

CMOS 4-BIT SINGLE CHIP MICROCOMPUTER
E0C6281 DEVELOPMENT TOOL MANUAL



NOTICE

No part of this material may be reproduced or duplicated in any form or by any means without the written permission of Seiko Epson. Seiko Epson reserves the right to make changes to this material without notice. Seiko Epson does not assume any liability of any kind arising out of any inaccuracies contained in this material or due to its application or use in any product or circuit and, further, there is no representation that this material is applicable to products requiring high level reliability, such as medical products. Moreover, no license to any intellectual property rights is granted by implication or otherwise, and there is no representation or warranty that anything made in accordance with this material will be free from any patent or copyright infringement of a third party. This material or portions thereof may contain technology or the subject relating to strategic products under the control of the Foreign Exchange and Foreign Trade Control Law of Japan and may require an export license from the Ministry of International Trade and Industry or other approval from another government agency. Please note that "E0C" is the new name for the old product "SMC". If "SMC" appears in other manuals understand that it now reads "E0C".

I. E0C6281 CROSS ASSEMBLER MANUAL

PREFACE

This manual mainly explains how to operate the ASM6281 cross-assembler for the E0C6281 4-bit, single-chip microcomputers, and how to generate source files.

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 "81".

62XX → 6281

C2XX → C281

For details on the E0C6281, refer to the "E0C6281 Technical Hardware Manual" and "E0C6281 Technical Software Manual". For such items as development procedure, refer to the "E0C62 Family Technical Guide".

CONTENTS

1. E0C6281 RESTRICTIONS	<i>I-1</i>
2. INTRODUCTUON	<i>I-2</i>
2.1 Outline of ASM62XX	<i>I-2</i>
2.2 ASM62XX Input/Output Files	<i>I-3</i>
3. ASM62XX OPERATION PROCEDURE	<i>I-4</i>
3.1 Starting ASM62XX	<i>I-4</i>
3.2 Selecting Auto-Page-Set Function	<i>I-7</i>
3.3 Generating a Cross-Reference Table	<i>I-8</i>
4. SOURCE FILE FORMAT	<i>I-9</i>
4.1 Source File Name	<i>I-9</i>
4.2 Statements	<i>I-10</i>
4.2.1 Label field	<i>I-11</i>
4.2.2 Mnemonic field	<i>I-11</i>
4.2.3 Operand field	<i>I-12</i>
4.2.4 Comment field	<i>I-12</i>
4.3 Index	<i>I-13</i>
4.3.1 Label	<i>I-13</i>
4.3.2 Symbol	<i>I-14</i>
4.4 Constant and Operational Expression	<i>I-15</i>
4.4.1 Numeric constant	<i>I-15</i>
4.4.2 Character constant	<i>I-16</i>
4.4.3 Operator	<i>I-16</i>
4.4.4 Location counter	<i>I-19</i>
4.5 Pseudo-Instructions	<i>I-20</i>
4.5.1 Data definition pseudo-instructions	<i>I-20</i>
4.5.2 Memory setting pseudo-instructions	<i>I-22</i>
4.5.3 Assembler control pseudo-instructions	<i>I-26</i>
4.6 Macro-Functions	<i>I-27</i>
4.6.1 Macro-instructions	<i>I-27</i>
4.6.2 Macro-definitions	<i>I-29</i>
4.6.3 Macro-calls	<i>I-31</i>
5. ERROR MESSAGES	<i>I-33</i>
APPENDIX ASM62XX EXECUTION EXAMPLE	<i>I-35</i>

1. E0C6281 RESTRICTIONS

Note the following when generating a program by the E0C6281:

1) ROM Area

The capacity of the E0C6281 ROM is 1K steps (0000H to 03FFH). The memory configuration is as follows.

Bank: Only bank 0

Page: 4 pages (0 to 3H), 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 03FFH
PAGE	pseudo-instruction:	00H to 03H
BANK	pseudo-instruction:	Only 0H
PSET	instruction:	00H to 03H

2) RAM Area

The capacity of the E0C6281 RAM is 160 words (000H to 05FH, 090 to 0AFH, and 0E0H to 0FFH, 4 bits/word). Memory access is invalid when the unused area of the index register is specified.

Example: LD X, 78H 78H 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, C7H C7H 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.

3) Undefined codes

The following instructions have not been defined in the E0C6281 instruction sets.

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

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.

Fig.1 shows the ASM62XX execution flow.

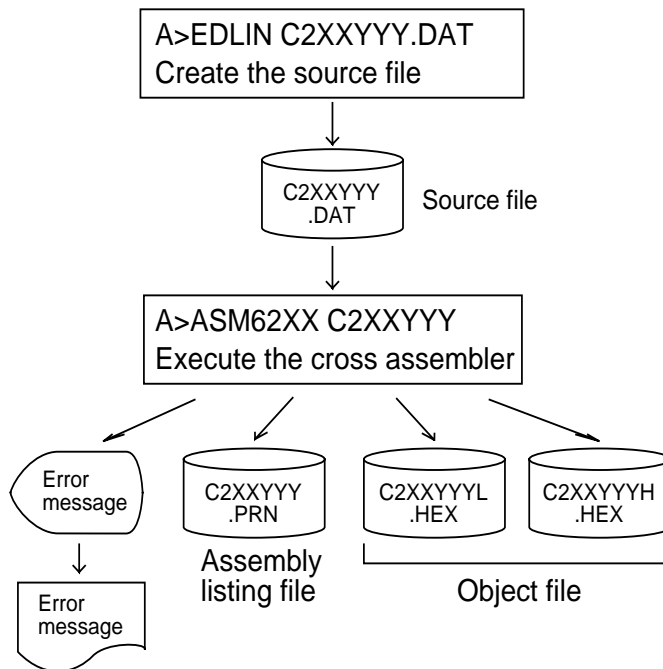


Fig.1 ASM62XX Execution Flow

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.

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 source file 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 character 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 P P P P P P P P S S S S S S S S O O O O O O O O N N N N N N N N
EEEEEEEEEE P P P P P P P P P P S S S S S S S S O O O O O O N N N N N N N N
EEE P P P P P P P P S S S S S S S S O O O O O O N N N N N N N N
EEE P P P P P P P P S S S S S S S S O O O O O O N N N N N N N N
EEEEEEEEEE P P P P P P P P P P S S S S S S S S O O O O O O N N N N N N N N
EEEEEEEEEE P P P P P P P P S S S S S S S S O O O O O O N N N N N N N N
EEE P P P P P P P P S S S S S S S S O O O O O O N N N N N N N N
EEE P P P P P P P P S S S S S S S S O O O O O O N N N N N N N N
EEEEEEEEEE P P P P P P P P S S S S S S S S O O O O O O N N N N N N N N
EEEEEEEEEE P P P P P P P P S S S S S S S S O O O O O O N N N N N N N N

(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.
```

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 "4.3.1 Label".

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.

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 Reference 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.

4.2.1 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.

4.2.2 Mnemonic field

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

4.2.3 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 ",".

4.2.4 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 Section 4.2.1, "Label field", for the restrictions on index descriptions.

4.3.1 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

```

4.3.2 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 4.5.1 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.

4.4.1 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 1FFFF.
	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.

4.4.2 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.

4.4.3 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. $\leftarrow b7<<<<<<b1 \leftarrow 0$ Example: 00000011B SHL 2 \rightarrow 00001100B
a_SHR_b:	Shifts a b bits to the right. $0 \rightarrow b7>>>>>>b0 \rightarrow$ Example: 11000011B SHR 2 \rightarrow 00110000B
HIGH_a:	Separates the high-order eight bits from a (13 bits). Example: HIGH 1234H \rightarrow 12H
LOW_a:	Separates the low-order eight bits from a (13 bits). Example: LOW 1234H \rightarrow 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 \rightarrow 00000011B
a_OR_b:	Logical sum Example: 00001111B OR 11110000B \rightarrow 11111111B
a_XOR_b:	Exclusive logical sum Example: 00001111B XOR 00000011B \rightarrow 00001100B
NOT_a:	Logical negation Example: NOT 00001111B \rightarrow 11110000B

– Relational operators

A logical operator compares two terms; if the relationship between the terms is as the operator specifies, 1FFFFH (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
HIGH, LOW, NOT - (monadic negative), + (monadic positive)	

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

```
LD  A,BCD*(ABC+1)      ;A <- 6
LD  A,ABC LT BCD       ;A <- 0FH (1111B)
OR  B,ABC SHL BCD      ;Set bit3 in B-register(=OR B,1000B)
AND B,ABC SHL BCD XOR 0FH ;Reset bit3 in B-register(=AND
B,0111B)
```

4.4.4 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.

4.5.1 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) ... To define a symbol

```
<Symbol>_EQU_<Expression>
```

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...To define a symbol

<Symbol>_SET_<Expression>

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) ... To preset data

<Label>_DW_<Expression>

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.

4.5.2 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) ... To set the location counter

ORG_<Expression>

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 "1. E0C62XX RESTRICTIONS".)

– BANK ... To set the bank (BNK)

BANK_<expression>

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 ... To set the page counter (PCP)

PAGE_<expression>

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
!      :      :      LD      B,MY
                                :      :
                                ]      Ineffective because
                                :      :      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 "1. E0C62XX RESTRICTIONS".)

– SECTION ... To change the section

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.

(BNK)	(PCP)	(PCS)		
0	1	00H	Section 1	} 16 steps
0	1	10H	Section 2	
0	1	20H		
:	:	:	:	:
0	1	F0H	Section 16	
0	2	00H	Section 17	
0	2	20H		
:	:	:	:	:
0	3	F0H	Section 48	

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
:      :      :      :      :

```

4.5.3 Assembler control pseudo-instructions

– END ... To terminate assembly

```

END

```

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.

4.6.1 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	09	08	07	06	05	04	03	02	01	00	
JPM ps	PSET p	1	1	1	0	0	1	0	p ₄	p ₃	p ₂	p ₁	p ₀	
	JP s	0	0	0	0	s ₇	s ₆	s ₅	s ₄	s ₃	s ₂	s ₁	s ₀	
JPM C, ps	PSET p	1	1	1	0	0	1	0	p ₄	p ₃	p ₂	p ₁	p ₀	
	JP C, s	0	0	1	0	s ₇	s ₆	s ₅	s ₄	s ₃	s ₂	s ₁	s ₀	
JPM NC, ps	PSET p	1	1	1	0	0	1	0	p ₄	p ₃	p ₂	p ₁	p ₀	
	JP NC, s	0	0	1	1	s ₇	s ₆	s ₅	s ₄	s ₃	s ₂	s ₁	s ₀	
JPM Z, ps	PSET p	1	1	1	0	0	1	0	p ₄	p ₃	p ₂	p ₁	p ₀	
	JP Z, s	0	1	1	0	s ₇	s ₆	s ₅	s ₄	s ₃	s ₂	s ₁	s ₀	
JPM NZ, ps	PSET p	1	1	1	0	0	1	0	p ₄	p ₃	p ₂	p ₁	p ₀	
	JP NZ, s	0	1	1	1	s ₇	s ₆	s ₅	s ₄	s ₃	s ₂	s ₁	s ₀	
CALLM ps	PSET p	1	1	1	0	0	1	0	p ₄	p ₃	p ₂	p ₁	p ₀	
	CALL s	0	1	0	0	s ₇	s ₆	s ₅	s ₄	s ₃	s ₂	s ₁	s ₀	

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  
      :
```

4.6.2 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

```

4.6.3 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
                                CBFIRST  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 , CBFIRST , 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

```

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          SSSSSS      OOOOOOOO      NNN  NN

```

(C) COPYRIGHT 1989 SEIKO EPSON CORP.

SOURCE FILE NAME IS " C2XXYY.DAT "

THIS SOFTWARE MAKES NEXT FILES.

```

C2XXYYH.HEX ... HIGH BYTE OBJECT FILE.
C2XXYYL.HEX ... LOW BYTE OBJECT FILE.
C2XXYY .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
:10002000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE0
:10003000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD0
:10004000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC0
:10005000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFB0
:10006000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA0
:10007000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF90
:10008000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF80
:10009000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF70
:1000A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF60
:1000B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF50
:1000C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF40
:1000D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF30
:1000E0001AA6851C5E39E003FFFF00FFFFFFFFF3C
:1000F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF10
:10010000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10011000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEF
:10012000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDF
:10013000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFCF
:10014000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBF
:10015000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFAF
:10016000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9F
:10017000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8F
:10018000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7F
:10019000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6F
:1001A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5F
:1001B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4F
:1001C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3F
:1001D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2F
:1001E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1F
:1001F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0F
:10020000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE
:10021000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEE
:10022000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDE
:10023000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFCE
:10024000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBE
:10025000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFAE
:10026000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9E
:10027000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8E
:10028000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7E
:10029000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6E
:1002A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5E
:1002B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4E
:1002C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3E
:1002D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2E
:1002E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1E
:1002F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0E
```

```
:10030000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD
:10031000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD
: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
:10000000E0B080EFFFFFFFFFFFFFFFFFFFFFFFFFC
:10001000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0
:10002000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE0
:10003000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD0
:10004000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC0
:10005000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFB0
:10006000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA0
:10007000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF90
:10008000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF80
:10009000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF70
:1000A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF60
:1000B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF50
:1000C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF40
:1000D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF30
:1000E0000C0E0C0F0F0F020E0F0F00FFFFFFFFF94
:1000F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0F00
:10010000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10011000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEF
:10012000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDF
:10013000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFCF
:10014000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBF
:10015000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFAF
:10016000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9F
:10017000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8F
:10018000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7F
:10019000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6F
:1001A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5F
:1001B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4F
:1001C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3F
:1001D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2F
:1001E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1F
:1001F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0F
:10020000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE
:10021000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEE
:10022000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDE
:10023000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFCE
:10024000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBE
:10025000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFAE
:10026000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9E
:10027000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8E
:10028000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7E
:10029000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6E
:1002A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5E
:1002B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4E
:1002C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3E
:1002D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2E
:1002E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1E
:1002F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0E
:10030000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD
:10031000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFED
```

```
:10032000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD  
: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 "1. E0C62XX RESTRICTIONS".*

II. ***E0C6281MELODY ASSEMBLER MANUAL***

PREFACE

This manual mainly explains how to operate the MLA6281 melody-assembler for the E0C6281 4-bit, single-chip microcomputers, and how to generate source files.

Chapter 1 and subsequent chapters provide information common to all E0C6280 models, the model name being denoted "8X". Read this manual, replacing "8X" with "81".

628X → 6281

C28X → C281

For details on the E0C6281, refer to the "E0C6281 Technical Hardware Manual" and "E0C6281 Technical Software Manual".

CONTENTS

1 INTRODUCTION	<i>II-1</i>
1.1 Outline of MLA628X	<i>II-1</i>
1.2 MLA628X Input/output File	<i>II-2</i>
2. Activating MLA628X	<i>II-3</i>
3. FORMAT OF SOURCE FILE	<i>II-7</i>
3.1 Source File Name	<i>II-7</i>
3.2 Statement (line)	<i>II-7</i>
3.2.1 Attack field	<i>II-8</i>
3.2.2 Note field	<i>II-8</i>
3.2.3 Scale field	<i>II-8</i>
3.2.4 End bit field	<i>II-8</i>
3.2.5 Comment field	<i>II-8</i>
3.2.6 Fields and corresponding melody data	<i>II-9</i>
3.3 Pseudo-instruction	<i>II-10</i>
3.3.1 Address-setting pseudo-instruction	<i>II-10</i>
3.3.2 Option-setting pseudo-instructions	<i>II-10</i>
4. ERROR MESSAGES.....	<i>II-12</i>
5. EXAMPLES OF INPUT-OUTPUT FILES	<i>II-13</i>
5.1 Example of Source File	<i>II-13</i>
5.2 Example of Assembly List	<i>II-14</i>
5.3 Example of Melody Hex File Data Format	<i>II-15</i>
5.4 Example of Assembly List When Error Occurs	<i>II-16</i>
5.5 Example of Melody Document File Format	<i>II-18</i>

1. INTRODUCTION

1.1 Outline of MLA628X

The Melody Assembler MLA628X (hereafter abbreviated as MLA628X) is an assembler that outputs melody ROM data of the 4-bit single-chip microcomputers E0C628X and 62L8X. MLA628X can be used with both MS-DOS and PC-DOS.

The MLA628X system disks come in two types: for the NEC PC9801 V series (5.25", 2HD) and the IBM PC/XT and PC/AT (5.25", 2D). Basically, the system is configured as follows:

– PC9801 V series

Host computer: PC9801 V series
 Disk drive: Housing at least one floppy disk (5.25", 2HD)
 Operating system: MS-DOS Ver. 3.1 or later
 Printer: For printing source files, assembly lists, error messages

– IBM PC/XT, PC/AT

Host computer: IBM PC/XT, PC/AT
 Disk drive: Housing at least one floppy disk (5.25", 2D)
 Operating system: PC-DOS (MS-DOS) Ver. 3.1 or later
 Printer: For printing source files, assembly lists, error messages

The Melody Assembler's program name is "MLA628X.EXE".

Figure 1.1 shows the flow of executing MLA628X.

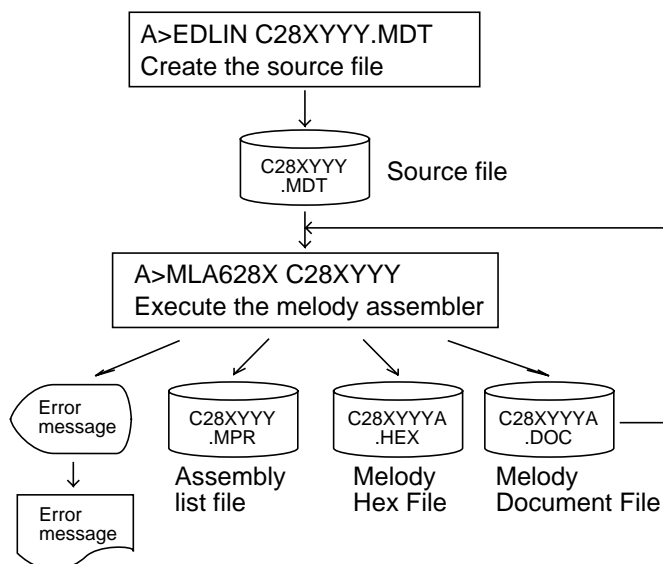


Fig. 1.1 Flow of execution of MLA628X

1.2 MLA628X Input/output File

MLA628X inputs the source file, and after assembly it outputs the object file, assembly list file and document file.

– Source file (C28XYYY.MDT)

Source program file created with an editor, such as EDLIN.

Create the file name using seven characters maximum, in the format "C28XYYY". For "YYY", use the model name from Seiko Epson.

Also, be sure to make the extension ".MDT".

– Melody HEX file (C28XYYA.HEX)

Program file of Intel hexadecimal format created by the assembler.

Note: When loaded with ICE, the file format is checked, and an error results when it is not following format.

· Main ROM high-order data = 00H or 01H

· Main ROM low-order data = 00H–FFH

· When 80 bytes are not used, the space is filled out with FF.

– Melody assembly list file (C28XYYY.MPR)

Melody ROM list file with melody ROM data and error messages (if any) added to each line of the source file. The scale ROM table can be created at the end of the file. The file specifier is "C28XYYY.MPR".

– Document file (C28XYYA.DOC)

File containing hexadecimal data of melody ROM and scale ROM as well as information on melody options. The file specifier is "C28XYYA.DOC".

2. ACTIVATING MLA628X

To activate MLA628X, input the following from the DOS command level (that is, when the prompt "A>" appears on the screen).

MLA628X_[drive name:]source filename[.shp]_[-H]↵

_ indicates where a space is to be input.

[] indicates that this parameter may be omitted.

↵ indicates where the Return key (Enter key) is pressed.

Drive name: When the source file is in a different drive from MLA628X.MDT, the drive name is input before the source filename. If in the same drive, then it may be omitted.

Source filename: The source file to input to MLA628X.

Note: Make the source filename up to seven characters long, and do not input the extension (".MDT").

.shp: s, h, and p are options specifying the file's input/output drive, as explained below. These may be omitted, and input is valid for both upper- and lower-case.

s: Specifies the drive from A through P that inputs the source file.

When "@" is specified, the source file on the current drive (directory) is input.

The "s" specification is valid when the drive name is input before the source filename.

h: Specifies the drive from A through P that outputs the melody HEX file and melody document file.

When "@" is specified, output is made to the current drive (directory).

When "Z" is specified, only assembly is performed and the melody HEX file and melody document file are not created.

p: Specifies the drive from A through P that outputs the assembly list file.

When "@" is specified, output is made to the current drive (directory).

When "X" is specified the list including error messages is output from the console.

When "Z" is specified, the assembly list file is not created.

Specify s, h and p at the same time. These cannot be specified separately.

-H: "-H" is the option to indicate activation of the conversion program from the melody document file to the melody HEX file.

When this option is specified, the [shp] option is disabled. The melody document file of the current drive is input and the melody HEX file is created for the current drive. Input can be in upper- and lower-case.

