next, the screen for entering the source file will be displayed. Pressing the <ESC> key here will return the previous screen.

The following sample screen is the screen which will be displayed when ASM6262 is selected.



```
      DMS6200 Version 1.0   Copyright(C) SEIKO EPSON CORP. 1990.

 1) C2620A0 .DAT
 2) C2620A0 .PRN
 3) C2620A0F.DOC
 4) C2620A0F.HEX
 5) C2620A0H.HEX
 6) C2620A0L.HEX
 7) C62620A0.PA0

 Input Number ? [1  ]

   Edit > [ASM6262 C2620A0                                             ]
```

When the source file is selected by number, the edit line enclosed in [ ] will appear; enter the option parameter if necessary. The <BS> key is valid on the edit line. Press the ENTER key when input is completed.

- ASM6262 will start.

When starting, press the RETURN key twice particularly for the following support tools which do not require source files.

Refer to the support manuals regarding operations after starting.

FOG6262
ICS6262
ICS6262W
ICS6262J
MDC6262

(5) When execution of the development support tool is completed, the following message will appear:

```
Input Any Key ...
```

Press any key and the first menu screen will be returned.

APPENDIX List of Software Development Tool Starting Command Formats and Input/Output Files

CROSS ASSEMBLER ASM6262		
Command	ASM6262_[drive-name:] source-file-name [.shp]_[-N] ↵	
Option	.shp option:	Specifies the file I/O drives.
		s Specifies the drive from which the source file is to be input. (A–P, @)
		h Specifies the drive to which the object file is to be output. (A–P, @, Z)
		p Specifies the drive to which the assembly listing file is to be output. (A–P, @, Z)
		@: Current drive, Z: File is not generated
	-N option:	The code (FFH) in the undefined area of program memory is not created.
Input file	C262XXX.DAT	(Source file)
Output files	C262XXXL.HEX	(Object file, low-order)
	C262XXXH.HEX	(Object file, high-order)
	C262XXX.PRN	(Assembly listing file)

FUNCTION OPTION GENERATOR FOG6262		
Command	FOG6262 ↵	
Input files	C262XXXF.DOC	(Function option document file, when modifying)
Output files	C262XXXF.DOC	(Function option document file)
	C262XXXF.HEX	(Function option HEX file)

ICE CONTROL SOFTWARE ICS6262		
Command	ICS6262 ↵	
Input files	C262XXXL.HEX	(Object file, low-order)
	C262XXXH.HEX	(Object file, high-order)
	C262XXXD.HEX	(Data RAM file)
	C262XXXC.HEX	(Control file)
Output files	C262XXXL.HEX	(Object file, low-order)
	C262XXXH.HEX	(Object file, high-order)
	C262XXXD.HEX	(Data RAM file)
	C262XXXC.HEX	(Control file)

MASK DATA CHECKER MDC6262			
Command	MDC6262↵		
Input files	C262XXXL.HEX	(Object file, low-order)	When packing
	C262XXXH.HEX	(Object file, high-order)	
	C262XXXF.DOC	(Function option document file)	
	C6262XXX.PAn	(Packed file)	When unpacking
Output files	C6262XXX.PAn	(Packed file)	When packing
	C262XXXL.PAn	(Object file, low-order)	When unpacking
	C262XXXH.PAn	(Object file, high-order)	
	C262XXXF.PAn	(Function option document file)	

_ indicates a blank.

↵ indicates the RETURN key.

A parameter enclosed by [ ] can be omitted.

II. E0C6262 Cross Assembler Manual

Chapter 2 and subsequent chapters provide information common to all E0C62 Family models, the model name being denoted "XX". Read this manual, replacing "XX" with "62".

$$62\underline{XX} \rightarrow 62\underline{62}$$

$$C2\underline{XX} \rightarrow C2\underline{62}$$

CONTENTS

CHAPTER 1 E0C6262 RESTRICTIONS

Note the following when generating a program by the E0C6262:

1.1 ROM Area

The capacity of the E0C6262 ROM is 2,048 steps (0000H to 07FFH, 12 bits/step). The memory configuration is as follows.

Bank: Only bank 0
Page: 8 pages (00H to 07H), each 256 steps

Therefore, the specification range of the memory setting pseudo-instructions and PSET instruction is restricted as follows:

	Significant specification range
ORG pseudo-instruction:	0000H to 07FFH
PAGE pseudo-instruction:	00H to 07H
BANK pseudo-instruction:	Only 0H
PSET instruction:	00H to 07H

ASM6262

1.2 RAM Area

The capacity of the E0C6262 RAM is 161 words (000H to 07FH, 0D0H to 0DAH, 0E0H to 0EAH and 0F0H to 0FAH, 4 bits / word).
Memory access is invalid when the unused area of the index register is specified.

Example: `LD X,080H` 80H is loaded into the IX register, but an unused area has been specified so that the memory accessible with the IX register (MX) is invalid.

`LD Y,0EBH` EBH is loaded into the IY register, but an unused area has been specified so that the memory accessible with the IY register (MY) is invalid.

1.3 Undefined Code

The following instructions have not been defined in the E0C6262 instruction sets.

```
SLP
PUSH   XP       PUSH   YP
POP    XP       POP    YP
LD     XP,r     LD     YP,r
LD     r,XP     LD     r,YP
```

CHAPTER 2 INTRODUCTION

2.1 Outline of ASM62XX

The ASM62XX cross assembler (the ASM62XX in this manual) is an assembler program for generating the machine code used by the E0C62XX and E0C62*XX 4-bit, single-chip microcomputers. It can be used under MS-DOS or PC-DOS. Two types of ASM62XX system disk are supplied: a 5.25", high-density, double-sided, one for the NEC PC-9801V Series, and a 5.25", double-sided, one for the IBM PC/XT and PC/AT. The basic system configurations are as follows:

– **PC-9801V Series**

Computer:	NEC PC-9801V Series
Disk drive:	5.25", high-density, double-sided, floppy disk drive × 1 or more
Operating system:	MS-DOS 3.1 or later
Printer:	For printing source listings, assembly listings, and error messages

– **IBM PC/XT or PC/AT**

Computer:	IBM PC/XT or PC/AT
Disk drive:	5.25", double-sided, floppy disk drive × 1 or more
Operating system:	PC-DOS (MS-DOS) 2.1 or later
Printer:	For printing source listings, assembly listings, and error messages

The program name of the assembler is ASM62XX.EXE.

Figure 2.1.1 shows the ASM62XX execution flow.

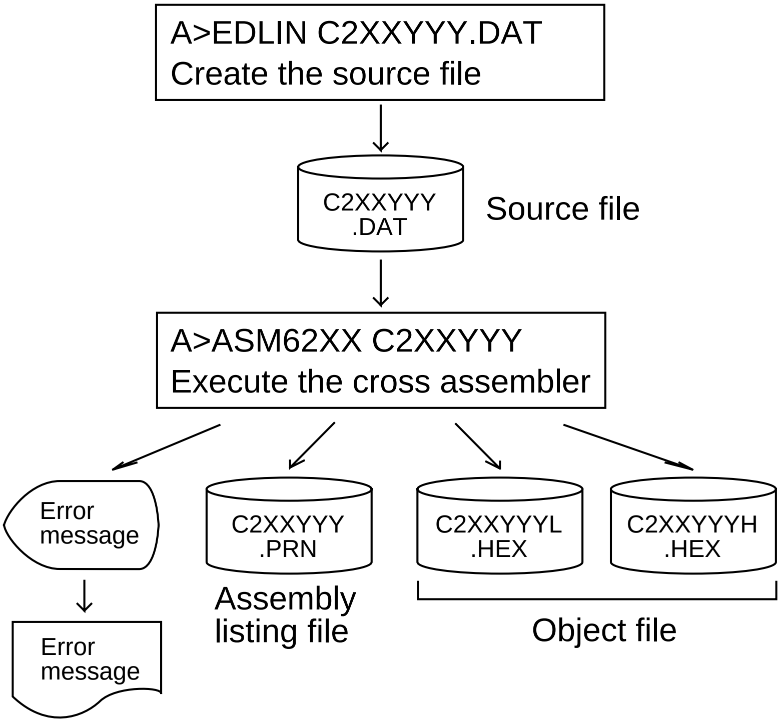


Fig. 2.1.1

ASM62XX Execution Flow

2.2 ASM62XX Input/Output Files

ASM62XX reads a source file, assembles it, and outputs object files and an assembly listing file.

– **Source file (C2XXYYY.DAT)**
  This is a source program file produced using an editor such as EDLIN. The file name format is C2XXYYY, and the file name must not exceed seven characters in length. Character string YYY should be determined by referencing the device name specified by Seiko Epson. The file extension must be added ".DAT".

– **Object file (C2XXYYYH.HEX, C2XXYYYL.HEX)**
  This is an assembled program file in Intel hex format. Because the machine code of the E0C62XX and E0C62*XX is 12-bit, the high-order bytes (bits 9 to 12 suffixed by high-order bits 0000B) are output to file C2XXYYYH.HEX, and the low-order bytes (bits 8 to 1) are output to file C2XXYYYL.HEX.

– **Assembly listing file (C2XXYYY.PRN)**
  This is a program listing file generated by adding an operation codes and error messages (if any errors have occurred) to respective source program statements. A cross-reference table is generated at the end of the file, depending on the label table and options. The file name is C2XXYYY.PRN.

See the Appendix for the contents of each file.

CHAPTER 3 ASM62XX OPERATION PROCEDURE

This section explains how to operate ASM62XX.

3.1 Starting ASM62XX

When starting ASM62XX, enter the following at DOS command level (when a prompt such as A> is being displayed):

```
ASM62XX _ [drive-name:] source-file-name [.shp] _ [-N]↵
```

_ indicates a blank.

A parameter enclosed by [ ] can be omitted.

↵ indicates the return (enter) key.

Drive name

If the source file is not on the same disk as ASM62XX.EXE, specify a disk drive mounted the floppy disk storing the source file before input the source file name. If the source file is on the same disk as ASM62XX.EXE, it does not need to specify the disk drive.

Source file name

This is the name of the source file to be entered for ASM62XX. The sourcefile name must not exceed seven characters in length. File extension .DAT must not be entered.

.shp Characters s, h, and p are options for specifying the file I/O drives, and can be omitted.

s: Specifies the drive from which the source file is to be input. A charac-ter from A to P can be specified. If @ is specified, the source file in the current drive (directory) is input. Even if a drive name is prefixed to the source file name, this option is effective.

h: Specifies the drive to which the object file (HEX) is to be output. A character from A to P can be specified. If @ is specified, the object file is output to the current drive (directory). If Z is specified, only assembly is executed; the object file is not generated.

p: Specifies the drive to which the assembly listing file is to be output. A character from A to P can be specified. If @ is specified, the object file is output to the current drive (directory). If X is specified, a listing containing error messages is output to the console. If Z is specified, the assembly listing file is not generated.

Characters s, h, p must all be specified; only one or two of them is not sufficient.

-N option The code (FFH) in the undefined area of program memory is not created.

Note *The program data to be provided does not use the "-N" option. The FFH data should be inserted into the undefined program area.*

Example 1: Basic assembly example

```
A>ASM62XX C2XXYYY
```

The source file "C2XXYYY.DAT" is input from drive A, and the object files "C2XXYYYH.HEX" and "C2XXYYYL.HEX" and the assembly listing file "C2XXYYY.PRN" are output to drive A.

```
A>ASM62XX B:C2XXYYY↵
```

The source file "C2XXYYY.DAT" is input from drive B, and the object files "C2XXYYYH.HEX" and "C2XXYYYL.HEX" and the assembly listing file "C2XXYYY.PRN" are output to drive B.

```
A>ASM62XX C2XXYYY.BBZ↵
```

The source file "C2XXYYY.DAT" is input from drive B, and the object files "C2XXYYYH.HEX" and "C2XXYYYL.HEX" are output to drive B. The assembly listing file is not generated.

Example 2: -N option use

```
A>ASM62XX C2XXYYY -N↵
```

No undefined program area is generated in the created object files (C2XXYYYH.HEX, C2XXYYYL.HEX).

```
A>ASM62XX C2XXYYY↵
```

In this case, FFH data is inserted into the undefined program area of the object files.

When ASM62XX is started, the following start-up message is displayed.

Example: When assembling C2XX0A0.DAT

```
A>ASM62XX C2XX0A0
          *** E0C62XX CROSS ASSEMBLER. --- VERSION 2.00 ***

EEEEEEEEEE   PPPPPPPP        SSSSSSS       OOOOOOOO     NNN     NNN
EEEEEEEEEE   PPPPPPPPPP     SSS   SSSS    OOO    OOO    NNNN    NNN
EEE          PPP     PPP   SSS     SSS   OOO      OOO   NNNNN   NNN
EEE          PPP     PPP    SSS          OOO      OOO   NNNNNN  NNN
EEEEEEEEEE   PPPPPPPPPP      SSSSSS      OOO      OOO   NNN NNN NNN
EEEEEEEEEE   PPPPPPPP           SSSS     OOO      OOO   NNN  NNNNNN
EEE          PPP                  SSS    OOO      OOO   NNN   NNNNN
EEE          PPP           SSS     SSS   OOO      OOO   NNN    NNNN
EEEEEEEEEE   PPP           SSSS   SSS     OOO    OOO    NNN     NNN
EEEEEEEEEE   PPP             SSSSSSS       OOOOOOOO     NNN      NN

              (C) COPYRIGHT 1990 SEIKO EPSON CORP.

          SOURCE FILE NAME IS " C2XXYYY.DAT "

          THIS SOFTWARE MAKES NEXT FILES.

             C2XXYYYH.HEX  ...  HIGH BYTE OBJECT FILE.
             C2XXYYYL.HEX  ...  LOW BYTE OBJECT FILE.
             C2XXYYY .PRN  ...  ASSEMBLY LIST FILE.
```

3.2 Selecting Auto-Page-Set Function

After the start-up message, the following message is displayed, prompting the user to select the auto-page-set function.

```
DO YOU NEED AUTO PAGE SET?(Y/N)
```

Press the "Y" key if selecting the auto-page-set function, or the "N" key if not selecting it. At this stage, the user can also return to the DOS command level by entering "CTRL" + "C" key.

– **Auto-page-set function**

When the program branches to another page through a branch instruction such as JP, the branch-destination page must be set using the PSET instruction before executing the branch instruction.
The auto-page-set function automatically inserts this PSET instruction. It checks whether the branch instruction page is the same as the branch-destination one. If the page is different,the function inserts the "PSET" instruction. If the page is the same, the function performs no operation.
Therefore, do not select the auto-page-set function if "PSET" instructions have been correctly included in the source file.

Note *When auto page set is selected, there are restricted items related to source programming. See "Label" in Section 4.3.*

3.3 Generating a Cross-Reference Table

After the auto-page-set function has been selected, the following message is output, prompting the user to select cross-reference table generation.

```
DO YOU NEED CROSS REFERENCE TABLE?(Y/N)
```

Press the "Y" key if generating the cross-reference table, or the "N" key if not generating it. At this stage, the user can also return to DOS command level by entering "CTRL" + "C" key.

Note *If the assembly listing file output destination (p option) is specified as Z (listing not generated) at the start of ASM62XX, the above message is not output and the cross-reference table is not generated.*

– **Cross-reference table**

The cross-reference table lists the symbols and their locations in the source file, and is output at the end of the assembly listing file in the following format:

```
 CROSS REFERENCE TABLE   PAGE X-  1
LABEL1  4#      29        36        ....
LABEL2  15#     40
   :     :       :
```

Symbol   Number of the program statement

(# indicates the number of the statement
at which the symbol was defined)

This table should be referenced during debugging. An error such as duplicate definition of a symbol can be easily detected.

CHAPTER 4 SOURCE FILE FORMAT

The source file contains the source program consisting of E0C62XX/62*XX instructions (mnemonics) and pseudo-instructions, and is produced using an editor such as EDLIN.

Refer to the "E0C6200 Core CPU Manual" and the "E0C62XX Technical Software Manual" for instruction sets.

4.1 Source File Name

A desired file name not exceeding seven characters in length can be assigned to each source file. The format must be as follows:

```
C2XXYYY.DAT
```

"YYY" of the "C2XXYYY.DAT" is an alphanumeric character string of up to three characters, and should be determined by referencing the device name specified by Seiko Epson. The file extension must be ".DAT".

4.2 Statements

Each source program statement must be written using the following format.

Basic format:

<Index>[:] <Instruction> <Expression> <; comment>

Example:

```
ON         EQU        1
           ORG        100H
START:     JP         INIT       ;To init.
 Label     Mnemonic   Operand    Comment
 field     field      field      field
```

A statement consists of four fields: label, mnemonic, operand, and comment. Up to 132 characters can be used for one statement. Fields must be delimited by one or more blanks or tabs.

The label and comment fields are optional. Blank lines consisting only of a carriage return (CR) code are also allowed.

Although each statement and field (excluding the label field) can begin at any desired column. the program becomes easier to understand if the heads of corresponding fields are aligned.

Label field

The label field can contain a label for referencing the memory address, a symbol that defines a constant, or a macro name. This field can be omitted if the statement name is not required. The label field must begin at column 1 and satisfy the following conditions.

- The length must not exceed 14 characters.
- The same name as a mnemonic or register name must not be used.
- The following alphanumeric characters can be used, but the first character must not be a digit:

  A to Z, a to z, 0 to 9, _, ?
- The uppercase and lowercase forms of a letter are not equivalent.
- ??nnnn (n is a digit) cannot be used as a name.

A colon ":" can be used as a delimiter between a label field and the mnemonic field. If a colon is used, neither blanks nor tabs need to be written subsequently.
Statements consisting of only a label field are also allowed.

Mnemonic field

The mnemonic field is used for an instruction mnemonic or a pseudo-instruction.

Operand field

The operand field is used for the operands of the instruction. The form of each operand and the number of operands depend on the kind of instruction. The form of expressions specifying values must be one of the following:

- A numeric constant, a character constant, or a symbol that defines a constant
- A label indicating a memory address
- An operational expression for obtaining the specified value

If the operand consists of two or more expressions, the expressions must be separated by commas
",".

Comment field

The comment field is used for comment data such as program headers and descriptions of processing. The contents of this field do not affect assembly or the object files generated by assembly.

The part of the statement from a semicolon ";" to the CR code at the end of the statement is considered to be the comment field. Statements consisting of only a comment field are also allowed. When a comment spans multiple lines, a semicolon must be written at the beginning of each line.

4.3 Index

ASM62XX allows values to be referenced by their indexes. Refer to "Label field" in Section 4.2, for the restrictions on index descriptions.

Label

A label is an index for referencing a location in the program, and can be used as an operand that specifies a memory address as immediate data in an instruction. For example, a label can be used as the operand of an instruction such as JP by writing the label in the branch-destination statement. The name written in the label field of an EQU or SET instruction is considered to be a symbol, not a label.

Example:

```
          :
          JP   NZ,LABEL1
          :
          :
LABEL1:   LD   A,0
```

A label can be assigned to any statement, but the label assigned to the following pseudo-instructions is ignored:

ORG, BANK, PAGE, SECTION, END, LABEL, ENDM

Note *When selecting the auto-page-set function (see Section 3.2), a statement consisting of only a label must be written immediately before the JP or CALL instructions.*

Example:

```
PGSET:
          JP   LABEL
```

Symbol

A symbol is an index that indicates a numeric or character constant, and must be defined before its value is referenced (usually at the beginning of the program). The defined symbol can be used as the operand that specifies immediate data in an instruction.

Example:

```
ON   EQU  1     (See Section 4.5 for EQU.)
OFF  EQU  0
     :
     LD   A,ON      ; = LD A,1
     :
     LD   A,OFF     ; = LD A,0
     :
```

4.4 Constant and Operational Expression

This section explains the immediate data description formats.

Numeric constant

A numeric constant is processed as a 13-bit value by ASM62XX. If a numeric constant greater than 13 bits is written, bit 13 and subsequent high-order bits are ignored. Note that the number of actual significant bits depends on the operand of each instruction. If the value of a constant is greater than the value that can be accommodated by the actual number of significant digits, an error occurs.

Example:

`ABC  EQU  0FFFFH`	→	ABC is defined as 1FFFH.
`     LD   A,65535`	→	An error occurs because it exceeds the significant digit count (4 bits).

The default radix is decimal. The radix description formats are as follows:

Binary numeral: A numeral suffixed with B, such as 1010B (=10) or 01100100B (=100).

Octal numeral: A numeral suffixed with O or Q, such as 012O (=10) or 144Q (=100).

Decimal numeral: A numeral alone or a numeral suffixed with D, such as 10 or 100D (=100).

Hexadecimal numeral: A numeral suffixed with H, such as 0AH (=10) or 64H (=100).
If the value begins with a letter from A to F, it must be prefixed with 0 to distinguish it from a name.

Character constant

A character constant is one or two ASCII characters enclosed by apostrophes (' '). A single ASCII character is processed as eight-bit data. If two or more ASCII characters are written, only the last two characters are significant as 13-bit data.

Examples:

'A' (=41H), 'BC' (=0243H), 'PQ' (=1051H)

'DEFGH' → 'GH' (=0748H; DEF is ignored.)

The apostrophe itself cannot be processed as a character constant, so it must be written as a numeric constant, such as 27H or 39.

Operator

When specifying a value for an item such as an operand, an operational expression can be written instead of a constant, and its result can be used as the value.

Labels and symbols as well as constants can be used as terms in expressions. These values are processed as 13-bit data (bit 14 and subsequent high-order bits are ignored); the operation result also consists of 13 bits. If the result exceeds the number of significant digits of the instruction operand, an error occurs.

There are three types of operator—arithmetic, logical, and relational—as listed below (a and b represent terms, and _ represents one or more blanks).

– **Arithmetic operators**

There are 11 arithmetic operators including the ones for addition, subtraction, multiplication, division, bit shifting, and bit separation.

+a: Monadic positive
(indicates the subsequent value is positive)

-a: Monadic negative
(indicates the subsequent value is negative)

a+b: Addition (unsigned)

a-b: Subtraction (unsigned)

a*b: Multiplication (unsigned)

a/b: Division (unsigned)

a_MOD_b: Remainder of a/b

a_SHL_b: Shifts a b bits to the left. ←[b7<<<<<<b1]←0
Example: 00000011B SHL 2 → 00001100B

a_SHR_b: Shifts a b bits to the right.0→[b7>>>>>>b0]→
Example: 11000011B SHR 2 → 00110000B

HIGH_a: Separates the high-order eight bits from a (13 bits).
Example: HIGH 1234H → 12H

LOW_a: Separates the low-order eight bits from a (13 bits).
Example: LOW 1234H → 34H

– **Logical operators**

There are four logical operators as listed below. The logical operator returns the result of logical operation on the specified terms.

a_AND_b: Logical product
Example:
00001111B AND 00000011B → 00000011B

a_OR_b: Logical sum
Example:
00001111B OR 11110000B → 11111111B

a_XOR_b: Exclusive logical sum
Example:
00001111B XOR 00000011B → 00001100B

NOT_a: Logical negation
Example:
NOT 00001111B → 11110000B

– **Relational operators**
A logical operator compares two terms; if the relationship between the terms is as the operator specifies, 1FFFH (true) is returned; if not, 0 (false) is returned.

a_EQ_b: True when a is equal to b
a_NE_b: True when a is not equal to b
a_LT_b: True when a is less than b
a_LE_b: True when a is less than or equal to b
a_GT_b: True when a is greater than b
a_GE_b: True when a is greater than or equal to b

Be sure to insert one or more blanks for symbol "_" between terms. All operators must be entered in uppercase letters.

An expression can contain one or more operators and pairs of parentheses. In this case, operators are basically evaluated from left to right. However, an operation stipulated by an operator with higher priority or by parentheses is executed earlier. Every left parenthesis must have a corresponding right parenthesis.

The following table shows the priority of operators.

Operator	Priority
(	Low
OR, XOR	:
AND	
EQ, NE, LT, LE, GT, GE	
+ (addition), - (subtraction)	
*, /, MOD, SHL, SHR	
(	
HIGH, LOW, NOT	:
- (monadic negative), + (monadic positive)	High

Examples: Operational expressions (ABC = 1, BCD = 3)

```
LD   A,BCD*(ABC+1)        ;A-register <- 6

LD   A,ABC LT BCD         ;A-register <- 0FH (1111B)

OR   B,ABC SHL BCD        ;Set bit 3 in B-register
                          ;(=OR B,1000B)

AND  B,ABC SHL BCD XOR 0FH;Reset bit 3 in B-register
                          ;(=AND B,0111B)
```

Location counter

The start address of each instruction code is set in the location counter when a statement is assembled. A label or $ can be used when referencing the location counter value in a program.

– **Location counter**

The location counter consists of 13 bits: one bit for the bank field, four bits for the page counter field, and eight bits for the step counter field.

	Bank	Page counter				Step counter							
Bit	12	11	10	9	8	7	6	5	4	3	2	1	0
Contents	Bank BNK	Page address PCP				Step address PCS							

Example:

```
          Location counter
          (BNK) (PCP) (PCS)
            0     1     02      JP    $+3
```

The location counter indicates the start address of the JP instruction, and the PCS value (02) is assigned to $. Consequently, the statement is assembled as "JP 5", and the program sequence jumps to the location three steps before (PCS=05) when it is executed.

4.5 Pseudo-Instructions

There are four types of pseudo-instruction: data definition, memory setting, assembler control, and macro.
These pseudo-instructions as well as operational expressions can be used to govern assembly, and are not executed in the developed program.

In the subsequent explanations, the items enclosed by < > in the pseudo-instruction format must be written in the statement (do not write the < > characters themselves). Symbol _ represents one or more blanks or tabs. One or more symbols and constants or an operational expression can be used in <expression>. See Section 4.6 for macro functions.

Data definition pseudo-instructions

There are three data definition pseudo-instructions: EQU, SET, and DW. The EQU and SET pseudo-instructions each define a symbol, and the DW pseudo-instruction presets data in program memory.

EQU

(Equate)

<Symbol>_EQU_<Expression>   To define a symbol

The EQU pseudo-instruction defines <symbol> (written in the label field) as having the value of <expression> (written in the operand field).
If a value greater than 13 bits is specified in <expression>, bit 14 and subsequent high-order bits are ignored.
This definition must be made before the symbol is referenced in the program. A U-error occurs if an attempt is made to reference a symbol that has not been defined.
The same symbol cannot be defined more than once. A P-error occurs if an attempt is made to define a symbol that has already been defined.

Examples:

```
ZERO    EQU     30H
ONE     EQU     ZERO+1
ONE     EQU     31H      ← P-error because ONE has been
                           defined more than twice
FOUR    EQU     TWO*2    ← U-error because TWO has not
                           been defined
```

SET

<Symbol>_SET_<Expression>   To define a symbol

Like EQU, the SET pseudo-instruction defines the value of <symbol> as being <expression>. The SET pseudo-instruction allows a symbol to be redefined.

Examples:

```
     BIT  SET  1
           :
     BIT  SET  2           ← Redefinition possible
           :
     BIT  SET  BIT SHL 1 ← Previously-defined items can be
                             referenced.
```

DW

(Define Word)

<Label>_DW_<Expression>   To preset data

The DW pseudo-instruction assigns the value of <expression> (the low-order 12 bits when the value is greater than 12 bits) to the current memory location, indicated by the location counter.

Examples:

```
     Location counter
     (BNK)(PCP)(PCS)
       0    2    0A    TABLE   DW    141H  ; = RETD 'A'
       0    2    0B            DW    142H  ; = RETD 'B'
       0    2    0C            DW    143H  ; = RETD 'C'
                                :
```

<label> can be omitted.

Memory setting pseudo-instructions

The program memory mounted at the E0C62XX/62*XX is divided into 256-step pages. Memory management (including the setting of the program location and page boundaries) during program generation must be controlled by the source program.

The memory setting pseudo-instructions are used to specify memory management. The assembler sets the location counter according to these pseudo-instructions.

If a memory area that has already been used is specified or a statement that exceeds the page is used without specifying that the statement is to exceed the page, the assembler displays an exclamation mark "!", indicating a warning, and ignores all subsequent statements until the next correct statement. This should be taken into account.

When using the auto-page-set function, the space for insertion of the "PSET" pseudo-instruction must be allocated in each page.

ORG

(Origin)

ORG_<Expression> **To set the location counter**

The ORG pseudo-instruction sets the location counter to the value of <expression>.

If the ORG pseudo-instruction is not written at the beginning of the program, the location counter is set to 0 (BNK=0, PCP=0, PCS=0) and assembly is started.

The ORG pseudo-instruction can be used at multiple locations in the program. However, it cannot be used to set the location to a value before the current location. If this is attempted, an exclamation mark "!", indicating a warning, is displayed, and all subsequent statements until the next correct statement are ignored.

A label can be written before the ORG statement, but it cannot be referenced because it is not cataloged in the label table. In this case, write the label in the statement following the ORG pseudo-instruction.

Example:

```
            ORG   0100H   ; BNK=0, PCP=1, PCS=00H
     START   :
```

An R-error occurs if a value is specified exceeding the ROM capacity.

Note The upper limit of program memory depends on the model. (See Chapter 1, "E0C62XX RESTRICTIONS".)

BANK

BANK_<expression> **To set the bank (BNK)**

The BANK pseudo-instruction sets the value of <expression> in the bank (BNK) field, and sets the page counter (PCP) and step counter (PCS) to 00H.
The BANK pseudo-instruction can be written at multiple locations in the program. However, it cannot be used to specify the current bank (excluding the specification in page 00, step 00) or a previous bank. If it is used to specify the current bank or a previous bank, an exclamation mark "!", indicating a warning, is displayed, and all subsequent statements until the next correct statement are ignored.
A label can be written before the BANK statement, but it cannot be referenced because it is not cataloged in the label table. In this case, write the label in the statement after the BANK pseudo-instruction.

PAGE

PAGE_<expression> To set the page counter (PCP)

The PAGE pseudo-instruction sets the value of <expression> in the page counter (PCP) and sets the step counter (PCS) to 00H.

The PAGE pseudo-instruction can be written at multiple locations in the program. However, it cannot be used to specify the current page (excluding the specification in step 00) or a previous page. If it is used to specify the current page or a previous page, an exclamation mark "!", indicating a warning, is displayed, and all subsequent statements until the next correct statement are ignored.

A label can be written before the PAGE statement, but it cannot be referenced because it is not cataloged in the label table. In this case, write the label in the statement after the PAGE pseudo-instruction.

Example:

```
    Location counter
    (BNK)(PCP)(PCS)
      :    :    :          :      :
      0    0   1AH         LD     X,0
      0    0   1BH         LD     Y,0
      :    :    :          :      :
      0    0   F0H         JP     xxx

                           PAGE   2
      0    2   00H  SUB1:  LD     A,MX
      0    2   01H         LD     B,MY
      :    :    :          :      :

                           PAGE   1
      !             SUB2:  LD     A,MX       Ineffective because
      !                    LD     B,MY       a previous page was
                           :      :          specified

                           PAGE   3
      0    3   00H  SUB3:  LD     A,0
      0    3   01H         LD     B,1        Effective
      :    :    :          :      :
```

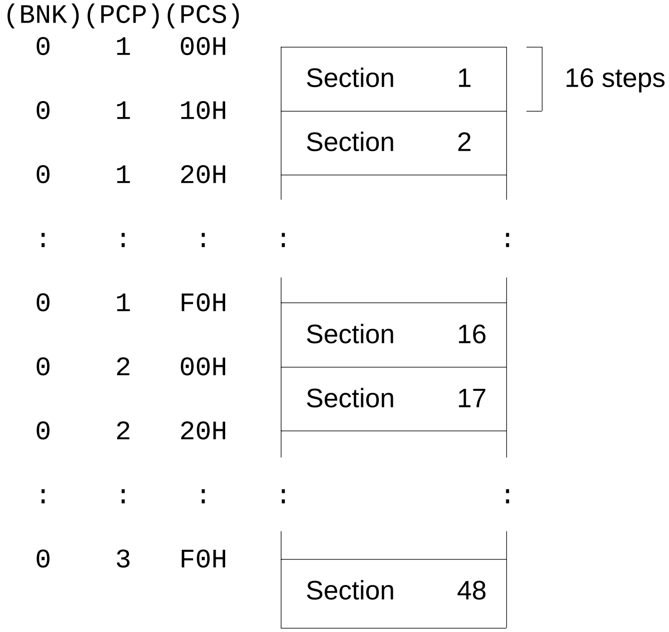
An R-error occurs if a value is specified that exceeds the last page.

Note *The last page depends on the model. (See Chapter 1, "E0C62XX RESTRICTIONS".)*

SECTION

SECTION — **To change the section**

The SECTION pseudo-instruction sets the first address of the subsequent section in the location counter. Sections are 16-step areas starting from the beginning of the program memory.



A SECTION pseudo-instruction written in the last section of the page not only clears the step counter but also updates the page counter, so a new page need not be specified.

A label can be written before the SECTION pseudo-instruction, but it cannot be referenced because it is not cataloged in the label table. In this case, write the label in the statement following the SECTION pseudo-instruction.

Example:

```
    Location counter
    (BNK)(PCP)(PCS)
      :    :    :               :        :
      0    1   09H              JPBA
      0    1   0AH              LD       X,0
      0    1   0BH              LD       Y,0
      0    1   0CH              LD       MX,4

                                SECTION
      0    1   10H      TABLE   LD       A,1
      0    1   11H              ADD      A,1
      :    :    :               :        :
      0    1   FAH              RET

                                SECTION
      0    2   00H      LOOP    SCF
      0    2   01H              ADD      A,MY
      :    :    :               :        :
```

Assembler control pseudo-instructions

END

END **To terminate assembly**

The END statement terminates assembly. All statements following the END statement are ignored. Be sure to write this statement at the end of the program. If it is missing, assembly may not terminate.

A label can be written before the END statement, but it cannot be referenced because it is not cataloged in the label table.

4.6 Macro-Functions

When using the same statement block at multiple locations in a program, the statement block can be called using a name defined beforehand. A statement block that has been so defined is called a macro.
Unlike a subroutine, the statement block is expanded at all locations where it is called, so the programmer should consider the statement block size and frequency of use and determine whether a macro or a subroutine is more appropriate.

Macro-instructions

ASM62XX provides the macro-instructions listed below so that branching between pages is possible without specifying the destination page using the PSET instruction.

Macro-instruction	Mnemonic after expansion	Code											
		11	10	9	8	7	6	5	4	3	2	1	0
`JPM ps`	`PSET p`	1	1	1	0	0	1	0	p4	p3	p2	p1	p0
	`JP s`	0	0	0	0	s7	s6	s5	s4	s3	s2	s1	s0
`JPM C,ps`	`PSET p`	1	1	1	0	0	1	0	p4	p3	p2	p1	p0
	`JP C,s`	0	0	1	0	s7	s6	s5	s4	s3	s2	s1	s0
`JPM NC,ps`	`PSWT p`	1	1	1	0	0	1	0	p4	p3	p2	p1	p0
	`JP NC,s`	0	0	1	1	s7	s6	s5	s4	s3	s2	s1	s0
`JPM Z,ps`	`PSET p`	1	1	1	0	0	1	0	p4	p3	p2	p1	p0
	`JP Z,s`	0	1	1	0	s7	s6	s5	s4	s3	s2	s1	s0
`JPM NZ,ps`	`PSET p`	1	1	1	0	0	1	0	p4	p3	p2	p1	p0
	`JP NZ,s`	0	1	1	1	s7	s6	s5	s4	s3	s2	s1	s0
`CALLM ps`	`PSET p`	1	1	1	0	0	1	0	p4	p3	p2	p1	p0
	`CALL s`	0	1	0	0	s7	s6	s5	s4	s3	s2	s1	s0

Character string ps represents 13-bit immediate data that indicates the branch-destination address. A label can be used for it.

Example:

Source file

```
        :
        JPM     LABEL2
        :
        PAGE    2
LABEL2  LD      A,0
        :
```

Assembly list file after expansion

```
          :
          JPM     LABEL2
+          PSET   LABEL2
+          JP     LABEL2
          :
          PAGE    2
  LABEL2  LD      A,0
          :
```

Macro-definitions

The macro-definition should be done by using the MACRO and the ENDM instructions (pseudo-instruction).

MACRO ENDM

```
<macro name>_MACRO_[<dummy argument>, ...]
                    Statement
                       :
                    ENDM
```

The statement block enclosed by a MACRO pseudo-instruction and an ENDM pseudo-instruction is defined as a macro. Any name can be assigned to the macro as long as it conforms to the rules regarding the characters, length, and label field.

A macro can have an argument passed to it when it is called. In this case, any symbol can be used as a dummy argument in the macro definition where the actual argument is to be substituted and the same symbol must be written after the MACRO pseudo-instruction. Multiple dummy arguments must be separated by commas (,).

Be sure to write the ENDM statement at the end of a macro-definition.