Examples:

```
A>MLA628X C28XYYY ↵
```

In this example, the source file C28XYYY.MDT is input from drive A, and the melody HEX file C28XYYYA.HEX, melody assembly list file C28XYYY.MPR, and melody document file C28XYYYA.DOC are created on drive A.

```
A>MLA628X B:C28XYYY ↵
```

In this example, the source file C28XYYY.MDT is input from drive B, and the melody HEX file C28XYYYA.HEX, melody assembly list file C28XYYY.MPR, and melody document file C28XYYYA.DOC are created on drive B.

```
A>MLA628X C28XYYY.BBZ ↵
```

In this example, the source file C28XYYY.MDT is input from drive B, and the melody HEX file C28XYYYA.HEX and melody document file C28XYYYA.DOC are created on drive B. The melody assembly list is not created.

When MLA628X is activated, the activation messages appear as shown below.

Example: When assembling C28X0A0.MDT (Basic assembly)

```

A>MLA628X C28X0A0 ↵
***  MLA628X MELODY ASSEMBLER. --- Ver 3.00 ***

EEEEEEEEEE  P P P P P P P P      S S S S S S S S      O O O O O O O O      N N N      N N N
EEEEEEEEEE  P P P P P P P P P P  S S S   S S S S   O O O   O O O      N N N N N      N N N
EEE         P P P   P P P      S S S   S S S      O O O   O O O      N N N N N N      N N N
EEE         P P P   P P P      S S S   S S S      O O O   O O O      N N N N N N      N N N
EEEEEEEEEE  P P P P P P P P P P  S S S S S S      O O O   O O O      N N N N N N      N N N
EEEEEEEEEE  P P P P P P P P      S S S S S S      O O O   O O O      N N N   N N N N N N
EEE         P P P           S S S   S S S      O O O   O O O      N N N   N N N N N
EEE         P P P           S S S   S S S      O O O   O O O      N N N   N N N N
EEEEEEEEEE  P P P           S S S S S S      O O O   O O O      N N N   N N N
EEEEEEEEEE  P P P           S S S S S S      O O O O O O      N N N   N N

          (C) COPYRIGHT 1989 SEIKO EPSON CORP.

SOURCE FILE NAME IS " C28XYYY.MDT ".

THIS SOFTWARE MAKES NEXT FILES.

C28XYYYA.HEX ... MELODY HEX FILE.
C28XYYYA.DOC ... MELODY DOCUMENT FILE.
C28XYYY .MPR ... MELODY ASSEMBLY LIST FILE.

          STRIKE ANY KEY

```

With the message "STRIKE ANY KEY", the program is requesting key input for confirmation.

The program will proceed when any key is pressed.

To cancel the program, press the CTRL and C keys together. This will return you to the DOS command level.

Example: -H option use

(activation of program to convert melody document file to melody HEX file)

```

A>MLA628X C28X0A0 -H ↵
***  MLA628X MELODY ASSEMBLER. --- 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  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           SSSSSS   OOOOOOOO  NNN  NN

          (C) COPYRIGHT 1989 SEIKO EPSON CORP.

SOURCE FILE NAME IS " C28XYYYA.DOC ".

THIS SOFTWARE MAKES NEXT FILES.

          C28XYYYA.HEX ... MELODY HEX FILE.

          STRIKE ANY KEY

```

With the message "STRIKE ANY KEY", the program is requesting key input for confirmation. Check the source filename and option that you have input.

The program will proceed when any key is pressed. To cancel the program, press the CTRL and C keys together. This will return you to the DOS command level.

3. FORMAT OF SOURCE FILE

Contents of the source file, created with an editor such as EDLIN, are configured from the E0C628X/62L8X melody codes and the pseudo-instructions described later.

3.1 Source File Name

The source file can be named with a maximum of any seven characters. As a rule, keep to the following format.

```
C28XYYY.MDT
```

Three alphanumeric characters are entered in the "YYY" part. Refer to the model name from Seiko Epson. The extension must be ".MDT".

3.2 Statement (line)

Write each of the source file statements (lines) as follows:

Basic format: <attack> <note> <scale> <end bit> <comment>

Example:

```

.TEMPC0=5
.TEMPC1=8
.OCTAVE=32
;
0          1          C3
0          4          D4
0          4          E4#
0          2          F5
0          3          G5#
1          7          A4
1          5          B4
0          6          A4#      1      ;1st Melody
;
ORG      10H
;
0          2          C3#
0          3          $45
0          7          $E3
1          6          $97
0          5          C6
0          7          A5#
1          3          $42      1      ;2nd Melody

```

Attack field	Note field	Scale field	End bit field	Comment field

The statement is made up of the five fields: attack field, note field, scale field, end bit field, and comment field. Up to 80 characters can be written in the statement. The fields are separated by one or more spaces or by inserting tabs.

The end bit fields and comment fields can be filled in on an as-needed basis.

A blank line is also permitted for the CR (carriage return) code only. However, it is not permitted on the last line. Each of the fields can be started from any column.

3.2.1 Attack field

Control of the attack output is written.

When "1" is written, attack output is performed. When "0" is written, attack output is not performed.

3.2.2 Note field

Eight notes can be specified with the melody data D5 through D7. Fill in the note field with numbers from 1 to 8.

Table 3.2.2 Notes and corresponding codes

No.	1	2	3	4	5	6	7	8
Note								

3.2.3 Scale field

The scale field can be filled in with any scale (C3 through C6#).

When inputting the scale data directly, prefix the data with "\$". In this case, the input data range is 00H through FDH.

3.2.4 End bit field

The instruction indicating the end of the melody is written in the end bit field. When "1" is written, the melody finishes with the melody data of that address. Otherwise, write "0", or omit it altogether.

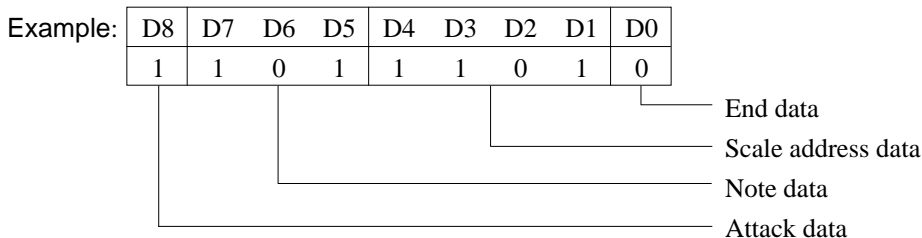
3.2.5 Comment field

Any comment, such as the program index or processing details, can be written in the comment field, with no affect on the object file created with the assembler.

The comment field is the area between the semicolon ";" and the CR code at the end of the line. A line can be made up of a comment field alone. However, if the comment extends into two or more lines, each line must be headed with a semicolon.

3.2.6 Fields and corresponding melody data

* Melody data



- End data

Becomes "0" when "0" is entered or no entry is made; otherwise, "1".

- Scale address data

The scale or scale data written in the scale field is loaded into the scale ROM, and the address of the loaded scale data becomes the scale address data.

Table 3.2.6.a Correspondence between scale and scale data

Scale	Scale Data								Hex.	Scale	Scale Data								Hex.
	S7	S6	S5	S4	S3	S2	S1	S0			S7	S6	S5	S4	S3	S2	S1	S0	
C3	0	0	0	0	0	1	0	0	04	G4	1	0	1	1	0	0	0	1	B1
C3#	0	0	0	1	0	0	1	0	12	G4#	1	0	1	1	0	1	0	1	B5
D3	0	0	1	0	0	0	0	0	20	A4	1	0	1	1	1	0	0	0	B8
D3#	0	0	1	0	1	1	1	1	2F	A4#	1	0	1	1	1	1	0	0	BC
E3	0	0	1	1	1	0	1	1	3B	B4	1	1	0	0	0	0	0	0	C0
F3	0	1	0	0	0	1	0	0	44	C5	1	1	0	0	0	1	0	0	C4
F3#	0	1	0	1	0	0	0	1	51	C5#	1	1	0	0	1	0	0	0	C8
G3	0	1	0	1	1	0	1	1	5B	D5	1	1	0	0	1	1	0	1	CD
G3#	0	1	1	0	0	1	0	1	65	D5#	1	1	0	0	1	1	1	0	CE
A3	0	1	1	0	1	1	0	0	6C	E5	1	1	0	1	0	0	1	1	D3
A3#	0	1	1	1	0	1	0	0	74	F5	1	1	0	1	0	1	0	0	D4
B3	0	1	1	1	1	1	1	0	7C	F5#	1	1	0	1	1	0	0	1	D9
C4	1	0	0	0	0	1	0	0	84	G5	1	1	0	1	1	0	1	1	DB
C4#	1	0	0	0	1	1	0	1	8D	G5#	1	1	0	1	1	1	0	0	DC
D4	1	0	0	1	0	0	1	0	92	A5	1	1	0	1	1	1	1	0	DE
D4#	1	0	0	1	1	0	0	0	98	A5#	1	1	1	0	0	0	0	0	E0
E4	1	0	0	1	1	1	1	0	9E	B5	1	1	1	0	0	0	1	0	E2
F4	1	0	1	0	0	1	0	0	A4	C6	1	1	1	0	0	1	0	0	E4
F4#	1	0	1	0	1	0	1	1	AB	C6#	1	1	1	0	0	1	1	0	E6

- Note data

The correspondence between notes and note data are as follows.

Table 3.2.6.b Correspondence between notes and note data

Note Data	111	110	101	100	011	010	001	000
Note								

- Attack data

"0" or "1" written in the attack field becomes the attack data.

3.3 Pseudo-instruction

The pseudo-instruction is for the assembler, and cannot be executed by the melody data after development.

In the explanations below, the symbols "<" and ">" used in the pseudo-instruction format indicate the contents of the statement. These symbols are not actually written. "_" indicates one or more spaces or tabs. The symbol, constant, arithmetic expression and so forth is written in "<expression>".

3.3.1 Address-setting pseudo-instruction

– **ORG (ORIGIN)**

ORG_<expression>	Sets location counter
-------------------------------	------------------------------

The ORG instruction sets the value of <expression> in the location counter. If the ORG instruction does not head the source file, the location counter is set to 0 and assembly is performed. The ORG instruction can be used in multiple places in the program. However, it cannot be set in a location ahead of the current location counter, otherwise all the statements will be invalid until the next correct setting is performed, and "!" (Warning) is displayed. When a value exceeding the ROM capacity is specified, an R error results.

3.3.2 Option-setting pseudo-instructions

– **.TEMPC0**

.TEMPC0=n	Sets TEMPC0 (n = 0–15)
------------------	-------------------------------

The TEMPC0 option is set by specifying n as an integer in the range 0 to 15. This setting cannot be omitted.

– **.TEMPC1**

.TEMPC1=n	Sets TEMPC1 (n = 0–15)
------------------	-------------------------------

The TEMPC1 option is set by specifying n as an integer in the range 0 to 15. This setting cannot be omitted.

- .OCTAVE

.OCTAVE=m**Sets scale range (m = 32 or 64)**

Decides the scale range by selecting the specification of the melody multiplier circuit.

The specification becomes 32 kHz for m=32, and the range becomes (C3–C6#).

The specification becomes 64 kHz for m=64, enabling output of notes one octave higher (C4–C7#) than can be done with the 32 kHz specification.

For instance, even if the scale in the source file is C5, the actual sound generated will be C6.

This setting cannot be omitted.

4. ERROR MESSAGES

When errors occur during assembly, MLA628X outputs the following error symbols or error messages to the console and assembly list file.

Just one error symbol is output at the head (first column) of the statement that generated an error. (When multiple errors have been generated, the symbol for the error of highest priority is output.)

The following error symbols are shown in order from highest priority.

– Error symbol (errors that can be assembled)

- S (Syntax error) Major syntax error

- Error in scale field (Exceeded scale range: C3–C6#)
- Error in note field (Exceeded note range: 1–8)
- Error in attack field (Number other than 0 or 1 was input.)
- Error in end bit field (Number other than 0 or 1 was input.)

- O (Scale ROM overflow)

The definition exceeded the scale ROM capacity.

- R (Range error)

The value of the location counter exceeded the upper limit of the melody ROM capacity. Otherwise, the specified location exceeded the upper limit.

– Error messages

(Fatal errors preventing assembly or output of assembly results.)

- OPTION COMMAND MISSING

Options cannot be set.

- FILE NAME ERROR

The source filename has eight or more characters.

- FILE NOT PRESENT

The specified source file is not there.

- DIRECTORY FULL

No more room in the directory of the specified disk.
















- FATAL DISK WRITE ERROR

The file cannot be written to the disk.

5. EXAMPLES OF INPUT-OUTPUT FILES

5.1 Example of Source File

```

.TEMPC0=5
.TEMPC1=8
.OCTAVE=32
;
0 1 C3      ; 
0 4 D4      ; 
0 4 E4#     ; 
0 2 F5      ; 
0 3 G5#     ; 
1 7 A4      ; 
1 5 B4      ; 
0 6 A4# 1   ;  1st Melody
;
ORG 10H
;
0 2 $C3     ; 
0 3 $45     ; 
0 7 $E3     ; 
1 6 $97     ; 
0 5 C6      ; 
0 7 A5#     ; 
1 3 $42 1   ;  2nd Melody

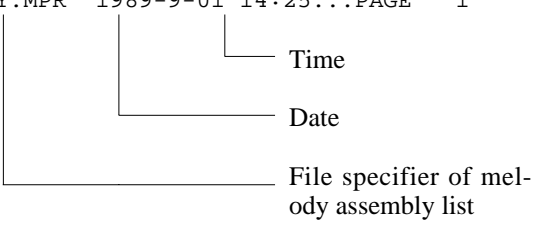
```

5.2 Example of Assembly List

```

LISTING OF MLA628X          C28XYYY.MPR  1989-9-01 14:25...PAGE  1

  ADRS  CODE      SOURCE
          .TEMPC0 = 5
          .TEMPC1 = 8
          .OCTAVE = 32
          ;
0      000      0 1 C3      ; 
1      062      0 4 D4      ; 
2      064      0 4 F4      ; 
3      026      0 2 F5      ; 
4      048      0 3 G5#     ; 
5      1CA      1 7 A4      ; 
6      18C      1 5 B4      ; 
7      0AF      0 6 A4# 1 ;  1st Melody
          ;
          ORG 10H
          ;
10     030      0 2 $C3     ; 
11     052      0 3 $45     ; 
12     0D4      0 7 $E3     ; 
13     1B6      1 6 $97     ; 
14     098      0 5 C6      ; 
15     0DA      0 7 A5#     ; 
16     159      1 3 $42 1 ;  2nd Melody
  
```



0 ERROR(S) DETECTED

SCALE ROM TABLE PAGE S-1

ADRS	SCALE	CODE
0000	C3	04
0001	D4	92
0010	F4	A4
0011	F5	D4
0100	G5#	DC
0101	A4	B8
0110	B4	C0
0111	A4#	BC
1000	\$C3	C3
1001	\$45	45
1010	\$E3	E3
1011	\$97	97
1100	C6	E4
1101	A5#	E0
1110	--	FF
1111	RR	C4

Example of scale ROM table
 - Hyphens "--" indicate unused code.
 - When unused, the code is FFH.
 - The last location, ADRS = "1111", of the scale ROM is fixed at SCALE = "RR" and CODE = "C4".

5.3 Example of Melody Hex File Data Format

```

:10000000000000000000010100FFFFFFFFFFFFFFFFFFFF6 ] (1)
:1000100000000000001000001FFFFFFFFFFFFFFFFFFFFE7 ]
:1001000000062642648CA8CAFFFFFFFFFFFFFFFFFFFFBF ] (2)
:100110003052D4B698DA59FFFFFFFFFFFFFFFFFFFFF12 ]
:100200000492A4D4DCB8C0BCC345E397E4E0FFFFF8E (3)
:10030000110010011111001010001110001001104A (4)
:00000001FF


























```

- (1) Main ROM high-order (D8)
(Of the 256 bytes, the 80 bytes from 0 to 4F are enabled.)
- (2) Main ROM low-order (D0–D7)
(Of the 256 bytes, the 80 bytes from 0 to 4F are enabled.)
- (3) Scale ROM (D0–D7)
(Of the 256 bytes, the 16 bytes from 0 to F are enabled.)
- (4) Option
 - Tempo
 - Octave

5.4 Example of Assembly List When Error Occurs

When an error occurs the code is made FFF forcibly. A value is not entered for the scale ROM.

LISTING OF MLA628X C28XYYY.MPR 1989-9-01 17:30...PAGE 1

ADRS	CODE	SOURCE	
		.TEMPC0 = 5	
		.TEMPC1 = 8	
		.OCTAVE = 32	
		;	
0	000	0 1 C3	<i>i</i> 
1	062	0 4 D4	<i>i</i> 
2	064	0 4 F4	<i>i</i> 
S 3	FFF	0 2 F6	<i>i</i> 
4	048	0 3 G5#	<i>i</i> 
S 5	FFF	2 7 A4	<i>i</i> 
S 6	FFF	1 9 B4	<i>i</i> 
7	0A9	0 6 A4# 1	<i>i</i>  1st Melody
		;	
		ORG 10H	
		;	
10	02A	0 2 \$C3	<i>i</i> 
11	06C	0 3 \$17	<i>i</i> 
12	0CE	0 7 \$E3	<i>i</i> 
13	1B0	1 6 \$97	<i>i</i> 
14	092	0 5 C6	<i>i</i> 
15	0D4	0 7 A5#	<i>i</i> 
16	056	0 3 E3	<i>i</i> 
17	078	0 4 G5	<i>i</i> 
18	09A	0 5 G4	<i>i</i> 
19	07C	0 4 G4#	<i>i</i> 
O 1A	FFF	0 3 A4	<i>i</i> 
1B	15F	1 3 RR 1	<i>i</i>  2nd Melody
		.	
		.	
		.	
4E	05A	0 3 G4	<i>i</i> 
4F	05A	0 3 G4	<i>i</i> 
R 50	FFF	0 3 G4	<i>i</i> 
S 51	FFF	4 3 G4	<i>i</i> 
R 52	FFF	0 3 G4 1	<i>i</i> 

7 ERROR(S) DETECTED

SCALE ROM TABLE PAGE S-1

ADRS	SCALE	CODE
0000	C3	04
0001	D4	92
0010	F4	A4
0011	G5#	DC
0100	A4#	BC
0101	\$C3	C3
0110	\$17	17
0111	\$E3	E3
1000	\$97	97
1001	C6	E4
1010	A5#	E0
1011	E3	3B
1100	G5	DB
1101	G4	B1
1110	G4#	B5
1111	RR	C4

***III.* E0C6281 FUNCTION OPTION GENERATOR MANUAL**

PREFACE

This manual explains how to operate the FOG6281 Function Option Generator for setting the hardware options of 4-bit single-chip E0C6281 microcomputer and details the specifications of their options.

Refer to "E0C6281 Technical Hardware Manual" for details about the E0C6281. Refer to "E0C62 Family Technical Guide" for details about the development procedure.

CONTENTS

1 GENERAL	III-1
1.1 Outline of Function Option Generator	III-1
1.2 Execution Flow and I/O Files	III-2
2 OPTION LIST GENERATION.....	III-4
2.1 Option List Recording Procedure	III-4
2.2 Option List	III-4
2.3 Option Specifications	III-6
2.3.1 Device type	III-6
2.3.2 Multiple key entry reset	III-7
2.3.3 Interrupt noise rejector	III-8
2.3.4 Input ports pull down resistor	III-9
2.3.5 Output port (R00–R03) output specification.....	III-10
2.3.6 R10 specification	III-11
2.3.7 R11 specification	III-12
2.3.8 R12 specification	III-13
2.3.9 I/O port specification	III-15
2.3.10 LCD common duty	III-16
2.3.11 OSC1 system clock	III-17
3 FUNCTION OPTION GENERATOR OPERATION PROCEDURE	III-18
3.1 Creating Work Disk	III-18
3.2 Starting FOG6281	III-19
3.3 Setting New Function Options	III-21
3.4 Modifying Function Option Settings	III-23
3.5 Selecting Function Options	III-25
3.5.1 Selecting the device type	III-26
3.5.2 Selecting multiple key entry reset function	III-26
3.5.3 Selecting input interrupt noise rejector	III-26
3.5.4 Selecting input port pull down resistor	III-27
3.5.5 Selecting output port (R00–R03) output specification.....	III-28
3.5.6 Selecting R10 terminal specification	III-29
3.5.7 Selecting R11 terminal specification	III-29
3.5.8 Selecting R12 terminal specification	III-30
3.5.9 Selecting I/O port specification	III-30
3.5.10 Selecting LCD common (drive) duty	III-31
3.5.11 Selecting OSC1 osillation circuit	III-31

3.6	HEX File Generation and EPROM Selection	<i>III-32</i>
3.7	End Procedure	<i>III-33</i>
4	SAMPLE FILE	<i>III-34</i>

1 GENERAL

1.1 Outline of Function Option Generator

With the 4-bit single-chip E0C6281 microcomputers, the customer may select 11 hardware options. By modifying the mask patterns of the E0C6281 according to the selected options, the system can be customized to meet the specifications of the target system.

The FOG6281 Function Option Generator (hereinafter called FOG6281) 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 FOG6281, the E0C6281 mask pattern is automatically generated by a general purpose computer. The HEX file for the evaluation board (EVA6281) 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 EVA6281, option functions can be executed on the EVA6281.

Two FOG6281 system 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) x 1 or more
Operating system: MS-DOS Ver. 3.1 or later
ROM writer: Required when using EVA6281

– IBM PC/XT and PC/AT

Host computer: IBM PC/XT and PC/AT
Disk drive: FD (5.25" 2D) x 1 or more
Operating system: PC-DOS Ver. 2.1 or later
ROM writer: Required when using EVA6281

The program name of FOG6281 is as follows:

FOG6281.EXE

1.2 Execution Flow and I/O Files

Figure 1.2 shows the FOG6281 execution flow.

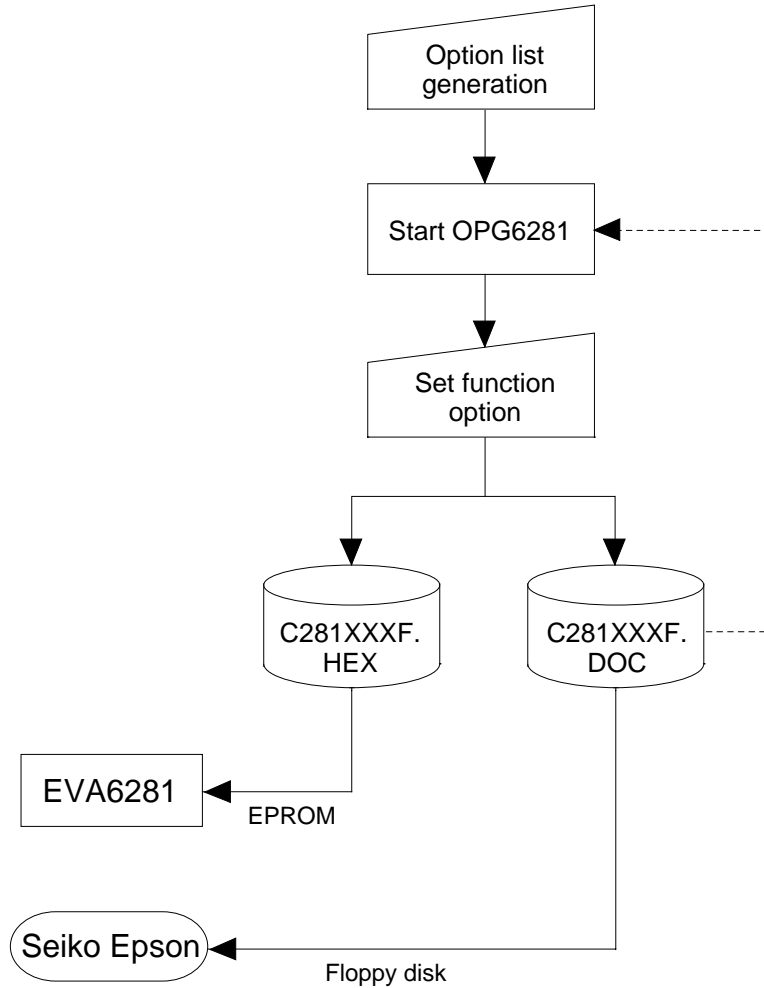


Fig. 1.2 Execution Flow

(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) FOG6281 execution

Start FOG6281 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.

FOG6281 outputs the following data files:

- Function option document file (C281XXXF.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 (C281XXXF.HEX)
This is a function option file (Intel hexa format) used for EVA6281. One EVA6281 function option ROM is generated by writing this file with the ROM writer.

- * File name "XXX" is specified for each customer by Seiko Epson.
- * Copy the document file, the program files (C281XXXH.HEX and C281XXXL.HEX) and the segment option document file (C281XXXS.DOC) in a batch to another disk and send it to Seiko Epson.
- * Set all unused ROM areas to FFH when writing the HEX file into the EPROM. (Refer to "EVA6281 Manual" for the ROM installation location.)

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 E0C6281 option list is as follows:

1. DEVICE TYPE

- E0C6281
- E0C62L81

2. MULTIPLE KEY ENTRY RESET

- Not Use
- Use K00, K01
 - K00, K01, K02
 - K00, K01, K02, K03

3. INTERRUPT NOISE REJECTOR

- K00–K03 Use Not Use
- K10 Use Not Use

4. INPUT PORT PULL DOWN RESISTOR

- K00 With Resistor Gate Direct
- K01 With Resistor Gate Direct
- K02 With Resistor Gate Direct
- K03 With Resistor Gate Direct
- K10 With Resistor Gate Direct

5. OUTPUT PORT OUTPUT SPECIFICATION (R00–R03)

- R00 Complementary Pch-OpenDrain
- R01 Complementary Pch-OpenDrain
- R02 Complementary Pch-OpenDrain
- R03 Complementary Pch-OpenDrain

6. R10 SPECIFICATION

- OUTPUT TYPE D.C.
 - 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 Pch-OpenDrain

7. R11 SPECIFICATION

- OUTPUT SPECIFICATION Complementary Pch-OpenDrain

8. R12 SPECIFICATION

- OUTPUT TYPE D.C.
 - Melody inverted output (fixed to Complementary)
 - Envelope (fixed to Nch-OpenDrain)
- OUTPUT SPECIFICATION (When D.C. is selected)
 - Complementary Pch-OpenDrain

9. I/O PORT SPECIFICATION

- P00 Complementary Pch-OpenDrain
- P01 Complementary Pch-OpenDrain
- P02 Complementary Pch-OpenDrain
- P03 Complementary Pch-OpenDrain

10. LCD COMMON DUTY

- 1/4 Duty
- 1/3 Duty

11. OSC1 SYSTEM CLOCK

- Crystal
- CR

2.3 Option Specifications

2.3.1 Device type

<ul style="list-style-type: none">· E0C6281 <input type="checkbox"/>· E0C62L81 <input type="checkbox"/>
--

Select the chip specification.

E0C6281 and E0C62L81 denote 3[V] power source voltage specification, and LOW POWER specification for 1.5[V] power source voltage, respectively.

When it uses E0C62A81, it selects E0C6281 and in the case of E0C62B81, it does E0C62L81.

2.3.2 Multiple key entry reset

- Not Use	<input type="checkbox"/>
- Use	<input type="checkbox"/> K00, K01
	<input type="checkbox"/> K00, K01, K02
	<input type="checkbox"/> 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 high at the same time. Similarly, the system is reset as soon as the K00 through K02 inputs or the K00 through K03 inputs go high. However, the system is reset when a high signal is input for more than a rule time (1-3[sec]).

The system reset circuit is shown in Figure 2.3.2.

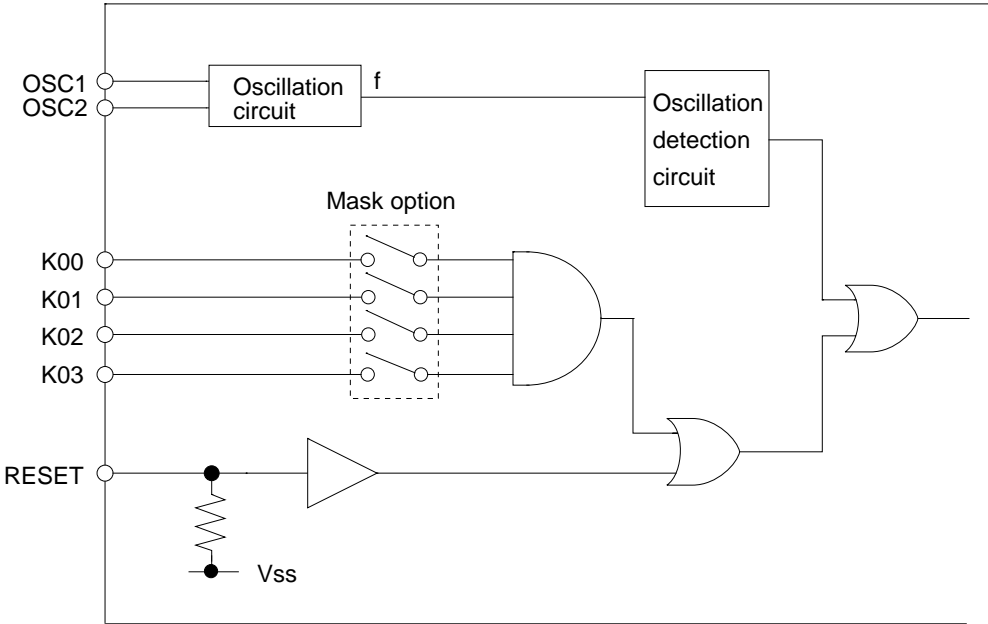


Fig. 2.3.2 The System Reset Circuit

2.3.3 Interrupt noise rejector

- | | | |
|-----------------|------------------------------|----------------------------------|
| · K00–K03 | <input type="checkbox"/> Use | <input type="checkbox"/> Not Use |
| · K10 | <input type="checkbox"/> Use | <input type="checkbox"/> Not Use |

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

2.3.4 Input ports pull down resistor

- | | | |
|-------------|--|--------------------------------------|
| · K00 | <input type="checkbox"/> With Resistor | <input type="checkbox"/> Gate Direct |
| · K01 | <input type="checkbox"/> With Resistor | <input type="checkbox"/> Gate Direct |
| · K02 | <input type="checkbox"/> With Resistor | <input type="checkbox"/> Gate Direct |
| · K03 | <input type="checkbox"/> With Resistor | <input type="checkbox"/> Gate Direct |
| · K10 | <input type="checkbox"/> With Resistor | <input type="checkbox"/> Gate Direct |

Select whether input ports (K00–K03 and K10) will each be supplemented with pull down resistors or not.

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

Moreover, the input port status is changed from "H" level (V_{DD}) to "L" (V_{SS}) with pull down resistors, a delay of approximately 1[ms] in waveform rise time will occur depending on the pull down resistor and entry load time constant. Because of this, when input reading is to be conducted, ensure the appropriate wait time with the program.

The configuration of the pull down resistor circuit is shown in Figure 2.3.4.

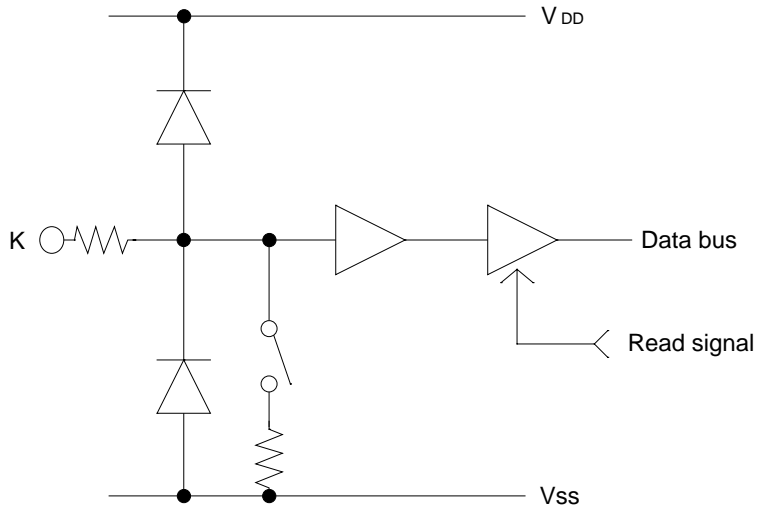


Fig. 2.3.4 Configuration of Pull Down Resistors

2.3.5 Output port (R00–R03) output specification

· R00	<input type="checkbox"/> Complementary	<input type="checkbox"/> Pch-OpenDrain
· R01	<input type="checkbox"/> Complementary	<input type="checkbox"/> Pch-OpenDrain
· R02	<input type="checkbox"/> Complementary	<input type="checkbox"/> Pch-OpenDrain
· R03	<input type="checkbox"/> Complementary	<input type="checkbox"/> Pch-OpenDrain

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

Either complementary output or Pch open drain output may be selected.

When output port is to be used on key matrix configuration, select Pch open drain output.

For unused output ports, select complementary output.

The output circuit configuration is shown in Figure 2.3.5.

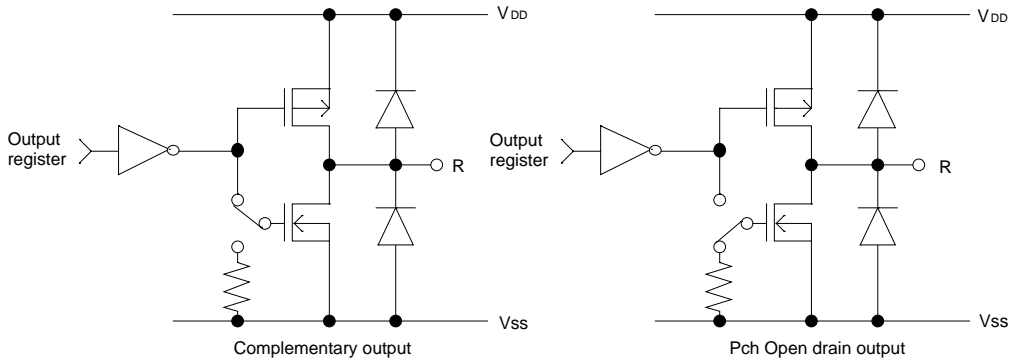


Fig. 2.3.5 Configuration of Output Circuit

2.3.6 R10 specification

· OUTPUT TYPE	<input type="checkbox"/> D.C.
	<input type="checkbox"/> FOUT 32768 [Hz]
	<input type="checkbox"/> FOUT 16384 [Hz]
	<input type="checkbox"/> FOUT 8192 [Hz]
	<input type="checkbox"/> FOUT 4096 [Hz]
	<input type="checkbox"/> FOUT 2048 [Hz]
	<input type="checkbox"/> FOUT 1024 [Hz]
	<input type="checkbox"/> FOUT 512 [Hz]
	<input type="checkbox"/> FOUT 256 [Hz]
· OUTPUT SPECIFICATION	<input type="checkbox"/> Complementary <input type="checkbox"/> Pch-OpenDrain

Select the output specification for R10 terminal.

Either complementary output or Pch open drain output may be selected.

When DC output is selected, R10 becomes a regular output port. When FOUT is selected, clock with frequency selected from R10 terminal is generated by writing "1" to the R10 register.

– When DC output is selected

When R10 register (F4 address, D0 bit) is set to "1", the R10 terminal output goes high (V_{DD}), and goes low (V_{SS}) when set to "0".

Output waveform is shown in Figure 2.3.6.a.

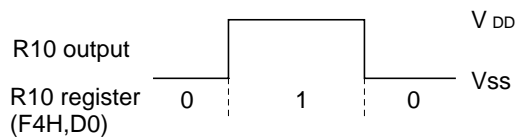


Fig. 2.3.6.a Output Waveform at DC Output Selection

– When FOUT output is selected

When FOUT bit (R10 register) is set to "1", 50% duty and $V_{DD}-V_{SS}$ amplitude square wave is generated at the specified frequency. When set to "0", the FOUT terminal goes low (V_{SS}). 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 2.3.6.b.

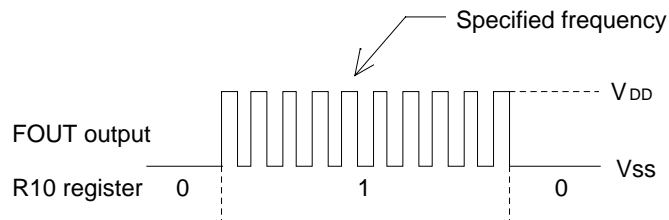


Fig. 2.3.6.b Output Waveform at R10 FOUT Output Selection

2.3.7 R11 specification

· OUTPUT SPECIFICATION <input type="checkbox"/> Complementary <input type="checkbox"/> Pch-OpenDrain
--

Select the output specification for R11 terminal.

Either complementary output or Pch open drain output may be selected.

The circuit configuration is the same as that of output ports (R00–R03 shown in Figure 2.3.5).

2.3.8 R12 specification

· OUTPUT TYPE	<input type="checkbox"/> D.C.
	<input type="checkbox"/> Melody inverted output (fixed to Complementary)
	<input type="checkbox"/> Envelope (fixed to Nch-OpenDrain)
· OUTPUT SPECIFICATION	<input type="checkbox"/> Complementary <input type="checkbox"/> Pch-OpenDrain

Select the specification for R12 terminal.

Any one of the following may be selected for the output type: DC output, melody inverted output, or envelope.

When DC output is selected, either complementary output or Pch open drain output may be selected for the output specification.

When melody inverted output is selected, output specification is fixed to complementary output.

When envelope is selected, output specification is fixed to Nch open drain output.

– When DC output is selected

When R12 register (F4 address, D2 bit) is set to "1", the R12 terminal output goes high (V_{DD}), and goes low (V_{SS}) when set to "0".

Output waveform is shown in Figure 2.3.8.a.

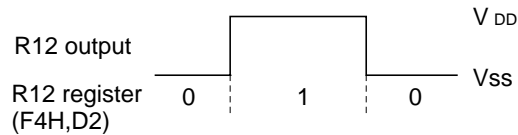


Fig. 2.3.8.a Output Waveform at DC Output Selection

– When melody inverted output is selected

When a melody is output from MO (R13), R12 terminal outputs the melody inverted waveform. Figure 2.3.8.b shows the output waveform when melody inverted output is selected.

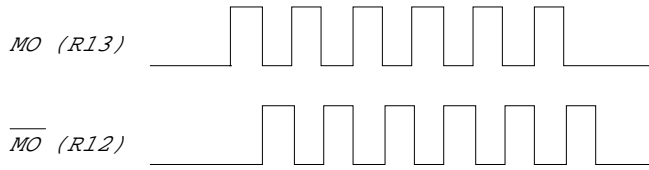


Fig. 2.3.8.b R12 Terminal Output Waveform when Melody Inverted Output is Selected

– When envelope is selected

Envelope waveform can be generated from MO terminal by connecting the capacitor to R12 terminal.

Figure 2.3.8.c shows the output waveform when envelope is selected.

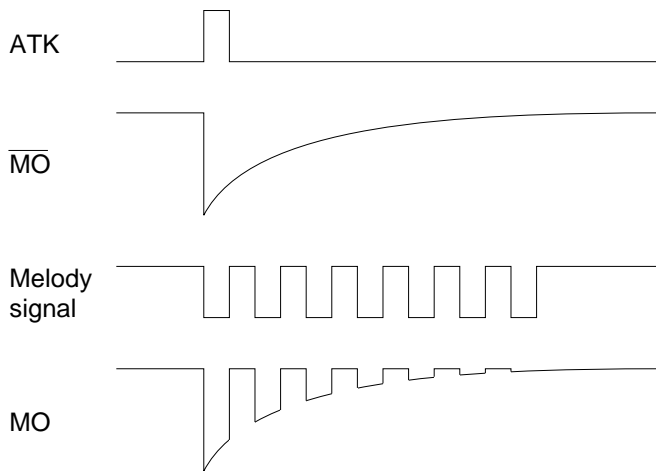


Fig. 2.3.8.c Output Waveform when Envelope is Selected

2.3.9 I/O port specification

- | | | |
|-------------|--|--|
| · P00 | <input type="checkbox"/> Complementary | <input type="checkbox"/> Pch-OpenDrain |
| · P01 | <input type="checkbox"/> Complementary | <input type="checkbox"/> Pch-OpenDrain |
| · P02 | <input type="checkbox"/> Complementary | <input type="checkbox"/> Pch-OpenDrain |
| · P03 | <input type="checkbox"/> Complementary | <input type="checkbox"/> Pch-OpenDrain |

Select the output specification to be used during I/O ports (P00–P03) output mode selection.

Either complementary output or Pch 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 2.3.5).

Select complementary output for unused ports.

The I/O ports can control the input/output direction according to the IOC bit (FC address, D0 bit); at "1" and "0" settings, it is set to output port and input port, respectively.

The pull down resistor of this port is turned on by the read signal and is normally turned off to minimize leak current.

Because of this, when the port is set for input, take care that a floating state does not occur in the terminal.

The circuit configuration I/O ports is shown in Figure 2.3.9.

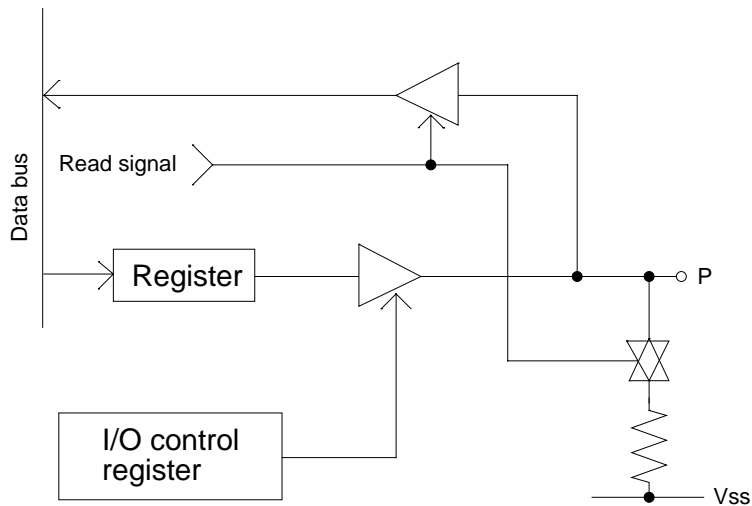


Fig. 2.3.9 The Circuit Configuration I/O Ports

2.3.10 LCD common duty

1/4 DUTY 1/4
 1/3 DUTY 1/3

Select the common (drive) duty for the LCD segment.

When 1/3 duty is selected, with 3 COM terminals and 26 SEG terminals, i.e., up to 78 segments may be driven; when 1/4 duty is selected, with 4 COM terminals and 26 SEG terminals, up to 104 segment drives will be possible.