Example: This macro loads data from the memory location specified by ADDR into the A or B register specified by REG. Sample call: LDM A,10H

```
LDM    MACRO    REG,ADDR
       LD       X,ADDR
       LD       REG,MX
       ENDM
```

These dummy arguments are replaced by actual arguments when the macro is expanded.

LOCAL

If a macro having a label is expanded at multiple locations, the label duplicates, causing an error. The LOCAL pseudo-instruction prevents this error occurring.

```
LOCAL_<label-name>[,<label-name>...]
```

The label specified by the LOCAL pseudo-instruction is replaced by "??nnnn" when the macro is expanded. Field nnnn is a four-digit decimal field, to which values 0001 to 9999 are assigned sequentially.

The LOCAL pseudo-instruction must be written at the beginning of the macro. The LOCAL pseudo-instruction is ignored if another instruction precedes it.

Example:

```
WAIT   MACRO   CNT
       LOCAL   LOOP
       LD      A,CNT
LOOP   SBC     A,1      ←Replaces LOOP with ??nnnn
       JP      NZ,LOOP   at expansion.
       ENDM
```

Macro-calls

The defined macro-name can be called from any location in the program by using the following format:

```
[<label>]_<macro-name>_[<actual-argument>, ...]
```

The MACRO can be called by using the macro-name.
When arguments are required, write actual arguments corresponding to the dummy arguments used in the macro-definition. Multiple actual arguments must be separated by commas (,).
Actual and dummy arguments correspond sequentially from left to right. If the number of actual arguments is greater than the number of dummy arguments, the excess actual arguments are ignored. If the number of actual arguments is less than the number of dummy arguments, the excess dummy arguments are replaced by nulls (00H).
Any label can be written before the macro-name.

Example:

Source file

```
        ORG     0200H

CTAS    EQU     00H
CTAE    EQU     02H
CAFSET  EQU     0101B
CAFRST  EQU     0000B
CTBS    EQU     10H
CTBE    EQU     08H
CBFSET  EQU     0001B
CBFRST  EQU     0100B

COUNT   MACRO   FSET,FRST,CTS,CTE
        LOCAL   LOOP1
        SET     F,FSET
        RST     F,FRST
        LD      A,0
        LD      X,CTS
LOOP1   ACPX    MX,A
        CP      XL,CTE
        JP      NZ,LOOP1
        ENDM

COUNTA  COUNT   CAFSET,CAFRST,CTAS,CTAE
        RET

COUNTB  COUNT   CBFSET,CBFRST,CTBS,CTBE
        RET

        END
```

The assembly listing file after assembly is shown on the next page.

Assembly listing file

```
LISTING OF ASM62XX     C2XX0A1.PRN  ........ PAGE   1
  LINE BANK PCP PCS    OBJ          SOURCE STATEMENT
     1                                  ORG     0200H
     2
     3                0000=     CTAS    EQU     00H
     4                0002=     CTAE    EQU     02H
     5                0005=     CAFSET  EQU     0101B
     6                0000=     CAFRST  EQU     0000B
     7                0010=     CTBS    EQU     10H
     8                0008=     CTBE    EQU     08H
     9                0001=     CBFSET  EQU     0001B
    10                0004=     CBFRST  EQU     0100B
    11
    12                          COUNT   MACRO   FSET,FRST,CTS,CTE
    13                                  LOCAL   LOOP1
    14                                  SET     F,FSET
    15                                  RST     F,FRST
    16                                  LD      A,0
    17                                  LD      X,CTS
    18                          LOOP1   ACPX    MX,A
    19                                  CP      XL,CTE
    20                                  JP      NZ,LOOP1
    21                                  ENDM
    22
    23                          COUNTA  COUNT   CAFSET,CAFRST,CTAS,CTAE
    24    0   2  00    F45    +         SET     F,CAFSET
    25    0   2  01    F50    +         RST     F,CAFRST
    26    0   2  02    E00    +         LD      A,0
    27    0   2  03    B00    +         LD      X,CTAS
    28    0   2  04    F28    + ??0001  ACPX    MX,A
    29    0   2  05    A52    +         CP      XL,CTAE
    30    0   2  06    704    +         JP      NZ,??0001
    31    0   2  07    FDF              RET
    32
    33                          COUNTB  COUNT   CBFSET,CBFRST,CTBS,CTBE
    34    0   2  08    F41    +         SET     F,CBFSET
    35    0   2  09    F54    +         RST     F,CBFRST
    36    0   2  0A    E00    +         LD      A,0
    37    0   2  0B    B10    +         LD      X,CTBS
    38    0   2  0C    F28    + ??0002  ACPX    MX,A
    39    0   2  0D    A58    +         CP      XL,CTBE
    40    0   2  0E    70C    +         JP      NZ,??0002
    41    0   2  0F    FDF              RET
    42
    43                                  END
```

CHAPTER 5 ERROR MESSAGES

If an error occurs during assembly, ASM62XX outputs the appropriate error symbol or error message listed below to the console and assembly listing file.

Only a single error symbol is output at the beginning (column 1) of the statement that caused the error. (If two or more errors occurred, only the error with highest priority is output.)

The following error symbols are listed in order of priority, starting with the one with the highest priority.

– **S (Syntax Error)**
An unrecoverable syntax error was encountered.

– **U (Undefined Error)**
The label or symbol of the operand has not been defined.

– **M (Missing Label)**
The label field has been omitted.

– **O (Operand Error)**
A syntax error was encountered in the operand, or the operand could not be evaluated.

– **P (Phase Error)**
The same label or symbol was defined more than once.

– **R (Range Error)**

- The location counter value exceeded the upper limit of the program memory, or a location exceeding the upper limit was specified.
- A value greater than that which the number of significant digits of the operand will accommodate was specified.

- **! (Warning)**
  - Memory areas overlapped because of a "PAGE" or "ORG" pseudo-instruction or both.
  - A statement exceeded a page boundary although its location was not specified.

- **FILE NAME ERROR**

  The source file name was longer than 8 characters.

- **FILE NOT PRESENT**

  The specified source file was not found.

- **DIRECTORY FULL**

  No space was left in the directory of the specified disk.

- **FATAL DISK WRITE ERROR**

  The file could not be written to the disk.

- **LABEL TABLE OVERFLOW**

  The number of defined labels and symbols exceeded the label table capacity (2000).

- **CROSS REFERENCE TABLE OVERFLOW**

  The label/symbol reference count exceeded the cross-reference table capacity (only when the cross-reference table is generated).

APPENDIX ASM62XX EXECUTION EXAMPLE

1) Source file (C2XX0A0.DAT)

```
A>TYPE C2XX0A0.DAT
;
;*******<< SAMPLE PROGRAM :E0C62XX >>*******
;
ABC     EQU     0F0H
TEN     EQU     10
;
START   LD      A,0
LD      X,8
LD      Y,3
LDPX    A,MX
;
ORG     0E0H
;
NEXT    ADD     B,TEN
LD      MX,XH
AND     A,101B
FAN     MY,A
RCF
SCPX    MX,B
JP      C,NEXT
;
;-------<<          ERROR          >>-------
        EQU     0CH-2
ERROR   EQU     4
ERROR   LD      A,3
        SBD     MX,A
        INC     Z
        JP      UNDEF
        ORG     11100000B
        NOP5
        SECTION
        ORG     ABC+0FH
        NOP7
        NOP7
        END
```

2) Running the assembler (display on the console)

```
A>ASM62XX C2XX0A0
          *** E0C62XX CROSS ASSEMBLER. --- VERSION 2.00 ***

    EEEEEEEEEE   PPPPPPPP        SSSSSSS       OOOOOOOO     NNN     NNN
    EEEEEEEEEE   PPPPPPPPPP     SSS   SSSS    OOO    OOO    NNNN    NNN
    EEE          PPP     PPP   SSS     SSS   OOO      OOO   NNNNN   NNN
    EEE          PPP     PPP    SSS          OOO      OOO   NNNNNN  NNN
    EEEEEEEEEE   PPPPPPPPPP      SSSSSS      OOO      OOO   NNN NNN NNN
    EEEEEEEEEE   PPPPPPPP           SSSS     OOO      OOO   NNN  NNNNNN
    EEE          PPP                  SSS    OOO      OOO   NNN   NNNNN
    EEE          PPP           SSS     SSS   OOO      OOO   NNN    NNNN
    EEEEEEEEEE   PPP           SSSS   SSS     OOO    OOO    NNN     NNN
    EEEEEEEEEE   PPP             SSSSSSS       OOOOOOOO     NNN      NN

           (C) COPYRIGHT 1989  SEIKO EPSON CORP.

          SOURCE FILE NAME IS " C2XXYYY.DAT "

          THIS SOFTWARE MAKES NEXT FILES.

             C2XXYYYH.HEX  ...  HIGH BYTE OBJECT FILE.
             C2XXYYYL.HEX  ...  LOW BYTE OBJECT FILE.
             C2XXYYY .PRN  ...  ASSEMBLY LIST FILE.


          DO YOU NEED AUTO PAGE SET?(Y/N) N
          DO YOU NEED CROSS REFERENCE TABLE?(Y/N) Y
          M   23                000A=              EQU       0CH-2
          P   24                0004=      ERROR   EQU       4
          P   25    0   0  E7    E03       ERROR   LD        A,3
          S   26    0   0  E8    FFF               SBD       MX,A
          O   27    0   0  E9    FFF               INC       Z
          U   28    0   0  EA    000               JP        UNDEF
          !   30                                   NOP5
          R   34    0   1  00                      NOP7

          8 ERROR OR WARNING(S) DETECTED
          Used : 6/2000  symbols
A>
```

3) Assembly listing file (C2XX0A0.PRN)

```
A>TYPE C2XX0A0.PRN
LISTING OF ASM62XX     C2XX0A0.PRN  ........  PAGE   1
  LINE BANK PCP PCS    OBJ          SOURCE STATEMENT
     1                          ;
     2                          ;*******<< SAMPLE PROGRAM :E0C62XX >>*******
     3                          ;
     4                00F0=     ABC     EQU     0F0H
     5                000A=     TEN     EQU     10
     6                          ;
     7    0   0  00    E00      START   LD      A,0
     8    0   0  01    B08              LD      X,8
     9    0   0  02    803              LD      Y,3
    10    0   0  03    EE2              LDPX    A,MX
    11                          ;
    12                                  ORG     0E0H
    13                          ;
    14    0   0  E0    C1A      NEXT    ADD     B,TEN
    15    0   0  E1    EA6              LD      MX,XH
    16    0   0  E2    C85              AND     A,101B
    17    0   0  E3    F1C              FAN     MY,A
    18    0   0  E4    F5E              RCF
    19    0   0  E5    F39              SCPX    MX,B
    20    0   0  E6    2E0              JP      C,NEXT
    21                          ;
    22                          ;-------<<          ERROR          >>-------
M   23                000A=             EQU     0CH-2
P   24                0004=     ERROR   EQU     4
P   25    0   0  E7    E03      ERROR   LD      A,3
S   26    0   0  E8    FFF              SBD     MX,A
O   27    0   0  E9    FFF              INC     Z
U   28    0   0  EA    000              JP      UNDEF
    29                                  ORG     11100000B
!   30                                  NOP5
    31                                  SECTION
    32                                  ORG     ABC+0FH
    33    0   0  FF    FFF              NOP7
R   34    0   1  00                     NOP7
    35                                  END
8 ERROR OR WARNING(S) DETECTED
  LABEL TABLE          PAGE L-  1
  ABC     =00F0       ERROR   =0004       NEXT    0-0-E0       START    0-0-00
  TEN     =000A     U UNDEF   0-0-00
 CROSS REFERENCE TABLE   PAGE X-  1
ABC    4#       32
ERROR  24#      25#
NEXT   14#      20
START  7#
TEN    5#       14
UNDEF  28
```

4) Object files (C2XX0A0H.HEX, C2XX0A0L.HEX)

```
A>TYPE C2XX0A0L.HEX
 :10000000000803E2FFFFFFFFFFFFFFFFFFFFFFFF0F
 :10001000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0
 :10002000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE0
 :10003000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD0
 :10004000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC0
 :10005000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFB0
 :10006000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA0
 :10007000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF90
 :10008000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF80
 :10009000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF70
 :1000A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF60
 :1000B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF50
 :1000C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF40
 :1000D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF30
 :1000E0001AA6851C5E39E003FFFF00FFFFFFFFFF3C
 :1000F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF10
 :10010000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
 :10011000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEF
 :10012000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDF
 :10013000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFCF
 :10014000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBF
 :10015000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFAF
 :10016000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9F
 :10017000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8F
 :10018000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7F
 :10019000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6F
 :1001A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5F
 :1001B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4F
 :1001C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3F
 :1001D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2F
 :1001E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1F
 :1001F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0F
 :10020000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE
 :10021000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEE
 :10022000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDE
 :10023000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFCE
 :10024000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBE
 :10025000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFAE
 :10026000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9E
 :10027000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8E
 :10028000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7E
 :10029000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6E
```

```
:1002A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5E
:1002B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4E
:1002C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3E
:1002D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2E
:1002E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1E
:1002F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0E
:10030000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD
:10031000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFED
:10032000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDD
:10033000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFCD
:10034000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBD
:10035000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFAD
:10036000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9D
:10037000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8D
:10038000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7D
:10039000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6D
:1003A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5D
:1003B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4D
:1003C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3D
:1003D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2D
:1003E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1D
:1003F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0D
:00000001FF
```

(When ROM capacity is in 1,024 steps)

```
A>TYPE C2XX0A0H.HEX
 :100000000E0B080EFFFFFFFFFFFFFFFFFFFFFFFFCD
 :10001000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0
 :10002000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE0
 :10003000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD0
 :10004000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC0
 :10005000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFB0
 :10006000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA0
 :10007000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF90
 :10008000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF80
 :10009000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF70
 :1000A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF60
 :1000B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF50
 :1000C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF40
 :1000D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF30
 :1000E0000C0E0C0F0F020E0F0F00FFFFFFFFFFFF94
 :1000F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0F00
 :10010000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
 :10011000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEF
 :10012000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDF
 :10013000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFCF
 :10014000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBF
 :10015000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFAF
 :10016000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9F
 :10017000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8F
 :10018000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7F
 :10019000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6F
 :1001A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5F
 :1001B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4F
 :1001C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3F
 :1001D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2F
 :1001E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1F
 :1001F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0F
 :10020000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE
 :10021000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEE
 :10022000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDE
 :10023000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFCE
 :10024000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBE
 :10025000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFAE
 :10026000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9E
 :10027000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8E
 :10028000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7E
 :10029000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6E
 :1002A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5E
 :1002B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4E
 :1002C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3E
```

```
:1002D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2E
:1002E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1E
:1002F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0E
:10030000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD
:10031000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFED
:10032000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDD
:10033000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFCD
:10034000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBD
:10035000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFAD
:10036000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9D
:10037000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8D
:10038000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7D
:10039000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6D
:1003A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5D
:1003B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4D
:1003C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3D
:1003D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2D
:1003E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1D
:1003F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0D
:00000001FF
```

(When ROM capacity is in 1,024 steps)

Note *The size of the object file differs depending on the device and the ROM capacity. See Chapter 1, "E0C62XX RESTRICTIONS".*

III. E0C6262 Function Option Generator Manual

CONTENTS

FOG6262

CHAPTER 1 GENERAL

1.1 Outline of Function Option Generator

With the 4-bit single-chip E0C6262 microcomputers, the customer may select 16 hardware options. By modifying the mask patterns of the E0C6262 according to the selected options, the system can be customized to meet the specifications of the target system.

The FOG6262 Function Option Generator (hereinafter called FOG6262) is a software tool for generating data files used to generate mask patterns. It enables the customer to interactively select and specify pertinent items for each hardware option. From the data file created with FOG6262, the E0C6262 mask pattern is automatically generated by a general purpose computer.

The HEX file for the evaluation board (EVA6262) hardware option ROM is simultaneously generated with the data file. By writing the contents of the HEX file into the EPROM and mounting it on the EVA6262, option functions can be executed on the EVA6262.

Two FOG6262 program disks are supplied by SEIKO EPSON: one for NEC PC-9801V series (5.25" 2HD) and one for IBM PC/XT and PC/AT (5.25" 2D). The basic configurations are as follows.

PC-9801V series	Host computer:	PC-9801V series
	Disk drive:	FD (5.25" 2HD) × 1 or more
	Operating system:	MS-DOS Ver. 3.1 or later
	ROM writer:	Required when using EVA6262
IBM PC/XT and PC/AT	Host computer:	IBM PC/XT and PC/AT
	Disk drive:	FD (5.25" 2D) × 1 or more
	Operating system:	PC-DOS Ver. 2.1 or later
	ROM writer:	Required when using EVA6262

The program name of FOG6262 is as follows:

FOG6262.EXE

1.2 Execution Flow and I/O Files

Figure 1.2.1 shows the FOG6262 execution flow.

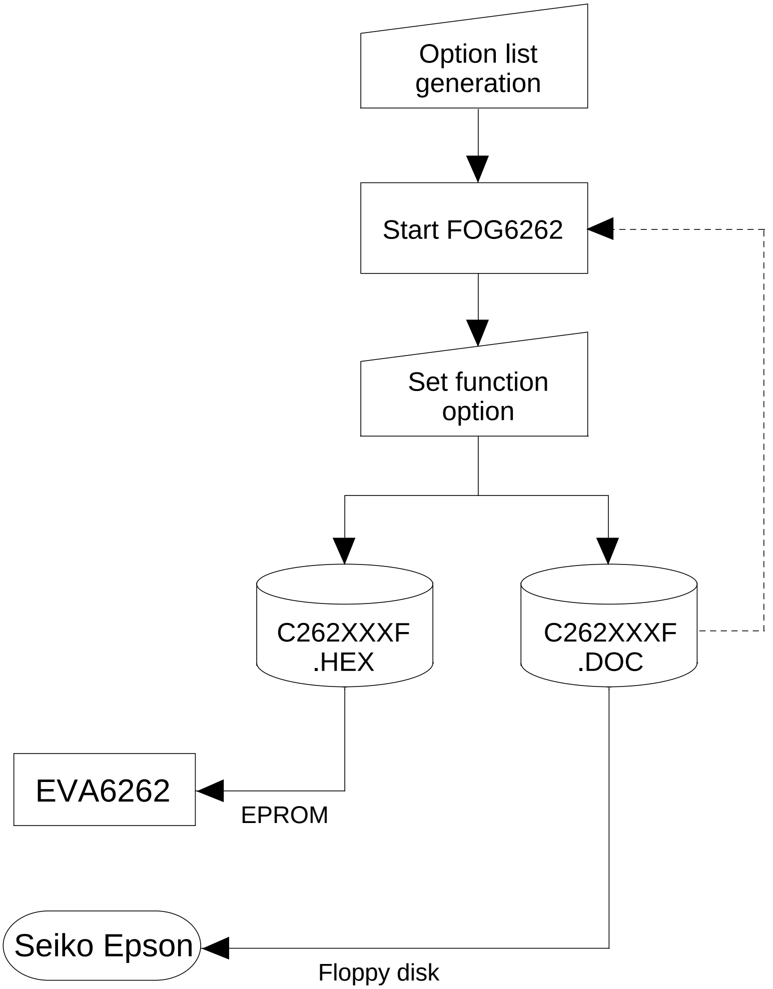


Fig. 1.2.1
Execution flow

(1) Option list generation

Select the hardware options that meet the specifications of the target system and record them in the option list (paper for recording items in preparation for input operation; explained later).

(2) FOG6262 execution

Start FOG6262 and select the required function options. Function options can be interactively selected, so an input file need not be generated. Already selected options can be modified.

FOG6262 outputs the following data files:

- Function option document file (C262XXXF.DOC)
  This is a data file used to generate the mask patterns for such items as I/O ports. This file must be sent with the completed program file.
- Function option HEX file (C262XXXF.HEX)
  This is a function option file (Intel hexa format) used for EVA6262. One EVA6262 function option ROM is generated by writing this file with the ROM writer.

Remarks:

- File name "XXX" is specified for each customer by Seiko Epson.
- Combine the document files with the program files (C262XXXH.HEX and C262XXXL.HEX) using the mask data checker (MDC6262): copy the combined file into another diskette and submit to Seiko Epson.
- Set all unused ROM areas to FFH when writing the HEX file into the EPROM. (Refer to "EVA6262 Manual" for the ROM installation location.)

CHAPTER 2 OPTION LIST GENERATION

2.1 Option List Recording Procedure

Multiple specifications are available in each option item as indicated in the Option List in Section 2.2. Using "2.3 Option Specifications" as reference, select the specifications that meet the target system and check the appropriate box. Be sure to record the specifications for unused ports too, according to the instructions provided.

2.2 Option List

The E0C6262 option list is as follows:

1. DEVICE TYPE
   - E0C6262 .... □
   - E0C62L62 .... □

     OSC3 Oscillator (See Note 1)
     - Ceramic Oscillator .... □
     - CR Oscillator .... □
     - Not Use .... □

2. V_{DE} POWER SUPPLY
   - Not Use .... □
   - Use .... □ K1, R1, P2, P3
□ All Pads

3. MULTIPLE KEY ENTRY RESET
   - Not Use .... □
   - Use .... □ K00, K01
     □ K00, K01, K02
     □ K00, K01, K02, K03

4. INPUT INTERRUPT NOISE REJECTOR
   - K00–K03 .... □ Use □ Not Use
   - K10–K13 .... □ Use □ Not Use

5. INPUT PORT PULL UP RESISTOR
   - K00–K03 .... □ With Resistor □ Gate Direct
   - K10–K13 .... □ With Resistor □ Gate Direct

6. OUTPUT PORTS OUTPUT SPECIFICATION (R00–R03)
   - R00–R03 .... □ Complementary □ Nch Open Drain

7. R10 SPECIFICATION
   - Output Type .... □ DC Output □ $\overline{\text{SRDY}}$ Output
   - Output Specification .... □ Complementary □ Nch Open Drain

8. R11 SPECIFICATION

- Output Type .... □ DC Output
  □ FOUT 32768 [Hz]
  □ FOUT 16384 [Hz]
  □ FOUT 8192 [Hz]
  □ FOUT 4096 [Hz]
  □ FOUT 2048 [Hz]
  □ FOUT 1024 [Hz]
  □ FOUT 512 [Hz]
  □ FOUT 256 [Hz]
- Output Specification .... □ Complementary □ Nch Open Drain

9. R12 SPECIFICATION

- Output Type .... □ DC Output □ Buzzer Output
- Output Specification .... □ Complementary □ Nch Open Drain

10. R13 SPECIFICATION

- Output Type .... □ DC Output
  □ Buzzer Inverted Output (See Note 2)
- Output Specification .... □ Complementary □ Nch Open Drain

11. I/O PORTS OUTPUT SPECIFICATION

- P00–P03 .... □ Complementary □ Nch Open Drain
- P10–P13 .... □ Complementary □ Nch Open Drain
- P20–P23 .... □ Complementary □ Nch Open Drain

12. I/O PORTS PULL UP RESISTOR

- P00–P03 .... □ With Resistor □ Gate Direct
- P10–P13 .... □ With Resistor □ Gate Direct
- P20–P23 .... □ With Resistor □ Gate Direct
  - P00–P03 Pull Up Resistance .. □ 150 kΩ □ 50 kΩ (See Note 1)
  - P10–P13 Pull Up Resistance .. □ 150 kΩ □ 50 kΩ (See Note 1)
  - P20–P23 Pull Up Resistance .. □ 150 kΩ □ 50 kΩ (See Note 1)

13. P30–P33 SPECIFICATION (See Note 3)

- Output Type .... □ I/O Port □ SI/O Port
- Output Specification .... □ Complementary □ Nch Open Drain

14. P30–P33 PULL UP RESISTOR (See Note 3)

- P30–P33 .... □ With Resistor □ Gate Direct
    - P30–P33 Pull Up Resistance .. □ 150 kΩ □ 50 kΩ (See Note 1)

15. WATCHDOG TIMER

- Use .... □
- Not Use .... □

16. SERIAL INTERFACE DATA PERMUTATION (See Note 4)

- LSB First.............................. □
- MSB First............................. □

FOG6262

Note 1: This option is valid only if "E0C6262" were selected for device type in Item 1.

Note 2: "Buzzer Inverted Output" (R13) may only be selected if "Buzzer Output" has been selected as the output type for R12 in Item 9.

Note 3: If "Use" were selected for the VDE power supply in Item 2, P30–P32 will be the terminals for which option may be selected.

Note 4: These options may be selected even if P30–P33 are not set to "SI/O Port" in Item 13; however, if P30–P33 are not set to "SI/O Port", the capability of the built-in serial interface circuit cannot be used.

2.3 Option Specification

Device type (OSC3 oscillator)

- E0C6262 .... □
- E0C62L62 .... □

Select the chip specification.
The E0C6262 has 3.0 V type power voltage while E0C62L62 has 1.5 V low power type power voltage.
When using the twin clock type E0C62A62, select "E0C6262".

If "E0C6262" were selected here, the following options may be selected:

OSC3 Oscillator

- Ceramic Oscillator .... □
- CR Oscillator .... □
- Not Use .... □

Select the oscillator which uses OSC3 and OSC4 terminals.
In case of twin clock type E0C62A62, select "Ceramic Oscillator" or "CR Oscillator".
In case of single clock type E0C6262, select "Not Use".

In case OSC3 oscillator is to be utilized (E0C62A62):
To minimize the number of external parts, CR oscillator is suitable; to obtain stable oscillation frequency, ceramic oscillator is suitable.
When "CR Oscillator" is selected, only resistor will be required as external parts because capacity is built in.
On the other hand, when "Ceramic Oscillator" is selected, external parts necessary are ceramic oscillator, gate capacity and drain capacity.
When "Ceramic Oscillator" is selected, the frequency will be fixed at 1 MHz whereas selecting "CR Oscillator" will allow modification of the frequency to a certain extent, depending on the external resistance.

VDE power supply

- Not Use .... □
- Use .... □ K1, R1, P2, P3
  □ All Pads

Select the power supply for the I/O driver of the input ports (K0 and K1), output ports (R0 and R1), and I/O ports (P0, P1, P2 and P3).

If "Not Use" were selected, the I/O driver of all ports will assume the power supply VDD (+) of IC to be the power supply on the positive side.

If VDE power supply is to be used, the P33 I/O port will serve as the power terminal VDE (+). Since the VDD/VDE power supply switching is done in units of (K0, R0, P0, P1) and (K1, R1, P2, P3), if "Use" were selected, the power supply of the ports will be as follows:

- Use .... □ K1, R1, P2, P3

K0, R0, P0, P1: VDD
K1, R1, P2, P3: VDE (See note below)

- Use .... □ All pads

K0, R0, P0, P1: VDE
K1, R1, P2, P3: VDE (See note below)

Since the electric potential of VDE (+) may be independently set from the power supply VDD (+) of the IC (with the VSS (-) being common), for example, if VDD = 1.5 V and VDE = 5.0 V settings were made, while the E0C62L62 main unit is being operated at low voltage, it can be interfaced with other ICs which are being operated at 5.0 V.

Note *K0, R0, etc., refer to K00–K03, R00–R03, etc., respectively.*
If VDE power supply were set to "Not Use", P3 may be used as a 4-terminal I/O port with P30–P33; if set to "Use", P3 may be used as a 3-terminal I/O port with P30–P32.

Multiple key entry reset

- Not Use .... □
- Use .... □ K00, K01

  □ K00, K01, K02

  □ K00, K01, K02, K03

The reset function is set when K00 through K03 are entered. When "Not Use" is selected, the reset function is not activated even if K00 through K03 are entered. When "Use K00, K01" is selected, the system is reset immediately the K00 and K01 inputs go low at the same time. Similarly, the system is reset as soon as the K00 through K02 inputs or the K00 through K03 inputs go low.
However, the system is reset when a low signal is input for more than a rule time (2 to 4 sec).

The system reset circuit is shown in Figure 2.3.1.

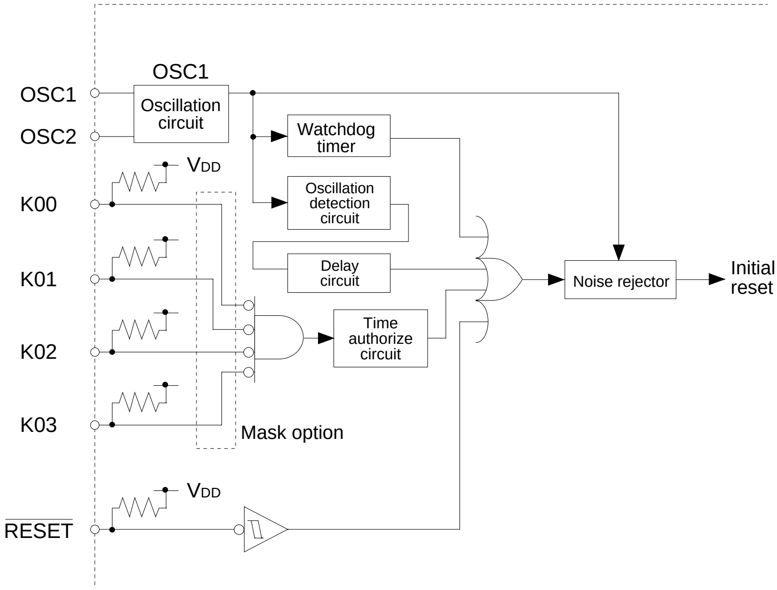


Fig. 2.3.1
System reset circuit

Input interrupt noise rejector

- K00–K03 .... □ Use □ Not Use
- K10–K13 .... □ Use □ Not Use

Select whether noise rejector will be supplemented to the input interrupt circuit of K00–K03 and K10–K13.

When "Use" is selected, the entry signal will pass the noise rejector, and occurrence of interrupt errors due to noise or chattering can be avoided. Note, however, that because the noise rejector performs entry signal sampling at 4 kHz, "Not Use" should be selected when high speed response is required.

Input ports pull up resistor

- K00–K03 .... □ With Resistor □ Gate Direct
- K10–K13 .... □ With Resistor □ Gate Direct

Select whether or not pull up resistors will be used for every 4 bits of the input ports (K00–K03 and K10–K13).

When "Gate Direct" is selected, see to it that entry floating state does not occur. Select "With Resistor" for unused ports.

The configuration of the pull up resistor circuit is shown in Figure 2.3.2.

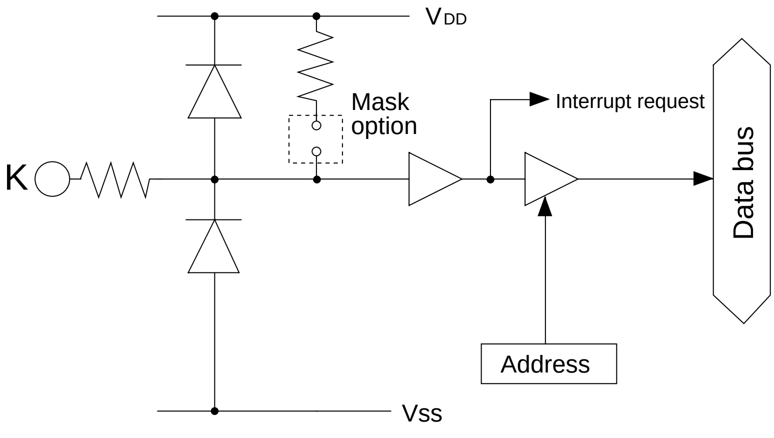


Fig. 2.3.2
Configuration of pull up resistor circuit

Output ports output specification (R00–R03)

- R00–R03 .... □ Complementary □ Nch Open Drain

Select the output specification for the output ports (R00–R03).

Either "Complementary" output or "Nch Open Drain" output may be selected.

When output port is to be used on key matrix configuration, select "Nch Open Drain" output.

If the output ports will not be used, select "Complementary" output.

The configuration of the output port is shown in Figure 2.3.3.

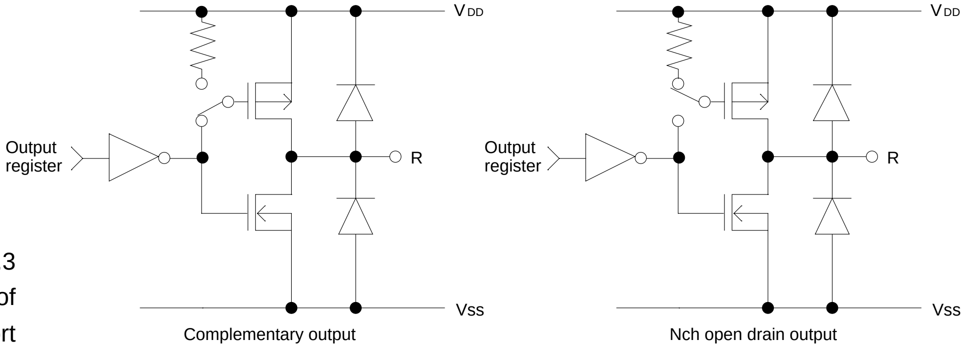


Fig. 2.3.3 Configuration of output port

R10 specification

• Output Type .... □ DC Output □ $\overline{\text{SRDY}}$ Output
• Output Specification ... □ Complementary □ Nch Open Drain

Select the output specification for the R10 terminal.
Either "Complementary" output or "Nch Open Drain" output may be selected.
When "DC Output" is selected, R10 becomes a regular output port.
When "$\overline{\text{SRDY}}$ Output" is selected, ready signal of the serial interface ($\overline{\text{SRDY}}$) is generated from R10 terminal.

The circuit configuration is the same as that of output ports (R00–R03 shown in Figure 2.3.3).

Refer to Figure 2.3.8 for $\overline{\text{SRDY}}$ output waveform.

R11 specification

• Output Type .... □ DC Output

□ FOUT 32768 [Hz]

□ FOUT 16384 [Hz]

□ FOUT 8192 [Hz]

□ FOUT 4096 [Hz]

□ FOUT 2048 [Hz]

□ FOUT 1024 [Hz]

□ FOUT 512 [Hz]

□ FOUT 256 [Hz]

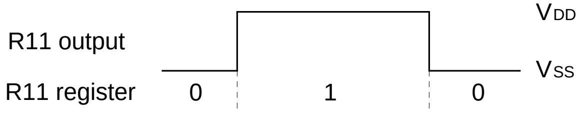
• Output Specification ... □ Complementary □ Nch Open Drain

Select the output specification for R11 terminal.
Either "Complementary" output or "Nch Open Drain" output may be selected.
When "DC Output" is selected, R11 becomes a regular output port.
When "FOUT" output is selected, clock with frequency selected from R11 terminal is generated by writing "1" to the R11 register.

– When "DC Output" is selected

When R11 register (D5 address, D1 bit) is set to "1", the R11 terminal output goes high (V_{DD}), and goes low (V_{SS}) when set to "0".
Output waveform is shown in Figure 2.3.4.

Fig. 2.3.4
Output waveform at R11 DC output selection



– When "FOUT" output is selected

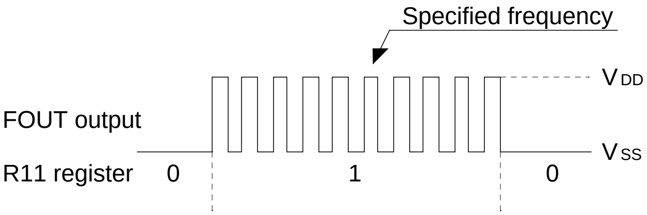
When FOUT bit (R11 register) is set to "1", 50% duty and VDD–VSS amplitude square wave is generated at the specified frequency. When set to "0", R11 terminal goes low (VSS).

A FOUT frequency may be selected from among 8 types, ranging from 256 Hz to 32,768 Hz.

FOUT output is normally utilized to provide clock to other devices but since hazard occurs at the square wave breaks, great caution must be observed when using it.

Output waveform is shown in Figure 2.3.5.

Fig. 2.3.5
Output waveform at R11
FOUT output selection



R12 specification

- Output Type .... □ DC Output □ Buzzer Output
- Output Specification ... □ Complementary □ Nch Open Drain

Select the output specification for the R12 terminal.
Either "Complementary" output or "Nch Open Drain" output may be selected.
When "DC Output" is selected, R12 becomes a regular output port.
When "Buzzer Output" is selected, by writing "1" to the R12 register, buzzer drive (oscillation output) signal is output from the R12 terminal.

* When "DC Output" is selected, R13 terminal output type (see "R13 specification") selection is limited to "DC Output" only.

The circuit configuration is the same as that of output ports (R00–R03 shown in Figure 2.3.3).

Refer to Figure 2.3.6 for buzzer output waveform.

R13 specification

- Output Type .... □ DC Output
  □ Buzzer Inverted Output
- Output Specification ... □ Complementary □ Nch Open Drain

Select the output specification for the R13 terminal.
Either "Complementary" output or "Nch Open Drain" output may be selected.
When "DC Output" is selected, R13 becomes a regular output port.
When "Buzzer Inverted Output" is selected, inverted waveform of R12 buzzer output is generated from the R13 terminal. Buzzer output and buzzer inverted output can be controlled simultaneously by the R12 register, and the data of R13 register will not affect buzzer output and buzzer inverted output.