When 1/3 duty is selected, COM terminals COM0–COM2 become effective and COM3 will always generate OFF signals.

For drive duty selection, please refer to Table 2.3.10.

Table 2.3.10 Common (Drive) Duty Selection Standard

Number of LCD segment drives	Common (Drive) duty
1 – 78	1/3
79 – 104	1/4

The drive waveform of the COM terminals is shown in Figure 2.3.10.

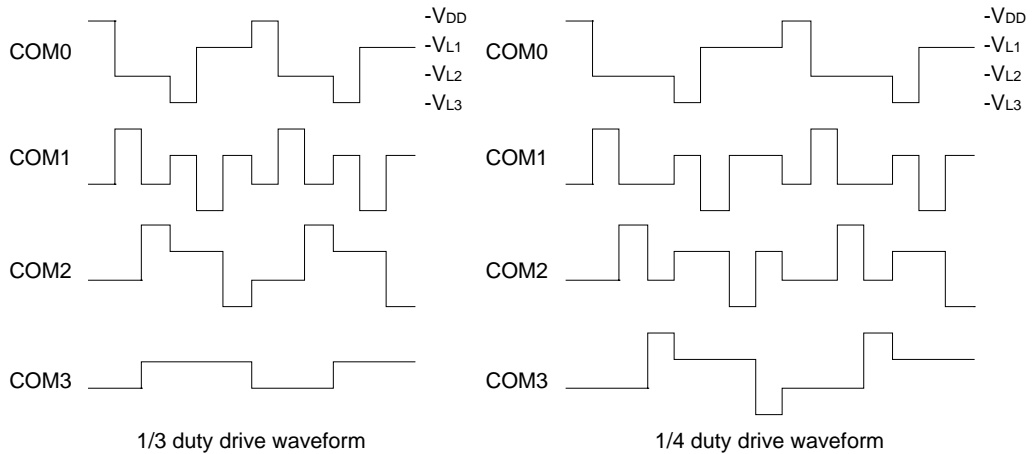


Fig. 2.3.10 Drive Waveform of COM Terminals

2.3.11 OSC1 system clock

- | |
|--|
| <ul style="list-style-type: none">· Crystal <input type="checkbox"/>· CR <input type="checkbox"/> |
|--|

Select oscillation circuit that uses OSC1 and OSC2.

To minimize external components, CR oscillation circuit would be suitable; to obtain a stable oscillation frequency, crystal oscillation circuit would be suitable.

When CR oscillation circuit is selected, only resistors are needed as external components since capacities are built-in.

On the other hand, when crystal oscillation circuit is selected, crystal oscillator and trimmer capacitor are needed as external components. Although when crystal oscillation circuit is selected, it is fixed at 32.768[kHz], when CR oscillation circuit is selected, frequency may be modified to a certain extent depending on the resistance of external components.

3 FUNCTION OPTION GENERATOR OPERATION PROCEDURE

3.1 Creating Work Disk

To prevent inadvertent destruction of the contents of the FOG6281 program disk, create a work disk by copying the program disk, place a write protection tab on the system disk, and keep the system 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 FOG6281 program disk.

Example: Insert the FOG6281 program disk into drive A and the 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 "FOG6281.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 FOG6281

To start FOG6281, 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.

- * In examples,  means press the RETURN key "↵".

```
A>FOG6281 
```

When FOG6281 is started, the following message is displayed.

```

*** E0C6281 FUNCTION OPTION GENERATOR. --- Ver 3.00 ***

EEEEEEEEEE P P P P P P P P P S S S S S S S S O O O O O O O O N N N N N N
EEEEEEEEEE P P P P P P P P P S S S S S S S S O O O O O O N N N N N N
EEE        P P P P P P P P P S S S S S S S S O O O O O O N N N N N N
EEE        P P P P P P P P P S S S S S S S S O O O O O O N N N N N N
EEEEEEEEEE P P P P P P P P P S S S S S S S S O O O O O O N N N N N N
EEEEEEEEEE P P P P P P P P P S S S S S S S S O O O O O O N N N N N N
EEE        P P P P P P P P P S S S S S S S S O O O O O O N N N N N N
EEE        P P P P P P P P P S S S S S S S S O O O O O O N N N N N N
EEEEEEEEEE P P P P P P P P P S S S S S S S S O O O O O O N N N N N N
EEEEEEEEEE P P P P P P P P P S S S S S S S S O O O O O O N N N N N N

          (C) COPYRIGHT 1989 SEIKO EPSON CORP.

THIS SOFTWARE MAKES NEXT FILES.

C232XXXF.HEX ... FUNCTION OPTION HEX FILE.
C232XXXF.DOC ... FUNCTION OPTION DOCUMENT FILE.

          STRIKE ANY KEY.
```

For "STRIKE ANY KEY," press any key to advance the program execution. To suspend execution, hold down CTRL and press C: 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.

```
*** E0C6281 USER'S OPTION SETTING. --- Ver 3.00 ***  
  
CURRENT DATE IS 89/07/13  
PLEASE INPUT NEW DATE : 89/07/20 
```

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 (C281XXXF.DOC) generated by "1. INPUT NEW FILE".
3. RETURN TO DOS: Used to terminate FOG6281 and return to the DOS command level.

3.3 Setting New Function Options

This section explains how to set new function options.

* In examples, means press the RETURN key "↵".

```

*** OPERATION SELECT MENU ***

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

PLEASE SELECT NO.? 1 .. (1)

PLEASE INPUT FILE NAME? C2810A0 .. (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 (C281XXXF.DOC) with the same name as the file name specified in the current drive exists, the user is asked whether overwriting is desired. Enter "Y" or "N" accordingly.

Example: PLEASE INPUT FILE NAME? C2810N0
 EXISTS OVERWRITE (Y/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. 11 sequentially and interactively. (See 3.5 for the option selection procedure.)

3.4 Modifying Function Option Settings

This section explains how to modify the function option settings.

* In examples, means press the RETURN key "".

```

*** OPERATION SELECT MENU ***

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

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

*** SOURCE FILE(S) ***

C2810A0          C2810B0          C2810C0          ... (2)

PLEASE INPUT FILE NAME? C2810A0        ... (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 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 (C281XXXXF.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? C2810N0
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 11) of the function option to be modified, then start setting the option contents (See 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.

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

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

PLEASE INPUT EDIT NO.? 9

- 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 1–11 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 "↵".

3.5.1 Selecting the device type

```

*** OPTION NO.1 ***

--- DEVICE TYPE ---

      1. E0C6281  ( Normal Type )
      2. E0C62L81 ( Low Power Type )

PLEASE SELECT NO.(1) ? 2 

      2. E0C62L81 ( Low Power Type )   SELECTED

```

3.5.2 Selecting multiple key entry reset function

```

*** OPTION NO.2 ***

--- 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) ? 2 

      2. Use  K00,K01   SELECTED

```

3.5.3 Selecting input interrupt noise rejector

```

*** OPTION NO.3 ***

--- INTERRUPT NOISE REJECTOR ---

      K00-K03          1. Use
                      2. Not Use

PLEASE SELECT NO.(1) ? 1 

      K10              1. Use
                      2. Not Use

PLEASE SELECT NO.(1) ? 1 

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

```

3.5.4 Selecting input port pull down resistor

*** OPTION NO.4 ***

--- INPUT PORT PULL DOWN RESISTOR ---

K00 1. With Resistor
 2. Gate Direct

PLEASE SELECT NO.(1) ? 1

K01 1. With Resistor
 2. Gate Direct

PLEASE SELECT NO.(1) ? 1

K02 1. With Resistor
 2. Gate Direct

PLEASE SELECT NO.(1) ? 1

K03 1. With Resistor
 2. Gate Direct

PLEASE SELECT NO.(1) ? 1

K10 1. With Resistor
 2. Gate Direct

PLEASE SELECT NO.(1) ? 1

K00	1. With Resistor	SELECTED
K01	1. With Resistor	SELECTED
K02	1. With Resistor	SELECTED
K03	1. With Resistor	SELECTED
K10	1. With Resistor	SELECTED

3.5.5 Selecting output port (R00–R03) output specification

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

--- OUTPUT PORT OUTPUT SPECIFICATION ( R00-R03 ) ---

      R00          1. Complementary
                  2. Pch-OpenDrain

PLEASE SELECT NO.(1)? 1 

      R01          1. Complementary
                  2. Pch-OpenDrain

PLEASE SELECT NO.(1)? 1 

      R02          1. Complementary
                  2. Pch-OpenDrain

PLEASE SELECT NO.(1)? 2 

      R03          1. Complementary
                  2. Pch-OpenDrain

PLEASE SELECT NO.(1)? 2 

      R00          1. Complementary   SELECTED
      R01          1. Complementary   SELECTED
      R02          2. Pch-OpenDrain   SELECTED
      R03          2. Pch-OpenDrain   SELECTED
```

3.5.6 Selecting R10 terminal specification

```

*** OPTION NO.6 ***

--- R10 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) ? 2 [↓]

      OUTPUT SPECIFICATION        1. Complementary
                                   2. Pch-OpenDrain

PLEASE SELECT NO.(1) ? 1 [↓]

      OUTPUT TYPE                2. Fout  32768 [Hz]  SELECTED
      OUTPUT SPECIFICATION        1. Complementary  SELECTED

```

3.5.7 Selecting R11 terminal specification

```

*** OPTION NO.7 ***

--- R11 SPECIFICATION ---

      1. Complementary
      2. Pch-OpenDrain

PLEASE SELECT NO.(1) ? 2 [↓]

      2. Pch-OpenDrain  SELECTED

```

3.5.8 Selecting R12 terminal specification

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

--- R12 PORT OUTPUT SPECIFICATION ---

      OUTPUT TYPE          1. D.C.
                          2. /MELODY OUTPUT
                          3. ENVELOPE

PLEASE SELECT NO.(1) ? 2 

      OUTPUT SPECIFICATION  1. Complementary
                          2. Pch-OpenDrain

PLEASE SELECT NO.(1) ? 

      OUTPUT TYPE          2. /MELODY OUTPUT   SELECTED
      OUTPUT SPECIFICATION 1. Complementary   SELECTED
```

3.5.9 Selecting I/O port specification

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

--- I/O PORT OUTPUT SPECIFICATION ---

      P00          1. Complementary
                  2. Pch-OpenDrain

PLEASE SELECT NO.(1) ? 1 

      P01          1. Complementary
                  2. Pch-OpenDrain

PLEASE SELECT NO.(1) ? 1 

      P02          1. Complementary
                  2. Pch-OpenDrain

PLEASE SELECT NO.(1) ? 1 

      P03          1. Complementary
                  2. Pch-OpenDrain

PLEASE SELECT NO.(1) ? 1 

      P00          1. Complementary   SELECTED
      P01          1. Complementary   SELECTED
      P02          1. Complementary   SELECTED
      P03          1. Complementary   SELECTED
```

3.5.10 Selecting LCD common (drive) duty

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

--- LCD COMMON DUTY ---

    1. 1/4 Duty
    2. 1/3 Duty

PLEASE SELECT NO.(1) ? 1 

    1. 1/4 Duty   SELECTED
```

3.5.11 Selecting OSC1 oscillation circuit

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

--- OSC 1 SYSTEM CLOCK ---

    1. Cristal
    2. CR

PLEASE SELECT NO.(1) ? 1 

    1. Cristal   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.? 2           ..... (2)

      2. 27C128   SELECTED
  
```

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

When debugging the program with EVA6281, HEX file C281XXXXF.HEX is needed, so enter "Y". If "N" is entered, no HEX file is generated and only document file C281XXXXF.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 EVA6281 options. This menu is not displayed when "N" is entered in Step (1).

One EPROM is required for setting function options (27C128 is selected in the above example).

When the above operation is completed, FOG6281 generates files and the sequence returns to the operation selection menu.

```

MAKING FILE(S) IS COMPLETED.
  
```

Note: The EPROM to be mounted on the EVA6281 must satisfy the following conditions:

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

3.7 End Procedure

This section explains how to end FOG6281 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 FOG6281 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.

FOG6281 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.)

4 SAMPLE FILE

Function Option Document File

```

* E0C6281 FUNCTION OPTION DOCUMENT V 3.00
*
* FILE NAME      C2810A0F.DOC
* USER'S NAME    SEIKO EPSON CORP.
* INPUT DATE     89/07/20
* COMMENT        TOKYO DESIGN CENTER
*                390-4 HINO HINO-SHI TOKYO 191 JAPAN
*                TEL 0425-83-7313
*                FAX 0425-83-7413
*
*
* OPTION NO.1
* < DEVICE TYPE >
*   E0C62L81 (LOW POWER TYPE) ----- SELECTED
OPT0101 02
*
* OPTION NO.2
* < MULTIPLE KEY ENTRY RESET >
*   COMBINATION USE  K00,K01 ----- SELECTED
OPT0201 02
*
* OPTION NO.3
* < INTERRUPT NOISE REJECTOR >
*   K00-K03  USE ----- SELECTED
*   K10      USE ----- SELECTED
OPT0301 01
OPT0302 01
*
* OPTION NO.4
* < INPUT PORT PULL DOWN RESISTOR >
*   K00 WITH RESISTOR ----- SELECTED
*   K01 WITH RESISTOR ----- SELECTED
*   K02 WITH RESISTOR ----- SELECTED
*   K03 WITH RESISTOR ----- SELECTED
*   K10 WITH RESISTOR ----- SELECTED
OPT0401 01
OPT0402 01
OPT0403 01
OPT0404 01
OPT0405 01
*
* OPTION NO.5
* < OUTPUT PORT OUTPUT SPECIFICATION ( R00-R03 ) >
*   R00 COMPLEMENTARY ----- SELECTED
*   R01 COMPLEMENTARY ----- SELECTED
*   R02 PCH-OPENDRAIN ----- SELECTED
*   R03 PCH-OPENDRAIN ----- SELECTED
OPT0501 01
OPT0502 01

```

```

OPT0503 02
OPT0504 02
*
* OPTION NO.6
* < R10 SPECIFICATION >
*   OUTPUT TYPE           FOUT  32768 [HZ]  -----  SELECTED
*   OUTPUT SPECIFICATION  COMPLEMENTARY -----  SELECTED
OPT0601 02
OPT0602 01
OPT0603 01
*
* OPTION NO.7
* < R11 SPECIFICATION >
*   PCH-OPENDRAIN -----  SELECTED
OPT0701 02
*
* OPTION NO.8
* < R12 PORT OUTPUT SPECIFICATION >
*   OUTPUT TYPE           /MELODY OUTPUT -----  SELECTED
*   OUTPUT SPECIFICATION  COMPLEMENTARY -----  SELECTED
OPT0801 02
OPT0802 01
*
* OPTION NO.9
* < I/O PORT OUTPUT SPECIFICATION >
*   P00  COMPLEMENTARY -----  SELECTED
*   P01  COMPLEMENTARY -----  SELECTED
*   P02  COMPLEMENTARY -----  SELECTED
*   P03  COMPLEMENTARY -----  SELECTED
OPT0901 01
OPT0902 01
OPT0903 01
OPT0904 01
*
* OPTION NO.10
* < LCD COMMON DUTY >
*   1/4 DUTY -----  SELECTED
OPT1001 01
*
* OPTION NO.11
* < OSC 1 SYSTEM CLOCK >
*   CRISTAL -----  SELECTED
OPT1101 01
*
*
* SEIKO EPSON'S AREA
*
*
OPT1201 01
OPT1202 01
OPT1203 01
OPT1204 01

```

```
*  
OPT1301 02  
OPT1302 02  
OPT1303 02  
OPT1304 02  
*  
OPT1401 01  
*  
OPT1501 01  
OPT1502 01  
*  
OPT1601 01  
OPT1602 01  
¥¥END
```

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

***IV.* E0C6281 SEGMENT OPTION GENERATOR MANUAL**

PREFACE

This manual explains how to operate the SOG6281 Segment Option Generator for setting the hardware options of 4-bit single-chip E0C6281 microcomputer and details the specifications of their options.

Refer to "E0C6281 Technical Hardware Manual" for details about the E0C6281. Refer to "E0C62 Family Technical Guide" for details about the development procedure.

CONTENTS

1 GENERAL	<i>IV-1</i>
1.1 Outline of Segment Option Generator	<i>IV-1</i>
1.2 Execution Flow and I/O Files	<i>IV-2</i>
2 OPTION LIST GENERATION	<i>IV-4</i>
2.1 Option List Recording Procedure	<i>IV-4</i>
2.2 Option List	<i>IV-4</i>
2.3 Segment Ports Output Specifications	<i>IV-5</i>
3 SEGMENT OPTION GENERATOR OPERATION PROCEDURE	<i>IV-6</i>
3.1 Creating Work Disk	<i>IV-6</i>
3.2 Creating Segment Option Source File	<i>IV-7</i>
3.3 Starting SOG6281	<i>IV-10</i>
3.4 Input File Selection	<i>IV-11</i>
3.5 HEX File Generation and EPROM Selection	<i>IV-13</i>
3.6 End Procedure	<i>IV-14</i>
4 ERROR MESSAGES	<i>IV-15</i>
5 SAMPLE FILES	<i>IV-19</i>
5.1 Segment Option Source File	<i>IV-19</i>
5.2 Segment Option Document File	<i>IV-20</i>

1 GENERAL

1.1 Outline of Segment Option Generator

With the 4-bit single-chip E0C6281 microcomputers, the customer may select the LCD segment options. By modifying the mask patterns of the E0C6281 according to the selected options, the system can be customized to meet the specifications of the target system.

The SOG6281 Segment Option Generator (hereinafter called SOG6281) is a software tool for generating data files used to generate mask patterns. From the data file created with SOG6281, the E0C6281 mask pattern is automatically generated by a general purpose computer.

The HEX file for the evaluation board (EVA6281) 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 EVA6281, option functions can be executed on the EVA6281.

Two SOG6281 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) x 1 or more
Operating system: MS-DOS Ver. 3.1 or later
ROM writer: Required when using EVA6281

– IBM PC/XT and PC/AT

Host computer: IBM PC/XT and PC/AT
Disk drive: FD (5.25" 2D) x 1 or more
Operating system: PC-DOS Ver. 2.1 or later
ROM writer: Required when using EVA6281

The program name of SOG6281 is as follows:

SOG6281.EXE

1.2 Execution Flow and I/O Files

Figure 1.2 shows the SOG6281 execution flow.

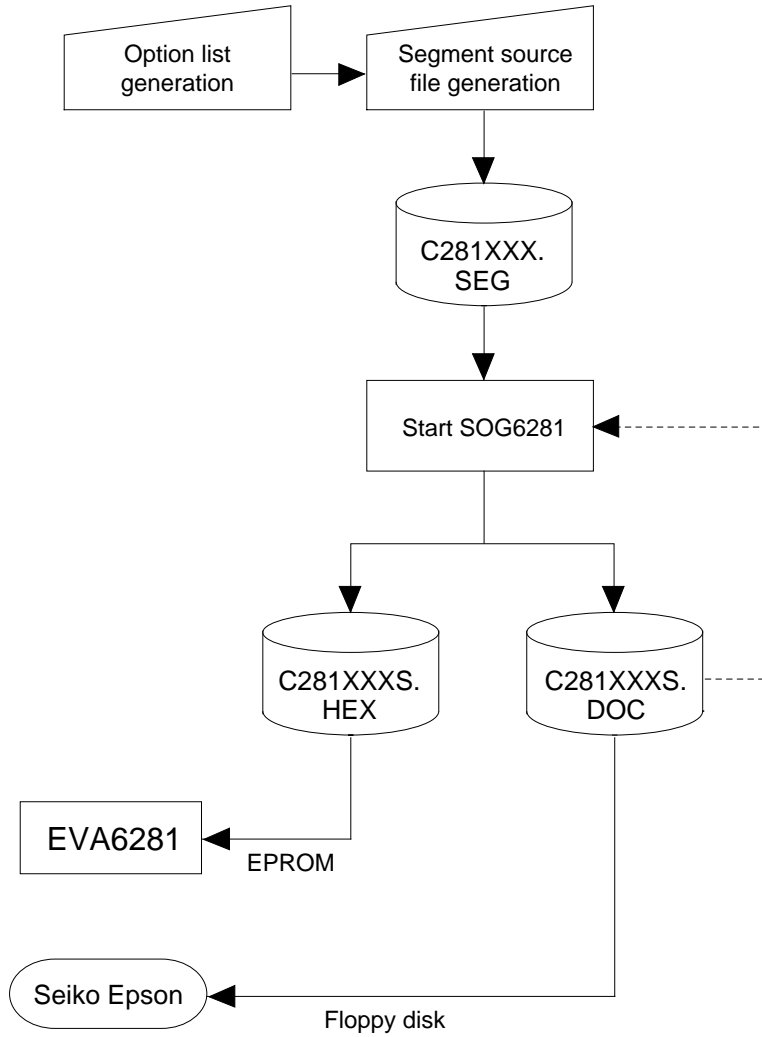


Fig. 1.2 Execution Flow

(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) Segment source file (C281XXX.SEG) generation

The specifications of segment ports must be set in the segment source file (input file for SOG6281). If the segment source file is not generated, SOG6281 stops execution.

Generate the segment source file using an editor such as EDLIN while referencing the option list generated by (1). The file name is C281XXX.SEG.

(3) SOG6281 execution

SOG6281 outputs the following data files:

- Segment option document file (C281XXXS.DOC)
This is a data file used to generate the mask patterns of the segment decoder and segment output port.
This file must be sent with the completed program file.
- Segment option HEX file (C281XXXS.HEX)
This is a segment option file for EVA6281 (Intel hexa format). Two EVA6281 segment option ROMs containing the same data are generated by writing this file with the ROM writer.

- * File name "XXX" is specified for each customer by Seiko Epson.
- * Copy the segment option document file, the program files (C281XXXH.HEX and C281XXXL.HEX) and the function option document file (C281XXXF.DOC) in a batch to another disk and send it to Seiko Epson.
- * Set all unused ROM areas to FFH when writing the HEX file into the EPROM. (Refer to "EVA6281 Manual" for the ROM installation location.)

2 OPTION LIST GENERATION

2.1 Option List Recording Procedure

Multiple specifications are available in segment option item as indicated in the Option List in Section 2.2. Using "2.3 Segment ports output 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.

Furthermore, write the segment memory addresses as well as the selected output specifications.

2.2 Option List

TERMINAL NAME	ADDRESS												OUTPUT SPECIFICATION				
	COM0			COM1			COM2			COM3							
	H	L	D	H	L	D	H	L	D	H	L	D					
SEG0															SEG output		
SEG1															DC output	C	P
SEG2															SEG output		
SEG3															DC output	C	P
SEG4															SEG output		
SEG5															DC output	C	P
SEG6															SEG output		
SEG7															DC output	C	P
SEG8															SEG output		
SEG9															DC output	C	P
SEG10															SEG output		
SEG11															DC output	C	P
SEG12															SEG output		
SEG13															DC output	C	P
SEG14															SEG output		
SEG15															DC output	C	P
SEG16															SEG output		
SEG17															DC output	C	P
SEG18															SEG output		
SEG19															DC output	C	P
SEG20															SEG output		
SEG21															DC output	C	P
SEG22															SEG output		
SEG23															DC output	C	P
SEG24															SEG output		
SEG25															DC output	C	P
Legend: <ADDRESS>												<OUTPUT SPECIFICATION>					
H: High order address within the page (9–A)												C: Complementary output					
L: Low order address within the page (0–F)												P: Pch open drain output					
D: Data bit (0–3)																	

Note: 1. Even if there are unused areas, set "---" (3 hyphens) in the COM0–COM3 such that there are no blank columns.

2. When DC output is selected, the segment memory of the COM0 column becomes.

2.3 Segment Ports Output Specifications

The E0C6281 as a segment port has 26 (SEG0–SEG25) output terminals; segment output and DC output can be selected in units of two terminals. When used for liquid crystal panel drives, select segment output; when used as regular output port, select DC output. When DC output is selected, either complementary output or Pch open drain output may further be selected. However, for segment output ports that will not be used, select segment output.

– When segment output is selected

The segment output port has a segment decoder built-in, and the data bit of the optional address in the segment memory area (090H–0AFH) can be allocated to the optional segment.

With this, up to 104 segments (78 segments when 1/3 duty is selected) of liquid crystal panel could be driven. However, the segment memory may be allocated only one segment and multiple setting is not possible. Accordingly, it is required that all of the 104 segments be allocated unique addresses and data bits.

The allocated segment displays when the bit for this segment memory is set to 1, and goes out when bit is set to 0.

Segment allocation is set to H for high address (9–A), to L for low address (0–F), and to D for data bit (0–3) and are recorded in their respective column in the option list. For segment ports that will not be used, write "–" (hyphen) in the H, L, and D columns of COM0–COM3.

– When DC output is selected

The DC output can be selected in units of 2 terminals and up to 26 terminals may be allocated for DC output. Also, either complementary output or Pch open drain output is likewise selected in units of 2 terminals. When the bit for the selected segment memory is set to 1, the segment output port goes high (V_{DD}), and goes low (V_{SS}) when set to 0. Segment allocation is the same as when segment output is selected but for the while the segment memory allocated to COM1–COM3 becomes ineffective. Write "–" (hyphen) in the COM1–COM3 columns in the option list.

3 SEGMENT OPTION GENERATOR OPERATION PROCEDURE

3.1 Creating Work Disk

To prevent inadvertent destruction of the contents of the SOG6281 program disk, create a work disk by copying the program disk, place a write protection tab on the program disk, and keep the system 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 SOG6281 program disk.

Example: Insert the SOG6281 program disk into drive A and the 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 "SOG6281.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 Creating Segment Option Source File

The segment option generator requires a segment option source file containing segment output port specifications to serve as an input file. Refer to the specification explanation in Section 2.3 and after recording out the necessary items in the option list, create a segment option source file according to the format described in below.

The SOG6281 needs, as an input file, a segment option source file containing the specifications for the segment output ports. Using the editor, generate this source file on the work disk by referencing the contents of the option list.

Use the following file name. For XXX, enter the string distributed by Seiko Epson.

C281XXX.SEG

(XXX: A string distributed to each customer from Seiko Epson)

Example: C2810A0.SEG

Write the output specifications (SEG output, DC complementary output, or DC Pch open drain output) and the segment memory–SEG ports correspondence data (data that associates segment memory addresses to SEG ports) in the file. Comments may also be written in the file. The description procedure is explained by using a sample segment option source file.

```

; C2810A0.SEG, VER.1.21
; EVA6281 LCD SEGMENT DECODE TABLE
;
0      901 900 932 A20 S      ;1st DIGIT
1      912 911 910 923 S
2      913 920 921 922 S
3      A00 902 930 931 S
4      941 940 972 A21 S      ;2nd DIGIT
5      952 951 950 963 S
6      953 960 961 962 S
7      A01 942 970 971 S
8      981 980 9B2 A22 S      ;3rd DIGIT
9      992 991 990 9A3 S
10     993 9A0 9A1 9A2 S
:      :      :      :      :
21     A03 A43 A83 AC3 S
22     933 973 9B3 9F3 S
23     A33 A73 AB3 AF3 S
24     AE0 --- --- --- C      ;DC OUTPUT
25     AF0 --- --- --- C

```

} Comment

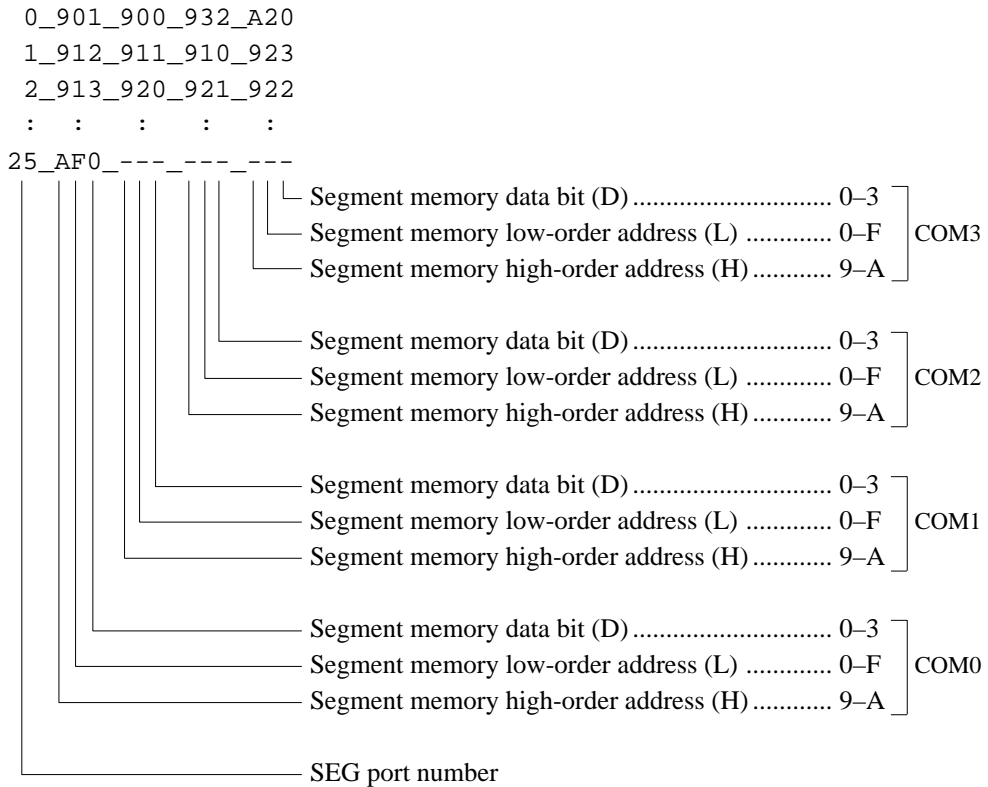
Segment memory–SEG ports
correspondence data
Output
specification data
Comment

(1) Comment

A statement beginning with a semicolon ";" is considered a comment. Such items as date, summary, and version may be written in such a line.

(2) Segment memory–SEG ports correspondence data

This data indicates correspondence between segment memory addresses and segment ports. The arrangement is the same as that of the option list, so write the data in the following format while referencing the option list.



- Each SEG port number corresponds to an actual device, so it must be unique. Moreover, data descriptions in accordance with the following format are required for segments SEG0–SEG25.
- Off areas COM0 to COM3, write three successive "----" (3 hyphens) as data for unused areas. SEG port numbers are needed even if the ports themselves will not be used, so write "----"(3 hyphens) for all areas COM0 to COM3.

Example: When not using COM2 in SEG8

```
8_981_980_---_A22
```

Example: When not using SEG12

```
12_---_---_---_---
```

- When "DC output" is selected, write the segment memory–SEG ports correspondence data for COM0; "---"(hyphen) for COM1 to COM3.

Example: When outputting SEG21 and SEG22 as DC output

```
21_933_---_---_---
```

```
22_A33_---_---_---
```

- Symbol "_" indicates a blank or tab. Be sure to write one or more blanks or a tab between the SEG port number, COM0, COM1, COM2, and COM3.

(3) Output specification selection data

This data is used to specify whether the SEG port will be used as a segment output port, a DC complementary output port, or a DC Pch open drain output port.

Write data after inserting one or more blanks or a tab after the segment memory–SEG ports correspondence data.

S: Segment output

C: DC complementary output

P: DC Pch open drain output

- The SEG port output specifications must be selected in units of two ports, so write the selection data carefully while referencing the option list.

Example: When outputting SEG23 and SEG24 as DC complementary output

```
23_AE0_---_---_--- C
```

```
24_AF0_---_---_--- C
```

- Select "SEG output" for the segment ports that will not be used.

Example: When not using SEG18

```
18_---_---_---_--- S
```

Note: Only complementary output is enabled as the DC output of the SEG ports of EVA6281. Therefore, complementary output is enabled even if Pch open drain output is selected. Respond to it by adding external circuits as required.

Generate the segment option source file according to the formats and restrictions above.

3.3 Starting SOG6281

To start SOG6281, 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. The work disk must contain the segment option source file (C281XXX.SEG).

- * In examples, means press the RETURN key (↵). _ indicates a blank.
[] indicates that this parameter may be omitted.

```
A>SOG6281_[ -H ] 
```

-H: Specifies the segment option document file (C281XXX.DOC) for input file of SOG6281.

When SOG6281 is started, the following message is displayed.

```

***  E0C6281 SEGMENT OPTION GENERATOR.  --- Ver 3.00  ***

EEEEEEEEEE  PPPPPPPP      SSSSSSS      00000000      NNN      NNN
EEEEEEEEEE  PPPPPPPPPP    SSS  SSSS     000  000      NNNN     NNN
EEE         PPP  PPP     SSS  SSS     000  000      NNNNNN    NNN
EEE         PPP  PPP     SSS          000  000      NNNNNN    NNN
EEEEEEEEEE  PPPPPPPPPP    SSSSSS      000  000      NNN NNN  NNN
EEEEEEEEEE  PPPPPPPP      SSSS        000  000      NNN  NNNNNN
EEE         PPP          SSS          000  000      NNN  NNNNN
EEE         PPP          SSS  SSS     000  000      NNN  NNNN
EEEEEEEEEE  PPP          SSSS  SSS     000  000      NNN  NNN
EEEEEEEEEE  PPP          SSSSSS      00000000      NNN      NN

      (C) COPYRIGHT 1989  SEIKO EPSON CORP.

SEGMENT OPTION SOURCE FILE NAME IS " C281XXX.SEG ".

THIS SOFTWARE MAKES NEXT FILES.

      C281XXXS.HEX  ...  SEGMENT OPTION HEX FILE.
      C281XXXS.DOC  ...  SEGMENT OPTION DOCUMENT FILE.

      STRIKE ANY KEY.
```

For "STRIKE ANY KEY," press any key to advance the program execution. To suspend execution, hold down CTRL and press C: 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.


```

*** E0C6281 USER'S OPTION SETTING. --- Ver 3.00 ***

CURRENT DATE IS 89/07/13
PLEASE INPUT NEW DATE : 89/07/20

```

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.

3.4 Input File Selection

This section explains how to select the input file.

* In examples, ↵ means press the RETURN key "↵".

```

*** SOURCE FILE(S) ***

C2810A0          C2810B0          C2810B1          C2810C0  .. (1)

PLEASE INPUT SEGMENT SOURCE FILE NAME? C2810A0
PLEASE INPUT USER'S NAME? SEIKO EPSON CORP.
PLEASE INPUT ANY COMMENT
(ONE LINE IS 50 CHR)? TOKYO DESIGN CENTER
                    ? 390-4 HINO HINO-SHI TOKYO 191 JAPAN
                    ? TEL 0425-83-7313
                    ? FAX 0425-83-7413
                    ?

```

(1) *** SOURCE FILE(S) ***

- H option use

Will display the segment option source files on the current drive.

If no source files exists, the following message will be displayed and the program will be terminated.

```
SEGMENT OPTION SOURCE FILE IS NOT FOUND.
```

- H option not use

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

If no document files exists, the following message will be displayed and the program will be terminated.

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

(2) PLEASE INPUT SEGMENT SOURCE FILE NAME?

• H option use

Enter the segment option source file name. Do not enter the extended part of the file name. If the specified file name is not found in the current drive, an error message like the one below is output, prompting entry of another file name:

Example: PLEASE INPUT SEGMENT SOURCE FILE NAME? C2810N0
SEGMENT OPTION SOURCE FILE IS NOT FOUND.

• H option not use

Enter the segment option document file name. Do not enter the extended part of the file name. If the specified file name is not found in the current drive, an error message like the one below is output, prompting entry of another file name:

Example: PLEASE INPUT SEGMENT DOCUMENT FILE NAME? C2810N0
SEGMENT OPTION DOCUMENT FILE IS NOT FOUND.

(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

When the above operations are complete, move to the confirmation procedure for HEX file generation.

3.5 HEX File Generation and EPROM Selection

When input file selection 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.? 2           ..... (2)

      2. 27C128   SELECTED

```

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

When debugging the program with EVA6281, HEX file C281XXXS.HEX is needed, so enter "Y". If "N" is entered, no HEX file is generated and only document file C281XXXS.DOC is generated.

However, when H option is used, HEX file is generated without any conditions. Therefore, this menu is not displayed.

(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 EVA6281 options. This menu is not displayed when "N" is entered in Step (1).

A total of two EPROMs of the same type are required for setting segment options (27C128 is selected in the above example).

When the above operation is completed, SOG6281 generates files. If no error is committed while setting segment options, the following message will be displayed and the SOG6281 program will be terminated.

```

MAKING FILE IS COMPLETED.

```

Note: The EPROM to be mounted on the EVA6281 must satisfy the following conditions:

EPROM for setting segment option: $T_{ACC} \leq 170 \text{ ns}$
(T_{ACC} : Access time)

3.6 End Procedure

When a series of operations are complete, the SOG6281 program will be terminated.

OPG6281 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.)

4 ERROR MESSAGES

If an error is detected in the segment option source file, an error message is displayed. In this case, the segment option HEX file is not generated, and the segment option document file consisting of the segment option source file and an error message is generated.

N	12	66	9B0	9B1	9B2	9B3	S
S	16	15	9F0MSD	9F1	9F2	9F3	S
D	20	19	A30	A31	A32	A31	S
N	22	42	A50	A51	A52	A53	S
D	23	22	A60	A61	A31	A31	S
R	25	24	A80	881	A82	A83	S

Duprication is SEG NO. 19 COM NO. 3

Duprication is SEG NO. 22 COM NO. 2

Duprication is SEG NO. 22 COM NO. 3

7 ERROR(S)

STRIKE ANY KEY.

MAKING SEGMENT OPTION FILES IS NOT COMPLETED BY SOURCE FILE ERROR-(S).

If one or more errors are detected, error symbols are output in column 0 and the source lists containing the errors are output in subsequent columns.

The following four error symbols are used for SOG6281:

S: Syntax error

N: Segment number selection error

R: RAM address selection error

D: Duplication error

The priority order is S, N, R, and D.

Each type of error is explained here.

– S: Syntax error

This type of error occurs when the data was written in an invalid format. Correct the segment option source file format.

Example: S 16 15 9F0MSD 9F1 9F2 9F3 S

↑
This format is invalid.

– N: Segment number selection error

This type of error occurs when a segment number outside the specificable range is specified. Correct the segment option source file so that all segment numbers are 0 to 25.

Example: N 12 66 9B0 9B1 9B2 9B3 S
 N 22 42 A50 A51 A52 A53 S

↑
This part is out of the range specified.

– R: RAM address selection error

This type of error occurs when the segment memory address or data bit outside the specificable range is specified. Correct the segment option source file so that all addresses are 90 to AF and all data bits are 0 to 3.

Example: R 25 24 A80 881 A82 A83 S

↑
This part is out of the range specified.

– D: Duplication error

This type of error occurs when the same data (SEG port No., segment memory address, or data bit) is specified more than once. Correct the segment option source file so that each data item is unique in the description.

Example: D 20 19 A30 A31 A32 A31 S
 D 23 22 A60 A61 A31 A31 S

"A31" is used more then once.

Duplication is SEG NO. 19 COM NO. 3

Duplication is SEG NO. 22 COM NO. 3

Message "Duplication is ..." is output only for the second and subsequent duplicated data items.

In some cases, the following error message is output.

– Out port set error

This error occurs when the output specifications were not set in units of two ports. Correct the segment option source file to satisfy this condition.

Example: Segment No. 18 - 19 Out Port Set Error

This error is not checked when one of the above four errors (S, N, R, or D) is detected. Therefore, this error may occur after the above error are corrected.

If an error occurs, the displayed message can be checked by referencing the segment option document file. Correct the segment option source file by comparing it with the option list, then rerun the program.

The following is an example of the segment option document file when some errors occurred.

LINE	SOURCE	STATEMENT				
1	0	900	901	902	903	S
2	1	910	911	912	913	S
3	2	920	921	922	923	S
4	3	930	931	932	933	S
5	4	940	941	942	943	S
6	5	950	951	952	953	S
7	6	960	961	962	963	S
8	7	970	971	972	973	S
9	8	980	981	982	983	S
10	9	990	991	992	993	S
11	10	9A0	9A1	9A2	9A3	S
N 12	66	9B0	9B1	9B2	9B3	S
13	12	9C0	9C1	9C2	9C3	S
14	13	9D0	9D1	9D2	9D3	S
15	14	9E0	9E1	9E2	9E3	S
S 16	15	9F0MSD	9F1	9F2	9F3	S
17	16	A00	A01	A02	A03	S
18	17	A10	A11	A12	A13	S
19	18	A20	A21	A22	A23	S
D 20	19	A30	A31	A32	A31	S
21	20	A40	A41	A42	A43	S
N 22	42	A50	A51	A52	A53	S
D 23	22	A60	A61	A31	A31	S
24	23	A70	A71	A72	A73	S
R 25	24	A80	881	A82	A83	S
26	25	A90	A91	A92	A93	S

S --- Syntax Error
N --- Segment No. Select Error
R --- RAM Address Select Error
D --- Duprication Error

Duprication is SEG NO. 19 COM NO. 3
Duprication is SEG NO. 22 COM NO. 3

5 SAMPLE FILES

5.1 Segment Option Source File

```

; C2810A0.SEG,VER.3.00
; EVA6281 LCD SEGMENT DECODE TABLE
;
0      901 900 932 AE0 S      ;1st DIGIT
1      912 911 910 923 S
2      913 920 921 922 S
3      AC0 902 930 931 S
4      941 940 972 AE1 S      ;2nd DIGIT
5      952 951 950 963 S
6      953 960 961 962 S
7      AC1 942 970 971 S
8      981 980 9B2 AE2 S      ;3rd DIGIT
9      992 991 990 9A3 S
10     993 9A0 9A1 9A2 S
11     AC2 982 9B0 9B1 S
12     9C1 9C0 9F2 AE3 S      ;4th DIGIT
13     9D2 9D1 9D0 9E3 S
14     9D3 9E0 9E1 9E2 S
15     AC3 9C2 9F0 9F1 S
16     A01 A00 A32 AF0 S      ;5th DIGIT
17     A12 A11 A10 A23 S
18     A13 A20 A21 A22 S
19     AD0 A02 A30 A31 S
20     A41 A40 A72 AF1 S      ;6th DIGIT
21     A52 A51 A50 A63 S
22     A53 A60 A61 A62 S
23     AD1 A42 A70 A71 S
24     AF3 --- --- --- C      ;DC OUTPUT
25     AF3 --- --- --- C

```

```
* E0C6281 SEGMENT OPTION DOCUMENT V 3.00
*
* FILE NAME      C2810A0S.DOC
* USER'S NAME   SEIKO EPSON CORP.
* INPUT DATE    89/07/20
* COMMENT       TOKYO DESIGN CENTER
*               390-4 HINO HINO-SHI TOKYO 191 JAPAN
*               TEL 0425-83-7313
*               FAX 0425-83-7413
*
*
* OPTION NO.12
*
* < LCD SEGMENT DECODE TABLE >
*
* SEG COM0 COM1 COM2 COM3 SPEC
*
  0  901  900  932  AE0  S
  1  912  911  910  923  S
  2  913  920  921  922  S
  3  AC0  902  930  931  S
  4  941  940  972  AE1  S
  5  952  951  950  963  S
  6  953  960  961  962  S
  7  AC1  942  970  971  S
  8  981  980  9B2  AE2  S
  9  992  991  990  9A3  S
 10  993  9A0  9A1  9A2  S
 11  AC2  982  9B0  9B1  S
 12  9C1  9C0  9F2  AE3  S
 13  9D2  9D1  9D0  9E3  S
 14  9D3  9E0  9E1  9E2  S
 15  AC3  9C2  9F0  9F1  S
 16  A01  A00  A32  AF0  S
 17  A12  A11  A10  A23  S
 18  A13  A20  A21  A22  S
 19  AD0  A02  A30  A31  S
 20  A41  A40  A72  AF1  S
 21  A52  A51  A50  A63  S
 22  A53  A60  A61  A62  S
 23  AD1  A42  A70  A71  S
 24  AF3  A80  AB2  AF2  C
 25  AF3  A91  A90  AA3  C
¥¥END
```

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

V. E0C6281 DEVELOPMENT TOOL USER'S MANUAL

PREFACE

This manual mainly explains the outline of the development support tool for the 4-Bit Single Chip Microcomputer E0C6281 and starting procedures through the DEV6281 menu.