* "Buzzer Inverted Output" may not be selected when the R12 terminal output type (see "R12 specification") is not set to "Buzzer Output". Moreover, when the output type of R12 terminal is reselected to "DC Output" by the B command after selecting "Buzzer Inverted Output" at this point, the output type of R13 terminal is reset to "DC Output".

Buzzer output waveform is shown in Figure 2.3.6.

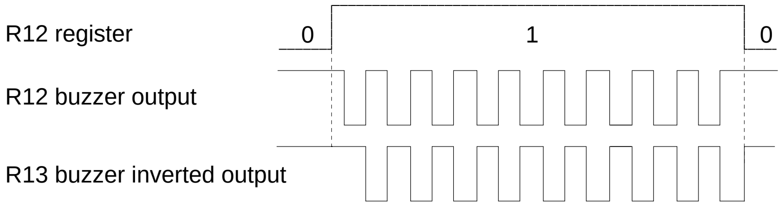


Fig. 2.3.6
Buzzer output waveform

I/O ports specification (P0, P1, P2)

- P00–P03 .... □ Complementary □ Nch Open Drain
- P10–P13 .... □ Complementary □ Nch Open Drain
- P20–P23 .... □ Complementary □ Nch Open Drain

Select the output specification to be used during I/O ports (P0, P1, P2) output mode selection.

Either "Complementary" output or "Nch Open Drain" output may be selected.

The circuit configuration of the output driver is the same as that of output ports (R00–R03 shown in Figure 2.3.3).

Select "Complementary" output for unused ports.

The circuit configuration I/O port is shown in Figure 2.3.7.

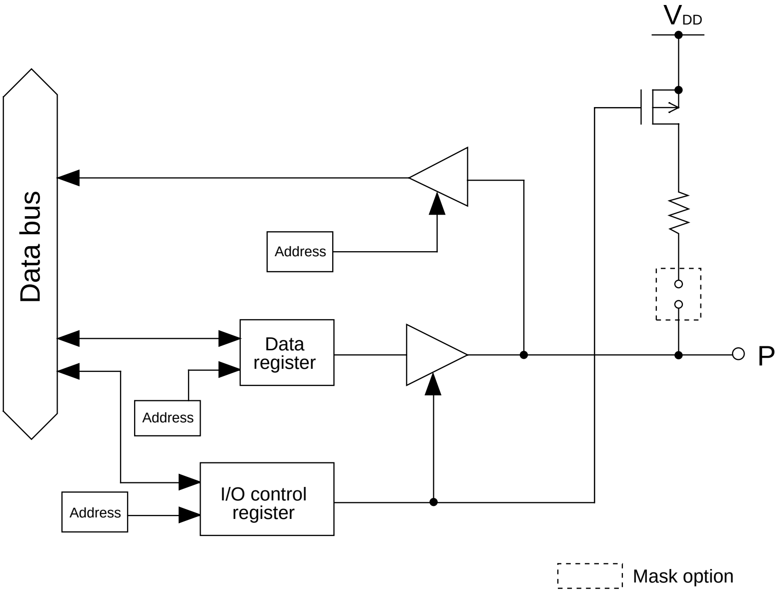


Fig. 2.3.7
Circuit configuration of I/O port

* P0, P1 and P2, refer to P00–P03, P10–P13 and P20–P23, respectively.

I/O ports pull up resistor (P0, P1, P2)

- P00–P03 .... □ With Resistor □ Gate Direct
- P10–P13 .... □ With Resistor □ Gate Direct
- P20–P23 .... □ With Resistor □ Gate Direct

Select whether or not pull up resistors will be used for every 4 bits of the I/O ports (P00–P03, P10–P13 and P20–P23). These pull up resistors will function when the I/O ports are set to the input mode (after resetting or through software). When "Gate Direct" is selected, see to it that entry floating state does not occur. Select "With Resistor" for unused I/O ports.

If "E0C6262" (3.0 V type) were selected in "Device type", selecting "With Resistor" in the above option will allow the selection of the following options:

- P00–P03 Pull Up Resistance ...... □ 150 kΩ □ 50 kΩ
- P10–P13 Pull Up Resistance ...... □ 150 kΩ □ 50 kΩ
- P20–P23 Pull Up Resistance ...... □ 150 kΩ □ 50 kΩ

(The resistance values indicated above are approximates and do not represent absolute accuracy.)

When the input port status is to be changed from low level (VSS), to high level (VDD) using a pull up resistor, a delay in the rise of waveform will occur due to the time constant of the pull up resistor and input load capacity.
In case high speed operation with "E0C62A62" (3.0 V type) is required, low resistance (50 kΩ) may be used to reduce this delay time.
Refer to Figure 2.3.7 for the configuration of the pull up resistor circuit.

* P0, P1 and P2, refer to P00–P03, P10–P13 and P20–P23, respectively.

P30–P33 specification

• Output Type .... □ I/O Port □ SI/O Port
• Output Specification ... □ Complementary □ Nch Open Drain

Select the output specification for the P30–P33 terminals.

– When "I/O Port" is selected

When "I/O Port" is selected as the output type, the output specification during the output mode setting P30–P33 must be selected.
Either "Complementary" output or "Nch Open Drain" output may be selected in 4 bits units (P30–P33).
Select "Complementary" output if these terminals will not be used.
The circuit configuration is the same as that of I/O ports (P00–P03, P10–P13 and P20–P23 shown in Figure 2.3.7).

– When "SI/O Port" is selected

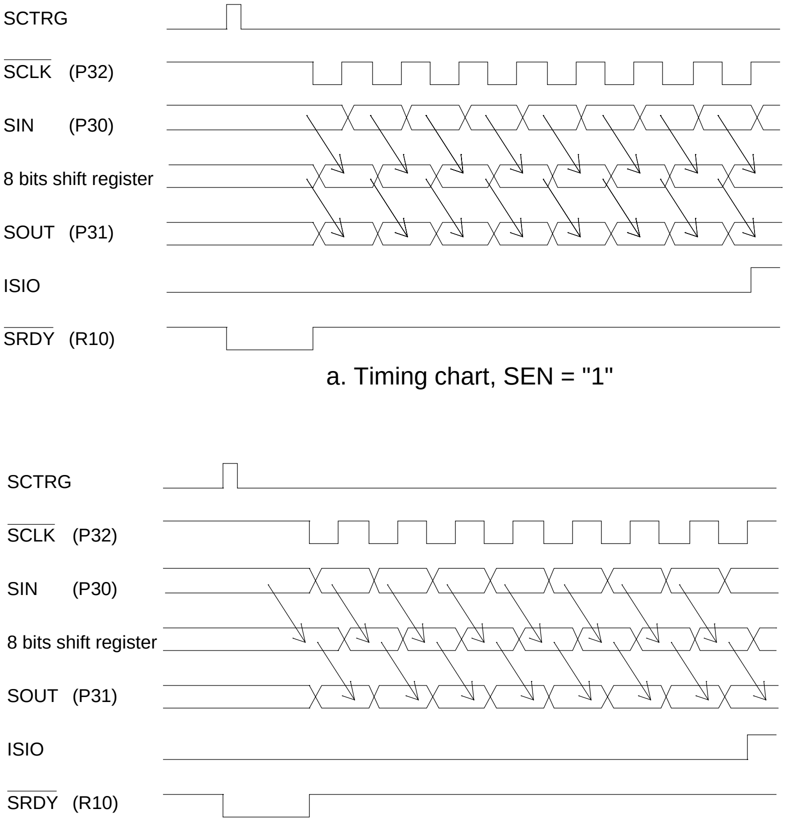
When "SI/O Port" is selected as the output type, the capability of the built-in serial interface circuit may be used. In this case, the specifications of the P30–P33 terminals will be as follows:

P30 = SIN : Serial data input terminal (input only)
P31 = SOUT : Serial data output terminal (output only)
P32 = $\overline{\text{SCLK}}$: Serial data transfer clock I/O terminal (input/output)
P33 = Exclusive output terminal

When "SI/O Port" is selected, the output specification for P31 (SOUT), P32 (SCLK, during transfer clock output) and P33 (exclusive output terminal) must be selected.
Either "Complementary" output or "Nch Open Drain" output may be selected in 3 bits units (SOUT, SCLK and P33).

The output waveform is shown in Figure 2.3.8.

Note *When "SI/O Port" is selected, the value of the P30–P32 data registers will have no effect on the SIN, SOUT and $\overline{\text{SCLK}}$ terminals. Moreover, P33 will become a general purpose output terminal and hence input will not be possible.*
If "Use" were selected in "V_{DE} power supply", P33 cannot be used as a regular terminal regardless of the output type ("I/O Port" or "SI/O Port") selection.



a. Timing chart, SEN = "1"

b. Timing chart, SEN = "0"

Fig. 2.3.8
Timing chart of serial interface

P30–P33 pull up resistor

- P30–P33 .... □ With Resistor □ Gate Direct

Select whether or not pull up resistors will be used with all 4 bits of P30–P33.

– In case of "I/O Port" selection as the output type of P30–P33

These pull up resistors will function when P30–P33 are set to the input mode (after resetting or through software). When "Gate Direct" is selected, see to it that entry floating state does not occur. Select "With Resistor" for unused I/O ports.

If "E0C6262" (3.0 V type) were selected in "Device type", selecting "With Resistor" in the above option will allow the selection of the following options:

- P30–P33 Pull Up Resistance ...... □ 150 kΩ □ 50 kΩ

(The resistance values indicated above are approximates and do not represent absolute accuracy.)

Note *If P30–P33 are to be used as I/O ports, all specification and option selections will be the same as those for P00–P03, P10–P13 and P20–P23.*

– In case of "SI/O Port" selection as the output type of P30–P33

When "With Resistor" is selected, pull up resistor will be added to SIN (P30) and $\overline{\text{SCLK}}$ (P32) terminals. The pull up resistor for SIN will always function, but the pull up resistor for $\overline{\text{SCLK}}$ will only function during serial data transfer clock input.

If "E0C6262" (3.0 V type) were selected in "Device type", selecting "With Resistor" in the above option will allow the selection of the following options:

• P30–P33 Pull Up Resistance ...... □ 150 kΩ □ 50 kΩ

(The resistance values indicated above are approximates and do not represent absolute accuracy.)

Note *If "Use" were selected in "V_{DE} power supply", P33 cannot be used as a regular terminal regardless of the output type ("I/O Port" or "SI/O Port") selection.*

Watchdog timer

- Use .... □
- Not Use .... □

Select whether the watchdog timer built-in to detect CPU runaways will be used or not.
When the watchdog timer is not reset by the program within 3 to 4 second cycles, the CPU is initially reset.

Serial interface data permutation

- LSB First .... □
- MSB First .... □

Select the serial data input/output permutation.

In case of LSB first,
data is input/output in the order SD0 ·····SD7.

In case of MSB first,
data is input/output in the order SD7 ·····SD0.

Select the one suitable to your programming needs.
Input/output permutation is shown in Figure 2.3.9.

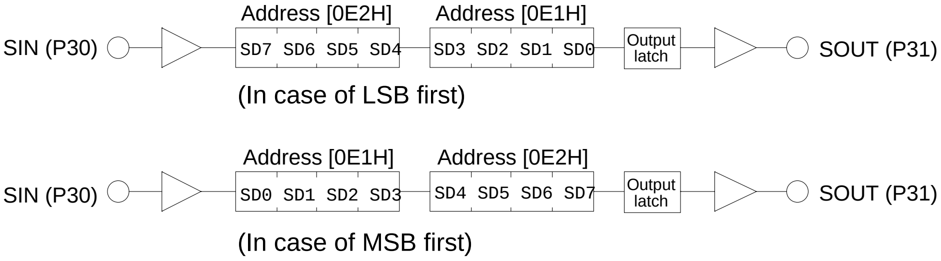


Fig. 2.3.9 Serial interface data permutation

Note *If P30–P33 are not set to "SI/O Port" ("P30–P33 specification"), the capability of the built-in serial interface circuit cannot be used.*

CHAPTER 3 FUNCTION OPTION GENERATOR OPERATION PROCEDURE

3.1 Creating Work Disk

To prevent inadvertent destruction of the contents of the FOG6262 program disk, create a work disk by copying the program disk, place a write protection tab on the program disk, and keep the program disk as a master disk in a safe place. Use the work disk for actual operation. The work disk creation procedure is as follows.

* In examples ↵ means press the RETURN key.

(1) Activate MS-DOS (Ver. 3.1 or later) or PC-DOS (Ver. 2.1 or later) and format a new floppy disk.

Example: Insert the DOS system disk into drive A and a new floppy disk (to be used as the work disk) into drive B, then format the disk in drive B.

```
A>FORMAT B:/S↵    ← The DOS is also copied.
```

(2) Copy the FOG6262 program.

Example: Insert the FOG6262 program disk into drive A and th formatted work disk into drive B, then copy the disk in drive A to the one in drive B.

```
A>COPY *.* B:↵
```

After copying, check that the "FOG6262.EXE" file has been copied onto the work disk. Copy the editor also when performing all operations with this disk.

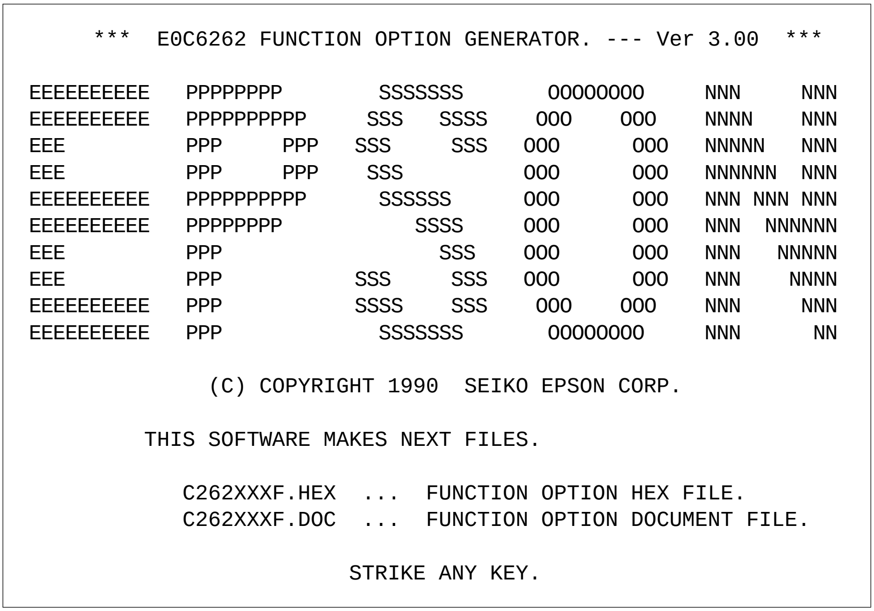
Now, the work disk is ready for use. The two required files are generated on this disk.

3.2 Starting FOG6262

To start FOG6262, insert the work disk into the current drive at the DOS command level (state in which a prompt such as A> is displayed), then enter the program name as shown below.

```
A>FOG6262↵
```

When FOG6262 is started, the following message is displayed.



```
     ***  E0C6262 FUNCTION OPTION GENERATOR. --- Ver 3.00  ***

EEEEEEEEEE   PPPPPPPP        SSSSSSS       OOOOOOOO     NNN     NNN
EEEEEEEEEE   PPPPPPPPPP     SSS   SSSS    OOO    OOO    NNNN    NNN
EEE          PPP     PPP   SSS     SSS   OOO      OOO   NNNNN   NNN
EEE          PPP     PPP    SSS          OOO      OOO   NNNNNN  NNN
EEEEEEEEEE   PPPPPPPPPP      SSSSSS      OOO      OOO   NNN NNN NNN
EEEEEEEEEE   PPPPPPPP           SSSS     OOO      OOO   NNN  NNNNNN
EEE          PPP                  SSS    OOO      OOO   NNN   NNNNN
EEE          PPP           SSS     SSS   OOO      OOO   NNN    NNNN
EEEEEEEEEE   PPP           SSSS    SSS    OOO    OOO    NNN     NNN
EEEEEEEEEE   PPP             SSSSSSS       OOOOOOOO     NNN      NN

               (C) COPYRIGHT 1990  SEIKO EPSON CORP.

         THIS SOFTWARE MAKES NEXT FILES.

            C262XXXF.HEX  ...  FUNCTION OPTION HEX FILE.
            C262XXXF.DOC  ...  FUNCTION OPTION DOCUMENT FILE.

                          STRIKE ANY KEY.
```

For "STRIKE ANY KEY," press any key to advance the program execution. To suspend execution, press the "CTRL" and "C" keys together: the sequence returns to the DOS command level. (It is possible by pressing STOP key depending on the PC used.)

Following the start message, the date currently set in the personal computer is displayed, prompting entry of a new date.

```
*** E0C6262 USER'S OPTION SETTING. --- Ver 3.00 ***

CURRENT DATE IS  90/10/03
PLEASE INPUT NEW DATE  :   90/10/10↵
```

When modifying the date, enter the 2-digit year, month, and day of the month by delimiting them with a slash ("/").

When not modifying the date, press the RETURN key "↵" to continue.

When the date is set, the following operation selection menu is displayed on the screen.

```
*** OPERATION SELECT MENU ***

        1. INPUT NEW FILE
        2. EDIT FILE
        3. RETURN TO DOS

PLEASE SELECT NO.?
```

Enter a number from 1 to 3 to select a subsequent operation. The items indicate the following.

1. INPUT NEW FILE: Used to set new function options.

2. EDIT FILE: Used to read the already-generated function option document file and set or modify the option contents. In this case, the work disk must contain the function option document file (C262XXXF.DOC) generated by "1. INPUT NEW FILE".

3. RETURN TO DOS: Used to terminate FOG6262 and return to the DOS command level.

3.3 Setting New Function Options

This section explains how to set new function options.

```
*** OPERATION SELECT MENU ***

        1. INPUT NEW FILE
        2. EDIT FILE
        3. RETURN TO DOS


PLEASE SELECT NO. ? 1↵                                          .. (1)
PLEASE INPUT FILE NAME   ? C2620A0↵                             .. (2)
PLEASE INPUT USER'S NAME ? SEIKO EPSON CORP.↵                   .. (3)
PLEASE INPUT ANY COMMENT
 ( ONE LINE IS 50 CHR )  ? TOKYO DESIGN CENTER↵                 .. (4)
                         ? 390-4 HINO HINO-SHI TOKYO 191 JAPAN↵
                         ? TEL 0425-83-7313↵
                         ? FAX 0425-83-7413↵
                         ? ↵
```

(1) PLEASE SELECT NO.?
Select "1. INPUT NEW FILE" on the operation selection menu.

(2) PLEASE INPUT FILE NAME?
Enter the file name. Do not enter the extended part of the file name. In case a function option document file (C262XXX.DOC) with the same name as the file name specified in the current drive exists, the user is asked whether overwrition is desired. Enter "Y" or "N" accordingly.

Example:
```
PLEASE INPUT FILE NAME  ? C2620A0↵
EXISTS OVERWRITE (Y/N)  ? N↵
```

(3) PLEASE INPUT USER'S NAME?

Enter the customer's company name.

(4) PLEASE INPUT ANY COMMENT

Enter any comment. Up to 50 characters may be entered in one line. If 51 or more characters are entered in one line, they are ignored. Up to 10 comment lines may be entered. To end entry of comments, press the RETURN key "↵". Include the following in comment lines:

- Company, department, division, and section names
- Company address, phone number, and FAX number
- Other information, including technical information

Next, start function option setting. For new settings, select function options from No. 1 to No. 16 sequentially and interactively.
Moreover, when you wish to modify previously set function options in the middle of a selection process, enter "B↵" to return 1 step back to the previous function option setting operation.

Example:

```
*** OPTION NO.4 ***
.....
PLEASE SELECT NO.(1) ? B↵
```

(See 3.5 for the option selection procedure.)

3.4 Modifying Function Option Settings

This section explains how to modify the function option settings.

```
*** OPERATION SELECT MENU ***

        1. INPUT NEW FILE
        2. EDIT FILE
        3. RETURN TO DOS


PLEASE SELECT NO. ? 2↵                              .. (1)

*** SOURCE FILE(S) ***

C2620A0        C2620B0        C2620C0             .. (2)


PLEASE INPUT FILE NAME   ? C2620A0↵                 .. (3)
PLEASE INPUT USER'S NAME ? ↵                        .. (4)
PLEASE INPUT ANY COMMENT
 ( ONE LINE IS 50 CHR )  ? ↵                        .. (5)
PLEASE INPUT EDIT NO. ? 4↵                          .. (6)
```

(1) PLEASE SELECT NO.?

Select "2. EDIT FILE" on the operation selection menu.

(2) *** SOURCE FILE(S) ***

Will display the function option document files on the current drive.

If no modifiable source exists, the following message is displayed and the program is terminated.

```
    FUNCTION OPTION DOCUMENT FILE IS NOT FOUND.
```

(3) PLEASE INPUT FILE NAME?

Enter a file name. Do not enter the extended part of the file name. If the function option document file (C262XXXF.DOC) is not in the current drive, an error message like the one below is output, prompting entry of other file name.

Example:

```
PLEASE INPUT FILE NAME  ? C2620N0↵
FUNCTION OPTION DOCUMENT FILE IS NOT FOUND.
```

(4) PLEASE INPUT USER'S NAME?

When modifying the customer's company name, enter a new name. The previously entered name may be used by pressing the RETURN key "↵".

(5) PLEASE INPUT ANY COMMENT

When modifying a comment, enter all the comment lines anew, beginning with the first line; comment data cannot be partially modified. Previously entered comment data can be used by pressing the RETURN key "↵". The input condition are the same as for new settings.

(6) PLEASE INPUT EDIT NO.?

Enter the number (1 to 16) of the function option to be modified, then start setting the option contents (See Section 3.5).

When selection of one option is complete, the system prompts entry of another function option number. Repeat selection until all options to be modified are selected.

If the "↵" key is pressed without entering a number, the option of the subsequent number can be selected.

Furthermore, if "T↵" is entered while the edit No. is ready for input, it will be possible to return to No.1 function option setting operation.

Enter "E↵" to end option setting. Then, move to the confirmation procedure for HEX file generation (See Section 3.6).

Example:

- When modifying the settings of the function option of No. 9

```
PLEASE INPUT EDIT NO.? 9↵
```

- To return to No. 1 function option setting

```
PLEASE INPUT EDIT NO.? T↵
```

- When ending setting

```
PLEASE INPUT EDIT NO.? E↵
```

3.5 Selecting Function Options

Selection procedure for function options are described below.

* Option selection is done interactively. For new settings, set Options 1–16 sequentially; to modify settings, the specified option number may be set directly.

* The selections for each option correspond one to one to the option list. While referring to the contents recorded in the option list, enter the selection number.

* In the message that prompts entry, the value in parentheses ( ) indicates the default value in case of new settings, or the previously set value in case of setting modification. This value is set when only the RETURN key "↵" is pressed.

* In examples, ↵ means press the RETURN key.

Selecting device type (OSC3 oscillator)

```
*** OPTION  NO.1 ***

--- DEVICE TYPE ---

        DEVICE TYPE          1. E0C6262  ( Type 3.0 V )
                             2. E0C62L62 ( Type 1.5 V )

PLEASE SELECT NO.(1) ? 1↵ *

        OSC3 OSCILLATOR      1. Ceramic
                             2. CR
                             3. Not Use

PLEASE SELECT NO.(1) ? 1↵

        DEVICE TYPE          1. E0C6262  ( Type 3.0 V )   SELECTED
        OSC3 OSCILLATOR      1. Ceramic                   SELECTED
```

* If "2" were selected, there is no option selection for "OSC3 OSCILLATOR".

Selecting VDE power supply

```
*** OPTION  NO.2 ***

--- EXTRA POWER SUPPLY VDE ---

        EX. POWER VDE           1. Not Use
                                2. Use K1,R1,P2,P3
                                3. Use All PAD

PLEASE SELECT NO.(1) ? 1↵

        EX. POWER VDE           1. Not Use   SELECTED
_
```

Selecting multiple key entry reset

```
*** OPTION  NO.3 ***

--- MULTIPLE KEY ENTRY RESET ---

        COMBINATION             1. Not Use
                                2. Use K00,K01
                                3. Use K00,K01,K02
                                4. Use K00,K01,K02,K03

PLEASE SELECT NO.(1) ? 1↵

        COMBINATION             1. Not Use   SELECTED
```

Selecting input interrupt noise rejector

```
*** OPTION  NO.4 ***

--- INPUT INTERRUPT NOISE REJECTOR ---

        K00-K03                 1. Use
                                2. Not Use

PLEASE SELECT NO.(1) ? 1↵

        K10-K13                 1. Use
                                2. Not Use

PLEASE SELECT NO.(1) ? 1↵

        K00-K03                 1. Use       SELECTED
        K10-K13                 1. Use       SELECTED
```

Selecting input ports pull up resistor

```
*** OPTION  NO.5 ***

--- INPUT PORTS PULL UP RESISTOR ---

        K00-K03                 1. With Resistor
                                2. Gate Direct

PLEASE SELECT NO.(1) ? 1↵

        K10-K13                 1. With Resistor
                                2. Gate Direct

PLEASE SELECT NO.(1) ? 1↵

        K00-K03                 1. With Resistor   SELECTED
        K10-K13                 1. With Resistor   SELECTED
```

Selecting output ports output specification (R00–R03)

```
*** OPTION  NO.6 ***

--- OUTPUT PORTS OUTPUT SPECIFICATION (R00-R03) ---

       R00-R03                  1. Complementary
                                2. Nch-OpenDrain

PLEASE SELECT NO.(1)? 1↵

       R00-R03                  1. Complementary   SELECTED
```

Selecting R10 specification

```
*** OPTION  NO.7 ***

--- R10 SPECIFICATION ---

        OUTPUT TYPE             1. D.C.
                                2. /SRDY Output

PLEASE SELECT NO.(1) ? 1↵

        OUTPUT SPECIFICATION    1. Complementary
                                2. Nch-OpenDrain

PLEASE SELECT NO.(1) ? 1↵

        OUTPUT TYPE             1. D.C.             SELECTED
        OUTPUT SPECIFICATION    1. Complementary    SELECTED
```

Selecting R11 specification

```
*** OPTION  NO.8 ***

--- R11 SPECIFICATION ---

        OUTPUT TYPE             1. D.C.
                                2. Fout  32768 [Hz]
                                3. Fout  16384 [Hz]
                                4. Fout   8192 [Hz]
                                5. Fout   4096 [Hz]
                                6. Fout   2048 [Hz]
                                7. Fout   1024 [Hz]
                                8. Fout    512 [Hz]
                                9. Fout    256 [Hz]

PLEASE SELECT NO.(1) ? 1↵

        OUTPUT SPECIFICATION    1. Complementary
                                2. Nch-OpenDrain

PLEASE SELECT NO.(1) ? 1↵

        OUTPUT TYPE             1. D.C.             SELECTED
        OUTPUT SPECIFICATION    1. Complementary    SELECTED
```

Selecting R12 specification

```
*** OPTION  NO.9 ***

--- R12 SPECIFICATION ---

        OUTPUT TYPE             1. D.C.
                                2. Buzzer Output

PLEASE SELECT NO.(1) ? 1↵

        OUTPUT SPECIFICATION    1. Complementary
                                2. Nch-OpenDrain

PLEASE SELECT NO.(1) ? 1↵

        OUTPUT TYPE             1. D.C.            SELECTED
        OUTPUT SPECIFICATION    1. Complementary   SELECTED
```

Selecting R13 specification

```
*** OPTION  NO.10 ***

--- R13 SPECIFICATION ---

        OUTPUT SPECIFICATION    1. Complementary
                                2. Nch-OpenDrain

PLEASE SELECT NO.(1) ? 1↵

        OUTPUT SPECIFICATION    1. Complementary   SELECTED
```

* If "Buzzer Output" were selected as the output type for R12 terminal, the output type for R13 terminal may be selected. Moreover, if "D.C." were selected as the output type for R12 terminal, the output type for R13 terminal will be fixed at DC output and hence there is no option selection for output type.

```
*** OPTION  NO.10 ***

--- R13 SPECIFICATION ---

        OUTPUT TYPE             1. D.C.
                                2. /Buzzer Output

PLEASE SELECT NO.(1) ? 1↵

        OUTPUT SPECIFICATION    1. Complementary
                                2. Nch-OpenDrain

PLEASE SELECT NO.(1) ? 1↵

        OUTPUT TYPE             1. D.C.            SELECTED
        OUTPUT SPECIFICATION    1. Complementary   SELECTED
```

Selecting I/O ports specification (P0, P1, P2)

```
*** OPTION  NO.11 ***

--- I/O PORTS OUTPUT SPECIFICATION ---

        P00-P03                 1. Complementary
                                2. Nch-OpenDrain

PLEASE SELECT NO.(1) ? 1↵

        P10-P13                 1. Complementary
                                2. Nch-OpenDrain

PLEASE SELECT NO.(1) ? 1↵

        P20-P23                 1. Complementary
                                2. Nch-OpenDrain

PLEASE SELECT NO.(1) ? 1↵

        P00-P03                 1. Complementary   SELECTED
        P10-P13                 1. Complementary   SELECTED
        P20-P23                 1. Complementary   SELECTED
```

Selecting I/O ports pull up resistor (P0, P1, P2)

```
*** OPTION  NO.12 ***

--- I/O PORTS PULL UP RESISTOR ---

        P00-P03                 1. With Resistor
                                2. Gate Direct

PLEASE SELECT NO.(1) ? 1↵

        P00-P03 RESISTANCE      1. 150K OHM
                                2.  50K OHM

PLEASE SELECT NO.(1) ? 1↵  *

        P10-P13                 1. With Resistor
                                2. Gate Direct

PLEASE SELECT NO.(1) ? 1↵

        P10-P13 RESISTANCE      1. 150K OHM
                                2.  50K OHM

PLEASE SELECT NO.(1) ? 1↵  *

        P20-P23                 1. With Resistor
                                2. Gate Direct

PLEASE SELECT NO.(1) ? 1↵

        P20-P23 RESISTANCE      1. 150K OHM
                                2.  50K OHM

PLEASE SELECT NO.(1) ? 1↵  *

        P00-P03                 1. With Resistor   SELECTED
        P00-P03 RESISTANCE      1. 150K OHM        SELECTED*
        P10-P13                 1. With Resistor   SELECTED
        P10-P13 RESISTANCE      1. 150K OHM        SELECTED*
        P20-P23                 1. With Resistor   SELECTED
        P20-P23 RESISTANCE      1. 150K OHM        SELECTED*
```

* These options may be selected if "1" (E0C6262) were selected for device type.

Selecting P30–P33 specification

```
*** OPTION  NO.13 ***

--- P30-33 SPECIFICATION --- *1

        INPUT/OUTPUT TYPE       1. I/O PORT
                                2. Serial I/O PORT

PLEASE SELECT NO.(1) ? 1↵

        OUTPUT SPECIFICATION    1. Complementary
                                2. Nch-OpenDrain

PLEASE SELECT NO.(1) ? 1↵

        INPUT/OUTPUT TYPE       1. I/O PORT         SELECTED
        OUTPUT SPECIFICATION    1. Complementary    SELECTED
```

Selecting P30–P33 pull up resistor

```
*** OPTION  NO.14 ***

--- P30-33 PULL UP RESISTOR --- *1

        P30-P33                 1. With Resistor
                                2. Gate Direct

PLEASE SELECT NO.(1) ? 1↵

        P30-P33 RESISTANCE      1. 150K OHM
                                2.  50K OHM

PLEASE SELECT NO.(1) ? 1↵ *2

        P30-P33                 1. With Resistor    SELECTED
        P30-P33 RESISTANCE      1. 150K OHM         SELECTED *2
```

*1 If either "2" or "3" (Use) were selected for VDE power supply, the terminals which may be set are P30–P32.

*2 This option may be selected if "1" (E0C6262) were selected for device type.

Selecting watchdog timer

```
*** OPTION  NO.15 ***

--- WATCHDOG TIMER ---

        WATCHDOG TIMER          1. Use
                                2. Not Use

PLEASE SELECT NO.(1) ? 1↵

        WATCHDOG TIMER          1. Use   SELECTED
```

Selecting serial interface data permutation

```
*** OPTION  NO.16 ***

--- SERIAL INTERFACE ---

        SERIAL INTERFACE        1. LSB First
                                2. MSB First

PLEASE SELECT NO.(1) ? 1↵

        SERIAL INTERFACE        1. LSB First   SELECTED
```

3.6 HEX File Generation and EPROM Selection

When setting function options setting is completed, the following message is output to ask the operator whether to generate the HEX file.

```
END OF OPTION SETTING.
DO YOU MAKE HEX FILE (Y/N) ? Y↵      ..... (1)

*** OPTION EPROM SELECT MENU ***

        1. 27C64
        2. 27C128
        3. 27C256
        4. 27C512

PLEASE SELECT NO. ? 3↵               ..... (2)

        2. 27C256   SELECTED
```

(1) DO YOU MAKE HEX FILE (Y/N) ?

When debugging the program with EVA6262, HEX file C262XXXF.HEX is needed, so enter "Y". If "N" is entered, no HEX file is generated and only document file C262XXXF.DOC is generated.

(2) PLEASE SELECT NO. ?

For the option ROM selection menu displayed when "Y" is entered in Step (1), select the EPROM to be used for setting EVA6262 options. This menu is not displayed when "N" is entered in Step (1).
One EPROM is required for setting function options (27C256 is selected in the above example).

When the above operation is completed, FOG6262 generates files. If no error is committed while setting segment options, the following message is output and the sequence returns to the operation selection menu.

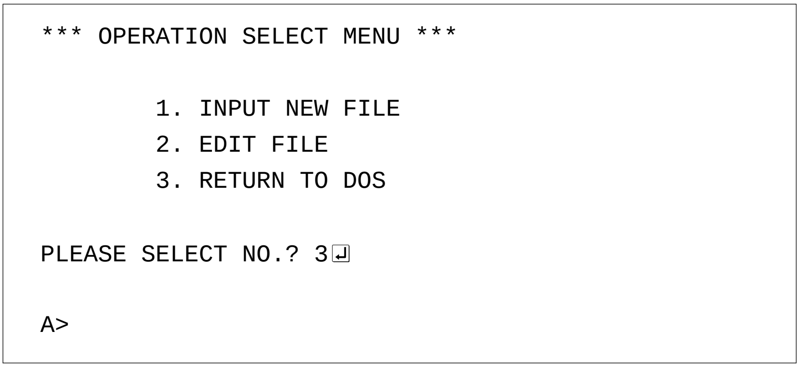
```
    MAKING FILE(S) IS COMPLETED.
```

Note *The EPROM to be mounted on the EVA6262 must satisfy the following conditions:*

EPROM for setting function options: $T_{ACC} \leq 250$ *ns*
(T_{ACC}*: Access time)*

3.7 End Procedure

This section explains how to end FOG6262 execution.



```
*** OPERATION SELECT MENU ***

        1. INPUT NEW FILE
        2. EDIT FILE
        3. RETURN TO DOS

PLEASE SELECT NO.? 3↵

A>
```

When a series of operations are complete, the sequence returns to the operation selection menu. Execution of FOG6262 can be ended by selecting "3. RETURN TO DOS" on this menu. If "1. INPUT NEW FILE" or "2. EDIT FILE" is selected, setting function options can be performed again.

FOG6262 can be forcibly terminated by pressing the "CTRL" and "C" keys together during program execution. (It is possible by pressing STOP key depending on the PC used.)