For details on the E0C6281, refer to the "E0C6281 Technical Manual"; for development procedures and related subject, see the "E0C62 Family Technical Guide".

CONTENTS

CHAPTER 1	OUTLINE OF THE E0C6281 DEVELOPMENT SUPPORT TOOL	V-1
1.1	Developmental Environment	V-1
1.2	Development Tool Management System (DMS6200)	V-2
1.3	Cross Assembler (ASM6281)	V-3
1.4	Function Option Generator (FOG6281)	V-4
1.5	Segment Option Generator (SOG6281)	V-5
1.6	Melody Assembler (MLA6281)	V-6
1.7	In-Circuit Emulator (ICE6200) & ICE Control Software (ICS6281)	V-7
1.8	Mask Data Checker (MDC6281)	V-8
1.9	Evaluation Board (EVA6281)	V-9
1.10	Demonstration Tool (DMT6281)	V-10
CHAPTER 2	CREATION OF DISK FOR DEV6281 EXECUTION	V-11
CHAPTER 3	DEV6281 STARTING PROCEDURES IN MENU FORM	V-12
APPENDIX	List of Development Support Tool (Software Products) Names and Starting Command Formats	V-17

CHAPTER 1

OUTLINE OF THE E0C6281 DEVELOPMENT SUPPORT TOOL

1.1

Developmental Environment

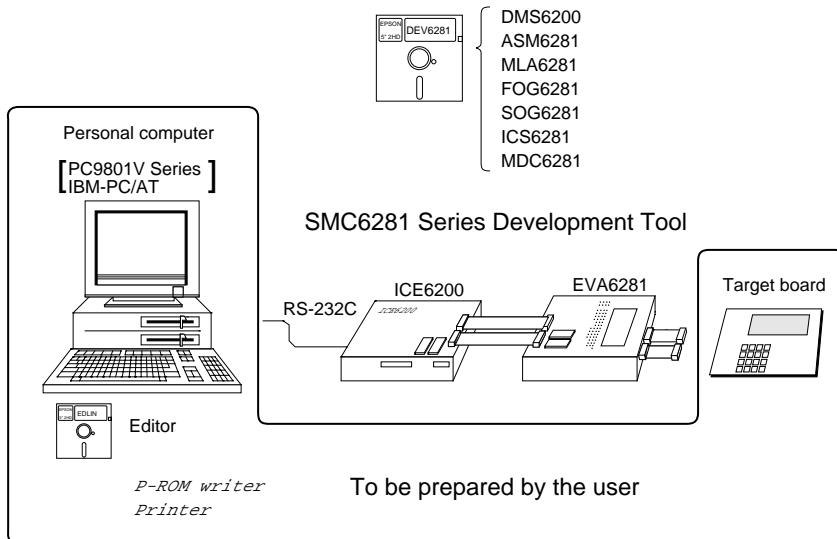
The software product of the E0C6281 development support tool (DEV6281) operates on the following host systems:

- 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 MDC6281 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 E0C6281 series, 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 Epson.



System Configuration

1.2

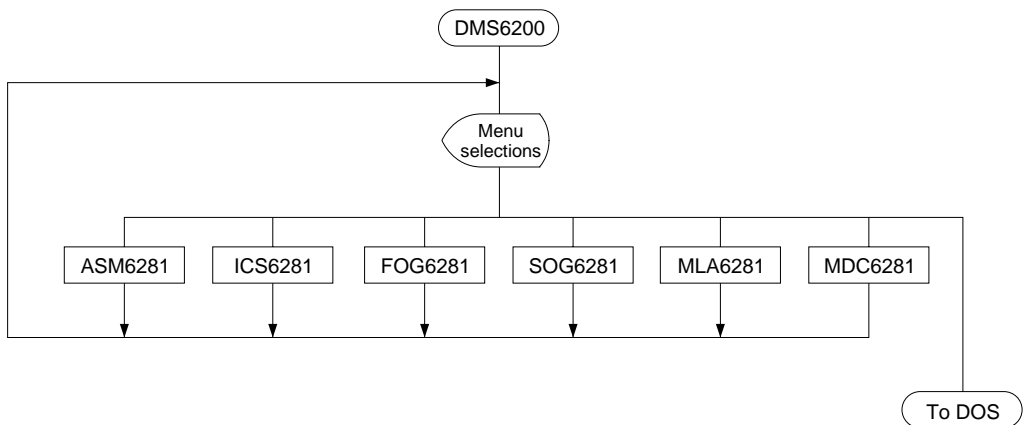
Development Tool Management System (DMS6200)

Outline:

This is a software which selects the DEV6281 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



1.3

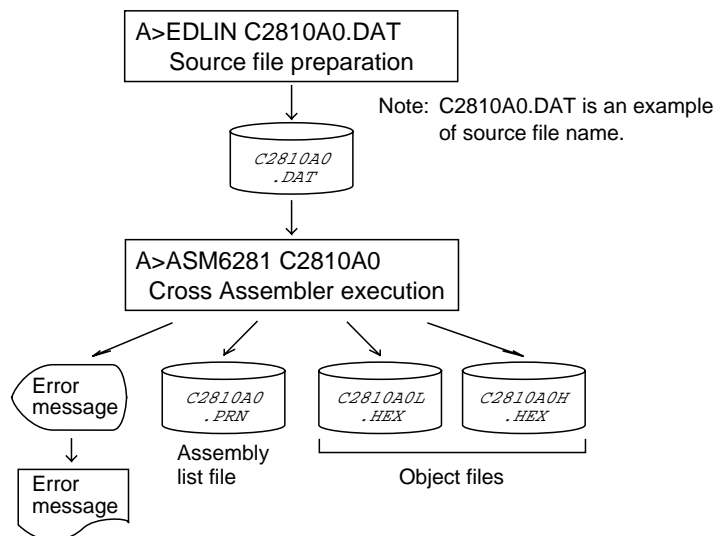
Cross Assembler (ASM6281)

Outline:

The Cross Assembler ASM6281 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 ASM6281
Execution Flow

1.4

Function Option Generator (FOG6281)

Outline:

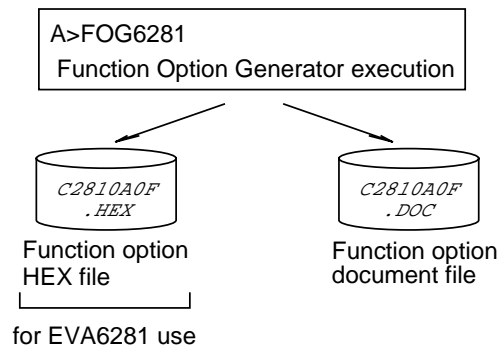
In the E0C6281 series, I/O port specifications may be selected with the hardware option and the mask pattern according to the setting is generated on the general-purpose computer.

The Function Option Generator FOG6281 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 EVA6281

Function Option Generator FOG6281 Execution Flow



1.5

Segment Option Generator (SOG6281)

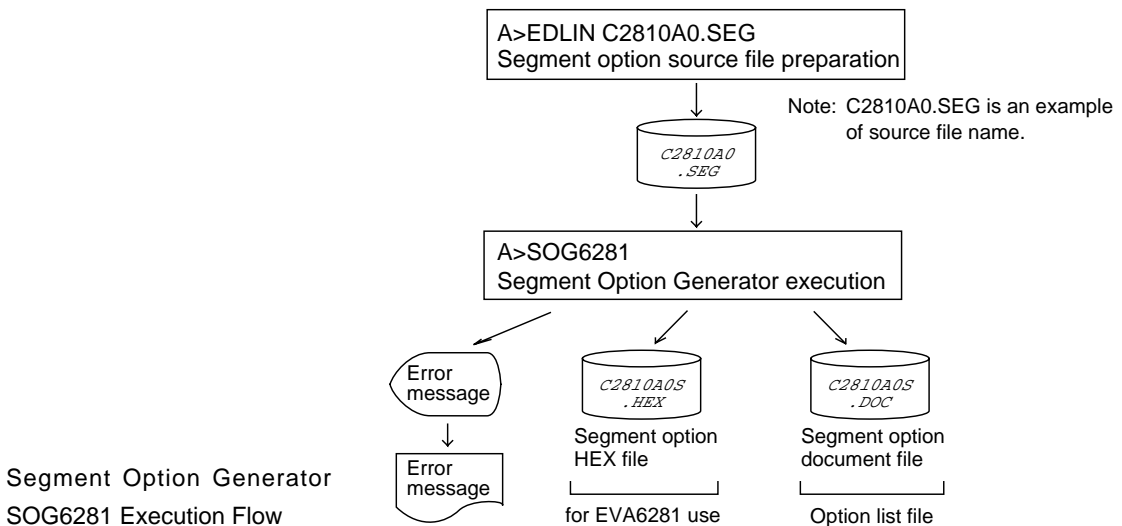
Outline:

In the E0C6281 series, the LCD segment structure may be selected with the hardware option and the mask pattern according to the setting is generated on the general-purpose computer.

The Segment Option Generator SOG6281 is a software that performs this hardware option selection on the personal computer and creates data files for mask pattern generation.

Features:

- Creates data in Intel-Hex format for the segment data ROM to be mounted on the EVA6281



1.6

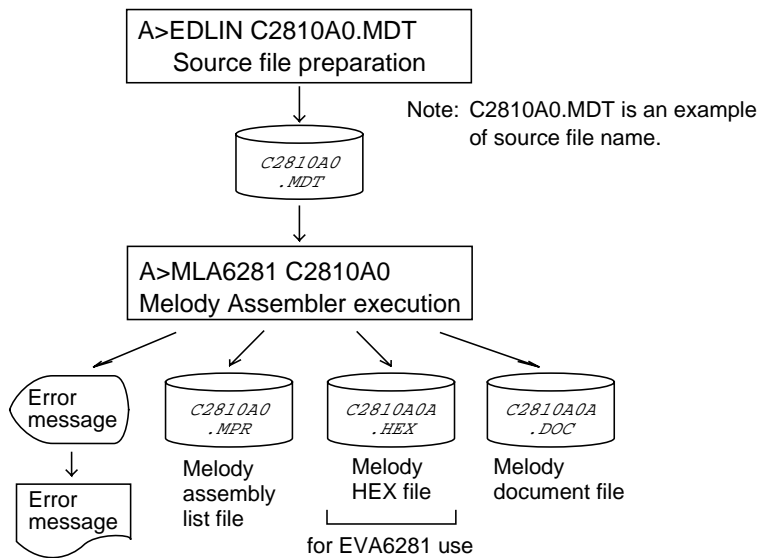
Melody Assembler (MLA6281)

Outline:

The Melody Assembler MLA6281 assembles the source file which has been input by the user's editor and outputs the object file in Intel-Hex format as well as the assembly list file and document file.

Features:

- Melody HEX file is generated which needs for melody emulation by assembling the scale data, note data and melody option contained in the source file
- Checks melody ROM and scale ROM capacity overflows



Melody Assembler MLA6281
Execution Flow

1.7

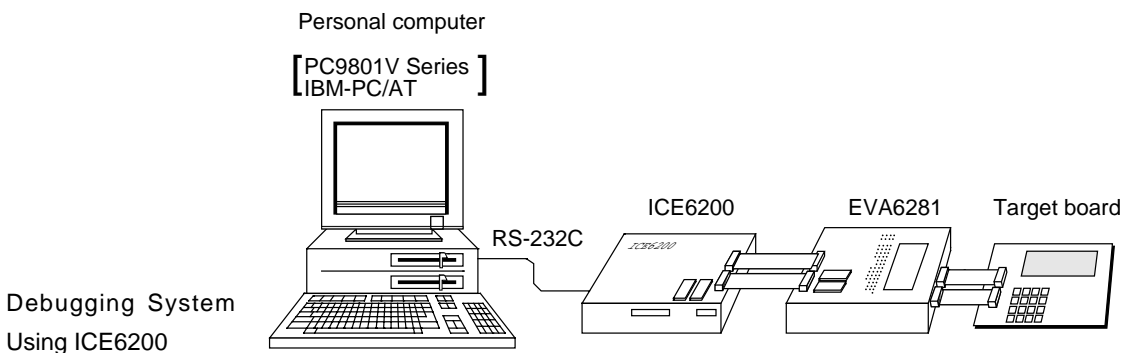
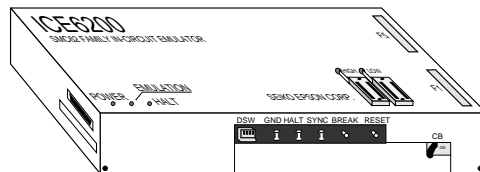
In-Circuit Emulator (ICE6200) & ICE Control Software (ICS6281)

Outline:

The In-circuit Emulator ICE6200 connects the target board produced by the user via the EVA6281 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 ICS6281.

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 EVA6281)
- Analysis of hardware is possible



1.8

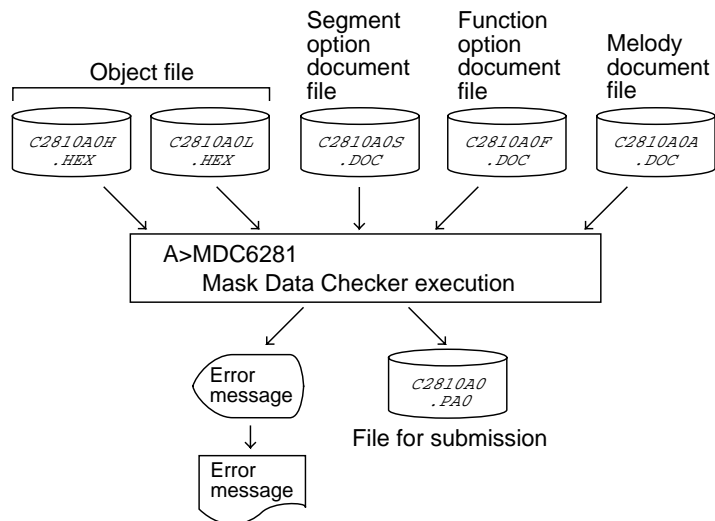
Mask Data Checker (MDC6281)

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/melody data)
- Performs packing and unpacking of program data and option data



Mask Data Checker MDC6281
Execution Flow

1.9

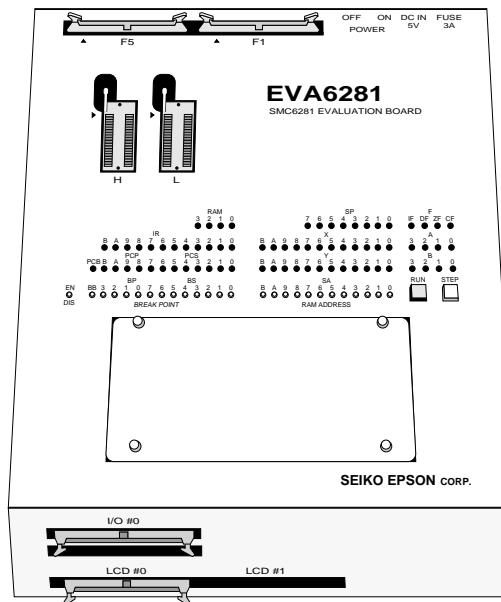
Evaluation Board (EVA6281)

Outline:

The Evaluation Board EVA6281 will implement almost the same functions as the actual CPU by creating ROM from the object files and option data files generated through ASM6281, MLA6281, FOG6281 and SOG6281 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
- EVA6281



1.10

Demonstration Tool (DMT6281)

Outline: This demonstration tool is intended for users who are currently planning applications using the E0C6281 series to better understand the E0C6281 as well as to evaluate its functions.

Features:

- Allows the E0C6281 user to evaluate the functions
- Allows the E0C6281 user to perform evaluation of electrical characteristics such as power current consumption and drive capabilities.

CHAPTER 2

CREATION OF DISK FOR DEV6281 EXECUTION

The DEV6281 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>

ASM6281.EXE Cross Assembler execution file

DMS6200.EXE Development Tool Management System
execution file

FOG6281.EXE Function Option Generator execution file

ICS6281B.BAT ICE Control Software batch file

ICS6281P.PAR ICE Control Software parameter file

ICS6281W.EXE ICE Control Software execution file

MDC6281.EXE Mask Data Checker execution file

MLA6281.EXE Melody Assembler execution file

SOG6281.EXE Segment Option Generator execution file

<MS-DOS Version>

ASM6281.EXE Cross Assembler execution file

DMS6200.EXE Development Tool Management System
execution file

FOG6281.EXE Function Option Generator execution file

ICS6281.BAT ICE Control Software batch file

ICS6281J.EXE ICE Control Software execution file

ICS6281P.PAR ICE Control Software parameter file

MDC6281.EXE Mask Data Checker execution file

MLA6281.EXE Melody Assembler execution file

SOG6281.EXE Segment Option Generator execution file

CHAPTER 3 DEV6281 STARTING PROCEDURES IN MENU FORM

DMS6200 (Development tool Management System) can start the DEV6281 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).

```
*** SMC6200 Development tool Management System. --- Ver 1.0 ***  
  
EEEEEEEEEE PPPPPPPP SSSSSSS OOOOOOOO NNN NNN  
EEEEEEEEEE PPPPPPPPPP SSS SSS 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 SSSSSS 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 DEV6281 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) ASM6281 .EXE
2) FOG6281 .EXE
3) ICS6281 .BAT
4) ICS6281J.EXE
5) MDC6281 .EXE
6) MLA6281 .EXE
7) SOG6281 .EXE

Input Number ? [  ]
```

<PC-DOS Version>

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

1) ASM6281 .EXE
2) FOG6281 .EXE
3) ICS6281B.BAT
4) ICS6281W.EXE
5) MDC6281 .EXE
6) MLA6281 .EXE
7) SOG6281 .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>

- ICS6281W.EXE, ICS6281J.EXE

To start ICS6281W.EXE or ICS6281J.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.

- ICS6281.BAT, ICS6281B.BAT

Since batch processing is programmed in the ICS6281.BAT and ICS6281B.BAT such that it will start with SPEED command or MODE command, ICS6281.BAT must be started after "PATH" of the disk containing SPEED command or MODE command and sub-directory has been specified.

Likewise, the ICS6281W.EXE requires the installation of RS-232C driver through the CONFIG.SYS.

At least 140K bytes are required for the RAM.

- MDC6281.EXE

Because the MDC6281.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 ASM6281 is selected.

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

1) C2810A0 .DAT
2) C2810A0 .MDT
3) C2810A0 .MPR
4) C2810A0 .PRN
5) C2810A0 .SEG
6) C2810A0A.DOC
7) C2810A0A.HEX
8) C2810A0F.DOC
9) C2810A0F.HEX
10) C2810A0H.HEX
11) C2810A0L.HEX
12) C2810A0S.DOC
13) C2810A0S.HEX
14) C62810A0.PA0

Input Number ? [ 1 ]

Edit > [ASM6281 C2810A0 ]
```

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.

- ASM6281 will start.
- MLA6281 will also start with the same operations.

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.

FOG6281
ICS6281
ICS6281W
ICS6281J
MDC6281
SOG6281

(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.

VI. SMC6281 ICE OPERATION MANUAL

PREFACE

This manual explains the function of ICE6200, a software development support system for the E0C6281 4-bit Single Chip Microcomputer, and the operation of ICS6281, its ICE control software.

Chapter 2 and subsequent chapters provide information common to all SMC62 Family models, the model name being denoted "XX". Read this manual, replacing "XX" with "81".

62XX → 6281

For details on the SMC6281, refer to the "SMC6281 Technical Hardware Manual" and "SMC6281 Technical Software Manual". For such items as development procedure, refer to the "SMC62 Family Technical Guide".

CONTENTS

1	ICS6281 RESTRICTIONS AND ADDITIONS	VI-1
1.1	ROM Area	VI-1
1.2	RAM Area	VI-1
1.3	Undefined Code	VI-1
1.4	OPTLD Command	VI-1
2	ICE6200 SPECIFICATIONS	VI-2
2.1	Features	VI-2
2.1.1	Description	VI-3
2.1.2	Software Configuration	VI-3
2.1.3	Function Table	VI-4
2.1.4	Function-differentiated Command List	VI-5
2.1.5	Alphabetical Listing of Commands	VI-7
2.2	Connecting and Starting the System	VI-9
2.2.1	HOST Settings	VI-10
2.2.2	Starting the ICS62XX	VI-12
2.3	ICE6200 Operation and Functions	VI-13
2.3.1	Operating Features	VI-13
2.3.2	Break Mode and Break Function	VI-15
2.3.3	SYNC Pin and HALT Pin Output	VI-17
2.3.4	Display During Run Mode and During Break	VI-18
2.3.5	Break Assigning Commands	VI-20
2.3.6	Target Interrupt and Break	VI-21
2.3.7	History Function	VI-22
2.3.8	Break Delay Function	VI-24
2.3.9	Coverage Function	VI-24
2.3.10	Measurement During Command Execution	VI-25
2.3.11	Self-diagnostic Function	VI-26
2.3.12	Starting the Printer	VI-27
2.3.13	Limitations During Emulation	VI-28
2.4	Command Details	VI-30
2.4.1	Display Command Group	VI-31
	- L Command	VI-32
	- DP Command	VI-34
	- DD Command	VI-36
	- DR Command	VI-38
	- H Command	VI-39
	- HB and HG Commands	VI-42
	- HS, HSR, and HSW Commands	VI-44
	- HP and HPS Commands	VI-45
	- CHK Command	VI-46
	- DXY Command	VI-47
	- CVD and CVR Commands	VI-48

2.4.2	Set Command Group	VI-49
	- A Command	VI-50
	- FP Command	VI-52
	- FD Command	VI-53
	- MP Command	VI-54
	- MD Command	VI-55
	- SP Command	VI-56
	- SD Command	VI-57
	- SR Command	VI-58
	- SXY Command	VI-59
	- HC Command	VI-60
	- HA, HAD, and HAR Commands	VI-61
2.4.3	Break and Go Command Group	VI-63
	- BA and BAR Commands	VI-64
	- BD and BDR Commands	VI-65
	- BR and BRR Commands	VI-66
	- BM and BMR Commands	VI-68
	- BC Command	VI-70
	- BRES Command	VI-71
	- G Command	VI-72
	- T Command	VI-75
	- U Command	VI-77
	- BE and BSYN Commands	VI-78
	- BT Command	VI-79
	- BRKSEL Command	VI-80
2.4.4	File Command Group	VI-81
	- RF and RFD Commands	VI-82
	- VF and VFD Commands	VI-83
	- WF and WFD Commands	VI-84
	- CL and CS Commands	VI-85
	- OPTLD Command	VI-86
2.4.5	ROM Command Group	VI-87
	- RP Command	VI-88
	- VP Command	VI-89
	- ROM Command	VI-90
2.4.6	Control Command Group	VI-91
	- I Command	VI-92
	- TIM Command	VI-93
	- OTF Command	VI-94
	- Q Command	VI-95
2.4.7	HELP Command	VI-97
2.5	Error Message Summary	VI-102
2.6	FD File Configuration	VI-103
2.7	Appendix HEX File Format	VI-104

1 ICS6281 RESTRICTIONS AND ADDITIONS

1.1 ROM Area

The ROM area is limited to a maximum address of 3FFH.
Assigning data above the 3FFH address causes an error.

1.2 RAM Area

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

060H to 08FH
0B0H to 0DFH

Memory 090H to 0AFH is display memory; 0E0H to 0FFH is I/O memory.
(Refer to the "E0C6281 Technical Hardware Manual" for details.)

1.3 Undefined Code

The instructions below are not specified for the E0C6281 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

1.4 OPTLD Command

This command is used to load melody HEX files (MLA6281 melody assembler output files) in the EVA6281 melody data memory with the ICE6200. It is also necessary in order to play a melody using the ICE6200 and EVA6281. For an explanation of the command, see page 86, reference items are described on pages 6, 8, 99, 100, and 103.

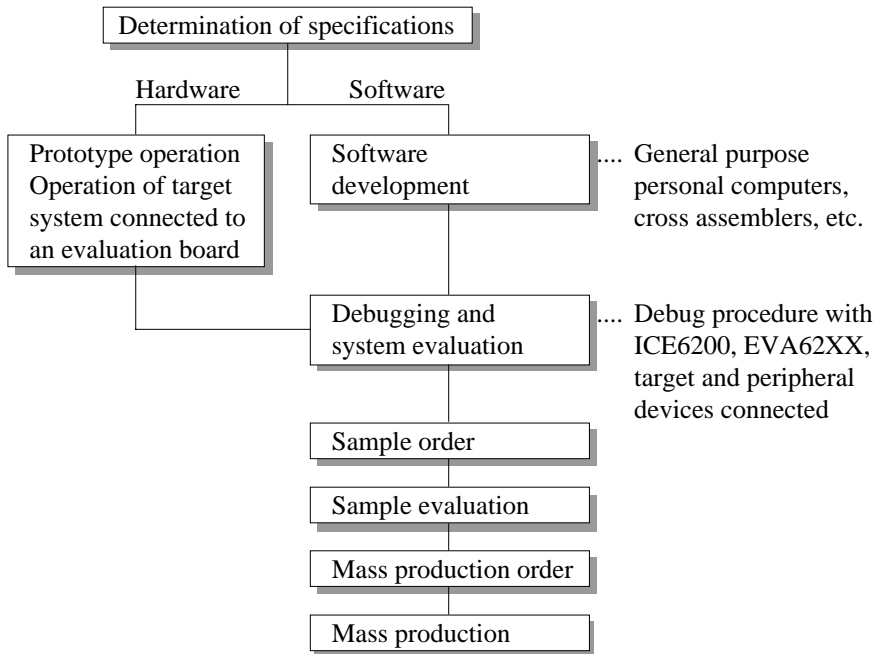
2 ICE6200 SPECIFICATIONS

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.



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

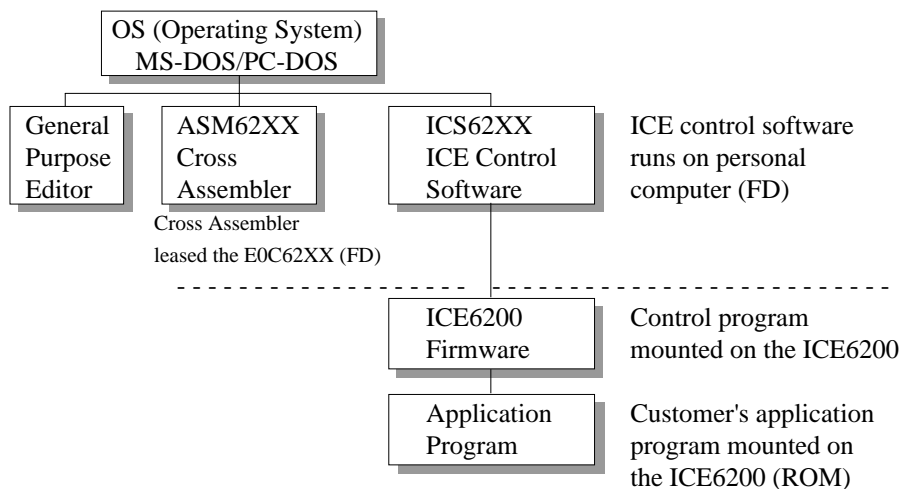
Refer to "E0C62 Family Technical Guide" to get more detailed information about "Sample order" to "Mass production" above mentioned flow chart.

2.1.1 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 V_{DD} 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.

2.1.2 Software Configuration



2.1.3 Function Table

Table 2.1.3 shows the functions supported by the ICE6200.

Table 2.1.3 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	

2.1.4 Function-differentiated Command List

Table 2.1.4 shows the function-differentiated command list for the ICE6200.

Table 2.1.4 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"	50
2	Disassemble	#L,a1,a2 ↵	Contents of addresses a1 to a2 are disassembled and displayed	32
3	Dump	#DP,a1,a2 ↵	Contents of program area a1 to a2 are displayed	34
		#DD,a1,a2 ↵	Content of data area a1 to a2 are displayed	36
4	Fill	#FP,a1,a2,d ↵	D is set in addresses a1 to a2 (program area)	52
		#FD,a1,a2,d ↵	D is set in addresses a1 to a2 (data area)	53
5	Set Run Mode	#G,a ↵	Program is executed from the "a" address	72
		#TIM ↵	Execution time and step counter selection	93
		#OTF ↵	On-the-fly display selection	94
6	Trace	#T,a,n ↵	Executes program while displaying results of step instruction from "a" address	75
		#U,a,n ↵	Displays only the final step of #T,a,n	77
7	Break	#BA,a ↵	Sets Break at program address "a"	64
		#BAR,a ↵	Cancels breakpoint	
		#BD ↵	Break condition is set for data RAM	65
		#BDR ↵	Breakpoint is canceled	
		#BR ↵	Break condition is set for EVA62XXCPU internal registers	66
		#BRR ↵	Breakpoint is canceled	
		#BM ↵	Combined break conditions set for program data RAM address and registers	68
		#BMR ↵	Cancel combined break conditions for program data ROM address and registers	
		#BRES ↵	All break conditions canceled	71
		#BC ↵	Break condition displayed	70
		#BE ↵	Enter break enable mode	78
#BSYN ↵	Enter break disable mode	78		
#BT ↵	Set break stop/trace modes	79		
#BRKSEL,REM ↵	Set BA condition clear/remain modes	80		
8	Move	#MP,a1,a2,a3 ↵	Contents of program area addresses a1 to a2 are moved to addresses a3 and after	54
		#MD,a1,a2,a3 ↵	Contents of data area addresses a1 to a2 are moved to addresses a3 and after	55
9	Data set	#SP,a ↵	Data from program area address "a" are written to memory	56
		#SD,a ↵	Data from data area address "a" are written to memory	57

Item number	Function	Command configuration	Description of operation	Reference page
10	Change CPU Internal Registers	#DR ↵	Display EVA62XXCPU internal registers	38
		#SR ↵	Set EVA62XXCPU internal registers	58
		#I ↵	Reset EVA62XXCPU	92
		#DXY ↵	Display X, Y, MX and MY	47
		#SXY ↵	Set data for X and Y display and MX, MY	59
11	History	#H,p1,p2 ↵	Display history data for pointer 1 and pointer 2	39
		#HB ↵	Display upstream history data	42
		#HG ↵	Display 21 line history data	42
		#HP ↵	Display history pointer	45
		#HPS ↵	Set history pointer	45
		#HC,S/C/E ↵	Sets up the history information acquisition before (S), before/after (C) and after (E)	60
		#HA,a1,a2 ↵	Sets up the history information acquisition from program area a1 to a2	61
		#HAR,a1,a2 ↵	Sets up the prohibition of the history information acquisition from program area a1 to a2	61
		#HAD ↵	Indicates history acquisition program area	61
		#HS,a ↵	Retrieves and indicates the history information which executed a program address "a"	44
		#HSW,a ↵	Retrieves and indicates the history information which wrote or read the data area address "a"	44
		#HSR,a ↵	Retrieves and indicates the history information which wrote or read the data area address "a"	44
12	File	#RF,file ↵	Move program file to memory	82
		#RFD,file ↵	Move data file to memory	82
		#VF,file ↵	Compare program file and contents of memory	83
		#VFD,file ↵	Compare data file and contents of memory	83
		#WF,file ↵	Save contents of memory to program file	84
		#WFD,file ↵	Save contents of memory to data file	84
		#CL,file ↵	Load ICE6200 set condition from file	85
		#CS,file ↵	Save ICE6200 set condition to file	85
		#OPTLD,n,file ↵	Load HEXA data from file	86
13	Coverage	#CVD ↵	Indicates coverage information	48
		#CVR ↵	Clears coverage information	48
14	ROM Access	#RP ↵	Move contents of ROM to program memory	88
		#VP ↵	Compare contents of ROM with contents of program memory	89
		#ROM ↵	Set ROM type	90
15	Terminate ICE	#Q ↵	Terminate ICE and return to operating system control	95
16	Command Display	#HELP ↵	Display ICE6200 instruction	98
17	Self Diagnosis	#CHK ↵	Report results of ICE6200 self diagnostic test	46

2.1.5 Alphabetical Listing of Commands

Table 2.1.5 shows an alphabetical listing of ICE6200 commands.

Table 2.1.5 Alphabetical Listing of Commands

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

Item number	Command configuration	Description of operation	Reference page
36	#HP ↵	Display history pointer	45
37	#HPS ↵	Set history pointer	45
38	#HS,a ↵	Retrieves and indicates the history information which executed the program address "a"	44
39	#HSR,a ↵	Retrieves and indicates the history information which read the data area address "a"	44
40	#HSW,a ↵	Retrieves and indicates the history information which wrote the data area address "a"	44
41	#I ↵	Reset EVA62XXCPU	92
42	#L,a1,a2 ↵	Display disassembled contents of addresses a1 to a2	32
43	#MD,a1,a2,a3 ↵	Move contents of data area addresses a1 to a2 to address a3 and after	55
44	#MP,a1,a2,a3 ↵	Move contents of program area addresses a1 to a2 to address a3 and after	54
45	#OTF ↵	Select on-the-fly display	94
46	#OPTLD,n,file ↵	Load HEXA data from file	86
47	#Q ↵	Terminate ICE and return to operating system control	95
48	#RF,file ↵	Move program file to memory	82
49	#RFD,file ↵	Move data file to memory	82
50	#ROM ↵	Select ROM type	90
51	#RP ↵	Move ROM contents to program memory	88
52	#SD,a ↵	Write data from address "a" of the data area	57
53	#SP,a ↵	Write data from address "a" of the program area	56
54	#SR ↵	Set EVA62XXCPU internal registers	58
55	#SXY ↵	display X, Y and set data to MX, MY	59
56	#T,a,n ↵	Execute while displaying n step instruction results from address "a"	75
57	#TIM ↵	Select execution time and step counter	93
58	#U,a,n ↵	Display only final step of #T,a,n	77
59	#VF,file ↵	Compare program file and memory contents	83
60	#VFD,file ↵	Compare data file and memory contents	83
61	#VP ↵	Compare contents of ROM and contents of program memory	89
62	#WF,file ↵	Save content of memory to the program file	84
63	#WFD,file ↵	Save content of memory to the data file	84

2.2 Connecting and Starting the System

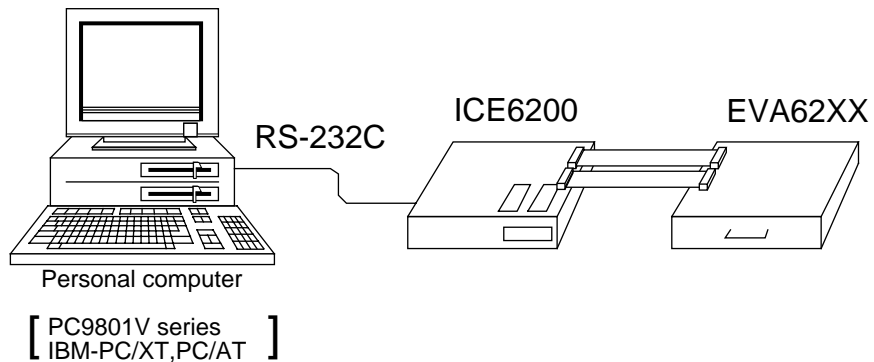


Fig. 2.2 System Connection Diagram

The ICE6200 connects to common personal computers and the E0C62 Family evaluation board EVA62XX for operation, as shown in Fig. 2.2. 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.

2.2.1 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

Enter settings (1) or (2) below. Item (2) is convenient since it has backup capability even after switching power OFF.

(1) Execute SPEED command soon after starting MS-DOS.

Setting:

```
A>SPEED RO 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)

(2) SWITCH command operates with the same settings as (1), but the settings become effective after the next boot.

Setting:

```
A>SWITCH RO[9600 B8 PN S1 NONE]↵
```

Verify settings:

```
A>SWITCH↵
```

```
SWITCH Version ??
```

Escape the command with:

```
RS232C-0:9600 BITS-8 PARITY-NONE STOP-1 NONE
```

```
:
```

```
:
```

– ↵ to escape the command

* 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 baudrate to 4800.

2.2.2 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 section "2.2.1".

– 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.

2.3.1 Operating Features

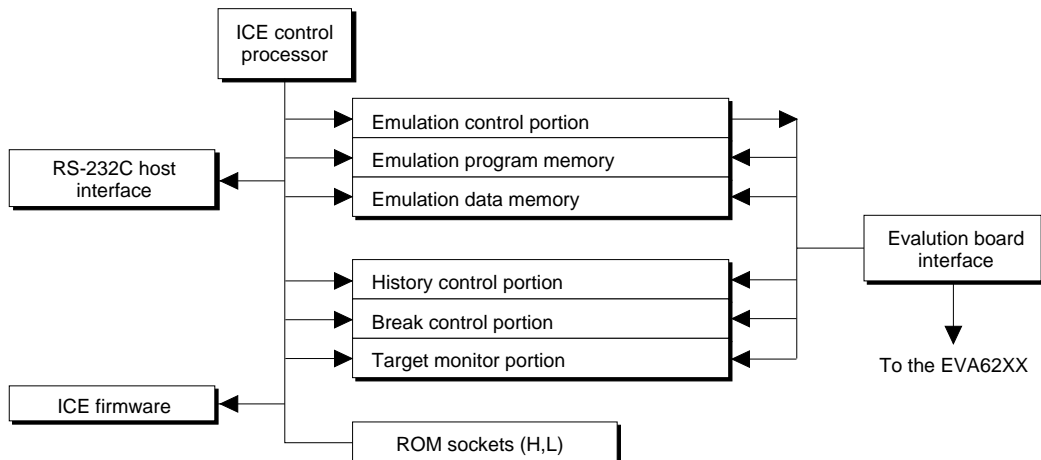


Fig. 2.3.1 Block Diagram of ICE6200 Functions

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 \times 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.

2.3.2 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 2.3.2 shows the break modes and break types.

Table 2.3.2 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 EVA62XX CPU 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.

2.3.3 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.

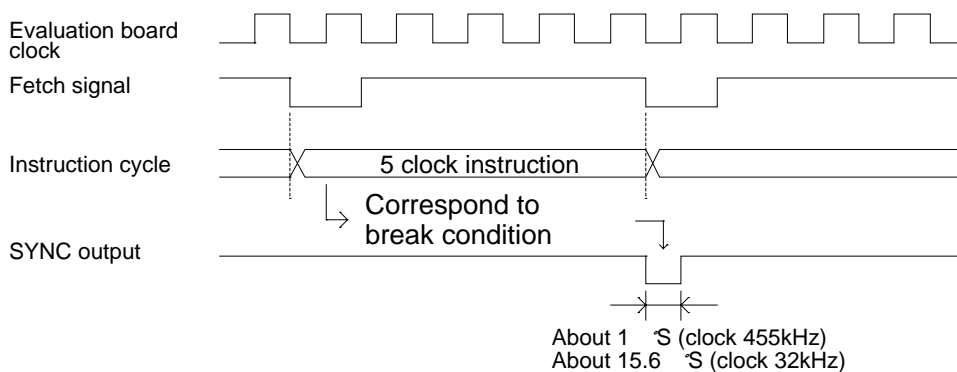


Fig. 2.3.3.a SYNC Pin Output

(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).

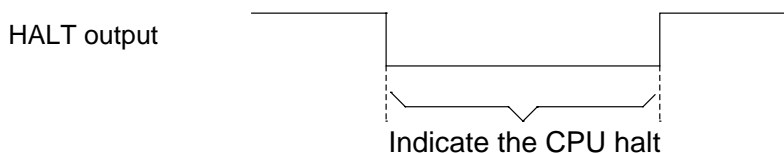


Fig. 2.3.3.b HALT Pin Output

2.3.4 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 section "2.3.10".

2.3.5 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.

2.3.6 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.

2.3.7 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.