CHAPTER 4   SAMPLE FILES

• Function Option Document File

```
* E0C6262 FUNCTION OPTION DOCUMENT V 3.00
*
* FILE NAME    C2620A0F.DOC
* USER'S NAME
* INPUT DATE   90/11/16
*
* OPTION NO.1
* < DEVICE TYPE >
*     DEVICE TYPE       E0C6262  ( TYPE 3.0 V)  -----   SELECTED
*     OSC3 OSCILLATOR   CERAMIC  --------------------   SELECTED
 OPT0101 01
 OPT0102 01
 OPT0103 01
*
* OPTION NO.2
* < EXTRA POWER SUPPLY VDE >
*     EX. POWER VDE     NOT USE ---------------------   SELECTED
 OPT0201 01
 OPT0202 01
*
* OPTION NO.3
* < MULTIPLE KEY ENTRY RESET >
*     COMBINATION       NOT USE  --------------------   SELECTED
 OPT0301 01
*
* OPTION NO.4
* < INPUT INTERRUPT NOISE REJECTOR >
*     K00-K03           USE  ------------------------   SELECTED
*     K10-K13           USE  ------------------------   SELECTED
 OPT0401 01
 OPT0402 01
*
* OPTION NO.5
* < INPUT PORTS PULL UP RESISTOR >
*     K00-K03           WITH RESISTOR  --------------   SELECTED
*     K10-K13           WITH RESISTOR  --------------   SELECTED
 OPT0501 01
 OPT0502 01
*
* OPTION NO.6
* < OUTPUT PORTS OUTPUT SPECIFICATION (R00-R03) >
*     R00-R03           COMPLEMENTARY  --------------   SELECTED
 OPT0601 01
*
* OPTION NO.7
* < R10 SPECIFICATION >
*     OUTPUT TYPE               D.C.  ---------------   SELECTED
*     OUTPUT SPECIFICATION      COMPLEMENTARY  ------   SELECTED
 OPT0701 01
 OPT0702 01
*
* OPTION NO.8
* < R11 SPECIFICATION >
*     OUTPUT TYPE               D.C.  ---------------   SELECTED
*     OUTPUT SPECIFICATION      COMPLEMENTARY  ------   SELECTED
 OPT0801 01
 OPT0802 01
*
* OPTION NO.9
* < R12 SPECIFICATION >
*     OUTPUT TYPE               D.C.  ---------------   SELECTED
*     OUTPUT SPECIFICATION      COMPLEMENTARY  ------   SELECTED
 OPT0901 01
 OPT0902 01
*
* OPTION NO.10
* < R13 SPECIFICATION >
*     OUTPUT SPECIFICATION      COMPLEMENTARY  ------   SELECTED
 OPT1001 01
 OPT1002 01
*
* OPTION NO.11
* < I/O PORTS OUTPUT SPECIFICATION >
*     P00-P03                   COMPLEMENTARY  ------   SELECTED
*     P10-P13                   COMPLEMENTARY  ------   SELECTED
*     P20-P23                   COMPLEMENTARY  ------   SELECTED
 OPT1101 01
 OPT1102 01
 OPT1103 01
*
* OPTION NO.12
* < I/O PORTS PULL UP RESISTOR >
*     P00-P03                   WITH RESISTOR  ------   SELECTED
*     P00-P03 RESISTANCE        150K OHM  -----------   SELECTED
*     P10-P13                   WITH RESISTOR  ------   SELECTED
*     P10-P13 RESISTANCE        150K OHM  -----------   SELECTED
*     P20-P23                   WITH RESISTOR  ------   SELECTED
*     P20-P23 RESISTANCE        150K OHM  -----------   SELECTED
 OPT1202 01
 OPT1204 01
 OPT1206 01
*
* OPTION NO.13
* < P30-33 SPECIFICATION >
*     INPUT/OUTPUT TYPE         I/O PORT  -----------   SELECTED
*     OUTPUT SPECIFICATION      COMPLEMENTARY  ------   SELECTED
 OPT1304 01
 OPT1305 01
*
* OPTION NO.14
* < P30-33 PULL UP RESISTOR >
*     P30-P33                   WITH RESISTOR  ------   SELECTED
*     P30-P33 RESISTANCE        150K OHM  -----------   SELECTED
 OPT1406 01
*
* OPTION NO.15
* < WATCHDOG TIMER >
*     WATCHDOG TIMER            USE  ----------------   SELECTED
 OPT1501 01
*
* OPTION NO.16
* < SERIAL INTERFACE >
*     SERIAL INTERFACE          LSB FIRST  ----------   SELECTED
 OPT1601 01
*
*
* SEIKO EPSON’S AREA
*
*
 OPT2001 01
 OPT2002 01
 OPT2101 01
 OPT2102 01
\\END
```

Note End mark "\\END" may be used instead of "¥¥END" depending on the PC used. (The code of \ and ¥ is 5CH.)

• Function Option HEX File

```
:10000000F0F0F0F0F0F1F0F0F0F0F0F1F1F1F0F0EC
:10001000F0F0F0F0F0F0F1F1F0F0F0F1F0F0F0F0DD
:10002000F1F0F0F1F1F0F0F0F1F0F0F0F1F0F0F0CB
:10003000F0F1F1F1F0F0F0F0F1F1F1F0F1F0F1F1B7
:10004000F0F0F0F0F0F0F0F0F0F1F1F1F0F0F0F0AD
:10005000F0F0F0F0F1F1F1F1F1F1F1F1F0F0F0F395
:00000001FF
```

IV. *EVA6262 Manual*

CONTENTS

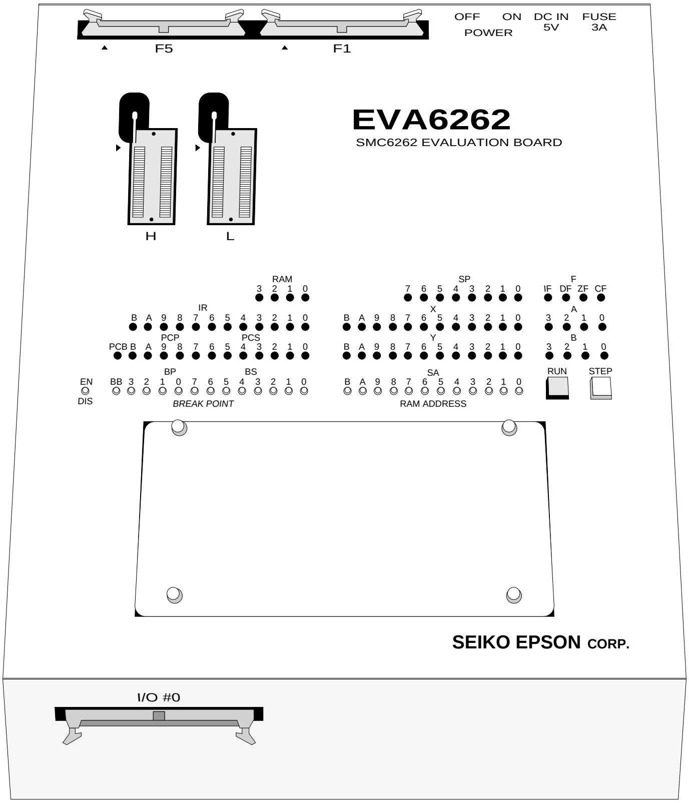
CHAPTER 1 INTRODUCTION

1.1 EVA6262 Outline

The EVA6262 is a debugging tool for the E0C6262, with various functions such as single step and program break. Almost the same functions that the E0C6262 CPU has can be implemented by writing application program and option data created by the option generator into EPROM, and installing it in the EVA6262.

Debugging and CPU monitoring can be done using the EVA6262 operation switches and LED indicators; therefore, debugging is possible with the EVA6262 alone.

In addition, the EVA6262 can interface with the ICE6200 in-circuit emulator, and so perform a higher level of debugging.



EVA6262

1.2 EVA6262 Components

When unpacking the EVA6262, check that the following goods are present:

(1)	EVA6262 main unit	1
(2)	I/O connection cable and connector (50-pin flat type)	1 set
(3)	Power cable (3-pin)	1 set
(4)	Diagnostic ROMs (two)	1 set
(5)	Fuse (3A)	1
(6)	EVA6262 Manual (this manual)	1
(7)	Warranty registration card	1
(8)	Warranty certificate	1
(9)	Notes for using	1

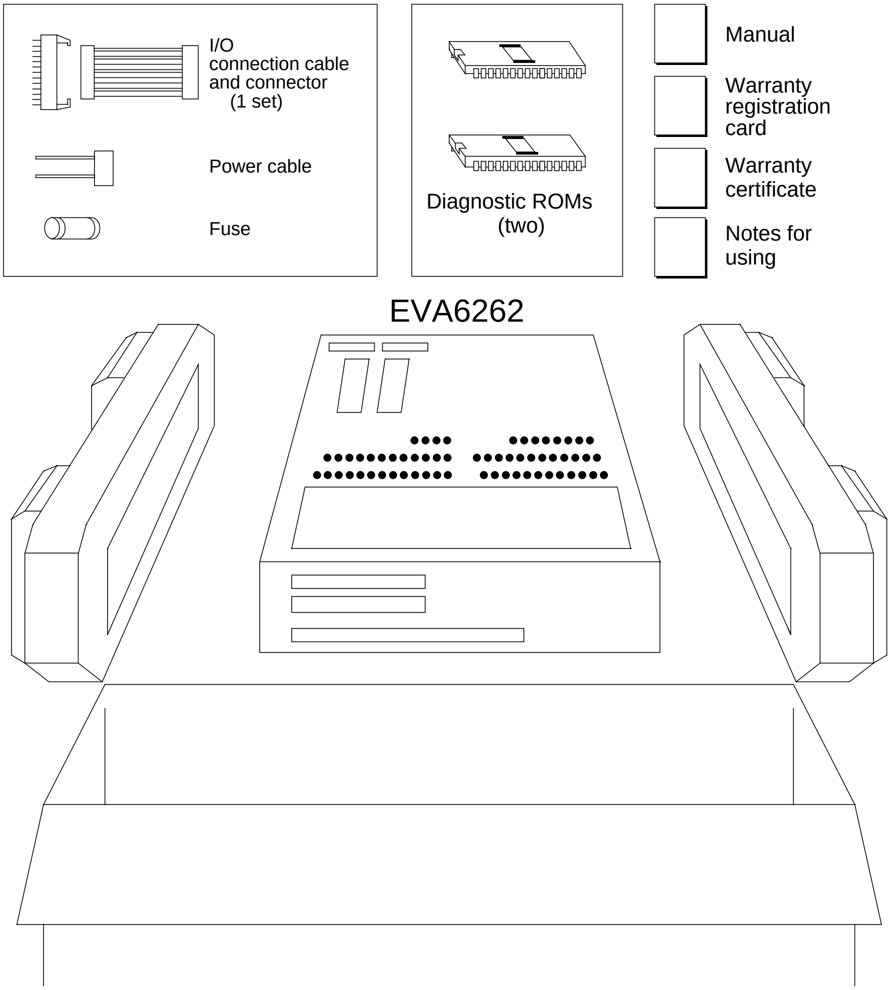


Fig. 1. 2.1
EVA6262
package

CHAPTER 2 PRECAUTIONS

Take the following precautions when using the EVA6262:

2.1 Precautions for Operation

- Turn the power of all equipment off before connecting or disconnecting cables.
- To turn the POWER switch of the EVA6262 off, then on again, wait for at least 10 seconds after turning off before turning on.
- When ROMs are inserted into the L and H ROM sockets, lock the lever securely by positioning it horizontally. After the ROMs have been removed from the sockets, lock the lever at the same position above. If the lever is left upright, poor contact may result.
- Confirm that the following ROMs have been installed correctly, then operate the EVA6262.

(Top panel)	Program ROM	2	L.HEX H.HEX
(Under top cover)	Function Option ROM	1	F.HEX

- If the EVA6262 does not operate normally, perform the operating test (Chapter 6).

2.2 Differences from Actual IC

There are some differences in functions between the EVA6262 and the actual IC.

I/O differences

- The response time has been changed by the differences in logic level (5 V for the EVA6262), output drive capability, and pull-up resistance. When creating key scan routines, especially, pay attention to the response time.

Power-on sequence differences

- The EVA6262 performs configuration and determines the internal state when the power is switched on. Then, it works as the IC does. Therefore, the I/O state of the EVA6262 is unstable until configuration has completed. This affects the power-on reset time.

Function differences

- The SVD function is implemented by varying the apparent power supply voltage with the VBLD control.
- The OSC1 crystal oscillation frequency is fixed at 32.768 kHz.
  For OSC3, ceramic oscillation (fixed at 1 MHz) or CR oscillation (1 MHz, slightry variable by changing the resistance) can be selected. Either OSC1 or OSC3 can be selected as the system clock.
- An external voltage is not applied to the VDE pin (P33) as the VDE power. (When pin P33 is set to VDE, its impedance becomes high.)
- The oscillation start and stop times are different from those of the IC. Because the logic level of EVA6262 is higher than it of actual IC.

CHAPTER 3 NAMES AND FUNCTIONS OF PARTS

This section describes the names and functions of the parts of the EVA6262.

3.1 Basic Functions

The EVA6262 has the following basic functions:

– **Program execution (Run function)**

Install the EPROM containing the application program and execute the program.

– **Single-step program operation (Single-step function)**

Programs may be run instruction by instruction to check the internal state of the CPU as it changes with each instruction.

– **Program execution suspension at a given address (Break function)**

A breakpoint may be set at an address at which it is desired to suspend program execution. After execution has stopped at the breakpoint, it can be restarted with the program run function.

– **Displaying program addresses and instruction codes during a break**

Program addresses and instruction codes may be displayed on the LED indicators.

- **Displaying the contents of RAM, registers, and flags during a break**

  The contents of RAM, the A, B, X, and Y registers, the stack pointers, and the flags may be displayed on the LED indicators during a break.

- **Interface with ICE6200**

  The EVA6262 can interface with the ICE6200 so that a higher level debugging environment may be established.

- **Setting hardware options by installing option ROM**

  Hardware options can be specified by writing option data created by the function option generator into EPROM, and installing the EPROM.

3.2 Operating Panel (Top view)

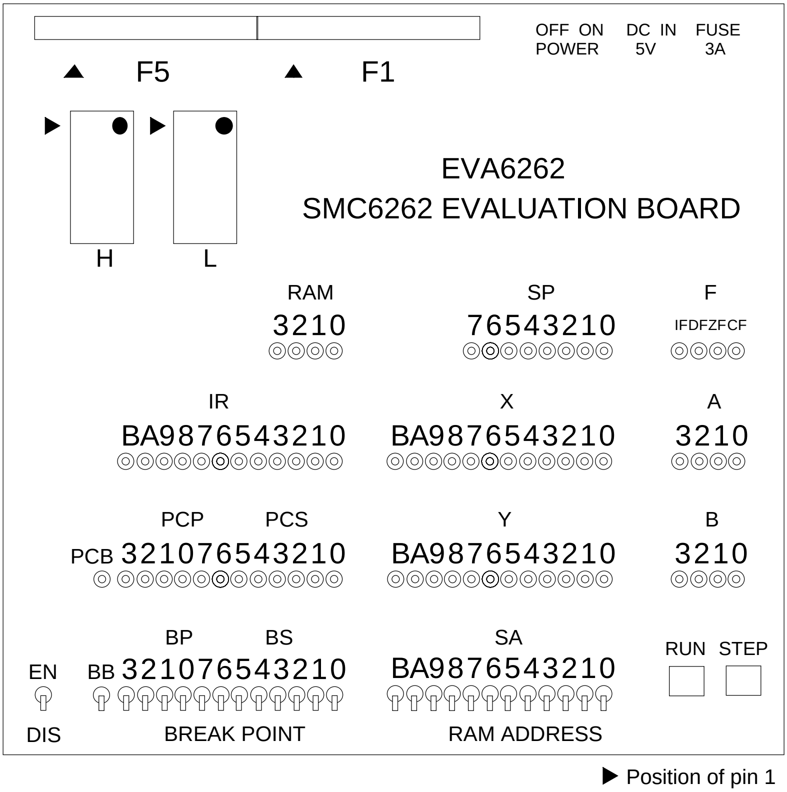


Fig. 3.2.1

Operating panel

Switches and keys

- **EN/DIS switch**
  This switch enables or disables the setting of breakpoints. When the switch is in the EN (Enable) position, the setting of breakpoints is enabled. When it is in the DIS (Disable) position, the setting of breakpoints is disabled. Normally, set the switch to the DIS position.

- **BREAK POINT switches (BB, BP, BS)**
  These switches set a breakpoint address at which program execution stops. BB, BP, and BS are switches that set the bank, page, and step, respectively, of the breakpoint address. When a switch is in the upper position, it represents "1"; when it is in the lower position, it represents "0".

The breakpoint address set with the BREAK POINT switches is valid when the EN/DIS switch is in the EN position. When the set address matches the current address of the program being executed, the program breaks, i.e., it stops immediately before executing the instruction at the current address. This function does not work when the EN/DIS switch is in the DIS position.

- **RAM ADDRESS switches (SA)**
  These switches are used to set RAM addresses and to check the contents of RAM after a program break. When a switch is in the upper position, it represents "1"; when it is in the lower position, it represents "0". The contents of the address set with these switches are displayed on the RAM display LEDs.

- **RUN key**
  This key restarts the program after a break. When it is pressed, the program continues, starting with the instruction at the break address.

- **STEP key**
  When this key is pressed, the program breaks immediately. If the key is pressed during a break, the instruction step at the break address is executed, and the program breaks again. Thus, the program can be executed step by step.

LEDs The internal state of the CPU is indicated by the LEDs. An LED lit indicates "1"; an LED not lit indicates "0".

- **RAM (3210)**
  The contents of the RAM address, which are fixed by the RAM ADDRESS switch, are displayed.

- **IR (BA9876543210)**
  The instruction at the current address is displayed. If the program has stopped at a breakpoint, the instruction is displayed before execution.

- **PCB**
  The bank address is displayed. The contents of the bank address, which are fixed by the BB switches, are displayed if the program has stopped at a breakpoint.

- **PCP (3210)**
  The page address is displayed. The contents of the page address, which are fixed by the BP switches, are displayed if the program has stopped at a breakpoint.

- **PCS (76543210)**
  The step address is displayed. The contents of the step address, which are fixed by the BS switches, are displayed if the program has stopped at a breakpoint.

- **SP (76543210)**
  The value of the stack pointer is displayed.

- **X (BA9876543210)**
  The contents of the X index register are displayed.

- **Y (BA9876543210)**
  The contents of the Y index register are displayed.

- **F/IF**
  The state of the interrupt flag is displayed.

- **F/DF**
  The state of the decimal flag is displayed.

- **F/ZF**
  The state of the zero flag is displayed.

- **F/CF**
  The state of the carry flag is displayed.

- **A (3210)**
  The contents of the A register are displayed.

- **B (3210)**
  The contents of the B register are displayed.

ROM sockets

- **L (low) and H (high)**
  These are IC sockets for target program ROMs. Insert the ROM (L.HEX) containing the 8 low-order bits (I7 to I0) of the machine code into the L socket, and the ROM (H.HEX) containing the 4 high-order bits (IB to I8) into the H socket.
  Insert the diagnostic ROM into a socket when an operation test is performed.

Connectors

- **F1 and F5**
  Connectors for the ICE6200 interface cable.

3.3 Under Top Cover (Top view after removal Top-cover)

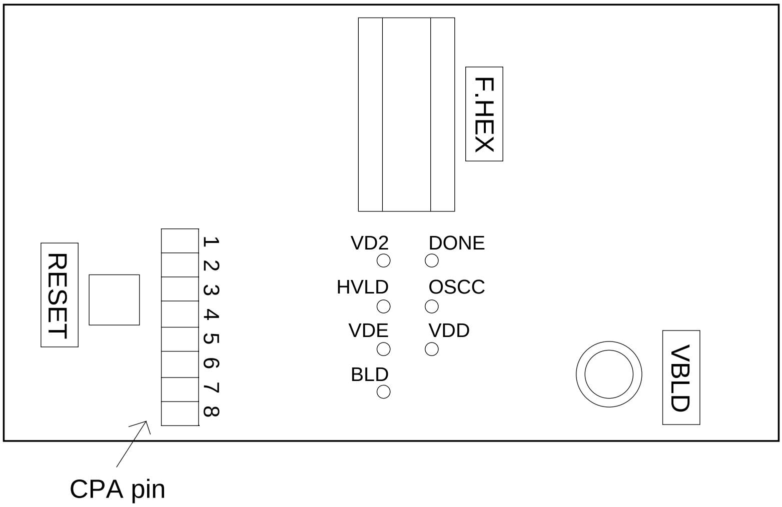


Fig. 3.3.1

Under top cover

- **RESET switch**
  This switch resets the CPU and starts the target program from bank 0, page 01H, step 00H.

- **VBLD**
  This is the control for varying the power supply voltage in simulation to check SVD operation.
  (Refer to section "2.2 Differences from Actual IC".)

- **HVLD, BLD, OSCC, VDE**
  These are LEDs to indicate the values ("1" or "0") of the HLMOD, SVDC, and OSCC registers, or select extra power VDE, respectively.

HVLD:	On while the HLMOD register (address FAH, D3) contains "1"; off while the register contains "0".
BLD:	On while the SVDC register (address FAH, D1) contains "1"; off while the register contains "0".

OSCC: On while the OSCC register (address E9H, D1) contains "1"; off while the register contains "0".

VDE: On while the extra power supply from P33; off while no select extra power from P33.

Take care of these registers or option switch for the power save.

- **VDD, VD2**

  These are LEDs to indicate the VD1 is generated by VDD or VD2.

  VDD: On while the VD1 is generated by VDD.

  VD2: On while the VD1 is generated by VD2.

- **DONE**

  This LED lights when the EVA6262 has completed configuration at power-on and is ready for debugging. If this LED is not lit several seconds after power-on, switch the power off and then on again.

- **F.HEX (ROM sockets)**

  This is the IC socket into which the ROM is inserted. This ROM includes the function options generated by option generator.

- **CPA pin**

  The values of the HLMOD, SVDC, and OSCC registers and the DONE, VDE, VDD, and VD2 signals can be checked by an oscilloscope or other instrument. The correspondence between the pins, registers, and the signals is shown below.

  Pin 1: No connection
  Pin 2: VDE
  Pin 3: VD2
  Pin 4: HLMOD
  Pin 5: OSCC
  Pin 6: VDD
  Pin 7: DONE
  Pin 8: SVDC

3.4 Front Panel

There is a connector on the front panel for connecting the EVA6262 to the target system.

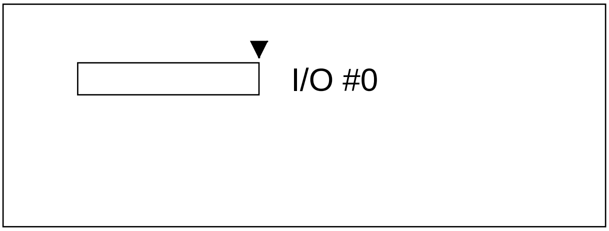


Fig. 3.4.1
Front panel

▼ Position of pin 1

- **I/O #0**
  Connector for the I/O cable. The I/O cable is used to connect the EVA6262 to the target system.

3.5 Rear Panel

The external power input section is on the rear panel.

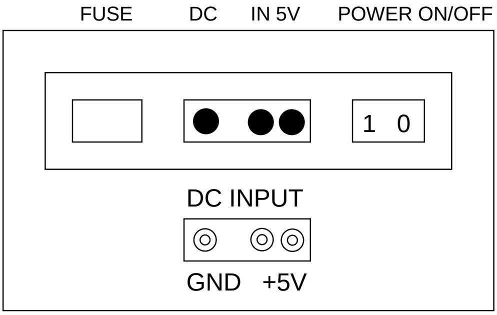


Fig. 3.5.1

Rear panel

- **POWER switch (on/off)**
  This is a switch to turn on or off the external power supply to EVA6262. (Please turn off the POWER switch when ICE6200 is connected.)

- **FUSE**
  This is 3A of the 3A-tubular fuse for external power supply, and is blown off by current of 3A or more.

- **DC IN 5 V**
  This is a connector with external power supply source. The external power supply should be in direct current of 5V for 3A or more.

Note *Be sure to disconnect external power source before connection with ICE6200, because power is supplied from ICE6200 when you connect EVA6262 to ICE6200.*

CHAPTER 4 CABLE CONNECTION

This section describes how to connect the power cable to the EVA6262, and the EVA6262 to the ICE6200 and the target system.

Note *Turn the power of all equipment off before connecting or disconnecting cables.*

4.1 Connection to ICE6200

The EVA6262 is connected to the ICE6200 by connecting the two interface cables (F1 and F5). Use EVA6262 connectors F1 and F5 with the projections facing outwards. Use ICE6200 connectors F1 and F5 with the projections facing inwards (cable side).
Figures 4.1.1 and 4.1.2 show the external view and connection diagram of the ICE6200 interface cable.

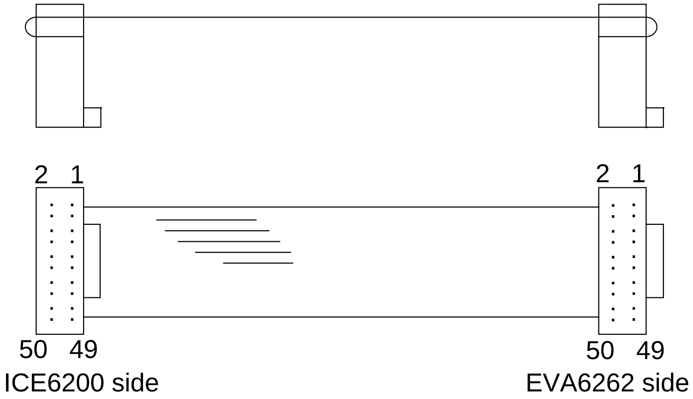


Fig. 4.1.1
External view of the ICE6200 interface cable

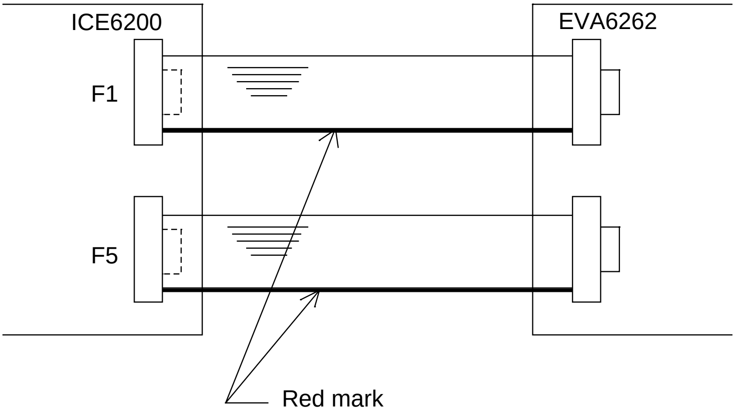


Fig. 4.1.2
Connection diagram

Note *The EVA6262 has an external power input connector for +5 V (V_{DD}) and GND (V_{SS}). Leave these connectors unconnected when the EVA6262 is connected to the ICE6200.*

4.2 Power Cable Connection

When using the EVA6262 on its own, it must be supplied with power (5V DC, 3A or more) from an external source through the power cable.

When the EVA6262 is connected to the ICE6200, power is supplied by the ICE6200; therefore, the power cable is not necessary. Disconnect the power cable if it is already connected.

Figure 4.2.1 shows the connection of the power cable pins.

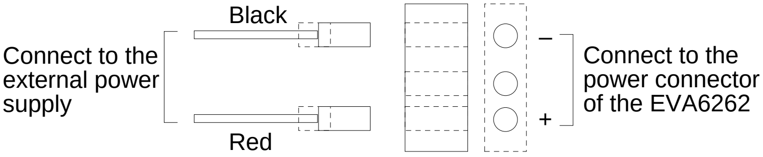

Black
Connect to the external power supply
–
+
Connect to the power connector of the EVA6262
Red

Fig. 4.2.1
Connection of power cable pins

4.3 Connection to Target System

The I/O #0 connector is used to connect the EVA6262 to the target system.

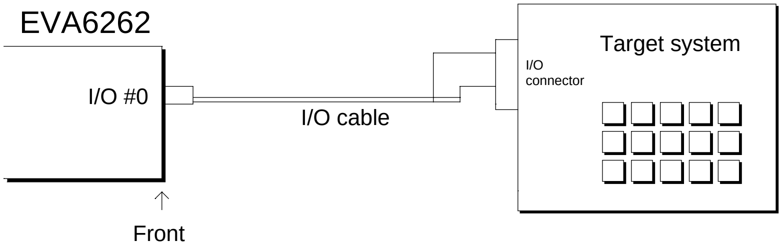


Fig. 4.3.1
Connection of target system

Take the following precautions when connecting the EVA6262 to the target system:

- Power is supplied to the EVA6262, unlike the chips. (See the "E0C6262 Technical Manual".)

- Do not use any pins that cannot be connected.

Table 4.3.1 lists the pins of the I/O #0 connector.

Table 4.3.1
I/O #0 connector pins

Pin No.	Signal Name	Pin No.	Signal Name
1	VDD (+5 V)	2	VDD (+5 V)
3	VDD (+5 V)	4	VDD (+5 V)
5	Cannot be connected	6	Cannot be connected
7	P00	8	P01
9	P02	10	P03
11	P10	12	P11
13	P12	14	P13
15	P20	16	P21
17	P22	18	P23
19	P30	20	P31
21	P32	22	P33
23	Cannot be connected	24	Cannot be connected
25	R00	26	R01
27	R02	28	R03
29	K00	30	K01
31	K02	32	K03
33	K10	34	K11
35	K12	36	K13
37	R10	38	R11
39	R12	40	R13
41	Cannot be connected	42	Cannot be connected
43	$\overline{\text{TEST}}$	44	$\overline{\text{RESET}}$
45	Cannot be connected	46	Cannot be connected
47	VSS (GND)	48	VSS (GND)
49	VSS (GND)	50	VSS (GND)

Note Do not use the pins that cannot be connected.

CHAPTER 5 OPERATION METHOD OF EVA6262

5.1 Preparation

This section describes the common preparation work necessary when the EVA6262 is used by itself and when it is connected to the ICE6200. Connection method, refer to Chapter 4 "CABLE CONNECTION".

Check the EVA6262 operation by mounting the supplied diagnostic ROMs as instructed in Chapter 6. It is recommended that this test be performed periodically.

Before doing the following, be sure to turn the POWER switch of the EVA6262 off.

Creation of target system

Mount the keys, and switches on the board to build a target system. Use the I/O connector supplied with the EVA6262 to connect the EVA6262 to the target system. (For the pin layout of connector, see Table 4.3.1 in Chapter 4.)

Note that there is some difference in specifications between the EVA6262 and the actual CPU. Refer to the "2.2 Differences from Actual IC" when building a target system.

Creation and installation of ROMs

Create the program ROMs and option ROM, and insert them into the sockets of the EVA6262. When the EVA6262 is delivered, the option ROM for a diagnostic program is already installed. Replace it with the created ROM.

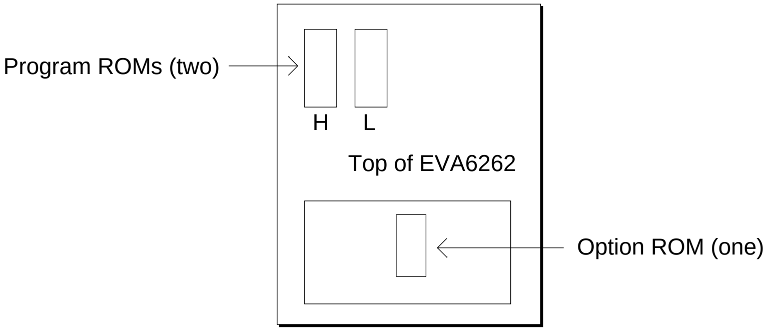


Fig. 5.1.1
Installation of ROMs

- **Program ROMs (two)**
  The program ROMs contain the application program machine code. Write the HEX files output by the ASM6262 cross-assembler into EPROMs to create program ROMs. Since two HEX files containing the high-order section (C262YYYH.HEX) and the low- order section (C262YYYL.HEX) of the machine code are output, two ROMs are created. Insert H.HEX into socket H and L.HEX into socket L on the top panel. These ROMs are not necessary when connecting the EVA6262 to the ICE6200.

- **Function Option ROM (one)**
  The function option ROM is used to specify function options, such as I/O ports. Create the function option ROM from the function option HEX file (C262YYYF.HEX) output by the function option generator, and insert it into the ROM socket (F.HEX) in the top cover.

- **EPROM specifications**
  Use EPROMs with the following specifications:

Program ROM:	27C64 to 27C512(250 ns or less access time)
Option ROM:	27C64 to 27C512(250 ns or less access time)

5.2 Independent Use of EVA6262

This section describes operation when using the EVA6262 by itself.

The EVA6262 may be used independently by connecting a power supply to it. Use a 5V DC regulator (more than 3A) as an external power supply. Connect it with the correct polarity (+ and -).

(Refer to the "4.2 Power Cable Connection".)

Power on/off

Before turning the POWER switch of the EVA6262 on, confirm the following:

1) The power cable is connected correctly.
2) The target system is connected correctly.
3) The three ROMs have been installed correctly.

After confirming the above items, turn the POWER switch of the EVA6262 on using the following procedure:

1) Turn the regulator on. If the regulator is of the variable-voltage type, set the output voltage to 5 V.
2) Turn the POWER switch of the EVA6262 on.

Note *To turn the POWER switch of the EVA6262 off, then on, turn it off, wait for 10 seconds or more, and then turn it on.*

After the POWER switch of the EVA6262 has been turned on, the DONE LED (green) on the top cover lights after several seconds to indicate that debugging may proceed. If the DONE LED is still off 10 seconds or more after the POWER switch has been turned on, do the following:

1) Turn the POWER switch of the EVA6262 off.
2) Confirm that the ROMs have been installed properly, and the cables connected properly.
3) Check the fuse.
4) Turn the POWER switch of the EVA6262 on.

If the DONE LED still does not light, do a self-diagnosis. For the self-diagnosis method, refer to the Chapter 6.

Debugging

When the EVA6262 is used alone, it provides the following debugging functions. The method of operation is given below.

- **Program free run**

  When the RESET switch (on the top cover) is pressed, the EVA6262 enters the program run state, and executes the application program from bank 0, page 1, step 0. Before pressing the RESET switch after the power to the EVA6262 has been switched on, make sure that the DONE LED is lit.

- **Program break**

  The program may be stopped at the address set by the BREAK POINT switches. This function is valid when the EN/DIS switch is in the EN position. The program stops at the program address where the breakpoint is set. It stops before the instruction at the breakpoint is executed. The program may be stopped by pressing the STEP key.

  When the program is stopped, the LED indicators for the internal state of the CPU show the current state. So debug by checking this state against the program.

  To restart the program after a break, set the next breakpoint, and press the RUN key.

  The single-step operation (described below) can be performed by pressing the STEP key instead of the RUN key.

- **Single step**

  By pressing the STEP key after a program break, the one instruction at the current address can be executed, and the program stopped at the next address (program break). Using this function, the program run state can be confirmed.

– **Other functions**

The operation of the SVD can be confirmed with the VBLD control.

The HLMOD, SVDC, and OSCC register values, and status of the VDE extra power supply selection, and VD1 is generated by VDD or VD2 can be confirmed with the LEDs displays and CPA pins.

5.3 Operation When ICE6200 is Connected

This section explains the operation and use of the EVA6262 when it is connected to the ICE6200.

Set up the EVA6262 as follows when it is connected to the ICE6200:

1) Do not connect the power supply.
2) Keep on turning the POWER switch off.
3) Set all the switches on the operation panel to their lower positions.

Power on/off

Power to the EVA6262 is supplied by the ICE6200, and the power is switched on and off by pressing the POWER switch of the ICE6200. Keep the POWER switch of the EVA6262 off.

Note *To turn the POWER switch of the ICE6200 off, then on, turn it off, wait for 10 seconds or more, and then turn it on.*

After the POWER switch of the ICE6200 has been turned on, the DONE LED (green) on the top cover of the EVA6262 lights after several seconds to indicate that debugging may proceed. If the DONE LED is still off 10 seconds or more after the POWER switch has been turned on, do the following:

1) Turn the POWER switch of the ICE6200 off.
2) Confirm that the circuit breaker of the ICE6200 is on.
3) Confirm that the ROMs have been installed properly and the cables connected properly.
4) Turn the POWER switch of the ICE6200 on.

If the DONE LED still does not light, do a self-diagnosis. For the self-diagnosis method, refer to the Chapter 6.

Debugging

Debugging is done with the host computer, and the EVA6262 is controlled by the ICE6200. For the method of operation, refer to the "E0C6262 ICE Operation Manual". The EVA6262 can control the following three functions:

1) Pseudo power supply voltage change with the VBLD control
2) RESET switch

The other switches and LEDs are invalid. Do not operate the switches of the EVA6262 side. The switches do not function when the target program ROM is installed.

CHAPTER 6 OPERATING TEST

Self-diagnosis of the EVA6262 can be performed with the following operating tests. To perform these tests, the option ROM and two program ROMs (supplied) are required. If these ROMs have not been installed, insert them into the sockets. To use the EVA6262 independently, connect the external power supply (5V DC, 3A).

This test checks the start and single-step operations and the program break function of the EVA6262 in three stages (1 to 3). If the EVA6262 does not operate normally, check whether the ROMs have been installed correctly and whether the external power is being input correctly. Then perform the test again from stage 1. If the test repeatedly fails after being retried several times, the EVA6262 is faulty.

<Stage 1> In stage 1, check whether the DONE LED lights correctly.

(1) Open the top cover.

(2) Confirm that the POWER switch is off.

(3) Insert the diagnostic ROMs (H.HEX and L.HEX) into sockets H and L (the sockets into which program ROMs are normally inserted) on the panel. Confirm that the option ROM has been installed properly.

(4) Set the EN/DIS switch to the DIS position.

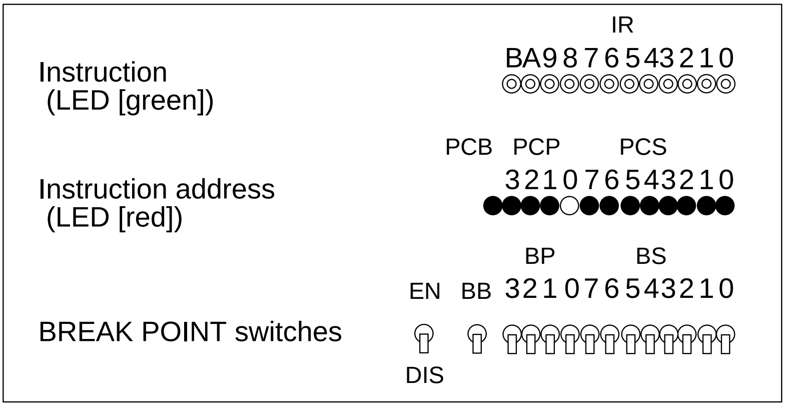
(5) Turn the POWER switch on. If the POWER switch has just been turned off, wait for at least 10 seconds before turning it on.

(6) Check whether the DONE LED (green) under the top cover lights correctly. After confirming that the DONE LED lights correctly, go to stage 2.

(7) If the LED does not light within 10 seconds of the power being switched on, turn the POWER switch off, and check the fuse and external power connector. Wait for at least 10 seconds, then perform the test again.

<Stage 2> In stage 2, single step operations are checked.

(1) Hold down the RESET switch under the top cover. Confirm that the instruction address LEDs (only PCP 0) light.



○: ON
●: OFF

(2) When the RESET switch is released, the LED blinks.

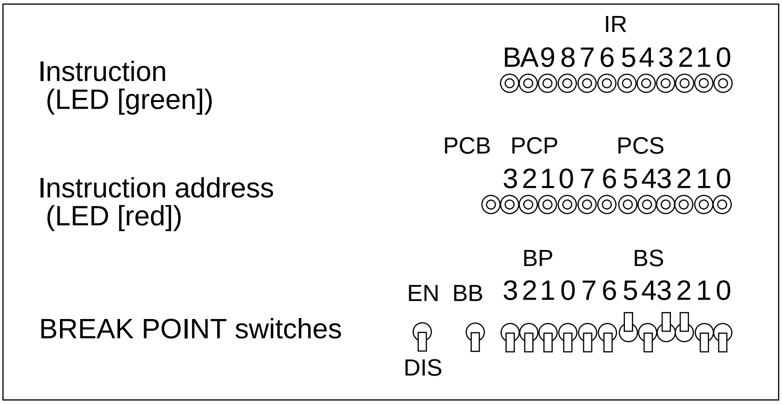
(3) When the STEP switch is pressed once lightly, the LED stops blinking.

(4) When the STEP switch is pressed again, the CPU performs a single-step operation. At the same time, the LED stops blinking. Press the STEP switch several times to confirm that single-step operations are performed properly, then go to stage 3.

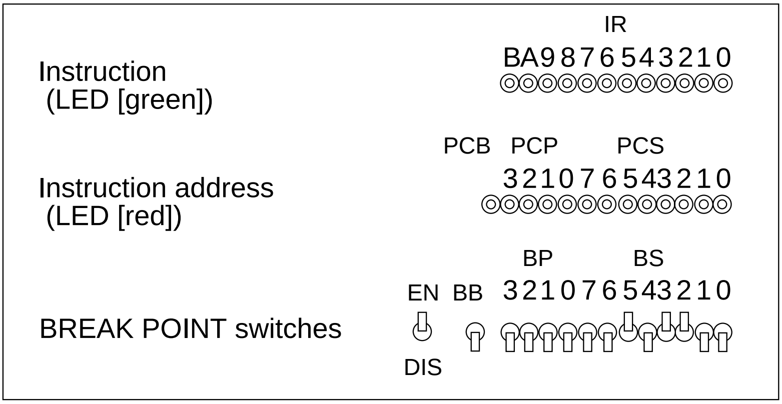
(5) When the RUN switch is pressed, the test returns to step 2 in stage 2.

<Stage 3> In stage 3, the setting of breakpoints is checked.

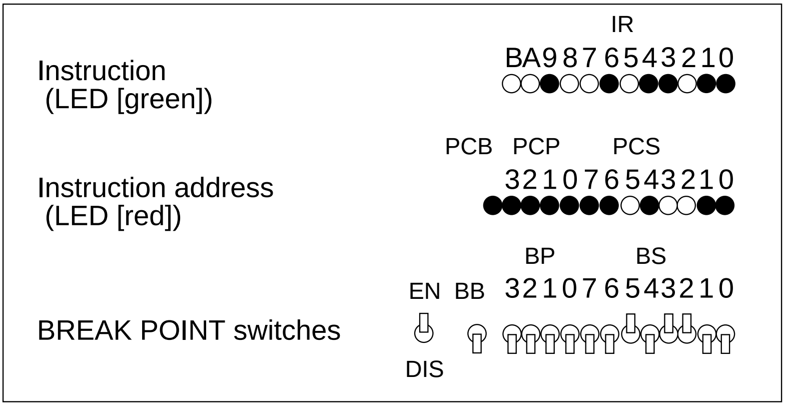
(1) Set the BREAK POINT switches as follows:



(2) Set the EN/DIS switch to the EN position.



(3) When the value indicated by PCP and PCS matches the setting of the BREAK POINT switches, the CPU stops, and the LEDs light as follows:



(4) When the RUN switch is pressed again, the test returns to step 3.

(5) Turn the POWER switch off.

Operating test is now complete. If the test ends normally, the basic functions of the EVA6262 are normal.

CHAPTER7 PRODUCT SPECIFICATIONS

The components specifications of the EVA6262 are listed below.

1. EVA6262

Dimensions:	(width) (depth) (height) 325 mm × 382 mm ×105 mm	(Including rubber feet)
Weight:	About 6 kg	(main unit only)
Color:	Cygnus white	
Power supply:	5V DC, 3A or more When connected to the ICE6200, power is supplied by the ICE6200.	(from external power supply)
Board:	Main board × 1 Sub board × 1	

2. ICE6200 interface cable (supplied with ICE6200)

EVA6262 connector:	3433-6002LCSC or equivalent	
Cable connector:	3425-6500SC	
Cable:	50-pin flat cable × 2	(Two cables are the same)
Interface:	CMOS interface	(5V)
Length:	About 50 cm	(Two cables are the same)

3. I/O cable (supplied with EVA6262)

EVA6262 connector:	3433-5002LCSC or equivalent	
Cable connector:	3425-6500SC	
Cable:	50-pin flat cable × 1	
Interface:	CMOS interface	(5V)
Length:	About 50 cm	

4. Power cable (supplied with EVA6262)

EVA6262 connector:	MOLEX 5276-03A or equivalent
Cable connector:	MOLEX 5196-03
Other side connector:	(According to power supply specifications)
Cable length:	About 80 cm
Capacity:	5V DC, 3A or more

5. Accessories

Fuse Type/rating:	MGC-ULCSA 250V 3A × 1	
50-pin connector:	For connecting to target system 3433-6002LCSC × 1	 (For I/O cable connection)

6. EPROM

For programs:	Intel i27C64–i27C512 or equivalent (Access time 250 ns or less)
For options:	Intel i27C64–i27C512 or equivalent (Access time 250 ns or less)

V. *E0C6262 ICE Operation Manual*

Chapter 2 and subsequent chapters provide information common to all E0C62 Family models, the model name being denoted "XX". Read this manual, replacing "XX" with "62".

$62\underline{XX} \rightarrow 62\underline{62}$

CONTENTS

CHAPTER 1 E0C6262 RESTRICTIONS

1.1 ROM Area

The ROM area is limited to a maximum address of 07FFH. Assigning data above the 07FFH address causes an error.

1.2 RAM Area

The RAM area is limited to a maximum address of 0FFH. Assigning data above the 0FFH address causes an error. However, as the following addresses are in the unused area, designation of this area with the ICE commands produces an error.

080H to 0CFH
0DBH to 0DFH
0EBH to 0EFH
0FBH to 0FFH

0D0H–0DAH, 0E0H–0EAH and 0F0H–0FAH becomes I/O memory.
(See "E0C6262 Technical Manual" for details.)

1.3 Undefined Code

The instructions below are not specified for E0C6262 and so cannot be used.

```
SLP
PUSH    XP          PUSH    YP
POP     XP          POP     YP
LD      XP,r        LD      YP,r
LD      r,XP        LD      r,YP
```

CHAPTER 2 ICE6200 SPECIFICATIONS

2.1 Features

The ICE6200 is a microcomputer software development support tool that increases the efficiency of software development for the E0C62 Family of 4-bit single chip microcomputers.

The ICE6200 and the E0C62 Family evaluation board EVA62XX, when used in combination, provide an exceptionally powerful hardware and software development support environment.

The following flow chart shows the creation sequence of the single chip microcomputer system from development through mass production.

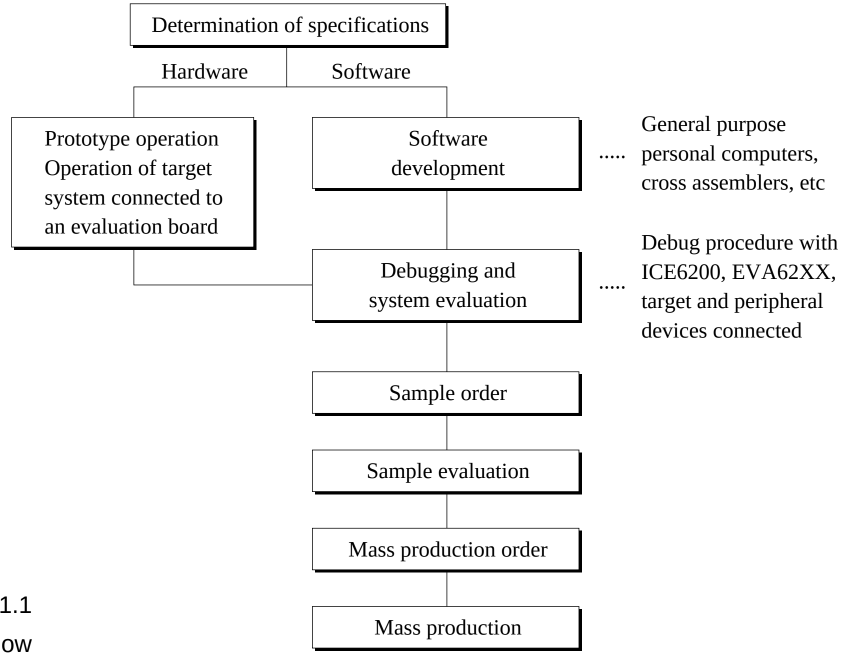

Determination of specifications
Hardware
Software
Prototype operation
Operation of target system connected to an evaluation board
Software development
General purpose personal computers, cross assemblers, etc
Debugging and system evaluation
Debug procedure with ICE6200, EVA62XX, target and peripheral devices connected
Sample order
Sample evaluation
Mass production order
Mass production

Fig. 2.1.1
Development flow

Use of the ICE6200 and EVA62XX can greatly shorten the development process time required for debugging and system evaluation procedures.

Refer to "E0C62 Family Technical Guide" to get more detailed information about "Sample order" to "Mass production" above mentioned flow chart.

Description

A description of the ICE6200 follows.

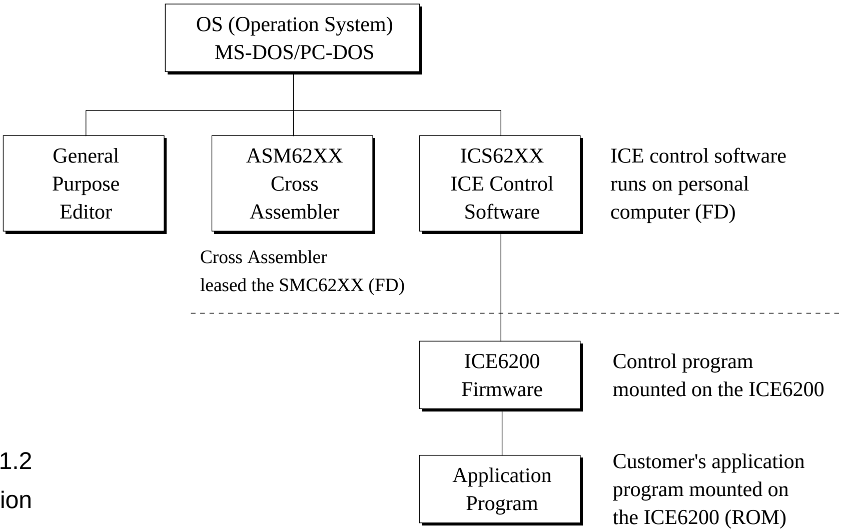
(1) The ICE6200 operates by connecting to a general purpose personal computer (NEC PC-9801V Series, IBM PC/XT, PC/AT). The debugging environment is constructed by the user's personal computer acting as the host system.

(2) High-performance emulation commands are provided. A variety of commands are supplied, such as a register value implemented break function, on-the-fly data display, history display, and other high-level functions.

(3) The ICE6200 is equipped with a special power supply. This power source supplies VDD to the evaluation board, making additional power supply from the user side unnecessary.

(4) The ICE6200 can also be used to analyze hardware. Hardware debugging is supported through the SYNC and HALT terminals.

Software configuration



OS (Operation System)
MS-DOS/PC-DOS
General Purpose Editor
ASM62XX Cross Assembler
ICS62XX ICE Control Software
ICE control software runs on personal computer (FD)
Cross Assembler leased the SMC62XX (FD)
ICE6200 Firmware
Control program mounted on the ICE6200
Application Program
Customer's application program mounted on the ICE6200 (ROM)

Fig. 2.1.2
Software configuration

Function table

Table 2.1.1 shows the functions supported by the ICE6200.

Table 2.1.1 ICE6200 functions

Item number	Item	Brief description of function	Comments
1	Real-time break	The target program is interrupted under optional conditions (1) Break by program counter (PC) (2) RAM address, data, R/W break (3) Break by register value (4) Break via a combination of items (1)–(3) (AND, OR) (5) Forced break by RESET or BREAK switch settings (6) Forced break by host system Escape key input	
2	History	EVA62XXCPU data collection during emulation (1) Collection of PC, instruction code, RAM R/W, or CPU register values (2) Approx. 2048 instruction bus data collections (3) Collects information up to the hit of break condition, or before or after the hit (4) Collects history information within the specified program area (5) Searches for history information	
3	Real-time execution	Target program is run in real time at frequencies up to 4MHz	
4	Real-time measurement	Emulation run in real time (up to approx. 425ms) or step number count	
5	Target memory referenced or modified	(1) ICE packaged target program memory is referenced, modified, or dumped (2) Target program memory-mapped I/O is referenced or modified (3) Internal CPU registers are referenced or modified	
6	Trace	Target program is executed step by step and register contents are displayed	
7	Assemble/ Disassemble	Mnemonic input is converted to machine language and stored in program memory; contents of memory are disassembled	
8	FD loaded, saved or verified	(1) Data from FD is loaded to the program or verified (2) Program data is saved to FD (3) ICE interim results are loaded or saved to FD (4) Data from FD memory is loaded, saved or verified	
9	ROM read or verify	Program is loaded to program memory from the ICE ROM socket and verified	
10	Execution supervision	During G command execution, the program counter and halt state are displayed	
11	Coverage	Acquire coverage information	
12	Other	(1) Printer start and stop (2) ICE command display (3) Evaluation board CPU reset (4) Evaluation board CPU status on LED display (5) Execution with SYNC pulse output at breakpoint, but without break (6) 2764 to 27512 EPROM (target) support (7) ICE6200 hardware check	

Function-differentiated command list

Tables 2.1.2.a–b show the function-differentiated command list for the ICE6200.

Table 2.1.2.a Function-differentiated command list

Item number	Function	Command configuration	Description of operation	Reference page
1	Assemble	#A,a ↵	Assemble command mnemonic code and store at address "a"	V-50
2	Disassemble	#L,a1,a2 ↵	Contents of addresses a1 to a2 are disassembled and displayed	V-32
3	Dump	#DP,a1,a2 ↵	Contents of program area a1 to a2 are displayed	V-34
		#DD,a1,a2 ↵	Content of data area a1 to a2 are displayed	V-36
4	Fill	#FP,a1,a2,d ↵	Data d is set in addresses a1 to a2 (program area)	V-52
		#FD,a1,a2,d ↵	Data d is set in addresses a1 to a2 (data area)	V-53
5	Set Run Mode	#G,a ↵	Program is executed from the "a" address	V-72
		#TIM ↵	Execution time and step counter selection	V-93
		#OTF ↵	On-the-fly display selection	V-94
6	Trace	#T,a,n ↵	Executes program while displaying results of step instruction from "a" address	V-75
		#U,a,n ↵	Displays only the final step of #T,a,n	V-77
7	Break	#BA,a ↵ #BAR,a ↵	Sets Break at program address "a" Breakpoint is canceled	V-64
		#BD ↵ #BDR ↵	Break condition is set for data RAM Breakpoint is canceled	V-65
		#BR ↵ #BRR ↵	Break condition is set for EVA62XXCPU internal registers Breakpoint is canceled	V-66
		#BM ↵ #BMR ↵	Combined break conditions set for program data RAM address and registers Cancel combined break conditions for program data ROM address and registers	V-68
		#BRES ↵	All break conditions canceled	V-71
		#BC ↵	Break condition displayed	V-70
		#BE ↵	Enter break enable mode	V-78
		#BSYN ↵	Enter break disable mode	V-78
		#BT ↵	Set break stop/trace modes	V-79
		#BRKSEL,REM ↵	Set BA condition clear/remain modes	V-80
8	Move	#MP,a1,a2,a3 ↵	Contents of program area addresses a1 to a2 are moved to addresses a3 and after	V-54
		#MD,a1,a2,a3 ↵	Contents of data area addresses a1 to a2 are moved to addresses a3 and after	V-55
9	Data set	#SP,a ↵	Data from program area address "a" are written to memory	V-56
		#SD,a ↵	Data from data area address "a" are written to memory	V-57

Table 2.1.2.b Function-differentiated command list

Item number	Function	Command configuration	Description of operation	Reference page
10	Change CPU Internal Registers	#DR ↵	Display EVA62XXCPU internal registers	V-38
		#SR ↵	Set EVA62XXCPU internal registers	V-58
		#I ↵	Reset EVA62XXCPU	V-92
		#DXY ↵	Display X, Y, MX and MY	V-47
		#SXY ↵	Set data for X and Y display and MX, MY	V-59
11	History	#H,p1,p2 ↵	Display history data for pointer 1 and pointer 2	V-39
		#HB ↵	Display upstream history data	V-42
		#HG ↵	Display 21 line history data	V-42
		#HP ↵	Display history pointer	V-45
		#HPS,a ↵	Set history pointer	V-45
		#HC,S/C/E ↵	Sets up the history information acquisition before (S), before/after (C) and after (E)	V-60
		#HA,a1,a2 ↵	Sets up the history information acquisition from program area a1 to a2	V-61
		#HAR,a1,a2 ↵	Sets up the prohibition of the history information acquisition from program area a1 to a2	V-61
		#HAD ↵	Indicates history acquisition program area	V-61
		#HS,a ↵	Retrieves and indicates the history information which executed a program address "a"	V-44
		#HSW,a ↵ #HSR,a ↵	Retrieves and indicates the history information which wrote or read the data area address "a"	V-44
12	File	#RF,file ↵	Move program file to memory	V-82
		#RFD,file ↵	Move data file to memory	V-82
		#VF,file ↵	Compare program file and contents of memory	V-83
		#VFD,file ↵	Compare data file and contents of memory	V-83
		#WF,file ↵	Save contents of memory to program file	V-84
		#WFD,file ↵	Save contents of memory to data file	V-84
		#CL,file ↵	Load ICE6200 set condition from file	V-85
		#CS,file ↵	Save ICE6200 set condition to file	V-85
13	Coverage	#CVD ↵	Indicates coverage information	V-48
		#CVR ↵	Clears coverage information	V-48
14	ROM Access	#RP ↵	Move contents of ROM to program memory	V-88
		#VP ↵	Compare contents of ROM with contents of program memory	V-89
		#ROM ↵	Set ROM type	V-90
15	Terminate ICE	#Q ↵	Terminate ICE and return to operating system control	V-95
16	Command Display	#HELP ↵	Display ICE6200 instruction	V-98
17	Self Diagnosis	#CHK ↵	Report results of ICE6200 self diagnostic test	V-46

Alphabetical listing of commands

Tables 2.1.3.a–b show an alphabetical listing of ICE6200 commands.

Table 2.1.3.a Alphabetical listing of commands

Item number	Command configuration	Description of operation	Reference page
1	#A,a↵	Assemble mnemonic instruction and store in address "a"	V-50
2	#BA,a↵	Set break at program address "a"	V-64
3	#BAR,a↵	Cancel breakpoint	V-64
4	#BC ↵	Display break condition	V-70
5	#BD↵	Set break condition for RAM data	V-65
6	#BDR↵	Cancels the data RAM break condition	V-65
7	#BE ↵	Break enable mode	V-78
8	#BM↵	Assign multiple break condition for program address, RAM data and registers	V-68
9	#BMR ↵	Cancels the multiple break condition	V-68
10	#BR ↵	Break condition set for EVA62XXCPU registers	V-66
11	#BRR ↵	Cancels the register break condition	V-66
12	#BRES ↵	All break conditions canceled	V-71
13	#BRKSEL,REM↵	Sets BA clear/remain modes	V-80
14	#BSYN↵	Break disable mode	V-78
15	#BT ↵	Sets break stop/trace mode	V-79
16	#CHK↵	Reports results of ICE6200 self diagnostic tests	V-46
17	#CL,file↵	Loads ICE6200 set condition from file	V-85
18	#CS,file↵	Saves ICE6200 set condition to file	V-85
19	#CVD↵	Indicates coverage information	V-48
20	#CVR↵	Clears coverage information	V-48
21	#DD,a1,a2↵	Displays contents of addresses a1 to a2 in the data area	V-36
22	#DP,a1,a2 ↵	Displays contents of addresses a1 to a2 in the program area	V-34
23	#DR↵	Displays EVA62XXCPU internal registers	V-38
24	#DXY↵	Displays X, Y and MX, MY	V-47
25	#FD,a1,a2,d ↵	Sets d to addresses a1 to a2 in the data area	V-53
26	#FP,a1,a2,d ↵	Sets d to addresses a1 to a2 in the program area	V-52
27	#G,a↵	Executes the program from the "a" address	V-72
28	#H,p1,p2 ↵	Displays history data for pointers 1 and 2	V-39
29	#HA,a1,a2↵	Sets up the history information acquisition from program area a1 to a2	V-61
30	#HAD↵	Indicates the history acquisition program area	V-61
31	#HAR,a1,a2↵	Sets up the prohibition of the history information acquisition from program area a1 to a2	V-61
32	#HB↵	Displays upstream history data	V-42
33	#HC,S/C/E↵	Sets up the history information acquisition before (S), before/after (C) and after (E) the break hit	V-60
34	#HELP↵	Display ICE6200 instructions	V-98
35	#HG↵	Display history data in 21 lines	V-42

Table 2.1.3.b Alphabetical listing of commands

Item number	Command configuration	Description of operation	Reference page
36	#HP ↵	Display history pointer	V-45
37	#HPS,a ↵	Set history pointer	V-45
38	#HS,a ↵	Retrieves and indicates the history information which executed the program address "a"	V-44
39	#HSR,a ↵	Retrieves and indicates the history information which read the data area address "a"	V-44
40	#HSW,a ↵	Retrieves and indicates the history information which wrote the data area address "a"	V-44
41	#I ↵	Reset EVA62XXCPU	V-92
42	#L,a1,a2 ↵	Display disassembled contents of addresses a1 to a2	V-32
43	#MD,a1,a2,a3 ↵	Move contents of data area addresses a1 to a2 to address a3 and after	V-55
44	#MP,a1,a2,a3 ↵	Move contents of program area addresses a1 to a2 to address a3 and after	V-54
45	#OTF ↵	Select on-the-fly display	V-94
46	#Q ↵	Terminate ICE and return to operating system control	V-95
47	#RF,file ↵	Move program file to memory	V-82
48	#RFD,file ↵	Move data file to memory	V-82
49	#ROM ↵	Select ROM type	V-90
50	#RP ↵	Move ROM contents to program memory	V-88
51	#SD,a ↵	Write data from address "a" of the data area	V-57
52	#SP,a ↵	Write data from address "a" of the program area	V-56
53	#SR ↵	Set EVA62XXCPU internal registers	V-58
54	#SXY ↵	Display X, Y and set data to MX, MY	V-59
55	#T,a,n ↵	Execute while displaying n step instruction results from address "a"	V-75
56	#TIM ↵	Select execution time and step counter	V-93
57	#U,a,n ↵	Display only final step of #T,a,n	V-77
58	#VF,file ↵	Compare program file and memory contents	V-83
59	#VFD,file ↵	Compare data file and memory contents	V-83
60	#VP	Compare contents of ROM and contents of program memory	V-89
61	#WF,file ↵	Save content of memory to the program file	V-84
62	#WFD,file ↵	Save content of memory to the data file	V-84

2.2 Connecting and Starting the System

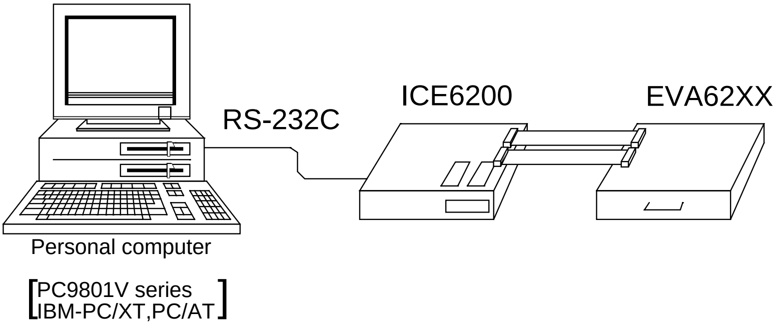


Fig. 2.2.1

System connection diagram

The ICE6200 connects to common personal computers and the E0C62 Family evaluation board EVA62XX for operation, as shown in Figure 2.2.1. The connection sequence described below should be followed.

(1) Verify Power OFF status

Make sure the power sources for the personal computer and ICE6200 are switched OFF. (The E0C62 Family evaluation board EVA62XX is powered by the ICE6200 power supply and thus has no power source.)

(2) Cable Connections

Connect cables in the manner prescribed in the "ICE6200 Hardware Manual".

(3) Power ON

Switch ON the power supplies for the personal computer and the ICE6200 in any order.

HOST settings

The ICE6200 is connected to a general purpose personal computer for operation.

The ICS62XX system program has an approximately 140KB capacity, and the personal computer must be set to proper operating parameters for the ICS62XX to operate. An example follows.

Program capacity

The ICS62XX system program requires a host system with a RAM capacity of about 140KB.

RS232C Settings

ICE Operation Using a PC9801V System with MS-DOS v.3.10

Execute SPEED command soon after starting MS-DOS.

Setting:

```
A>SPEED R0 9600 B8 PN S1 NONE↵
```

Verify settings:

```
A>SPEED↵

SPEED Version ?.?
```

Escape the command with:

```
RS232C-0 9600 BITS-8 PARITY-NONE STOP-1 NONE↵
(end)
```

ICE Operation Using a PC/XT, PC/AT System with PC-DOS v.2.10

Execute MODE command soon after starting PC-DOS.

Setting:

```
A>MODE COM1:4800,n,8,1,P↵
COM1:4800,n,8,1,P    .... Settings can be confirmed.
A>
```

Set the ICE6200 baud rate to 4800.

Starting the ICS62XX

Start the Operating System

First, call up the operating system (abbreviated OS below) for your general purpose personal computer. The ICS62XX can operate in the following OS environments.

(1) MS-DOS version 3.10 or higher
(2) PC-DOS version 2.10 or higher

Refer to your OS manual for procedures on loading the system. After loading the system, set the HOST setting as described in "HOST setting" (page V-10).

Starting the ICS62XX

(1) Insert the ICS62XX system software (supplied on 5.25" floppy disk) to the assigned floppy disk drive in your personal computer.

(2) Input the following information through the keyboard.

```
B>ICS62XX↵
...The Epson logo is displayed for about one second...
* ICE POWER ON RESET *
* DIAGNOSTIC TEST OK *
# _
  └──   Cursor position
```

When the ICS62XX system program is loaded in the computer as described above, control of the computer is given to the ICS62XX system program. ICS62XX commands are awaited when the program is properly loaded and the # mark is displayed.

Quitting ICS62XX Control

The ICS62XX program is terminated by entering the Q command; control is then returned to the computer's operating system.

```
#Q↵
B>
```

2.3 ICE6200 Operation and Functions

ICE6200 operations, details on functions and emulation limitations are discussed in this section.

Operating features

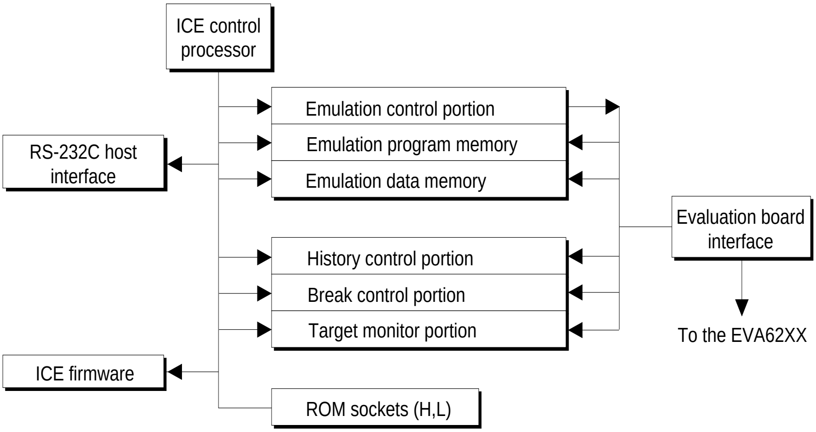


Fig. 2.3.1
Block diagram of ICE6200 functions

Figure 2.3.1 shows a block diagram of ICE6200 functions. The ICE6200 has a built-in control processor which processes ICE commands.

Emulation consists of executing and terminating functions of the EVA62XXCPU and is controlled via the emulation control portion. The EVA62XXCPU is halted unless the run (G command) or single step (T command) operations are invoked. In this condition the emulation lamp on the ICE6200 display is OFF and the HALT lamp is ON to indicate the set-up mode. Thus, the A command, etc., are executed during the set-up mode.

The emulation program memory is set-up by instructions which activate the EVA62XXCPU.
In the set-up mode, such operations as loading from the ROM sockets by the ICE control processor and program setting by the host processor are executed.
Similarly, the EVA62XXCPU data RAM is allocated to the emulation data memory.

The history control portion records the execution bus cycles of the EVA62XXCPU and consists of a 8192 word × 88 bit memory. The large memory capacity allows EVA62XXCPU register values to be recorded in real time. The history is written in target run mode, and is analyzed by the ICE6200 control processor in the set-up mode.

The break control portion has the functions which check the EVA62XXCPU bus condition whether it is at a break point or not, and will stop the execution at the break point. Breaking at CPU register values is also possible in real time. The ICE6200 control processor monitors the EVA62XXCPU on the target monitor during target run mode. Results are displayed as on-the-fly information.

Break mode and break function

Breaks are supported in many modes.

(1)**Break enable mode:**
Makes the break function valid. Actions during break are decided according to the mode setting of break-trace/ stop.

(2)**Break disable mode:**
Makes the break function invalid. ICE6200 SYNC pin pulse output mode which does not terminate the G command when in break condition. This function can be used as an oscilloscope synchronous signal to measure the target circuit timing using the pulse as a reference.

(3)**Break trace mode:**
Temporarily stops the target run during break condition, and quickly restarts the program after displaying the CPU register and execution time. Effective for viewing the program operation timing, but not in true real time.

(4)**Break stop mode:**
A mode to break programs when they are consistent with break conditions.

Different types of breaks are described below.

(1)**Reset switch:**
Need not be in break mode to break. Used to reset the ICE6200; does not display the target register during break.

(2)**Break switch:**
Need not be in break mode to break. EVA62XXCPU register is properly displayed during break.

(3)**ESC key:**
Break induced by ESC key input from the host. Need not be in break mode to break. EVA62XXCPU register is properly displayed during break.

(4)**Break set command:**
Break induced when CPU conditions and conditions set by BA, BD, BR or BM commands agree. Causes a break in break enable mode and break stop mode, but does not cause break in break disable mode. Cannot be set in break trace mode after completion of the instruction.

Table 2.3.1 shows the break modes and break types.

Table 2.3.1
Break modes and break types

Item	Break mode	Break method	Description
1	Break enable & break stop	Reset switch Break switch ESC key Break instruction	Normal use mode. Start up mode at power on. EVA62XXCPU runs in real time by entering GO command after setting this mode.
2	Break enable & break trace	Reset switch Break switch ESC key	Activates the break trace function. This mode is set by the BE command or BT command. Register data is displayed when the EVA62XXCPU agrees with the conditions set by the break set instruction. EVA62XXCPU does not run in real time when GO command is entered after setting this mode.
3	Break disable & break stop	Reset switch Break switch ESC key	The SYNC output function is executed. A pulse is output to the SYNC pin via the BSYN command when the CPU agrees with the condition set by the break set instruction. EVA62XXCPU runs in real time by entering GO command after setting this mode.
4	Break disable & break trace	——	Automatically sets to break disable and break trace. Break enable mode is automatically set when break trace is set.

SYNC pin and HALT pin output

(1) SYNC Pin Output

When the instruction cycle conforms to a break condition, a low level pulse is output by the first half of the subsequent instruction fetch cycle.

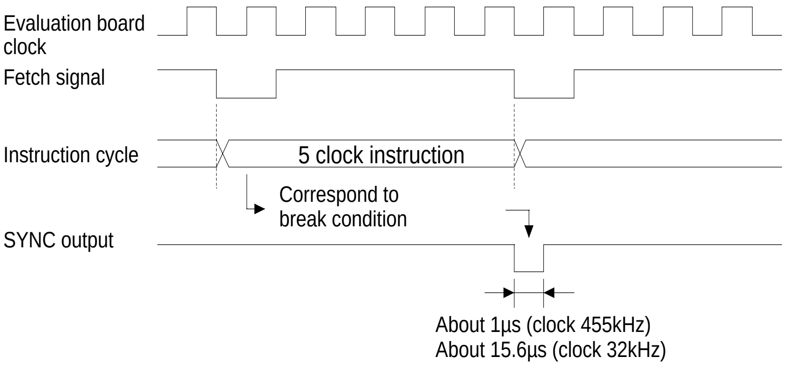


Fig. 2.3.2

SYNC pin output

(2) HALT Pin Output

A low level pulse is output when the evaluation board CPU is stopped (e.g., when the HALT or SLEEP instructions are executed).

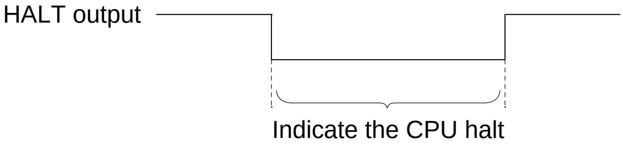


Fig. 2.3.3

HALT pin output

Display during run mode and during break

During run mode, the ICE6200 control processor monitors the state of the EVA62XXCPU. Monitored data EVA62XXCPU's executed program are displayed at intervals of about 500 ms when the on-the-fly display mode is set (by the OTF command).

```
#G↵
 *PC=0120   Underlined portion is displayed in succession.
 *PC=HALT   Enter HALT mode, line feed, and HALT is displayed.
 *PC=0200   HALT is canceled, operation is restarted, and PC is redisplayed.
```

Note *HALT indicates execution of the HALT or SLEEP instruction.*
When the printer is online and started, the PC values are printed in succession. PC is not displayed during on-the-fly inhibit mode.
During a break, the cause of the break, post break PC (the next executed program address), the contents of the CPU registers, and execution time are displayed.

```
#G↵
 *PC=xxxx
 *EMULATION END STATUS=BREAK HIT               .....(1)
 *PC=0201 A=0 B=0 X=070 Y=071 F=IDZC SP=10    .....(2)
 *RUN TIME=425.097mS                          .....(3)
```

(1) There are three statuses possible after completing the emulation: BREAK HIT, ESC KEY, OR BREAK SW. When a number of conditions prevail, only the highest priority position is displayed in the following priority ranking: BREAK SW > ESC KEY > BREAK HIT. A break may also be initiated by the reset switch; a reset switch break causes
" *ICE6200 RESET SW TARGET* "
to be displayed and instructions are awaited. The register display and execution time display are not active in this mode.

(2) The displayed PC shows the next executed value. Register values following "A" indicate the values during a break. In the above example, the values (indicated 2) results from completing to execute the instruction of address 0200.

(3) Execution time mode and step number mode can be set during run time (using the #TIM command). Millisecond is abbreviated to "mS". In step number mode, decimal values describe the run time, as in : " *RUN TIME=501 STEPS ".

When the execution time or step counters overflow, the message
" *RUN TIME=TIMEOVER "
is displayed. For more details, see page V-25, "Measurement during command execution".

Break assigning commands

The ICE6200 has a variety of break setting functions.

(1) Set break by PC:

Set by the BA command. The instruction is executed when the EVA62XXCPU PC and the set values agree, thus inducing a break. When the PSET command is entered at the set address, the PSET and subsequent instruction are executed, then processing is halted. (When multiple PSET commands are specified, the instructions are executed until a command other than PSET is encountered.)
Breaks can be set for multiple PC's (to the maximum capacity of program memory).

(2) Set break by RAM data:

Set by the BD command. A break is induced by the RAM data address, data, or R/W AND condition. Also, masks can be set for address, data and R/W respectively.
When a break is induced by writing F data at address 10, the settings are: address=10, data=F, R/W=W. Any data can be used with the following settings: address=10, data=mask, R/W=W. A break will occur after execution of the memory access instruction which equals the set conditions. The break point can be set to one point through these settings.

(3) Set break by register value:

Set by BR command. When the register values of the EVA62XXCPU coincide with the set break values, a break is initiated following execution of the instruction.
A break is induced by and AND condition set in the A, B, FI, FD, FZ, FC, X, or Y registers. Also, a mask can be set in any of the registers. When a break is induced with register A=5, X=70, and Y=0A, the other registers may be masked.

Example:

```
LD   A,5
LD   X,70
LD   Y,0A ........ A break is induced when the above
                   instruction is executed.
```

These settings will allow the operation to run in real time. The break point can be set at only one point.

Items (1), (2) and (3) above can be set independently. When BA, BD and BR are set concurrently, a break will occur when any of the conditions coincide.

(4) Set compound break:

Set by BM command. A compound break occurs when breaks (1), (2) and (3) include AND statements. Breaks can have the following elements masked: (coincide with PC), (coincide with RAM data address, data, R/W), (register value). The break point can be set at only one point. At the current setting, setting (1) through (3) are automatically canceled. If settings (1) through (3) follow the current setting, the BM condition is canceled.

Note *Since the RAM data condition is a break element, the break will not be initiated without instructions which access the RAM data.*

Target interrupt and break

When a target interrupt occurs the moment of a break it is given priority over the break. The break is then induced after the interrupt process is stacked. Next, the interrupt routine is executed from the top when the run mode commences.

The PC displayed during a break is the top interrupt address.

When a break is set by the BR command with FI=1, the break and interrupt are generated simultaneously, but due to the interrupt process, the register values after the break are:

```
*PC=0000  A=....  F=.DZC  X=000  Y=010
                     |
                  FI reset
```

so as to reset the FI flag status.

History function

The EVA62XXCPU information (PC, instruction code, RAM data address and data content, and CPU internal registers) while running an emulation are fetched to the history memory region with each CPU bus cycle. The history memory has a capacity of 8291 cycles, and can store 2730 (5 clock instructions only) to 1365 (12 clock instructions only) new instructions executed by the evaluation board.

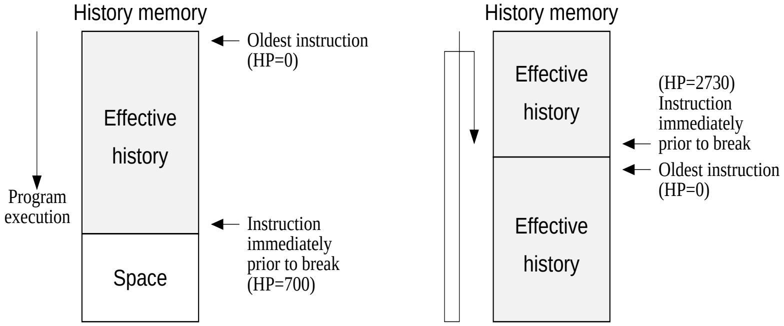
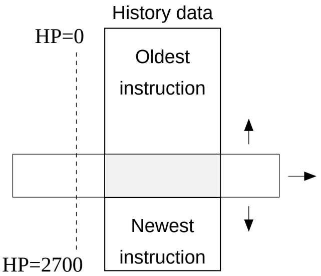

History memory
Oldest instruction (HP=0)
Effective history
Program execution
Instruction immediately prior to break (HP=700)
Space
History memory
Effective history
(HP=2730) Instruction immediately prior to break
Oldest instruction (HP=0)
Effective history

Fig. 2.3.4

History function diagram

Figure 2.3.4 shows a diagram of the history function. When the history memory is filled, old data is overwritten by new data.

The history pointer (HP) normally displays the oldest instruction at position 0, but during a break it displays the newest instruction. The maximum value of the HP is about 2730 when 5 clock instructions are executed.



The HP can display optional positions via the H, HB, and HG commands.
HP data from 1980 to 1986 is displayed by entering:
#H, 1980, 1986↵

Fig. 2.3.5

History data display

```
  #H, 1980, 1986↵
  LOC   PC  IR OP   OPR. A B   X   Y IDZC MEMORY OPERATION    OTHER
  1980 0200 FC1 PUSH B    0 0 03F 03F 1111 W010=0
  1981 0201 423 CALL 23   0 0 03F 03F 1111 W00F=8 W00E=0 W00D=2    ....(1)
  1982 0223 FDF RET       0 0 03F 03F 1111 R00D=2 R00E=0 R00F=8
  1983 0202 FD1 PDP  B    0 0 03F 03F 1111 R010=0
* 1984                                     W010=8 W00F=0 W00E=2 INT1
  1985                                                          INT2
  1986 00FE FFF NOP7      0 0 0FF 0FF 0111
   (a)  (b) (c)    (d)        (e)      (f)         (g)          (h)
```

(a) History pointer displayed

(b) Executed instruction address displayed

(c) Instruction code displayed

(d) Mnemonic instruction displayed

(e) Register value displayed when instruction completed

(f) When each flag is set, 1 is reset to 0 and displayed

(g) When a data memory R/W operation occurs during execution of an instruction, the data sequence write 8 to 0F address write 0 to 0E address write 2 to 0D address is sequentially displayed (1).

(h) During the interrupt process, INT1 (stack) and INT2 (vector) are displayed. The INT1 memory operation indicates the stack cycle.

Note * *During interrupt processing, two HP are renewed. Otherwise, HP is renewed by the instruction unit.*

Break delay function

Users can refer to the programs until break by the history function mentioned in the previous section. In the ICE6200 this function has been expanded so that the history information before hitting the break condition or before and after hitting break condition can be acquired and referred. To realize this function, this system is designed not to terminate the program right after the hit of break condition, but to terminate the program after acquiring specified history data. This specification is executed by the #HC command.

Note *When specifying the break delay by using the break enable & break stop mode (see page V-15, "Break mode and break function"), be sure that break is not made at the specified break condition.*

Coverage function

ICE6200 can acquire and indicate the address information of the program which was accessed during the execution of the program. One can confirm which parts have completed troubleshooting and debugging by referring to coverage information which is a result of executing programs for a long period of time. This coverage function is specified by #CVD, and #CVR commands.

Measurement during command execution

The ICS62XX possesses a counting function which counts the time or the number of steps from starting the target program to the occurrence of a break.

The counting range is described below.

(1) Time counting mode

6.5µs to 6.5 × 65535µs (=425.977ms)

Measurement error : ±6.5µs
(The display is in millisecond units: ms)

(2) Step counting mode

Step 1 to step 65535

Measurement error : 0 steps
(error of 1 step may be presumed during interrupt process)

When the measurement range is exceeded, the following message is displayed:

```
*RUN TIME=TIMEOVER.
```

Self-diagnostic function

The ICE6200 performs a self-check at power ON. When a check instruction (#CHK↵) is input from the host system, the self-test results are sent to the host.

```
#CHK↵
#          …System awaits instruction unless an error occurs.
```

A check instruction is automatically input when the ICS62XX system program is loaded.

```
B>ICS62XX↵                   (Epson logo appears)
* ICE POWER ON RESET *
* DIAGNOSTIC TEST OK *      (Check instruction is automatically input;
                            if no anomaly occurs, the following
                            message appears)
#
```

When the above display appears, it indicates that the ICE6200 and host are connected properly and the ICE6200 is operating correctly.

If the ICE6200 is power supply is OFF or the the cable to the host is not connected at the prompt, the following message appears:

```
B>ICS62XX↵
*COMMUNICATION ERROR OR ICE NOT READY*
```

Then, when the ICE6200 power supply is switched ON, a self-test is automatically performed and the following message is displayed:

```
* ICE POWER ON RESET *
* DIAGNOSTIC TEST OK *
#
```

When an error message is displayed after entering the check instruction, it is likely to be due to hardware failure. Contact customer support.

Starting the printer

The printer is controlled by the operating system. The printer can be started and stopped by entering "CTRL"+"P" key even while the ICS62XX system is running.

```
#BA,100↵
#"CTRL"+"P" T↵        ........  The monitor display following the
                                "CTRL"+"P" key input is printed.
PC=300  IR=FFF        ........  SP=010
     :
     :
     :
#"CTRL"+"P"           ........  Stops the printer
```

Limitations during emulation

When running emulations with the ICE6200 and evaluation board connected, the EVA62XXCPU is normally stopped, as described in page V-13, "Operating features" (set up mode).

In the set up mode, the EVA62XXCPU and peripherals are stopped, and inappropriate operations cannot be initiated. Until the set up mode is canceled and the target program is executed, the EVA62XXCPU executes instructions provided by the command program of the ICE6200. The command program continues to operate when the emulation is completed and returns to the set up mode.

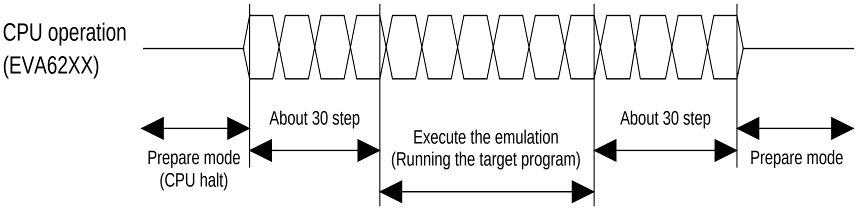


Fig. 2.3.6
EVA62XXCPU operation

You should be aware that when the command program takes over, the timers and counters are enabled and started from initial settings. Also, the watchdog timer is cleared immediately prior to the ICE6200 switching to emulation mode while under command program control.

Accordingly, the following points should be noted when using the ICE6200.

(1) When execution of the trace instruction (T,U) is prolonged

Evaluation board timer values can be renewed while the command program is operative.

(2) When the run is halted and restarted

The watchdog timer is cleared by the ICE6200 before and after the emulation, thus the watchdog timer is not continuous. The target program operates in real time when the run time is sufficiently long.

The command program runs approximately 30 steps before and after an emulation. When operating at 32kHz clock speed, these steps require 6ms + 6ms = 12ms. While at a clock speed of 455kHz, the command program steps before and after emulation require 400µs + 400µs = 800µs.

When the dump data command (#DD) is invoked, the I/O area interrupt condition flag is read but not cleared.

CHAPTER 3 COMMAND DETAILS

Detailed particulars on ICE6200 commands and explanations of functions are described in this section. Commands are divided into six categories.

DISPLAY: This command group displays the contents of program memory and data memory, and history information.

SET: This group of commands modifies the contents of memory (program and data memories).

BREAK and GO: Sets break conditions and starts emulations.

FILE: Controls transfer of files from the host to the ICE6200.

ROM: Controls the transfer of program memory and ROM (high and low) used by the evaluation board CPU.

CONTROL: Sets the ICE6200 operation mode (including initialization of the target system).

An E0C6231/62L31 program is used in the examples, but output error messages may differ with the type of device used.

The methods for entering instructions described in section 3.1 are as follows:

– A # mark is displayed when the program awaits instructions.

– Upper and lower case letters may be used to enter instructions.

– Individual instructions delineated by <> marks in the text should be separated by a comma when entering instructions.

– Interactive instructions imbeded in commands are displayed by key input. The interactive portions of instructions in the following examples are underlined in the text.

– The toggle instruction is set to reverse upon each command input.

– Notes indicates points for caution when using the described commands.

3.1 Display Command Group

L *DISASSEMBLE LIST*

Format

```
#L,<address 1>,<address 2>↵
#L,<address 1>↵
#L↵
```

Function

The program area (emulation program memory) is displayed disassembled from <address 1> to <address 2>.

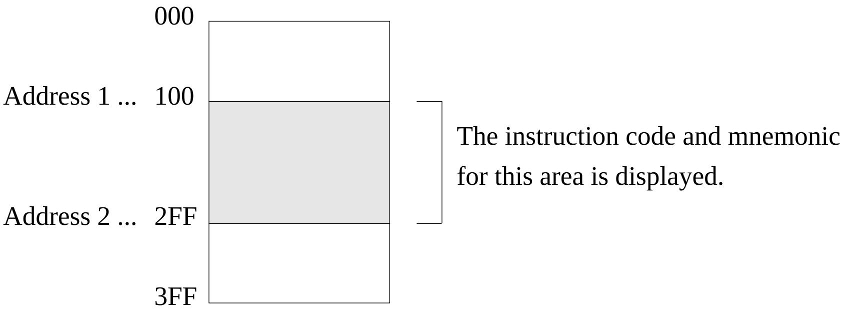
(1) When <address 2> defaults, a single screen (22 lines) is displayed disassembled.

(2) When <address 1> and <address 2> default, a single screen is displayed disassembled from the previous address plus one (one more than the previous address).
With only L↵ input after power on, the data from address 0 onward is displayed.

(3) When more than a single screen is displayed disassembled, a single line space appears between each 22 lines with about a one second pause.

(4) The instruction can be interrupted by hitting the "ESC" key.

Program area (for E0C6231/62L31)



The instruction code and mnemonic for this area is displayed.

DISASSEMBLE LIST L

Format

```
#L,<address 1>,<address 2>↵
#L,<address 1>↵
#L↵
```

Examples

```
#L,100,1FF↵              . . . . . Contents of addresses 100 to 1FF of the program
 0100 FDF RET                      are displayed disassembled
 0101 2FF JP C,FF
   :   :   :
 01FF FFF NOP7

#L,200↵                  . . . . . Contents from address 200 onward (22 lines)
 0200 E00 LD A,0                   are displayed
 0201 E6F LDPX MX,F
   :   :   :
 0215 FFF NOP7
#L↵                      . . . . . One more than the previous address at which the
 0216 FDF RET                      program stopped are displayed
 0217 E05 LD A,5
   :   :   :
 022B FFB NOP5

#L,100,FFF↵
 0100 FDF RET
   :   :   :
 0201 E6F LDPX MX,F
                         . . . . . Interrupt via "ESC" key input

#L,100,50↵               . . . . . Address 1 > address 2 error
 * COMMAND ERROR *

#L,100,100↵              . . . . . Contents of address 100 are disassembled,
 0100 FDF RET                      and executed normally

#L,3FC↵
 03FC E00 LD A,0
   :
 03FF 20F JP C,F         . . . . . Last program area (3FF address in the  case of
                                   E0C6231/62L31) is passed, and instruction
                                   terminates

#
```

DP *DUMP PROGRAM*

Format

```
#DP,<address 1>,<address 2>↵
#DP,<address 1>↵
#DP↵
```

Function

The program area (emulation program memory) from <address 1> to <address 2> is displayed in hexadecimal format.

(1) When <address 2> defaults, the contents of <address 1> are displayed in a single screen (21 lines, 21×8=168 addresses).

(2) When <addresses 1> and <2> default, a single screen is displayed from the previous address plus one (one more than the previous address).
When DP↵ alone is entered after power on, the data from address 0 are displayed.

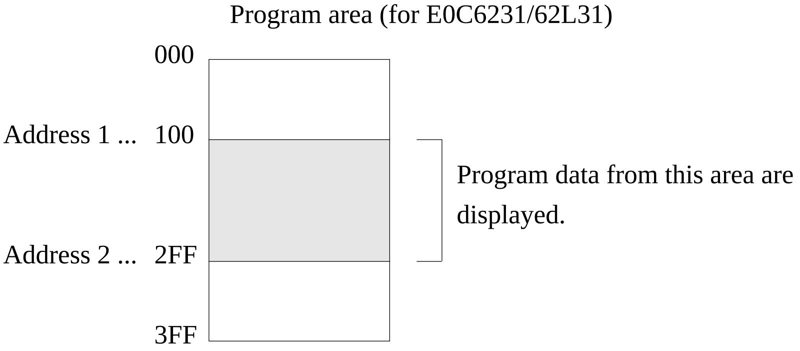
(3) When more than one screen of data is displayed, a one line space appears between every 21 lines with about a one second pause.

(4) Hexadecimal and ASCII codes can be displayed together, but the ASCII data operands are converted by the RETD and LBPX instructions before display.

Example: Data content 142 ... ASCII display B
(Instruction: RETD 42)

(5) When the last program area passes, the operation terminates.

(6) Commands can be interrupted by input from the "ESC" key.


Program area (for E0C6231/62L31)
000
Address 1 ... 100
Address 2 ... 2FF
3FF
Program data from this area are displayed.

DUMP PROGRAM DP

Format

```
#DP,<address 1>,<address 2>↵
#DP,<address 1>↵
#DP↵
```

Examples

```
#DP,104,121↵         ..... Specified area is displayed
 ADDR   0    1    2    3    4    5    6    7   ASCII
 0100                      FFF  FFB  930  142      ..0B
 0108  FFF  FFF  FFF  FFF  FFB  931  142  944  .....1BD
   :    :    :    :    :    :    :    :    :
 0118  FFF  FFF  FFF  FFF  FFB  FFB  FFB  FFB  ........
 0120  131  145                                1E

#DP↵                 ..... 21 lines are displayed
 ADDR   0    1    2    3    4    5    6    7   ASCII
 0120            131  132  145  FFF  FFB  FFB    12E...
   :    :    :    :    :    :    :    :    :
   :    :    :    :    :    :    :    :    :
   :    :    :    :    :    :    :    :    :
                     21 line display

#DP,0,FFF↵
ADDR   0    1    2    3    4    5    6    7   ASCII
 0000  FFF  FFF  FFF  FFF  FFF  FFF  FFF  FFF  .......
   :    :    :    :    :    :    :    :    :
   :    :    :    :    :    :    :    :    :
   :    :    :    :    :    :    :    :    :
                     ..... Command interrupt via "ESC" key input

#DP,100,50↵          ..... Address 1 > address 2 error
 * COMMAND ERROR *

#DP,400,FFF↵         ..... Error due to exceeding maximum value of program
 * COMMAND ERROR *         area (3FF address in the case of E0C6231/62L31)
```

DD *DUMP DATA RAM*

Format

```
#DD,<address 1>,<address 2>↵
#DD,<address 1>↵
#DD↵
```

Function

Data in the RAM area from <address 1> to <address 2> are displayed in hexadecimal format.

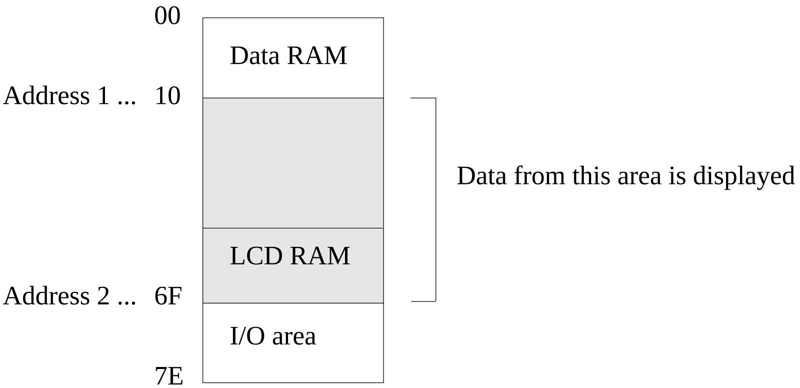
(1) When <address 2> defaults, the contents of <address 1> are displayed in a single screen (21 lines or the last RAM address).

(2) When <addresses 1> and <2> default, a single screen is displayed from the previous address plus one (one more than the previous address). When DD alone is entered after power on, the data from address 0 are displayed.

(3) The contents from the WRITE ONLY I/O area cannot be read.

(4) The I/O address with mixed R/W data is read and displayed with a ! mark.

(5) Commands can be interrupted by input from the "ESC" key.



(for E0C6231/62L31)

Examples

```
#DD,40,7E↵
 ADDR 0 1 2 3 4 5 6 7 8 9 A B C D E F
 0040 5 2 3 4 A B B C D 0 F F F F F F
 0050 - - - - - - - - - - - - - - - -
 0060 - - - - - - - - - - - - - - - -   ..... Write only area is displayed
 0070 5 A 3 F 0 5 6 F 4 4 4 0 5 A A

#DD,100,FFF↵              ..... Error results when RAM address exceeds 7E
 * COMMAND ERROR *              (in the case of E0C6231/62L31)
```

Format

```
#DD,<address 1>,<address 2>↵
#DD,<address 1>↵
#DD↵
```

Examples

```
#DD,0↵
 ADDR 0 1 2 3 4 5 6 7 8 9 A B C D E F
 0000 F F F F F 0 0 0 0 0 0 1 1 1 2 3
   :                                :
   :                                :
 0070 5 A 3 F 0 5 6 F 4 4 4 0 5 A A
                     . . . . . 21 lines or last RAM address is displayed

#DD↵                 . . . . . Display again from address 0 since last address exceeded
                                         (same as above)

#DD,50,40↵
 * COMMAND ERROR *   . . . . . Address 1 > address 2 error

#DD,0,7E↵
 ADDR 0 1 2 3 4 5 6 7 8 9 A B C D E F
 0000 F F F F F 0 0 0 0 0 0 1 1 1 2 3
   :
                     . . . . . Instruction terminated by "ESC" key input
#DD,E40,F1F↵
ADDR 0 1 2 3 4 5 6 7 8 9 A B C D E F
0E40 F 0 1 5 7 4 A 0 0 0 E F 3 2 0 1
                                      . . . . . When the unused area is one
0E80 0 0 3 2 7 6 C 1 1 2 0 0 6 5 4 9            entire line, the display skips
0E90 1 5 7 6 C F 3 2 0 1 0 1 E A C 0            that line (for E0C6246)
0EA0 0 0 0 1 4 0 5 0 0 0 3 0 0 1 5 2
0EBC 4 3 2 7 6 B A 0 1 5 D 3 2 7 4 3
0EC0 5 5 4 1 0 2 3 6 0 0 0 1 5 6 7 F

0F00 ! ! ! ! ! ! / / / / / / / / / /  . . . . . When addresses in the displayed
0F10 F 0 1 0 F F / / / / / / / / / /            lines are unused they are displayed
#                                               as slashes (for E0C6246)
```

Note

The read operation is invalid when the I/O address is set to write only.

DR *DISPLAY CPU REGISTER*

Format

#DR↵

Function

Displays the value of the current register of the EVA62XXCPU.

(1) PC: Displays the address which starts the next emulation.

(2) A, B, X, Y, F, SP: Displays the current value (break or after break value).

(3) IR, Mnemonic: Displays the mnemonic code for the PC program area command code.

Example

```
#DR↵
 * PC=0100 IR=FFF NOP7 A=0 B=0 X=06F Y=03A F=IDZC SP=10
#                                              |
```

Displays characters when F is set,
or . mark when F is reset

Format

```
#H,<pointer 1>,<pointer 2>↵
#H,<pointer 1>↵
```

Function

Displays history data.

(1) Displays history data from <pointer 1> to <pointer 2>.

(2) When <pointer 2> defaults, displays history data of <pointer 1> in 21 lines.

(3) Numerals displayed in <pointers 1> and <2> are decimal, from 0 to 9999.

(4) The following contents are displayed for each instruction:

LOC: History pointer (decimal)
PC: Program counter (hexadecimal) When a break, "[PC]" is displayed.
IR: Command code (hexadecimal)
OP: Command mnemonic
OPR: Command operand
A,B,X,Y: Contents of A, B (Xp, Xh, Xl), (Yp, Yh, Yl) registers
IDZC: Binary display of flag bit (1 when set, 0 when clear)
Other: During execution of an instruction, the memory R/W cycle and data are displayed. Also, data interrupts INT1 (stack data) and INT2 are displayed

(5) History memory has a capacity of 8192 bus cycles. One the other hand, the E0C6200 has 5, 7 and 12 clock instructions. The 5 clock instructions require three bus cycles, 7 clock instructions require four bus cycles, and 12 clock instructions require six bus cycles. Thus, the final value of the history pointer is changed according to the executed instruction. The maximum final value of the execution time for only a 5 clock instruction is approximately 2700, while the execution time for a 12 clock instruction is about 1300. When a break occurs before the history memory reaches the end, the last value of the history pointer is reduced.

(6) The history memory receives new data until a break occurs. Old data is erased when number of executed GO commands exceeds 2700.

(7) The top of the history pointer is 0. When the last value of <address 2> is set, the values are displayed to the last value.

(8) When there are no history data (Before GO command, after GO command execution, during T command execution, or during HAR command execution), the following message is displayed:

```
* NO HISTORY DATA *
```

(9) The HB command can be used to view history data immediately prior to a break.

H HISTORY DATA DISPLAY

Format

```
#H,<pointer 1>,<pointer 2>↵
#H,<pointer 1>↵
```

Examples

```
#H,200,205↵              ..... Set range displayed
  LOC   PC  IR OP   OPR.  A B   X   Y IDZC MEMORY OPERATION   OTHER
 0200  0128 FDO POP  A     F 0 020 021 0011 R01F=0
 0201  0129 F70 DEC  M0    0 0 020 021 0010 R000=1 W000=0
 0202  012A 722 JP   NZ,22 0 0 020 021 0010
 0203  012B F71 DEC  M1    0 0 020 021 0000 R001=2 W001=1
 0204  012C 721 JP   NZ,21 0 0 020 021 0000
 0205  0121 F80 LD   M0,A  0 0 020 021 0000 W000=0

#300↵                    ..... 21 lines displayed
  LOC   PC  IR OP   OPR.  A B   X   Y IDZC MEMORY OPERATION   OTHER
 0300  000F C1F ADD  B,OF  F 4 02D 031 0001
 0301  0010 70E JP   NZ,OE F 3 02D 031 0001
 0302  000E EE8 LDPX MX,A  F 3 02D 031 0001 W02D=F
   :     :   :     :              :
 0319  0124 E10 LD   B,00  F 0 030 031 0001
 0320  0125 BD0 LD   X,D0  F 0 010 031 0001

#H,0,100↵
  LDC   PC  IR OP   OPR.  A B   X   Y IDZC MEMORY OPERATION   OTHER
 0000  0000 E1C LD   A,B   5 4 000 024 0000
 0001  0001 E16 LD   B,06  4 4 000 024 0000
 0002  0002 822 LD   Y,22  4 6 000 022 0000
 0003  0003 EF0 INC  Y     4 6 000 022 0000
 0004  0004 EF3 LDPY A,MY  4 6 000 023 0000 R023=0
 0005  0005 90A LBPX MX,0A 0 6 001 024 0000 W000=A W001=0
 0006  0006 C05 ADD  A,05  0 6 002 024 0000
 0007  0007 D52 SBC  B,02  5 6 002 024 0000
 0008* 0008 17F RETD 7F    5 4 003 024 0000 R01A=C R01B=9 R01C=1 W002=F W003=7
```

* Instruction terminates after exceeding last history memory

Format

```
#H,<pointer 1>,<pointer 2>↵
#H,<pointer 1>↵
```

Examples

```
#H,310,3000↵
  LDC    PC  IR OP   OPR.  A B   X   Y IDZC MEMORY OPERATION   OTHER
 0310  0010 70E JP   NZ,0E F 0 020 021 0011
 0311  0011 8F1 LD   Y,21  F 0 020 021 0011
 0312  0012 E38 LD   MY,08 F 0 020 021 0011 W021=8
   :     :   :   :         : :  :   :    :
 2430  0172 E32 LD   MY,02 7 6 024 026 0000 W026=2
 2431  0173 F48 EI         7 6 024 026 0000
 2432  0174 FF8 HALT       7 6 024 026 1000
 2433                                       W01F=1 W01E=7 W01D=5 INT1
 2434                                                            INT2
 2435* 0108 0E6 JP   E6    7 6 024 026 0000
                              ..... INT1 or INT2 displayed when interrupt only occurs

#H,0,500↵
  LOC    PC  IR OP   OPR.  A B   X   Y IDZC MEMORY OPERATION   OTHER
 0000  0010 70E JP   NZ,0E F B 015 021 0001
 0001  000E EE8 LDPX MX,A  F B 015 021 0001 W015=F
 0002  000F C1F ADD  B,0F  F B 016 021 0001
 0003  0010 70E JP   NZ,0E F A 016 021 0001
 0004  000E EE8 LDPX MX,A  F A 016 021 0001 W016=F
 0005  000F C1F ADD  B,0F  F A 017 021 0001
 0006  0010 70E JP   NZ,0E F 9 017 021 0001
 0007  000E EE8 LDPX MX,A  F 9 017 021 0001 W017=F
 0008  000F C1F ADD  B,0F  F 9 018 021 0001
 0009  0010 70E JP   NZ,0E F 8 018 021 0001
 0010  000E EE8 LDPX MX,A  F 8 018 021 0001 W018=F
                              ..... Instruction terminated by "ESC" key input
#
```

Note

The history data register value is changed by the line following the instruction execution (limited to "LD X,x" and "LD Y,y").

HB, HG *HISTORY DATA DISPLAY BACKWARD/FORWARD*

Format

```
#HB↵
#HG↵
```

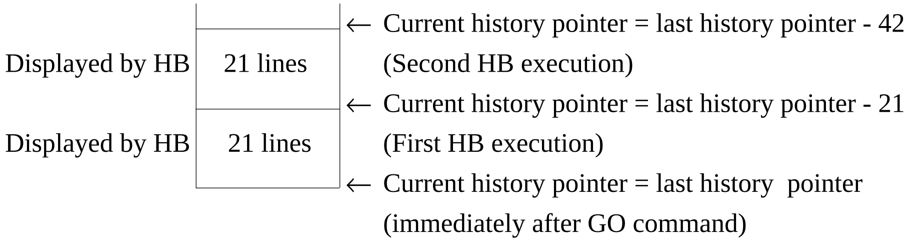
Function

Indicates the history information before and after the history pointer.

(1) HB: 21 instructions displayed from the current history pointer. The current pointer decrements 21 after display. (Validated in vicinity of last displayed history value.)

(2) HG: 21 instructions displayed from the current history pointer. The current pointer increments 21 after display. (Validated from old displayed history value by a screen.)

(3) The current history pointer indicates the last pointer after GO command completion.



Displayed by HB | 21 lines

Displayed by HB | 21 lines

← Current history pointer = last history pointer - 42 (Second HB execution)

← Current history pointer = last history pointer - 21 (First HB execution)

← Current history pointer = last history pointer (immediately after GO command)

Examples

```
#BA,108↵

#G,R↵
 *PC=
 *PC=HALT
 *EMULATION END STATUS = BREAK HIT
 *PC=01E6  A=7 B=6 X=024 Y=026 F=.... SP=4D
 *RUN TIME=TIMEOVER

#HB↵
  LOC    PC  IR OP   OPR.  A B   X   Y IDZC MEMORY OPERATION   OTHER
 2415  0423 83A LD   Y,3A  7 6 056 03A 0010
 2416  0424 CF1 OR   MY,01 7 6 056 03A 0000 R03A=0 W03A=1
 2417  0425 FDF RET        7 6 056 03A 0000 R01D=6 R01E=6 R01F=1
   :      :   :   :        : :  :   :    :
 2432  0174 FF8 HALT       7 6 024 026 1000
 2433                                       W01F=1 W01E=7 W01D=5 INT1
 2434                                                            INT2
 2435* 0108 0E6 JP   E6    7 6 024 026 0000
```

..... When an HB command is executed after a break hit, 21 lines are displayed from the break address onward

Format

#HB↵

#HG↵

Examples

```
#HPS,200↵

#HG↵                 ..... 21 history pointer instructions displayed from 200
  LOC    PC  IR OP   OPR.  A B   X   Y IDZC MEMORY OPERATION   OTHER
 0200  0128 FD0 POP  A     F 0 020 021 0011 R01F=0
 0201  0129 F70 DEC  M0    0 0 020 021 0010 R000=1 W000=0
 0202  012A 722 JP   NZ,22 0 0 020 021 0010
 0203  012B F71 DEC  M1    0 0 020 021 0000 R001=2 W001=1
   :     :   :   :         : :  :   :    :
 0218  000F C1F ADD  B,0F  F E 013 011 0001
 0219  0010 70E JP   NZ,0E F D 013 011 0001
 0220  000E EE8 LDPX MX,A  F D 013 011 0001 W013=F

#HPS,200↵

#HB↵                 ..... 21 history pointer instructions displayed from 200
  LDC    PC  IR OP   OPR.  A B   X   Y IDZC MEMORY OPERATION   OTHER
 0180  000F C1F ADD  B,0F  F 6 03B 021 0001
 0181  0010 70E JP   NZ,0E F 5 03B 021 0001
 0182  000E EE8 LDPX MX,A  F 5 03B 021 0001 W03B=F
 0183  000F C1F ADD  B,0F  F 5 03C 021 0001
   :     :   :   :         : :  :   :    :
 0198  0012 E38 LD   MY,08 F 0 020 021 0011 W021=8
 0199  0013 FDF RET        F 0 020 021 0011 R01C=8 R01D=2 R01E=1
 0200  0128 FDO POP  A     F 0 020 021 0011 R01F=0

#HG↵
  LDC    PC  IR OP   OPR.  A B   X   Y IDZC MEMORY OPERATION   OTHER
 2418  0166 B3A LD   Y,3A  7 6 03A 03A 0000
 2419  0167 CAE AND  MX,0E 7 6 03A 03A 0010 R03A=1 W03A=0
 2420  0168 BFE LD   X,2E  7 6 02E 03A 0010
 2421  0169 E20 LD   MX,00 7 6 02E 03A 0010 W02E=0
 2422  016A BF0 LD   X,20  7 6 020 03A 0010
 2423  016B 980 LBPX MX,B0 7 6 021 03A 0010 W020=0 W021=8
 2424  016C 9C1 LBPX MX,C1 7 6 023 03A 0010 W022=1 W023=C
                          ..... Instruction terminated by "ESC" key input
#
```

ICS6262

HS, HSR, HSW *HISTORY SEARCH PC/MEMORY READ/MEMORY WRITE*

Format

`#HS,<address>`↵
`#HSR,<address>`↵
`#HSW,<address>`↵

Function

Retrieves and indicates history information under the following conditions.

(1) HS: Indicates the history information of the PC address specified by <address>.

(2) HSR: Indicates the history information which read the memory specified by <address>.

(3) HSW: Indicates the history information which wrote the memory specified by <address>.

Examples

```
#HS,0700↵  ..... Retrieves and indicates the history information of PC = 700
  LOC    PC   IR OP   OPR.   A B   X   Y IDZC MEMORY OPERATION   OTHER
 1980  0700 FC1 PUSH B      0 0 0FE 0FF 1111 W0F0=0
 2038  0700 FC1 PUSH B      5 1 0FE 0F0 1001 W0FE=1
   :
   :
```

```
#HSR,30↵   ..... Retrieves and indicates the history information which read address 30
  LOC    PC   IR OP   OPR.   A B   X   Y IDZC MEMORY OPERATION   OTHER
 0820  0640 EC2 LD   A,MX   0 0 030 0FF 1111 R030=0
 0950  084F EC6 LD   B,MY   0 F 030 0FF 1111 R030=F
   :
   :
```

```
#HSW,30↵   ..... Retrieves and indicates the history information which wrote address 30
  LOC    PC   IR OP   OPR.   A B   X   Y IDZC MEMORY OPERATION   OTHER
 0838  0650 E60 LDPX MX,0   0 0 030 0FF 1111 W030=0
 0950  084F E71 LDPY MY,1   0 0 0FF 030 1111 W030=1
   :
   :
```

Format

```
#HP↵
#HPS,<history pointer>↵
```

Function

(1) HP: Displays current history pointer value.

(2) HPS: Sets the displayed history pointer value in the current history pointer. When a value is input which exceeds the last history pointer, the last pointer value is set to the current history pointer.

(3) The history pointer is displayed in four lines of decimal code, and set.

Examples

```
#HP↵
 * LOC=2058                    . . . . . Pointer (last value) displayed at break

#HPS,1000↵                . . . . . Pointer set to 1000

#HP↵
 * LOC=1000                    . . . . . Pointer value = 1000

#HPS,9999↵
 * LOC=2058                    . . . . . Return to last pointer value
                                         Last pointer value is validated when last value is
     #HP↵                                exceeded
 * LOC=2058
```

CHK *CHECK ICE6200 HARDWARE*

Format

#CHK↵

Function

Displays the results of the ICE6200 initial test.
(ICE6200 executes the initial test at power on.)

The test consists of the following:

(1) Sum check test of ICE6200 firmware

(2) ICE6200 RAM R/W test

Examples

```
#CHK↵
 * ROM CHECK ERROR 5F=>FF *                 ┐      Message is displayed when an
 * RAM CHECK ERROR 001111 55=>FF *          ┘      error is detected

#CHK↵

#                             . . . . . A waits command under normal conditions
```

Note

When an error message is displayed, avoid further use of the device since it is likely due to hardware failure.

DISPLAY X, Y REGISTER, MY CONTENT DXY

Format

```
#DXY↵
```

Function

Displays current X register (Xp, Xh, Xl) and Y register (Yp, Yh, Yl), as well as MX and MY (contents of memory specified by codes X and Y).

Examples

```
#DXY↵
 X=070  MX=  5
 Y=07C  MY=  F

#DXY↵
 X=200  MX=-:OV .....Indicates the RAM area has  been  exceeded;
 Y=050  MY=-         read operation not viable
           :.........Indicates write only area;  read operation not viable

#DXY↵
 X=E73  MX=  /  .....  Shows that E73 is unused area
 Y=252  MY=  F  .....  Read operation not viable

#
```

CVD, CVR *DISPLAY/RESET COVERAGE*

Format

```
#CVD,<address 1>,<address 2>↵
#CVD↵
#CVR↵
```

Function

Indicates and clears coverage information.

(1) CVD: Indicates the coverage information ranging from <address1> to <address2>.
Indicates all coverage information when address are omitted.

(2) CVR: Clears coverage information.

Examples

```
#CVD,100,110↵      .....Indicates the coverage information ranging
 *CV 0100               from address 100 to 110
 *CV 0109..0110
#

#CVD↵              .....Indicates the whole coverage information
 *CV 0100
 *CV 0109..02FF
 *CV 0400..04FF
#

#CVR↵              .....Clear coverage information
#
```

3.2 Set Command Group

A ASSEMBLE PROGRAM

Format

`#A,<address>↵` (With guidance)

Function

The mnemonic command is assembled and stored at the address indicated by <address>.

(1) Supports the mnemonics and operands in the instruction list used in the E0C62 Family.

(2) Operand expressions follow the configurations below:

p:	00 to 03 values
s:	00 to FF values
l:	00 to FF values
i:	00 to 0F values
r,q:	A, B, MX or MY

In general, hexadecimal expressions do not have "H" appended at the end.

Three digit data can be input starting from the 0 column.

0FF input:	Validates FF
00FF input:	Causes an error

An error is generated by invalidated values entered for p, s, l or i.

Only binary expressions (xxxxB) are allowed in the input area. The x in this case has a fixed length of from one to four digits comprised either of 0 or 1, with "B" input last.

When less than three digits are input, the expression is handled as a binary expression or an error.

(3) Either upper or lower case letters may be used for input.

(4) Mnemonic and operand codes should be separated by one or more character spaces or by a tab code.

(5) An error is generated when an unsupported instruction is entered.

(6) A or B input gains register priority. Input 0A or 0B when entering immediate value settings.

`LD   A,B`	Contents of B register are input to A register.
`LD   B,0A`	Immediate value A is loaded to B register.

Format

`#A,<address>↵` (With guidance)

Examples

```
#A,100↵                    ..... Instruction entered by key input
 0100 LD A,0F↵             ..... Address displayed; mnemonic input awaited (mnemonic
                                      instruction, operand input)
 0101 /↵                   ..... /↵ input cancels instruction

#A,200↵
 0200 PUSH XP↵             ..... Error generated by unapproved mnemonic input
     * ERROR *                   (for E0C62XX/62*XX); same address is  redisplayed
                                      with mnemonic request
 0200 NOP5↵
 0201 JJJ 0FF↵
     * ERROR *
 0201 LD A,FF↵             ..... Error generated when valid operand range is exceeded
     * ERROR *
 0201 LD A,0F↵
 0202 /↵

#A,202↵
 0202 ^↵                   ..... Return to previous address (current address less one) via
 0201 /↵                         ^ key input

#
```

Note

"ESC" key nonfunctional; cancel operation by entering /↵.

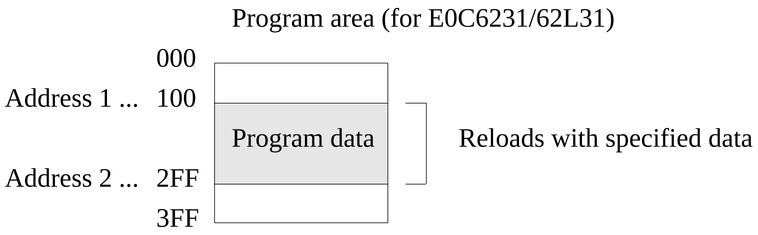
FP *FILL PROGRAM*

Format

```
#FP,<address 1>,<address 2>,<program data>↵
```

Function

The contents of <address 1> and <address 2> of the program area (ICE emulation memory) are stacked in the program data area.



Examples

```
#FP,0,3FF,FFB↵
```

..... Data from addresses 000 to 3FF of the program area are stacked to the FFB (NOP5 code)

```
#FP,100,200,FF9↵
 * COMMAND ERROR *
```

..... When undefined code is detected, an error message is displayed and the instruction will not execute

```
#FP,200,100,FFF↵
 * COMMAND ERROR *
```

..... Address 1 > address 2 error

```
#FP,200,200,FFF↵
```

..... Address 200 is modified to instruction code FFF (NOP7); instruction completes normally

```
#
```

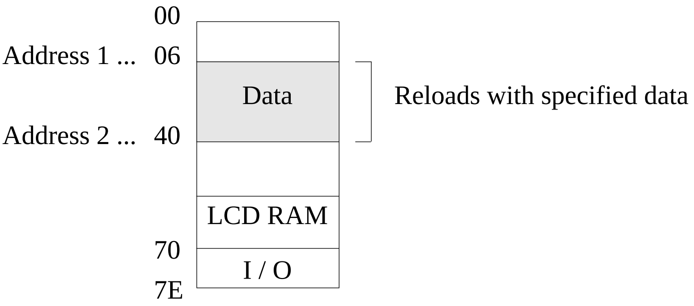
Format

`#FD,<address 1>,<address 2>,<data>↵`

Function

Data is stacked in the data RAM area at addresses 1 to 2 in hexadecimal or binary code.

Data RAM area (for E0C6231/62L31)



Examples

```
#FD,60,7E,A↵              ..... Reloads the contents of the data RAM addresses 60
                                to 7E to A
#FD,10,2F,0101B↵          ..... Reloads address 10 to 2F with
                                data 0101 (binary) = 5 (hexadecimal)
#FD,50,1FF,0↵             ..... Error is generated because settings exceed the RAM area
 * COMMAND ERROR *              (address 7E for E0C6231/62L31) and the instruction
                                will not execute
#FD,70,60,0↵
 * COMMAND ERROR *        ..... Address 1 > address 2 error

#FD,0,7E,B↵               ..... Reloads the entire RAM area (for E0C6231/62L31)
                                with data B (hexadecimal)
#FD,40,40,0↵              ..... 0 written to 40 address
```

Notes

(1) For binary expressions, four digit 0 (or 1) and B input (total of five characters) only are accepted.

(2) Write operation is not performed to the read only address of the I/O area.

(3) When there is an unused area in the specified address, the data is rewritten except for the unused area.

MP *MOVE PROGRAM*

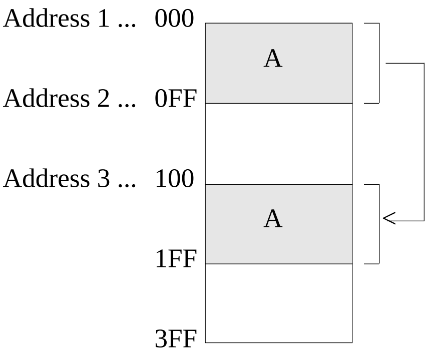
Format

```
#MP,<address 1>,<address 2>,<address 3>↵
```

Function

Contents of program area addresses 1 to 2 are transferred to addresses 3 and above.

Program area (for E0C6231/62L31)



Examples

```
#MP,0,FF,100↵                 . . . . . Contents of program area addresses 000 to 0FF are
                                        transferred to addresses 100 to 1FF
#MP,100,2FF,300↵              . . . . . When the transfer area surpasses address 3FF, an error
 * COMMAND ERROR *                      message is displayed and the instruction will not
                                        execute
#MP,200,100,300↵
 * COMMAND ERROR *            . . . . . Address 1 > address 2 error

#MP,200,200,300↵              . . . . . Contents of address 200 are copied  to address 300, then
                                        the instruction is executed normally
#
```

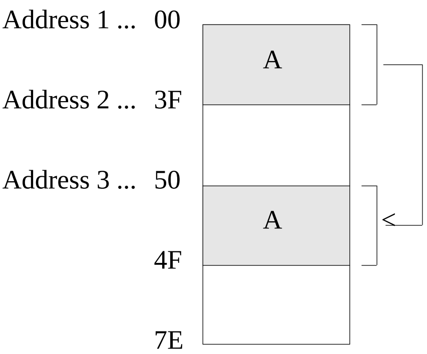
Format

```
#MD,<address 1>,<address 2>,<address 3>↵
```

Function

Contents of addresses 1 to 2 in the data RAM area are transferred to addresses 3 and above.

Data RAM area (for E0C6231/62L31)



Examples

`#MD,10,1F,30↵`	. . . . .	Contents of data RAM addresses 10 to 1F are moved to addresses 30 to 3F

`#MD,00,3F,70↵`
` * COMMAND ERROR *`
	. . . . .	When the transfer area exceeds the RAM area (7E for E0C6231/62L31), an error is indicated and commands are not executed

`#MD,30,20,50↵`
` * COMMAND ERROR *`
	. . . . .	Address 1 > address 2 error

`#MD,30,30,50↵`	. . . . .	Contents of address 30 are copied to address 50, then instruction is executed normally

`#MD,E00,E1F,E60↵`
` * UNUSED AREA *`
	. . . . .	When there is an unused area in the transfer area (either sending or receiving side), an unused area error message is displayed (for E0C6246)

Notes

(1) A write operation cannot execute when the top transferred address coincides with the I/O area read only region.

(2) A read operation cannot execute when the bottom transferred address coincides with the I/O area write only region. In this case a 0 is written to the top address.

(3) When the transfer address coincides with an I/O address of mixed readable bits and write only bits, either read or write operations can execute.

SP *SET PROGRAM*

Format

#SP,<address>↵ (With guidance)

Function

Contents of the specified program area address are displayed or modified.

Examples

```
#SP,100↵
 0100 FFF:↵                 ..... Contents of address 100 are read, and cannot be modified
                                  by a ↵ alone
 0101 FFF:FFB↵              ..... New data is written
 0102 FFF:FF9↵
 * CODE ERROR *             ..... Error message is displayed when undefined code is
                                  detected; contents are written unchanged to the same
                                  address
 0102 FFF:FO5↵
 0103 FFF:A6B↵
 0104 FFF:^↵                ..... Operation returns to previous address (one less than
 0103 A6B:^↵                       current address) via input by entering ^↵
 0102 F05:F06↵
 0103 A6B:↵
 0104 FFF:ABx↵
 * COMMAND ERROR *          ..... Error is generated by data setting error; message displayed
 0104 FFF:ABC↵
 0105 FFF:/↵                ..... /↵ input terminates instruction

#SP,400↵
 * COMMAND ERROR *          ..... Since it exceeds the program area (3FF for E0C6231/
                                  62L31), an error is indicated
#SP,3FE↵
 3FE FFF:011↵
 3FF FFF:FFB↵               ..... Instruction is completed after last address in input
#
```

Format

```
#SD,<address>↵                                        (With guidance)
```

Function

Contents of the data RAM are addresses are displayed or modified.

(1) Data cannot be written to the read only area.

(2) Data in the write only area cannot be read.

Examples

```
#SD,20↵
 20 5:A↵                 ..... Contents of address 20 are modified and stored to A
 21 5:^↵                 ..... Return to previous address (one less than the current
 20 A:B↵                       address) by entering ^↵
 21 5:F↵
 22 5:/↵                 ..... Instruction terminated by /↵

#SD,FFF↵
 * COMMAND ERROR *       ..... When specification exceeds the maximum value of the
                               RAM area (7F for E0C6231/62L31), an error is indicated
#SD,70↵
 70 4:-↵
 71 F:-↵                 ..... Hyphen only displayed due to read only address;
 72 5:-↵                       data input not accepted
 73 6:-↵
 74 6:5↵
 75 8:4↵
 76 5:A↵
 77 8:9↵
 78 8:5↵
 79 A:-↵
 7A B:-↵
 :  :  :
 7E F:-↵                 ..... Command terminates after last address  entered

#SD,E50↵
 * UNUSED AREA *         ..... When an unused area has been specified, "UNUSED
                               AREA" is displayed (for E0C6246)
#SD,ECE↵
 ECE 0:F↵
 ECF 4:F↵
 * UNUSED AREA *         ..... When an unused area is entered into during data setting,
                               "UNUSED AREA" is displayed (for E0C6246)
#
```

SR *SET REGISTER*

Format

```
#SR↵                                           (With guidance)
#SR,<register name>,<data>↵
```

Function

EVA62XXCPU registers are displayed and modified.

(1) Specified data is set in specified registers.

(2) Register names can be specified as: PC, A, B, X, Y, FI, FD, FZ, FC, and SP.

Examples

```
#SR↵
 PC=0100:0105↵          . . . . . Input data and ↵ to registers you wish to modify enter
  A=   5:↵                        ↵ only to skip to the next register
  B=   A:5↵
  X= 02F:20↵
  Y= 010:1A↵
 FI=   0:1↵
 FD=   1:↵
 FZ=   0:↵
 FC=   1:0↵
 SP=  4F:^↵             . . . . . Entering the ^↵ returns operation to previous  register
 FC=   0:1↵                       (one less than the current register)
 SP=  4F:↵

#SR,X,AA↵               . . . . . X register only is changed to AA

#SR↵
 PC= 105:↵              . . . . . Current value is saved with ↵ key input
  A=   5:↵
  B=   5:↵
  X=  2A:↵
  Y=  2A:↵
      :
      :
 SP=  4F:↵
#
```

Note

Instruction will not complete with /↵ input; use ↵ up to the last register.

Format

#SXY↵ (With guidance)

Function

Current contents of the X register (Xp, Xh, Xl), Y register (Yp, Yh, Yl), and MX and MY (contents specify memory X, Y) are displayed. Contents of MX and MY can also be modified.

Examples

```
#SXY↵                            ..... Display only; ↵ alone continues operation
 X=040 MX=5:↵
 Y=030 MY=A:↵

#SXY↵
 X=040 MX=5:0↵                   ..... Sets new data to MX, MY
 Y=030 MY=A:F↵

#SXY↵
 X=070 MX=3:-                    ..... Data to read only area not accepted
 Y=FFF MY=-:OV                   ..... Input not accepted if RAM area is exceeded

#SXY↵
 X=E52 MX * UNUSED AREA *        ..... An unused area error message is displayed
 Y=1A7 MY=1:3↵                         for E52 (for E0C6246)

#
```

HC SET HISTORY CONDITION

Format

#HC,S/C/E↵

Function

Sets up the area for history extraction by means of the break point.
"[ ]" is added to the break point.

Examples

#HC,S↵	. . . . . Extracts the history from the break point
#HC,C↵	. . . . . Extracts the history before and after the break point
#HC,E↵	. . . . . Extracts the history up to the break point (default value)

SET/DISPLAY/RESET HISTORY RANGE HA, HAD, HAR

Format

```
#HA,<address 1>,<address 2>/ALL↲
#HAD↲
#HAR,<address 1>,<address 2>/ALL↲
```

Function

Sets up, indicates and clears PC address within the history extraction area.

(1) HA: Extract the range specified by <address>.
When specifying ALL, all addresses will be specified.

(2) HAD: Indicates the address of history extraction area.

(3) HAR: Do not extract the range specified by <address>.
When specifying ALL, history isn't extracted.

Examples

```
#HAR,ALL↲                  . . . . . Clears the entire history extraction area

#HA,300,400↲               . . . . . Specifies history extraction area

#HA,100,200↲

#HA,500,500↲

#HAD↲                      . . . . . Indicates history extraction area
 *HA 0100..0200
 *HA 0300..0400
 *HA 0500
#
```

3.3 Break and Go Command Group

BA, BAR *SET/RESET BREAK ADDRESS CONDITION*

Format

```
#BA,<address 1>,<address 2>,<address 3>,<address 4>↵
#BAR,<address 1>,<address 2>,<address 3>,<address 4>↵
```

Function

Sets break condition for the PC.

(1) BA: The value indicated at the specified address is set to the break condition. Multiple addresses are set by using commas to divide them. Consecutive addresses are set by separating entries with two period marks (.). Entering <address 3>..<address 4> sets a break condition such that <address 3> ≤ PC ≤ <address 4>.

(2) BAR: Can be cleared separately from break condition set by BA.

(3) Addresses which can be entered by a single BA or BAR instruction can be set multiple times in a single line (80 columns).

(4) When the BA command is executed several times, previous settings are valid.

(5) When the BM command is executed, all BA conditions are canceled.

(6) When entering the GO command at a break, the BA condition may enter the clear mode or a condition retaining mode. (Refer to the BRKSEL command.)

Examples

```
#BA,100,200,101,1FF↵ ..... Break condition set at addresses 100, 200, 101 and 1FF

#BA,300..3FF↵        ..... Break conditions set at addresses 300 to 3FF

#BAR,100,200..3FF↵   ..... Break conditions canceled at address 100 and addresses
                           200 to 3FF (although break conditions were not set at
                           addresses 201 to 2FF, no error occurs even with BAR
                           setting)
#BC↵
 BA 0201             ..... BA condition is displayed by BC command
 BA 02FF
 BD NONE
 BR NONE
   :
#
```

ET/RESET BREAK DATA CONDITION BD, BDR

Format

#BD↵ (With guidance)

#BDR↵

Function

Break condition set for data RAM read/write area.

(1) BD: Break condition set for RAM data address, data, and R/W. Address can be set at one point, data set from addresses 0 to F or masked, and the R/W area set to read, write, or masked. A break is generated when the three conditions specified by address, data, and R/W coincide.

(2) BDR: Cancels the condition set by BD command.

(3) A break condition set by the BD command is functional at one point only, but can be mixed with BA and BR commands.

(4) A BD condition can be canceled by executing the BM command.

Examples

```
#BD↵
 ADDR ---:074↵          ..... A hyphen (-) is displayed when the BD condition is
 DATA   -:5↵                  absent.  At address 74, the number 5 is entered as data
 R/W    -:*↵                  and the R/W is masked (*)
```

In the above example, a break is set for when the number 5 is written to or read from the data RAM address 074.

```
#BD↵
 ADDR 074:↵             ..... When no setting modification is made, hitting the ↵
                              key continues the operation to the next setting
 DATA  5 :1*1*B↵        ..... Data is masked
 R/W   * :W↵            ..... Sets the R/W function to write
```

At the current settings, a break is generated when 1 is written to 2^3 bit and 2^1 bit at data RAM address 74.

```
#BDR↵                   ..... All BD conditions are cleared
#BD↵
 ADDR ---:↵             ..... Entering ↵ after canceling BD setting confirms
#                             cancellation
```

BR, BRR *SET/RESET BREAK REGISTER CONDITION*

Format

#BR↵ (With guidance)

#BRR↵

Function

A break condition is set in the EVA62XXCPU registers A, B, FLAG, X (Xp, Xh, Xl,) or Y (Yp, Yh, Yl).

(1) BR: A break condition is set in the target registers A, B, FLAG, X (Xp, Xh, Xl,) or Y (Yp, Yh, Yl). The break condition in each register can be masked (a masked register can generate a break in another register, whatever the specified value). Break is induced when the values of each register correspond to the set values in the internal CPU registers.

(2) BRR: Cancels a break condition set by BR command.

(3) A break set by the BR command is operative at one point. BA and BD settings can be mixed.

(4) A BR condition can be canceled by executing the BM command.

Examples

```
#BR↵
 A        -:C↵                 . . . . . A hyphen (-) is displayed when a BR condition is not
 B        -:*↵                           set. Break condition is sequentially set
 FI       -:1↵
 FD       -:*↵                 . . . . . Enter an asterisk (*) mark to indicate masking
 FZ       -:0↵                           This induces  a break unrelated to the FD value
 FC       -:*↵
 X      ---:040↵
 Y      ---:^↵                 . . . . . If a parameter is mis-set, entering the ^ key will return
 X      ---:041↵                         the operation to the previous setting (one less than the
 Y      ---:030↵                         current setting)
```

A break condition set as described above, where A=C, FI=1, FZ=0, X=41, and Y=30.

```
#BR↵
 A        C:↵                  . . . . . Reads a previously set break condition
 B        *:↵                            When no setting modification is made, hitting the ↵
 FI       1:*↵                           key continues the operation to the next setting
 FD       *:↵
 FZ       0:*↵
 FC       *:↵
 X      041:042↵
 Y      030:*↵
```

Two break conditions where A=C and X=42 are described above.

SET/RESET BREAK REGISTER CONDITION **BR, BRR**

Format

#BR↵ (With guidance)
#BRR↵

Examples

```
#BRR↵                        . . . . . A BR condition is cleared by the BRR command
#BR↵
 A       -:↵                 . . . . . Entering ↵ after canceling BR setting confirms
                                       cancellation
#BR↵
 A       -:0↵
 B       -:0↵
 FI      -:*↵
 FD      -:*↵
 FZ      -:*↵
 FC      -:*↵
 X     ---:40↵
 Y     ---:30↵
```

A break condition is set wherein A=0, B=0, X=40, and Y=30.

```
#BR↵
 A       0:↵
 B       0:5↵
 FI      *:/↵                . . . . . Entering / when no further setting changes are desired
#                                      completes the instruction
```

A break condition is set where A=0, B=5, X=40, and Y=30.

Notes

(1) The target system operates in real time even when a GO command is executed after setting a BR condition.

(2) Each model (E0C62XX/62*XX) has a different RAM area, and XY settings in a BR command can be set to FFF.

BM, BMR *SET/RESET BREAK MULTIPLE CONDITION*

Format

```
#BM↵                                    (With guidance)
#BMR↵
```

Function

Sets the compound break function for multiple breaks when all conditions for the EVA62XXCPU PC, data RAM access, and register values coincide.

(1) Although the BA, BD and BR instructions can be set independently, the BM command generates a break when all conditions for the PC, data RAM access, and register values coincide. In other words, it can be thought of as the AND setting for the BA, BD and BR commands.

(2) Previously set BA, BD and BR conditions are canceled by the BM instruction. Also, the BM setting is canceled when the BA, BD and/or BR instructions are set after the BM instruction is set.

(3) The BMR command cancels the BM instruction.

(4) A break is set at only one point by the BM command. Each register setting can be masked.

Example

```
#BM↵
 PC    ----:100↵
 ADDR   ---:70↵
 DATA     -:A↵
 R/W      -:*↵
 A        -:*↵
 B        -:*↵
 FI       -:*↵
 FD       -:1↵
 FZ       -:*↵
 FC       -:1↵
 X      ---:*↵
 Y      ---:3E↵
```

. . . . . A hyphen (-) is displayed when a BM condition is canceled.
Break condition is set where PC=100, RAM access=70, RAM data=A, D and C flags=1, and Y register=3E.
During execution of the instructions at address 100, a break occurs when the following conditions coincide: RAM at address 70 is accessed, read/write data A, FD and FC are set, and Y register is 3E. (Valid for break during program loop.)

. . . . . The point at which the break is placed is masked by an asterisk (*) mark.

Format

```
#BM↵                                                  (With guidance)
#BMR↵
```

Examples

```
#BM↵
 PC     100:*↵           ..... PC mask
 ADDR    70:71↵
 DATA     A:^↵           ..... Enables return to previous operation when ^ key is
 ADDR    71:72↵                entered
 DATA     A:↵            ..... Previous setting retained when ↵ alone is entered
 R/W      *:W↵
 A        *:↵
 B        *:↵
 FI       *:↵
 FD       1:↵
 FZ       *:↵
 FC       1:↵
 X        *:70↵
 Y       7E:↵
```

As shown above, a break is generated when data A is written to RAM address 72 if CPU register X=70, Y=7E, FD=1 and FC=1.

```
#BM↵
 PC       *:100↵
 ADDR    71:/↵           ..... Entering/↵ does not alter later settings; adds PC=100 to
                               above conditions
#BMR↵                    ..... Cancels condition set by BM command

#BM↵
 PC    ----:↵            ....  Entering ↵ after canceling BM setting confirms
#                              cancellation
```

Notes

(1) Use of the BM command automatically cancels BA, BD and BR commands.

(2) This instruction runs a break comparison only during execution with memory access. The above described limitations remain even when ADDR, data and R/W are masked. Therefore, a break will not occur when the instruction does not access data memory even if the PC and register values coincide.

(3) Each model (E0C62XX/62*XX) has a different RAM area, and XY settings in a BM command can be set to FFF.

BC BREAK CONDITION DISPLAY

Format

```
#BC↵
```

Function

Displays the current break condition.

Examples

```
#BC↵                     . . . . . Break condition is verified after power on. All break
 * BA NONE                         conditions are canceled.
 * BD NONE
 * BR NONE
 * BM NONE
 * BREAK ENABLE MODE     . . . . . Enters break enable mode
 * BREAK STOP MODE       . . . . . Enters break stop mode
 * TIME COUNT MODE       . . . . . Enters real-time mode

#BA,100,101↵

#BC↵                     . . . . . Reads after address break condition set Break condition
 * BA 0100..0101                   confirmed
 * BD NONE
 * BR NONE
 * BM NONE
 * BREAK ENABLE MODE
 * BREAK STOP MODE
 * TIME COUNT MODE

#BRES↵

#BA,100,102↵

#BC↵
 * BA 0100               . . . . . Displays multiple executions of BA condition when
 * BA 0102                         addresses are not consecutive
   :
   :
#
```

SRESET ALL BREAK CONDITION BRES

Format

#BRES↵

Function

All break conditions (BA, BD, BR, or BM settings) are canceled.

Example

```
#BRES↵
#BC↵
 * BA NONE
 * BD NONE
 * BR NONE
 * BM NONE
 * BREAK ENABLE MODE
 * BREAK STOP MODE
 * TIME COUNT MODE
#
```

Note

Although the break condition is canceled, the break mode (enable/disable, trace, stop, time/stop) is still operative.

G GO TARGET PROGRAM

Format

```
#G↵
#G,<address>↵
#G,R↵
```

Function

This instruction runs the target program. When a break condition is detected, program execution is halted and the break status is displayed to complete the instruction.

1. Setting the starting address

(1) When an address is entered, the run starts from that address.

(2) With an R setting the EVA62XXCPU is reset, and the run starts from the reset address 0100.

(3) When the address and R setting are defaulted, the run starts from the current address (PC which displays the status during the previous break).
When G↵ is entered after power on, the run starts from address 0100, but the EVA62XXCPU is not reset.

2. Break Mode and Break Condition

Item	Break mode(note)	Break condition	Comments
1	BE mode and Break stop mode	* Reset switch * Break switch * Break set commands (BA, BD, BR, BM) * ESC input	Mode at power on.
2	BE mode and Break trace mode	* Reset switch * Break switch * ESC input	When the break condition and EVA62XXCPU executed cycle coincide, the break status alone is displayed and the GO command is restarted.
3	BSYN mode and Break stop mode	* Reset switch * Break switch * ESC input	When the break condition and EVA62XXCPU executed cycle coincide, a pulse is output to the SYNC pin.

(Note) Refer to "Break mode and break function" in section 2.3 for more information on the break mode.

Format

```
#G↵
#G,<address>↵
#G,R↵
```

Function

3. Display During Execution of GO Instruction

Item	Display mode (note)	Display method
1	On-the-fly display mode	#G↵ *PC=xxxx ... Sampling of the PC is displayed about every 500ms. HALT message is displayed during halt.
2	On-the-fly inhibit mode	#G↵ Execution status is not displayed.

(Note) Refer to "Display during run mode and during break" in section 2.3 for information on the display modes.

4. Break Display

```
#G↵
 *PC=xxxx
 *EMULATION END STATUS = BREAK HIT          ..... (A)
 *PC=0100 A=0 B=0 X=70 Y=00 F=ID.C SP=10    ..... (B)
 *RUN TIME=xxx mS                           ..... (C)
```

> The break status is displayed.

(A) BREAK HIT, ESC KEY, BREAK SW displays appear in parts. When the reset switch is depressed, the message, `*ICE6200 RESET SW TARGET*`, is displayed without displaying the break status, and the next instruction is awaited.

(B) Register contents are displayed in part when PC (next executed address) is stopped.

(C) The execution time or executed number of steps set by TIM command are displayed in part. (Refer to page V-93 for details of the TIM command.)

G GO TARGET PROGRAM

Format

```
#G↵
#G,<address>↵
#G,R↵
```

Examples

```
#OTF↵                     ..... On-the-fly set command
 * ON THE FLY ON *

#BE↵                      ..... Break enable set command
 * BREAK ENABLE MODE *

#BT↵                      ..... Break stop mode set command
 * BREAK STOP MODE *
```

These settings are set at power on; default is command input

```
#G,R↵                     ..... Target and evaluation board is reset; run starts from reset
                                address (0100)
 *PC=xxxx                 ..... PC display is cyclic
 *EMULATION END STATUS = BREAK HIT        ..... (A)
 *PC=01FF A=5 B=0 X=70 Y=05 F=..ZC SP=20  ..... (B)
 *RUN TIME=100mS                          ..... (C)
```

(A) Break displayed through break condition (BA condition set at 01FE)
(B) F is expresses reset bit and (.) bit as English letter
(C) Run time is 100ms

Format

```
#T,<address>,<step number>↵
#T,<address>↵
#T,,<step number>↵
#T↵
```

Function

Executes trace, and single step actions of programs.

(1) The specified portion of the target program executes with a frequency indicated by the number of steps from the specified address (65535 possible in decimal code). The PC, instruction word and register contents are displayed with each execution.

(2) When the step number is defaulted, only one step is executed.

(3) When the address is defaulted, the specified number of steps is executed from the current PC (PC at which the previous T command completed).

(4) When both address and step number are defaulted, only one step is executed from the current PC. When this setting occurs after power on, one step is executed from PC=0100.

(5) When the step number is one (#T, <address> or #T), the instruction does not terminate after one step, but a further step is executed by the "SP" key input, at which time the instruction can be terminated by the "ESC" key input.

(6) In (1) above, the instruction is terminated by "ESC" key input.

Example

```
#T,100,3↵
 *PC=0100 IR=FFF NOP7     A=0 B=0 X=00F Y=00F F=IDZC SP=10
 *PC=0101 IR=E05 LD   A,5 A=5 B=0 X=00F Y=00F F=IDZC SP=10
 *PC=0102 IR=B05 ADC XH,5 A=5 B=0 X=051 Y=00F F=IDZC SP=10
     |           |                              |
Executed PC       Command code           Correctors displayed when the flag is set
is displayed      and mnemonic           and/or reset (After executing three steps,
                  are displayed          the current PC is 0103)
#
```

T SINGLE STEP TRACE

Format

```
#T,<address>,<step number>↵
#T,<address>↵
#T,,<step number>↵
#T↵
```

Examples

```
#T↵            Program executes sequentially in steps from current PC (=103) via "SP" key.
 *PC=0103 IR=FDF RET    A=5 B=0 X=04F Y=03F F=IDZC SP=013 ..... "SP"
 *PC=01AA IR=AD1 OR A,B A=5 B=0 X=04F Y=03F F=ID.C SP=013 ..... "ESC"
                   Instruction is terminated by "ESC" key.

#T↵
 *PC=01AB IR=xxx PSET  2 A=x B=x X=xxx Y=xxx F=xxxx SP=013
 *PC=01AC IR=xxx JP   10 A=x B=x X=xxx Y=xxx F=xxxx SP=013 ..... "ESC"

#              Because the PSET command is used in relation to the subsequent instruction,
               two command executions can be set by invoking the T command once.
#T↵
 *PC=01AD IR=xxx HALT _
                      └─ Cursor
```

When the HALT command is executed by the T command, the command mnemonics are displayed until the target interrupt as described above, but the register value is not displayed. When an interrupt is properly input, the register is displayed and the next "SP" is awaited. The SP input restarts the program after the interrupt routine.

When the target interrupt never occurs, the instruction can be forced to terminate by using the "ESC" key. At that point, the HALT and T commands terminate, but the HALT command executes from the next address when the T command is operative.

Notes

(1) The T command does not operate in real time. Therefore, the target timer is renewed.(For details refer to "Limitations during emulation" in section 2.3.)

(2) When the H command is input after executing this command, the message, `*NO HISTORY DATA*`, is displayed. Therefore, the G command must be used to analyze history data.

SINGLE STEP TRACE & LAST INFORMATION DISPLAY U

Format

```
#U,<address>,<step number>↵
#U,,<step number>↵
```

Function

Executes trace and single step actions of programs and indicates final results alone.

(1) The target program is executed from the address specified in <address> for the frequency specified in <step number> (65535 possible in decimal code), but the results are not displayed until after the final instruction is completed.

(2) When the address is defaulted, execution starts from the current PC for the specified number of steps.

Examples

```
#U,100,5↵
 *PC=01AA IR=ADI OR   A,B   A=5 B=0 X=04F T=03F F=ID.C SP=13

#U,,1↵
 *PC=01AB IR=FFF NOP7       A=5 B=0 X=04F Y=03F F=ID.C SP=13

#
```

Notes

(1) The U command does not run in real time, so the target timer is renewed. (For details refer to "Limitations during emulation" in section 2.3.)

(2) When the H command is input after executing this command, the message, `*NO HISTORY DATA*`, is displayed. Therefore, the G command must be used to analyze history data.

BE, BSYN *BREAK ENABLE MODE SET/BREAK DISABLE & SYNC MODE SET*

Format

#BE↵
#BSYN↵

Function

Sets the break enable mode and break disable mode.

(1) BE: Sets the break enable mode. A break is generated when the BA, BD, BR or BM conditions coincide with the EVA62XXCPU state.

(2) BSYN: Sets the break disable (synchronous) mode. When the BA, BD, BR or BM conditions coincide with the EVA62XXCPU state, a pulse is output to the ICE6200 SYNC pin and a break is not generated.

(3) At power on, the break enable mode is operative.

Examples

```
#BE↵

 * BREAK ENABLE MODE

#BSYN↵

 * BREAK DISABLE MODE
 * BREAK STOP MODE
```

Note

Refer to "Break mode and break function" in section 2.3 for details of break enable/disable functions.

Format

#BT↵ (Toggle)

Function

Selects the break stop mode or the break trace mode. Setting is reversed with each command input. At power on, the break stop mode is operative.

Examples

```
#BT↵
 * BREAK TRACE MODE   .....Since the stop mode is operative at power on, the trace
 * BREAK ENABLE MODE       mode is set by command input

#BT↵
 * BREAK STOP MODE    .....The setting is reversed by command input

#
```

Note

Refer to "Break mode and break function" in section 2.3 for details of break stop and trace modes.

BRKSEL *BREAK ADDRESS MODE SELECT*

Format

```
#BRKSEL,REM↵
#BRKSEL,CLR↵
```

Function

After setting the break address condition (BA), the program runs until stopped by a break hit; the settings then remain or clear the previously set BA condition. The clear mode (CLR mode) is operative at power on. The BA condition remain mode (REM mode) is used when multiple break conditions are set and the program runs to consecutive break points. The BA condition clear mode is used to debug when the break point is changed with each break.

Examples

```
#BA,0100↵
#BRKSEL,REM↵                                ..... Remain mode is set
#BC↵
 BA 0100
    :
#G↵
 *PC=100
 *EMULATION END STATUS = BREAK HIT ..... Break is generated when break
 *RUN TIME=10mS                                  condition hits
#BA,200↵                                    ..... New break condition is set
#BC↵
 BA 0100                                    ..... Pre-break condition remains
 BA 0200
    :
#BRKSEL,CLR↵                                ..... Clear mode is set
#G↵
 *PC=101
 *EMULATION END STATUS = BREAK HIT ..... Break condition hits
 *RUN TIME=30mS
#BA,300↵                                    ..... New break condition is set
#BC↵
 BA 0300                                    ..... Pre-break condition is canceled
    :
#BA,350,3A0↵
#BC↵
 BA 0300                                    ..... After break condition remains
 BA 0350
 BA 03A0
#
```

3.4 File Command Group

RF, RFD *READ PROGRAM/DATA FILE*

Format

```
#RF,<file name>↵
#RFD,<file name>↵
```

Function

Loads files onto the emulation memories.

(1) RF: The hex file specified in <file name> is loaded in the emulation program memory.

(2) RFD: The hex file (data RAM) specified in <file name> is loaded in the data memory.

Examples

```
#RF, C6200A0↵         ..... C6200A0H.HEX file and C6200A0L.HEX file are loaded
                            in the program memory
#RFD,WORK↵            ..... WORKD. HEX file is loaded in the data memory
```

Notes

(1) When the memory area is overreached (address 3FF in program memory; address 7E in data memory for E0C6231/62L31) or an FD file format error is detected, an error message, `*FILE DATA FORMAT ERROR*`, is displayed and the instruction terminates. The contents of the emulation program memory and data memory are not secured.

(2) I/O memory, segment memory and unused area are not loaded into data memory.

(3) The files are in hexadecimal format. (For details, refer to appendix B.)

(4) The file format is created by the E0C62XX/62*XX cross assembler. (For details, refer to the "E0C62XX/62*XX Cross Assembler Manual".)

(5) "ESC" key is invalid during instruction execution.

(6) When an input error (FD error, not drive error) is detected on the PC side, control is returned to the operating system, and therefore, the ICS62XX is terminated.

(7) When an undefined instruction is detected, an error message is displayed and the ICS62XX program terminates. (For details, refer to Chapter 4.)

VF, VFD

Format

```
#VF,<file name>↵
#VFD,<file name>↵
```

Function

Compares the contents of the emulation memories with those of files.

(1) VF: The contents of the emulation program memory and the hex file specified in <file name> are collated.

(2) VFD: The contents of the emulation data memory (data RAM) and the hex file specified in <file name> are collated.

Examples

```
#VF,C6200A0↵                 ..... C6200A0H.HEX and C6200A0L.HEX files and the
 ADDR  FD:ICE  ┐                   program memory are collated
 0100 FFF:FFC  │             ..... The contents of the FD address and the memory are
 0300 FFC:FFB  ┘                   displayed only when the collated data do not agree.

#VFD,DATA↵
 ADDR  FD:ICE
  001   1:3
 * ESC *                     ..... Display can be interrupted by "ESC" key input
```

Notes

(1) Notes (1), (3), (4) and (6) in page V-82 are applicable to these instructions.

(2) "ESC" key is valid during error message display; "ESC" key input terminates the instruction.

(3) I/O memory, segment memory and unused area in data memory cannot be compared.

WF, WFD *WRITE PROGRAM/DATA FILE*

Format

```
#WF,<file name>↵
#WFD,<file name>↵
```

Function

Saves the contents of the emulation memories to files.

(1) WF: The contents of the emulation program memory are saved to the file specified in <file name>.

(2) WFD: The contents of the emulation data memory (data RAM) are saved to the file specified in <file name>.

Examples

```
#WF,C6200A0↵              ..... Program memory is saved to C6200A0H.HEX and
                                C6200A0L.HEX files.
#WFD,WORK↵                ..... Data memory is saved to WORKD.HEX file.

#WF,ABCDEFGH↵
 * COMMAND ERROR *        ..... An error occurs if the file name exceeds seven characters.
```

Notes

(1) Notes (3), (4), (5) and (6) of page V-82 are applicable to these commands.

(2) I/O memory, segment memory and unused area in data memory cannot be saved.

Format

```
#CL,<file name>↵
#CS,<file name>↵
```

Function

Loads the contents of the emulation memories of ICE6200 and the contents of each setting from files or save them to files.

(1) CL: The program and data from the file specified in <file name> are loaded into the program and data memories respectively. Each type of command set condition is loaded, also.

(2) CS: The contents of the current ICE6200 emulation program memory and data memory as well as each command set condition (break state, etc.) are saved to the file specified in <file name>.

The loaded and saved contents are as follows:

- Target program (emulation program)
- Target data (emulation data)
- Current register values of the EVA62XXCPU (A, B, X, Y, F, SP, PC)
- Current break data (conditions set by BA, BD, BR and/or BM commands)
- Break mode data (execution time/steps, break stop/break trace, break enable/break SYNC, with/without on-the-fly).

These instructions are valid when power is switched off and reapplied.

Examples

```
#CS,TEST↵            ..... Current ICE6200 set conditions are saved to the
     :                     TESTC.HEX file; contents of emulation program
             :             memory are saved to the TESTH.HEX file, while
Power OFF                  contents of data memory are saved to the TESTD.HEX
Power ON                   file
             :
#CL,TEST↵            ..... Contents  saved in CS  are loaded;  ICE6200 returns to
                           the status prior to power OFF
```

Notes

(1) Notes (1), (2), (3), (4), (5), and (6) of page V-82 are applicable to these commands.

(2) A file name of up to seven characters may be specified as <file name> for #CS,<file name>.

3.5 ROM Command Group

RP LOAD ROM PROGRAM

Format

#RP↵

Function

The program is loaded to the ICE6200 emulation memory from the ROM at the ICE ROM socket (high and low). The FF ROM data is unassembled.

Examples

```
#RP↵
 * NO ROM H/L *         . . . . . Error is generated because high and low ROM are
                                  unassembled
#RP↵
 * NO ROM H *           . . . . . Error generated because high side ROM is unassembled

#RP↵
                        . . . . . Contents of ROM are properly loaded
#
```

Notes

(1) Refer to the ROM commands for information on the valid loading region.

(2) When undefined code is detected, the ICS62XX program is terminated and control returns to the operating system.

Format

#VP↵

Function

The contents of the ICE6200 ROM socket (high and low) and the ICE emulation memory are compared. When they do not agree, the data contents are displayed.

Examples

```
#VP↵

#                                  When the results of the comparison are acceptable,  the
 :                                 program execution is at waiting until ordering the next
 :                                 instruction

#VP↵
 ADDR ROM:ICE
 0100 FFF:FFC               .....  All non-agreeing data (ROM address, ROM contents,
      0300 0FF:0FC                 emulation memory contents) are displayed
  :    :   :
 03FF 000:001

#VP↵
 * NO ROM H *               .....  Error because high side ROM is unassembled

#VP↵
 ADDR ROM:ICE
 0100 FFF:FFC
 0300 0FF:0FC
  :    :   :
 * ESC *                    .....  Processing is interrupted by "ESC" key input, and the
                                   program execution is at waiting until entering the next
#                                  command
```

ROM ROM TYPE SELECT

Format

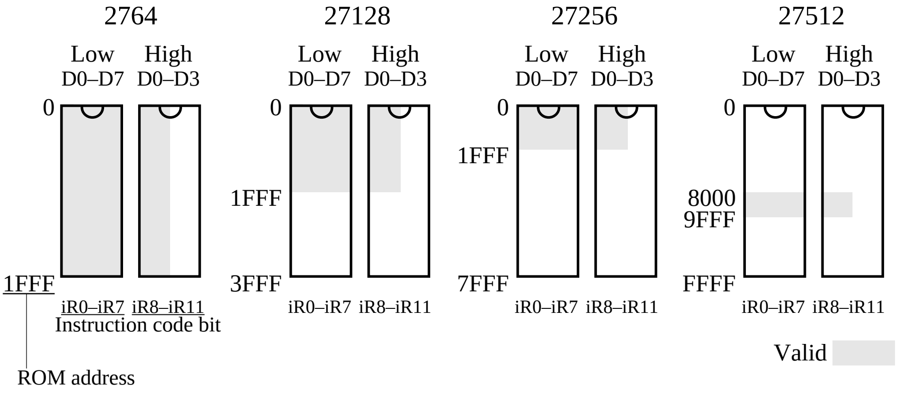
#ROM↵ (With guidance)

Function

The ROM type which is assembled to the ICE6200 ROM socket is set.

(1) 2764, 27128, 27256 or 27512 can be selected.

(2) The region to which the ROM type is loaded is described below.



Examples

```
#ROM↵
 *ROM 64:↵                 ..... Initial value set at 64
                                 When ↵ input alone is entered without modification of
                                 data, the execution is at waiting until entering the next
                                 command
#ROM↵
 *ROM 64:256↵              ..... Setting changed to 27256
#ROM↵
 *ROM 256:FF↵              ..... Setting other than 64, 128, 256 or 512 results in an error
 * COMMAND ERROR *
#ROM↵
 *ROM 256:↵
#
```

Note

ROM which is assembled to the high and low IC sockets should be the same types.

3.6 Control Command Group

I   *INITIALIZE TARGET CPU*

Format

```
#I↵
```

Function

Resets the EVA62XXCPU.

Resets the EVA62XXCPU, but the ICE6200 set conditions (break, etc.) are affected.

Example

```
#I↵

#                               The execution is at waiting until entering the next command
```

Format

#TIM↵ (Toggle)

Function

When the GO command is entered, the execution time counter, execution time count mode or step count mode is operative. The execution time count mode is the default at power on. The setting is reversed at each command input.

Examples

```
#TIM↵
 * STEP COUNT MODE    .....Since the mode after power supply is  the time count
                           mode, entering a command toggles the setting to step
                           mode
#TIM↵
 * TIME COUNT MODE    .....Setting is reversed with each command input

#
```

Note

Refer to "Measurement during command execution" in section 2.3 for more details on the time count and step count modes.

OTF ON THE FLY MODE SET

Format

#OTF↵ (Toggle)

Function

Selects whether or not to run the on-the-fly display during GO execution.
On-the-fly display mode is the default at power on. Use the display off mode when the host is connected to a printer.

Examples

```
#OTF↵
 * ON THE FLY OFF      ..... Since the display mode is the default at power on, a
                             command input toggles to the display off mode
#OTF↵
 * ON THE FLY ON       ..... On-the-fly display mode is operative

#G↵
 * PC=xxxx             ..... Displays fixed cycle of EVA62XXCPU's executed PC
   :
   :
   :
#OTF↵
 * ON THE FLY OFF

#G↵
                       ..... PC is not displayed
```

Note

For more details about the on-the-fly function, refer to "Display during run mode and during break" in section 2.3.

Format

```
#Q↵
```

Function

Terminates the ICS62XX program and returns control to the operating system.

Example

```
#Q↵

B>                          ..... Awaits control by host computer operating system

B>ICS62XX↵                  ..... Reloads the ICE
... Epson logo is displayed for about one second ...
 * ICE POWER ON RESET *
 * DIAGNOSTIC TEST OK *

#                           ..... Awaits ICE instruction
```

3.7 HELP Command

# HELP

Format

#HELP↵ (With guidance)

#HELP,n↵ (n=1 to 8)

Function

Displays the ICS62XX commands.

(1) All commands are displayed on a single screen when no option (,n) is set.

(2) Displays the related commands when an option (,n) is set.
Explanations for commands of the same group are displayed.

n value	Command group
1	DISPLAY COMMAND
2	SET COMMAND
3	BREAK and GO COMMAND
4	FILE COMMAND
5	ROM COMMAND
6	CONTROL COMMAND
7	ALL COMMAND DISPLAY
8	BASIC COMMAND DISPLAY

Examples

```
#HELP↵
```

Refer to HELP messages on next page

```
 KEY IN 1.8 ENTER OR ENTER ONLY : 1↵
```

Displays DISPLAY COMMAND
(Refer to next page)

```
#HELP,F↵
 * COMMAND ERROR *

#
```

..... Error is generated if a value other than 1 to 8 is entered

# HELP

Format

#HELP↵ (With guidance)

#HELP,n↵ (n=1 to 8)

Examples

```
#HELP↵
 1.DISPLAY COMMAND           #L   #DP  #DD  #DR  #H   #HB  #HG  #HS  #HSW
#HSR
                             #HP  #CHK #DXY #CVD #HAD
 2.SET COMMAND               #A   #FP  #FD  #MP  #MD  #SP  #SD  #SR  #SXY #HC
                             #HA  #HAR #HPS #CVR
 3.BREAK and GO COMMAND      #BA  #BD  #BR  #BM  #BAR #BDR #BRR #BMR #BRES
                             #BC  #G   #T   #U   #BSYN #BE #BT  #BRKSEL
 4.FILE COMMAND              #RF  #VF  #WF  #RFD #VFD #WFD #CL  #CS
 5.ROM COMMAND               #RP  #VP  #ROM
 6.CONTROL COMMAND           #I   #TIM #OTF #Q
 7.ALL COMMAND DISPLAY
 8.BASIC COMMAND DISPLAY

 KEY IN 1..8 ENTER or ENTER ONLY :↵
#
```

```
#HELP,1↵
 1.DISPLAY COMMAND
 (1)#L,addr1,addr2   program code and mnemonic display.
 (2)#DP,addr1,addr2  program area HEX display.
 (3)#DD,addr1,addr2  data area HEX display.
 (4)#DR              register data display.
 (5)#H,addr1,addr2   history data display.
 (6)#HB or #HG       history data display BACK or GO NEXT.
 (7)#HS,addr         history serch and display.
 (8)#HSW,addr        memory write history serch and display.
 (9)#HSR,addr        memory read history serch and display.
(10)#HP              current history pointer display.
(11)#CHK             ice initial self test information display.
(12)#DXY             X,Y register and MX,MY data display.
(13)#CVD,addr1,addr2 coverage area display.
(14)#HAD             history PC area information display.
```

# HELP

Format

#HELP↵ (With guidance)

#HELP,n↵ (n=1 to 8)

Examples

```
#
#HELP,2↵
 2.SET COMMAND
 (1)#A,addr                  assemble program.
 (2)#FP,addr1,addr2,data     fill program addr1 to addr2 by data.
 (3)#FD,addr1,addr2,data     fill data addr1 to addr2 by data.
 (4)#MP,addr1,addr2,addr3    move program from addr1..addr2 to addr3.
 (5)#MD,addr1,addr2,addr3    move data from addr1..addr2 to addr3.
 (6)#SP,addr                 program area patch.
 (7)#SD,addr                 data area patch.
 (8)#SR or #SR,reg,data      register patch.
 (9)#SXY                     MX,MY patch.
(10)#HC,S/C/E                history Start/Center/End set.
(11)#HA,addr1,addr2          set PC addr1..addr2 save to history memory.
     (#HA,ALL)               (all data save.)
(12)#HAR,addr1,addr2         inhibit PC addr1..addr2 save to history memory.
     (#HAR,ALL)              (all reset.)
(13)#HPS,addr                set history pointer.
(14)#CVR                     reset coverage information.

#

#HELP,3↵
 3.BREAK and GO COMMAND
 (1)#BA,addr,...    set break address.
 (2)#BD             set break data condition.
 (3)#BR             set break register condition.
 (4)#BM             set break address,data,register multiple condition.
 (5)#BAR            reset break address.
 (6)#BDR            reset break data condition.
 (7)#BRR            reset break register condition.
 (8)#BMR            reset break address,data,register multiple condition.
 (9)#BRES           reset all break condition.
(10)#BC             break condition display.
(11)#G or #G,addr   GO current address or GO from set addr.
(12)#G,R            GO after reset cpu.
(13)#T,addr,step    single step run and display break information.
(14)#U,addr,step   single step run in ICE. and display last break information.
(15)#BSYN           set break disable mode.
(16)#BE             set break enable mode.
(17)#BT             set and reset break trace made. (alternate)
(18)#BRKSEL,CLR/REM set break address clear mode or remain mode.

#
```

# HELP

Format

#HELP↵ (With guidance)

#HELP,n↵ (n=1 to 8)

Examples

```
#HELP,4↵
 4.FILE COMMAND
 (1)#RF,file    program load.
 (2)#VF,file    program verify.
 (3)#WF,file    program save.
 (4)#RFD,file   RAM data load.
 (5)#VFD,file   RAM data verity.
 (6)#WFD,file   RAM data save.
 (7)#CL,file    program,RAM data,break condition load.
 (8)#CS,file    program,RAM data,break condition save.

#

#HELP,5↵
 5.ROM COMMAND
 (1)#RP       program load from ROM.
 (2)#VP       program verify ice:ROM.
 (3)#ROM      ROM type select. (64,128,256,512)

#

#HELP,6↵
 6.CONTROL COMMAND
 (1)#I        reset target CPU.
 (2)#TIM      set step count mode or time count mode. (alternate)
 (3)#OTF      set on-the-fly display mode or inhibit mode. (alternate)
 (4)#Q        program exit.

#

#HELP,8↵
 8.BASIC COMMAND
 (1)#L,addr1,addr2   program code and mnemonic display.
 (2)#DD,addr1,addr2  data area HEX display.
 (3)#DR              register data display.
 (4)#BC              break condition display.
 (5)#H,addr1,addr2   history data display.
 (6)#A,addr          assemble program.
 (7)#SP,addr         program area patch.
 (8)#SD,addr         data area patch.
 (9)#SR              register patch.
(10)#BA,abbr,...     set break address.
(11)#BD              set break data condition.
(12)#BR              set break register condition.
(13)#BM              set break address,data,register multiple condition.
(14)#BRES            reset all break condition.
(15)#G or #G,addr    GO current address or GO from set address.
(16)#T,addr,step     single step run and display break information.
(17)#CL,file         program,RAM data,break condition load.
(18)#CS,file         program,RAM data,break condition save.
(19)#I               reset target CPU.
(20)#Q               program exit.

#
```

CHAPTER 4 ERROR MESSAGE SUMMARY

Error message *** COMMUNICATION ERROR OR ICE NOT READY ***

Meaning ICE6200 is disconnected or power is OFF.

Recovery procedure Switch OFF the host power supply, connect cable, and reapply power. Or switch ON power to ICE6200.

Error message *** TARGET DOWN(1) ***

Meaning Evaluation board is disconnected. (Check at power ON)

Recovery procedure Switch OFF power to ICE, and connect the evaluation board. Then, apply power to ICE6200.

Error message *** TARGET DOWN(2) ***

Meaning Evaluation board disconnected. (Check at command execution)

Recovery procedure Switch OFF power to ICE, and connect the evaluation board. Then, apply power to ICE6200.

Error message *** UNDEFINED PROGRAM CODE EXIST ***

Meaning Undefined code is detected in the program loaded from ROM or FD. (ICE program terminates)

Recovery procedure Convert ROM and FD data with the E0C62XX cross assembler, then restart the ICE6200.

Error message *** COMMAND ERROR ***

Meaning A miss occurs by command input.

Recovery procedure Reenter the proper command.

Error **No response after power on.**

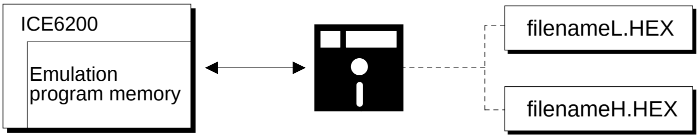
Meaning The ICE-to-HOST cable is disconnected on the host side.

Recovery procedure Connect the cable.

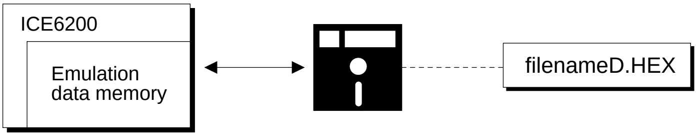
APPENDIX A. FD FILE CONFIGURATION

The ICE6200 uses the types of FD files listed below. All are in hexadecimal file format. For more details on hex file format, refer to appendix B.

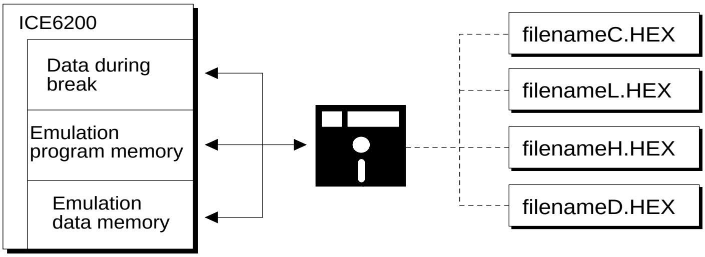
<table>
<tr><th>Command</th><th>File</th></tr>
<tr><td><b>WF,&lt;filename&gt;↵</b>
<b>RF,&lt;filename&gt;↵</b>
<b>VF,&lt;filename&gt;↵</b></td><td>
The high order 4 bits and low order 8 bits of program memory are output (or input, or compared) to two files: filenameH.HEX and filenameL.HEX. The output object file of the E0C62XX/62*XX cross assembler is loaded the emulation program memory via these commands.



</td></tr>
<tr><td><b>WFD,&lt;filename&gt;↵</b>
<b>RFD,&lt;filename&gt;↵</b>
<b>VFD,&lt;filename&gt;↵</b></td><td>
The contents of data RAM (4 bits. high order 4 bits are meaningless) are output (or input, or compared) to filenameD.HEX.



</td></tr>
<tr><td><b>CS,&lt;filename&gt;↵</b></td><td>
Contents of program memory and data RAM are output (or input) via the WF and/or WFD commands. During a break, data is output (or input) to the file specified by filenameC.HEX.

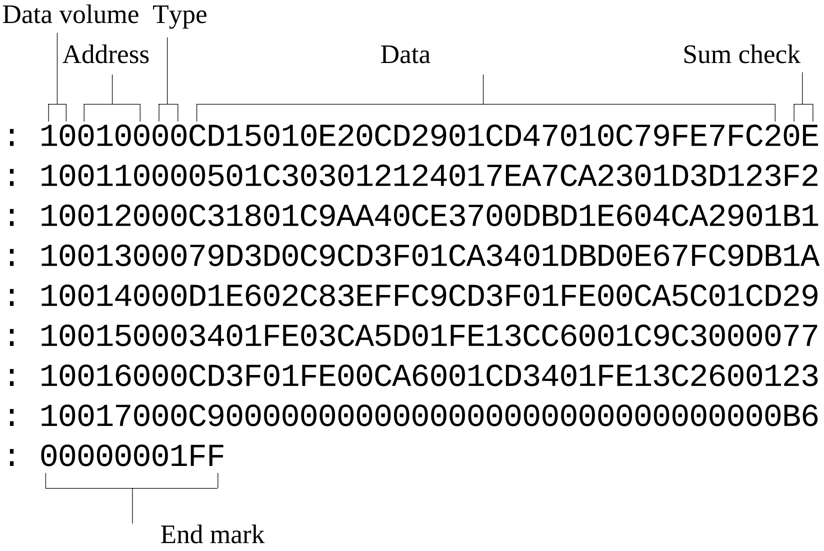


</td></tr>
</table>

APPENDIX B. HEX FILE FORMAT

Description of HEX file format

Example:



```
: 10010000CD15010E20CD2901CD47010C79FE7FC20E
: 100110000501C303012124017EA7CA2301D3D123F2
: 10012000C31801C9AA40CE3700DBD1E604CA2901B1
: 1001300079D3D0C9CD3F01CA3401DBD0E67FC9DB1A
: 10014000D1E602C83EFFC9CD3F01FE00CA5C01CD29
: 100150003401FE03CA5D01FE13CC6001C9C3000077
: 10016000CD3F01FE00CA6001CD3401FE13C2600123
: 10017000C9000000000000000000000000000000B6
: 00000001FF
```

a) Data volume (1 byte): Indicates the quantity of data contained in the data area. Maximum capacity is 10H (sixteen entries).

b) Address (2 bytes) : Indicates the top line of data at each address.

c) Type (1 byte) : Indicates the type of hexadecimal format, currently only 00.

d) Data (16 bytes max.) : Data is shown in hexadecimal format.

e) Sum check (1 byte) : Two complements resulting from adding all bytes from "data volume bytes" to "final data byte" are expressed as hexadecimal values.

f) End mark : Required to mark the end of the hex file.

VI. E0C6262 Mask Data Checker Manual

CONTENTS

CHAPTER 1 INTRODUCTION

1.1 Outline of the Mask Data Checker

The Mask Data Checker MDC6262 is a software tool which checks the program data (C262XXXH.HEX and C262XXXL.HEX) and option data (C262XXXF.DOC) created by the user and creates the data file (C6262XXX.PAn) for generating mask patterns.
The user must send the file generated through this software tool to Seiko Epson.

Moreover, MDC6262 has the capability to restore the generated data file (C6262XXX.PAn) to the original file format (C262XXXH.HEX, C262XXXL.HEX, and C262XXXF.DOC).

Two MDC6262 system disks are supplied by Seiko Epson: one for NEC PC-9801 series (5.25" 2HD) and one for IBM PC/XT and PC/AT (5.25" 2D).
The basic configurations are as follows.

– NEC PC-9801 series

Host computer:	PC-9801 series
Disk drive:	FD (5.25" 2HD) ¥ 1 or more
OS:	MS-DOS Ver. 3.1 or later

– IBM PC/XT or PC/AT

Host computer:	IBM PC/XT or PC/AT
Disk drive:	FD (5.25" 2D) ¥ 1 or more
OS:	PC-DOS Ver. 2.1 or later

The Mask Checker program name is as follows:

MDC6262.EXE

Note *In OS environment setup file CONFIG.SYS, the number of files that can be opened at the same time must be set at least 10.*

Example: *FILES = 20*

1.2 Execution Flow and Input/Output Files

The execution flow for MDC6262 is shown in Figure 1.2.1.

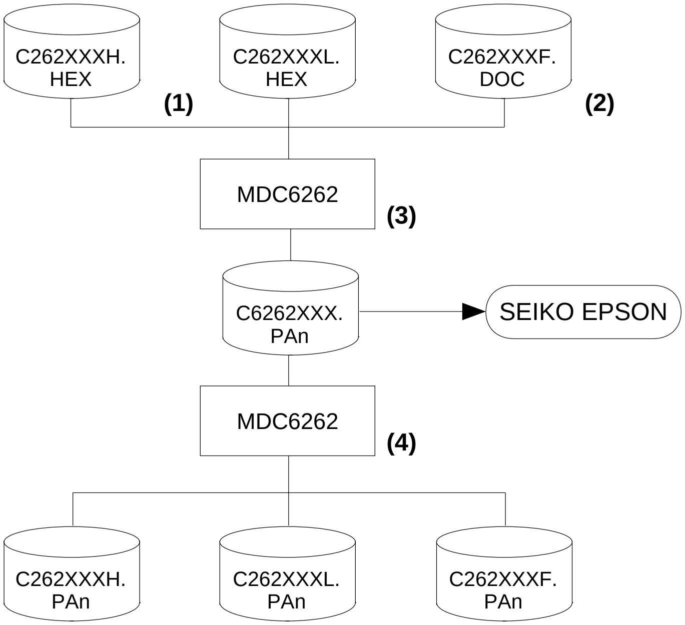


Fig. 1.2.1

MDC6262 Execution Flow

(1) Preparation of program data files

(C262XXXH.HEX and C262XXXL.HEX)

Prepare the program data files generated from the Cross Assembler (ASM6262).

(2) Preparation of option data files

(C262XXXF.DOC)

Prepare the option data file (function option) generated from the Function Option Generator (FOG6262).

(3) Packing of Data

Using the Mask Data Checker (MDC6262), compile the program data and option data in one mask data file (C6262XXX.PAn). This file must be sent to Seiko Epson.

(4) Unpacking of Data

The mask data file (C6262XXX.PAn) may be restored to the original program data and option data files using the Mask Data Checker (MDC6262).

CHAPTER 2 MASK DATA CHECKER OPERATION

2.1 Creating a Work Disk

In order to prevent accidents due to misoperations such as program erasures, place a write protection tab on the Mask Data Checker and keep it as master disk; actual operation should be conducted on other disks.
Create a work disk and copy "MDC6262.EXE" on it.

2.2 Copying the Data File

When submitting data to Seiko Epson, copy on the work disk the data generated from Cross Assembler (ASM6262) and Function Option Generator (FOG6262).

Be sure to assign the following file names (the XXX portion of the file name should be as designated by Seiko Epson):

–	Program data	(HIGH side)	:	C262XXXH.HEX
		(LOW side)	:	C262XXXL.HEX
–	Option data	(function option)	:	C262XXXF.DOC

2.3 Execution of MDC6262

Starting MDC6262

To start MDC6262, insert the work disk into the current drive at the DOS command level (state in which a prompt such as A> is displayed) and then enter the program name as follows:

```
A>MDC6262 ↵
```

*↵ means press the RETURN key.

When MDC6262 is started, the following message is displayed:

```
                *** E0C6262 PACK / UNPACK PROGRAM Ver 1.00 ***

EEEEEEEEEE    PPPPPPPP         SSSSSSS        OOOOOOOO      NNN     NNN
EEEEEEEEEE    PPPPPPPPPP      SSS   SSSS     OOO    OOO     NNNN    NNN
EEE           PPP     PPP    SSS     SSS    OOO      OOO    NNNNN   NNN
EEE           PPP     PPP     SSS           OOO      OOO    NNNNNN  NNN
EEEEEEEEEE    PPPPPPPPPP       SSSSSS       OOO      OOO    NNN NNN NNN
EEEEEEEEEE    PPPPPPPP            SSSS      OOO      OOO    NNN  NNNNNN
EEE           PPP                   SSS     OOO      OOO    NNN   NNNNN
EEE           PPP            SSS     SSS    OOO      OOO    NNN    NNNN
EEEEEEEEEE    PPP            SSSS   SSS      OOO    OOO     NNN     NNN
EEEEEEEEEE    PPP              SSSSSSS        OOOOOOOO      NNN      NN

                (C) COPYRIGHT 1990 SEIKO EPSON CORPORATION

                        --- OPERATION MENU ---

                                1. PACK
                                2. UNPACK

                        PLEASE SELECT NO.? 1
```

Here, the user is prompted to select operation options. When creating mask data for submission to Seiko Epson, select "1"; when the mask data is to be split and restored to the original format (C262XXXH.HEX, C262XXXL.HEX, and C262XXXF.DOC), select "2".

Packing of data

When generating data for submission to Seiko Epson, selecting "1" in the above section, "Starting MDC6262" will prompt for the name of the file to be generated as follows:

```
   C262XXXH.HEX --------+
                        |
   C262XXXL.HEX --------+-------- C6262XXX.PAn (PACK FILE)
                        |
   C262XXXF.DOC --------+

PLEASE INPUT PACK FILE NAME (C6262XXX.PAn) ? C62620A0.PA0 ↵
```

The XXX portion is as specified for the user by Seiko Epson. Moreover, after submitting the data to Seiko Epson and there is a need to re-submit the data for reasons such as faulty programs, etc., increase the numeric value of "n" by one when the input is made. (Example: When re-submiting data after "C62620A0.PA0" has been submitted, the pack file name should be entered as "C62620A0.PA1".

When data is packed, there is need to create ROM data file and option data file in the work disk beforehand.

When the file name has been input, mask data is generated and the corresponding file names are displayed.

```
     C2620A0H.HEX --------+
                          |
     C2620A0L.HEX --------+-------- C62620A0.PA0
                          |
     C2620A0F.DOC --------+
```

With this, the mask file (C6262XXX.PAn) is generated. Submit this file to Seiko Epson.

Note *Don't use the data generated with the -N option of the Cross Assembler (ASM6262) as program data. If the program data generated with the -N option of the Cross Assembler is packed, undefined program area is filled with FFH code.*
In this case, following message is displayed.

```
WARNING: FILLED <file_name> FILE WITH FFH.
```

Unpacking of data

In the process of restoring the packed data to the original file, when "2" is selected in the step described in "Starting MDC6262", the user is prompted for the input file name as follows:

```
PLEASE INPUT PACKED FILE NAME (C6262XXX.PAn) ? C62620A0.PA0_ ↵
```

When the file name has been entered, the unpacking process is executed and the corresponding file names are displayed.

```
                       +-------- C2620A0H.PA0
                       |
   C62620A0.PA0 --------+-------- C2620A0L.PA0
                       |
                       +-------- C2620A0F.PA0
```

With this, the mask data file (C6262XXX.PAn) is restored to the original file format, making it possible to make comparison with the original data.

The restored data file names will be as follows:

–	Program data	(HIGH side)	:	C262XXXH.PAn
		(LOW side)	:	C262XXXL.PAn
–	Option data	(function option)	:	C262XXXF.PAn

CHAPTER 3 ERROR MESSAGES

3.1 Data Error

The program data file and option data file are checked during packing; the packed data file is checked during unpacking.

If there are format problems, the following error messages are displayed.

Program data error

Error Message	Explanation
1. HEX DATA ERROR : NOT COLON.	There is no colon.
2. HEX DATA ERROR : DATA LENGTH. (NOT 00–20h)	The data length of 1 line is not in the 00–20H range.
3. HEX DATA ERROR : ADDRESS.	The address is beyond the valid range of the program ROM.
4. HEX DATA ERROR : RECORD TYPE. (NOT 00)	The record type of 1 line is not 00.
5. HEX DATA ERROR : DATA. (NOT 00–FFh)	The data is not in the range between 00H and 0FFH.
6. HEX DATA ERROR : TOO MANY DATA IN ONE LINE.	There are too many data in 1 line.
7. HEX DATA ERROR : CHECK SUM.	The checksum is not correct.
8. HEX DATA ERROR : END MARK.	The end mark is not : 00000001FF.
9. HEX DATA ERROR : DUPLICATE.	There is duplicate definition of data in the same address.

Function option data error

Error Message	Explanation
1. OPTION DATA ERROR : START MARK.	The start mark is not "¥OPTION". * (during unpacking)
2. OPTION DATA ERROR : OPTION NUMBER.	The option number is not correct.
3. OPTION DATA ERROR : SELECT NUMBER.	The option selection number is not correct.
4. OPTION DATA ERROR : END MARK.	The end mark is not "¥¥END" (packing) or "¥END" (unpacking). *

* ¥ sometimes appears as \, depending on the personal computer being used.

3.2 File Error

Error Message	Explanation
1. <File_name> FILE IS NOT FOUND.	The file is not found or the file number set in CONFIG.SYS is less than 10.
2. PACK FILE (File_name) ERROR.	The packed input format for the file name is wrong.
3. PACKED FILE NAME (File_name) ERROR.	The unpacked input format for the file name is wrong.

3.3 System Error

Error Message	Explanation
1. DIRECTORY FULL.	The directory is full.
2. DISK WRITE ERROR.	Writing on the disk is failed.

CHAPTER 4 PACK FILE CONFIGURATION

The pack file is configured according to the following format:

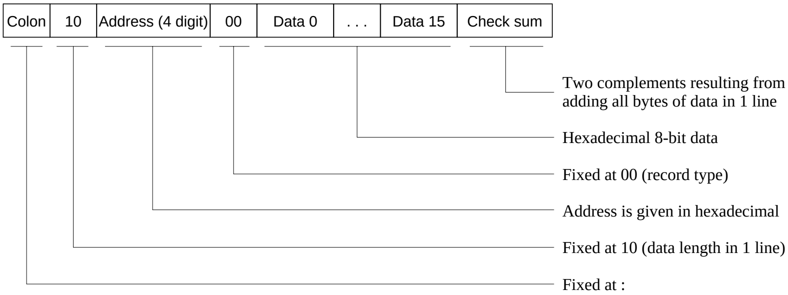
```
                                    *
                                    * SMC6262 MASK DATA VER 1.00
                                    *
Program Data Header               ─ ¥ROM
Model Name                        ─ SMC6262XXX
                                 ┌─ :100000000.................................
Program Data                     │  :100010000.................................
  High Side (Intel Hexa Format)  │      :    :    :    :    :    :    :    :
                                 └─ :00000001FF
                                 ┌─ :100000000.................................
Program Data                     │  :100010000.................................
  Low Side (Intel Hexa Format)   │      :    :    :    :    :    :    :    :
                                 └─ :00000001FF
End Mark                          ─ ¥END
Function Option Header            ─ ¥OPTION
                                 ┌─ * SMC6262 FUNCTION OPTION DOCUMENT  Ver 3.00
                                 │  *
                                 │  * FILE NAME    C262XXXF.DOC
                                 │  * USER'S NAME  SEIKO EPSON CORP.
                                 │  * INPUT DATE   90/12/20
                                 │  * COMMENT      TOKYO DESIGN CENTER
                                 │  *              390-4 HINO HINO-SHI TOKYO 191 JAPAN
Function Option Data             │  *              TEL 0425-83-7313
                                 │  *              FAX 0425-83-7413
                                 │  *
                                 │  *
                                 │  * OPTION NO.1
                                 │  * < .................. >
                                 │  *     ................  ----------------- SELECTED
                                 │   OPT.... ..
                                 │      :    :    :    :    :    :    :    :    :
                                 └─  OPT....  ..
End Mark                          ─ ¥END
```

* ¥ sometimes appears as \, depending on the personal computer being used.

- **Program data**

The program data is expressed as follows, using Intel hexa format:

(1)Data line


Colon
10
Address (4 digit)
00
Data 0
. . .
Data 15
Check sum
Two complements resulting from adding all bytes of data in 1 line
Hexadecimal 8-bit data
Fixed at 00 (record type)
Address is given in hexadecimal
Fixed at 10 (data length in 1 line)
Fixed at :

(2)End mark

```
: 00000001FF
```