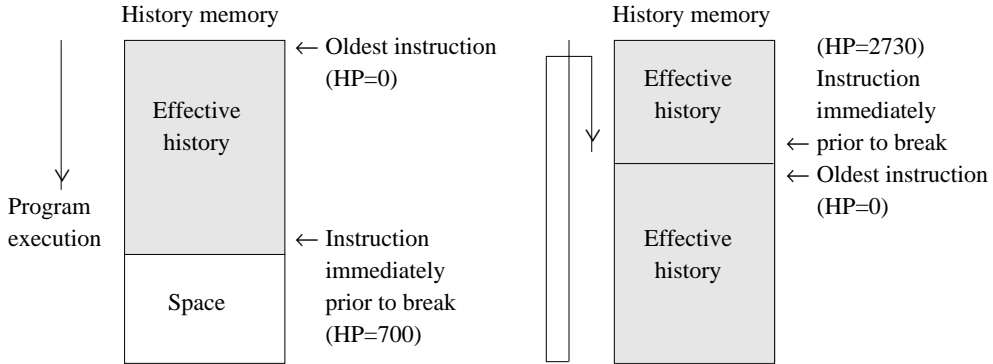
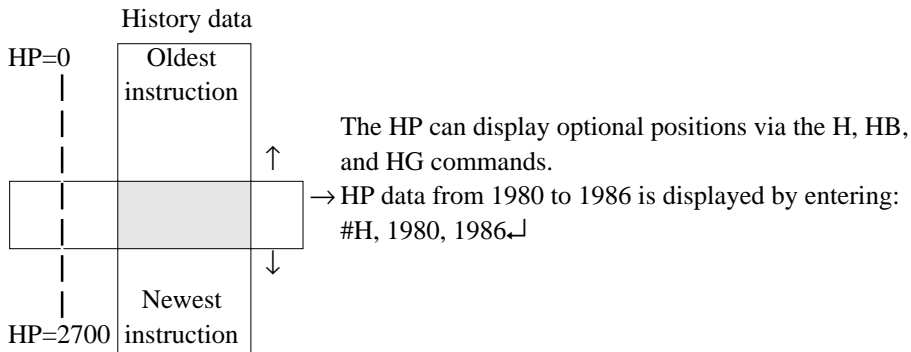


Fig. 2.3.7 History Function Diagram

Figure 2.3.7 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.



```

#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 OFF OFF 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.*

2.3.8 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 Section 2.3.2.), be sure that break is not made at the specified break condition.

2.3.9 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.

2.3.10 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

6.5 μ S to 6.5 \times 65535 μ S (=425.977mS)

Measurement error : \pm 6.5 μ S

(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.
```

2.3.11 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 power supply is OFF or 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.

2.3.12 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
```

2.3.13 Limitations During Emulation

When running emulations with the ICE6200 and evaluation board connected, the EVA62XXCPU is normally stopped, as described in section 2.3.1 (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.

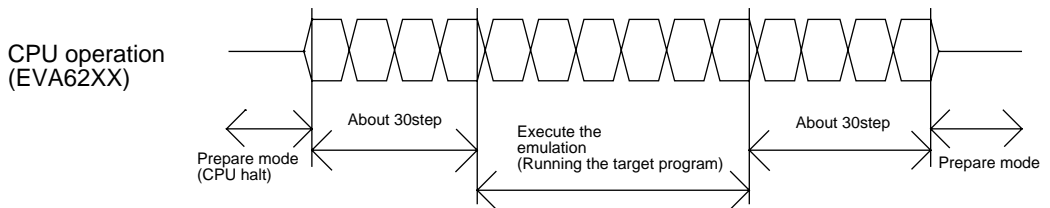


Fig. 2.3.13 EVA62XXCPU operation

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 $6\text{ms} + 6\text{ms} = 12\text{ms}$. While at a clock speed of 455kHz, the command program steps before and after emulation require $400\mu\text{s} + 400\mu\text{s} = 800\mu\text{s}$.

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

2.4 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 2.4.1 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 imbedded 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 indicate points for caution when using the described commands.

2.4.1 Display Command Group

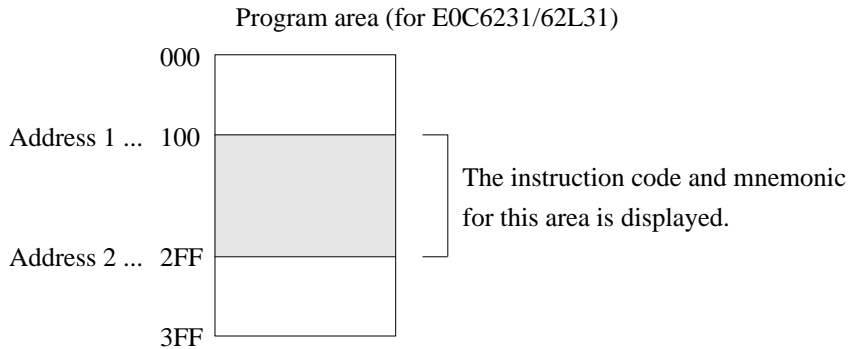
L	DISASSEMBLE LIST	VI-32
DP	DUMP PROGRAM	VI-34
DD	DUMP DATA RAM	VI-36
DR	DISPLAY CPU REGISTER	VI-38
H	HISTORY DATA DISPLAY	VI-39
HB	HISTORY DATA DISPLAY BACKWARD	VI-42
HG	HISTORY DATA DISPLAY FORWARD	VI-42
HS	HISTORY SEARCH PC	VI-44
HSR	HISTORY SEARCH MEMORY READ	VI-44
HSW	HISTORY SEARCH MEMORY WRITE	VI-44
HP	HISTORY POINTER DISPLAY	VI-45
HPS	HISTORY POINTER SET	VI-45
CHK	CHECK ICE6200 HARDWARE	VI-46
DXY	DISPLAY X, Y REGISTER and MX, MY COUNT	VI-47
CVD	DISPLAY COVERAGE	VI-48
CVR	RESET COVERAGE	VI-48

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.



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

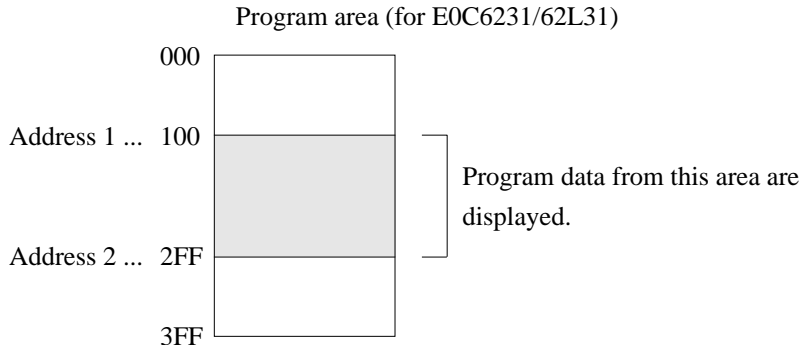
Format #DP,<address 1>,<address 2>↵
#DP,<address 1>↵
#DP↵

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.



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 *

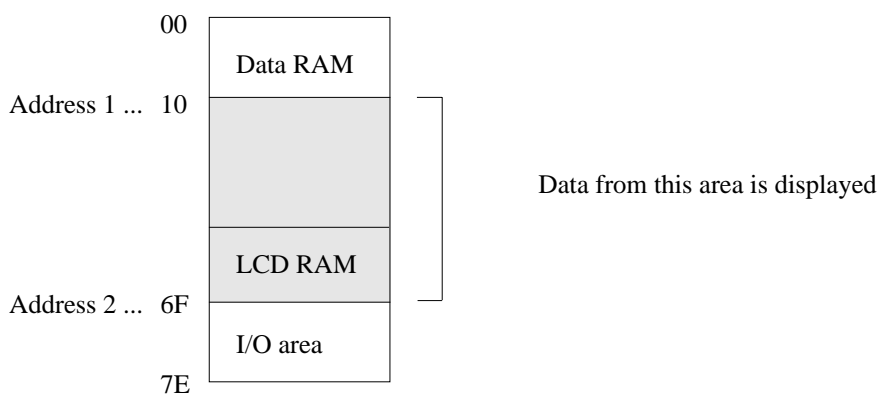
```
#DP,400,FFF↵          . . . . . Error due to exceeding maximum value of program
   * COMMAND ERROR *          area (3FF address in the case of E0C6231/62L31)
```

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)

DUMP DATA RAM**DD****Format**

#DD,<address 1>,<address 2>↵

#DD,<address 1>↵

#DD↵

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)

```

#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
0F10 F 0 1 0 F F / / / / / / / / / displayed 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.

HISTORY DATA DISPLAY**H****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. On 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.

#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

HISTORY DATA DISPLAY**H**

Format #H,<pointer 1>,<pointer 2>↵
 #H,<pointer 1>↵

Example #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

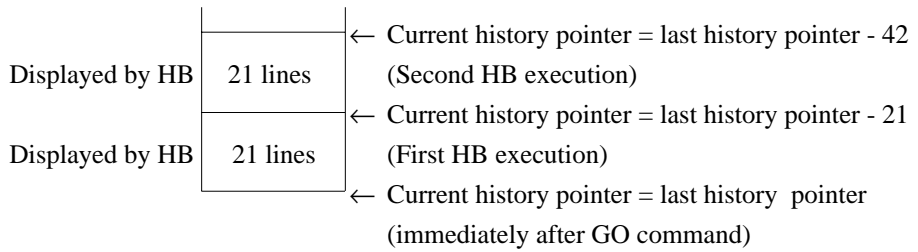
Format #HB↵
 #HG↵

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.



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.

HISTORY DATA DISPLAY BACKWARD/FORWARD**HB, HG****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

#

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
      :
      :
```

HISTORY POINTER DISPLAY/SET**HP, HPS****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
                       exceeded

#HP↵
* 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 *
 * RAM CHECK ERROR 001111 55=>FF * ] Message is displayed when an
                                     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
 #

2.4.2 Set Command Group

A	ASSEMBLE PROGRAM	VI-50
FP	FILL PROGRAM	VI-52
FD	FILL DATA RAM	VI-53
MP	MOVE PROGRAM	VI-54
MD	MOVE DATA RAM	VI-55
SP	SET PROGRAM	VI-56
SD	SET DATA RAM	VI-57
SR	SET REGISTER	VI-58
SXY	SET MX, MY DATA	VI-59
HC	SET HISTORY CONDITION	VI-60
HA	SET HISTORY RANGE	VI-61
HAD	DISPLAY HISTORY RANGE	VI-61
HAR	RESET HISTORY RANGE	VI-61

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.

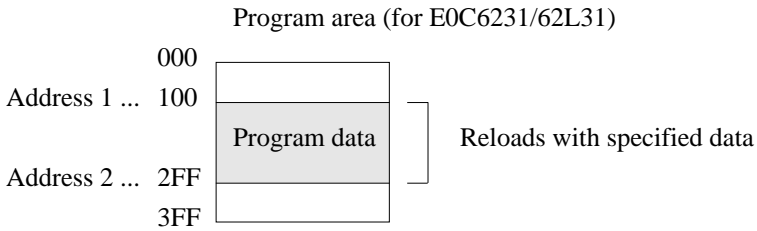
ASSEMBLE PROGRAM**A**

Format	#A,<address>	(With guidance)
Examples	<pre>#A,100 0100 LD A,0F 0101 / #A,200 0200 PUSH XP * ERROR * 0200 NOP5 0201 JJJ OFF * ERROR * 0201 LD A,FF * ERROR * 0201 LD A,0F 0202 / #A,202 0202 ^ 0201 / #</pre>	<pre>. Instruction entered by key input Address displayed; mnemonic input awaited (mnemonic instruction, operand input) / input cancels instruction Error generated by unapproved mnemonic input (for E0C62XX/62*XX); same address is redisplayed with mnemonic request Error generated when valid operand range is exceeded Return to previous address (current address less one) via ^ key input</pre>
Note	"ESC" key nonfunctional; cancel operation by entering /.	

FP *FILL PROGRAM*

Format #FP,<address 1>,<address 2>,<program data>↓

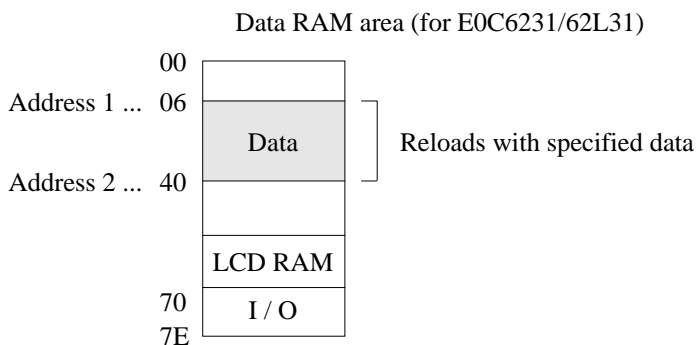
Function The contents of <address 1> and <address 2> of the program area (ICE emulation memory) are stacked in the program data area.



- 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↓ When undefined code is detected, an error message is displayed and the instruction will not execute
 - * COMMAND ERROR *
 - #FP,200,100,FFF↓ Address 1 > address 2 error
 - * COMMAND 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.



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 (address 7E for E0C6231/62L31) and the instruction will not execute
* COMMAND ERROR *		
#FD,70,60,0↵	Address 1 > address 2 error
* COMMAND 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

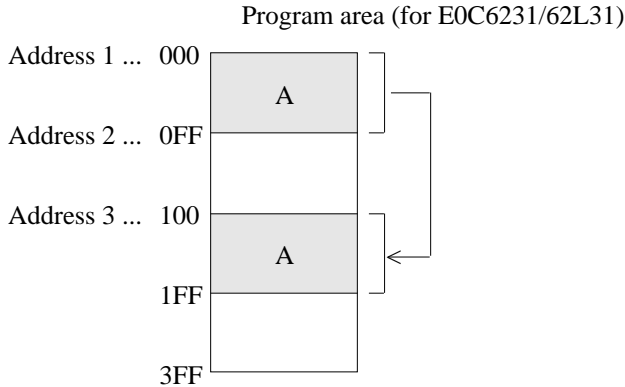
Notes

- (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>↵

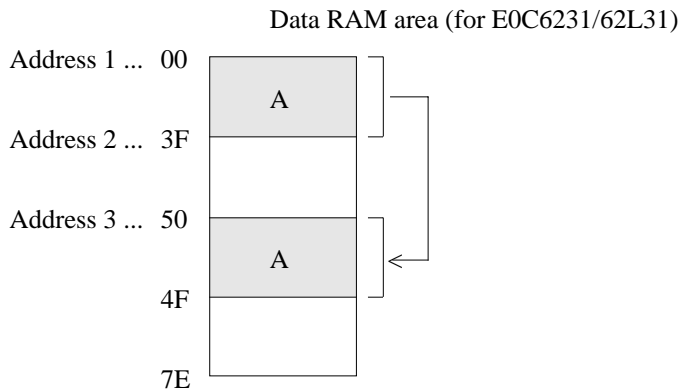
Function Contents of program area addresses 1 to 2 are transferred to addresses 3 and above.



- 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 message is displayed and the instruction will not execute
 - * COMMAND ERROR *
 - #MP,200,100,300↵ Address 1 > address 2 error
 - * COMMAND 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.



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:F05↵
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	<p>Contents of the data RAM are addresses are displayed or modified.</p> <p>(1) Data cannot be written to the read only area.</p> <p>(2) Data in the write only area cannot be read.</p>	
Examples	<pre>#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:- Hyphen only displayed due to read only address; 71 F:- data input not accepted 72 5:- 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). #</pre>	

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.

SET MX, MY DATA**SXY**

Format	#SXY <u>↵</u>	(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	<pre>#SXY↵ X=040 MX=5:↵ Y=030 MY=A:↵ #SXY↵ X=040 MX=5:0↵ Y=030 MY=A:F↵ #SXY↵ X=070 MX=3:- Y=FFF MY=-:OV #SXY↵ X=E52 MX * UNUSED AREA * Y=1A7 MY=1:3↵ #</pre>	<pre>. Display only; ↵ alone continues operation Sets new data to MX, MY Data to read only area not accepted Input not accepted if RAM area is exceeded An unused area error message is displayed for E52 (for E0C6246)</pre>

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	<pre>#HA,<address 1,<address 2>/ALL↵ #HAD↵ #HAR,<address 1,<address 2>/ALL↵</pre>
Function	<p>Sets up, indicates and clears PC address within the history extraction area.</p> <p>(1) HA: Extract the range specified by <address>. When specifying ALL, all addresses will be specified.</p> <p>(2) HAD: Indicates the address of history extraction area.</p> <p>(3) HAR: Do not extract the range specified by <address>. When specifying ALL, history isn't extracted.</p>
Examples	<pre>#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 #</pre>

2.4.3 Break and Go Command Group

BA	SET BREAK ADDRESS CONDITION	VI-64
BAR	RESET BREAK ADDRESS CONDITION	VI-64
BD	SET BREAK DATA CONDITION	VI-65
BDR	RESET BREAK DATA CONDITION	VI-65
BR	SET BREAK REGISTER CONDITION	VI-66
BRR	RESET BREAK REGISTER CONDITION	VI-66
BM	SET BREAK MULTIPLE CONDITION	VI-68
BMR	RESET BREAK MULTIPLE CONDITION	VI-68
BC	BREAK CONDITION DISPLAY	VI-70
BRES	RESET ALL BREAK CONDITION	VI-71
G	GO TARGET PROGRAM	VI-72
T	SINGLE STEP TRACE	VI-75
U	SINGLE STEP TRACE & LAST INFORMATION DISPLAY	VI-77
BE	BREAK ENABLE MODE SET	VI-78
BSYN	BREAK DISABLE & SYNC MODE SET	VI-78
BT	BREAK TRACE MODE SET	VI-79
BRKSEL	BREAK ADDRESS MODE SELECT	VI-80

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 $\text{<address 3>} \leq \text{PC} \leq \text{<address 4>}$.
- (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
:
#
```

SET/RESET BREAK DATA CONDITION BD, BDR

Format	<pre>#BD↵ #BDR↵</pre>	(With guidance)
Function	<p>Break condition set for data RAM read/write area.</p> <p>(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.</p> <p>(2) BDR: Cancels the condition set by BD command.</p> <p>(3) A break condition set by the BD command is functional at one point only, but can be mixed with BA and BR commands.</p> <p>(4) A BD condition can be canceled by executing the BM command.</p>	
Examples	<pre>#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 (*)</pre> <p>In the above example, a break is set for when the number 5 is written to or read from the data RAM address 074.</p> <pre>#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</pre> <p>At the current settings, a break is generated when 1 is written to 2³ bit and 2¹ bit at data RAM address 74.</p> <pre>#BDR↵ All BD conditions are cleared #BD↵ ADDR ---:↵ Entering ↵ after canceling BD setting confirms # cancellation</pre>	

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	<pre>#BR␣ #BRR␣</pre>	(With guidance)
Examples	<pre>#BRR␣ #BR␣ A -:␣ #BR␣ A -:0␣ B -:0␣ FI -:*␣ FD -:*␣ FZ -:*␣ FC -:*␣ X ---:40␣ Y ---:30␣</pre> <p>A break condition is set wherein A=0, B=0, X=40, and Y=30.</p> <pre>#BR␣ A 0:␣ B 0:5␣ FI *:/␣ #</pre> <p>A break condition is set where A=0, B=5, X=40, and Y=30.</p>	<p>..... A BR condition is cleared by the BRR command</p> <p>..... Entering ␣ after canceling BR setting confirms cancellation</p> <p>..... Entering / when no further setting changes are desired completes the instruction</p>
Notes	<p>(1) The target system operates in real time even when a GO command is executed after setting a BR condition.</p> <p>(2) Each model (E0C62XX/62*XX) has a different RAM area, and XY settings in a BR command can be set to FFF.</p>	

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←      . . . . . A hyphen (-) is displayed when a BM condition is
ADDR  ---: 70←          canceled.
DATA  -: A←            Break condition is set where PC=100, RAM access=70,
R/W   -: *←            RAM data=A, D and C flags=1, and Y register=3E.
A     -: *←            During execution of the instructions at address 100, a
B     -: *←            break occurs when the following conditions coincide:
          FI      -: *←          RAM at address 70 is accessed, read/write data A,
FD and
FD    -: 1←            FC are set, and Y register is 3E. (Valid for break during
FZ    -: *←            program loop.)
FC    -: 1←
X     ---: *←      . . . . . The point at which the break is placed is masked by an
Y     ---: 3E←      asterisk (*) mark.
```

SET/RESET BREAK MULTIPLE CONDITION **BM, BMR**

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 conditions are canceled.

- * BA NONE
- * 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 confirmed

- * BA 0100..0101
- * BD NONE
- * BR NONE
- * BM NONE
- * BREAK ENABLE MODE
- * BREAK STOP MODE
- * TIME COUNT MODE

#BRES↵

#BA,100,102↵

#BC↵ Displays multiple executions of BA condition when addresses are not consecutive

- * BA 0100
- * BA 0102
- :
- :

#

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 section 2.3.2 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 section 2.3.4 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 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 *

#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)

These settings
are set at power
on; default is
command input

- (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

SINGLE STEP TRACE**T****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.

Example

(6) In (1) above, the instruction is terminated by "ESC" key input.

```
#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
is displayed.

Command code
and mnemonic
are displayed.

Correctors displayed when the flag is set
and/or reset (After executing three steps,
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=FD F 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

Notes

- (1) The T command does not operate in real time. Therefore, the target timer is renewed.(For details refer to section 2.3.13.)
- (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 section 2.3.13.)
- (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

Details of break enable/disable functions can be found in section 2.3.2 Break Mode.

BREAK TRACE MODE SET**BT**

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	<pre>#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 #</pre>	
Note	Refer to section 2.3.2 for details on 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 cancel the previously set BA condition. The cancel 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 cancel 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
#
```

2.4.4 File Command Group

RF	READ PROGRAM FILE	VI-82
RFD	READ DATA FILE	VI-82
VF	VERIFY PROGRAM FILE	VI-83
VFD	VERIFY DATA FILE	VI-83
WF	WRITE PROGRAM FILE	VI-84
WFD	WRITE DATA FILE	VI-84
CL	CONDITION LOAD	VI-85
CS	CONDITION SAVE	VI-85
OPTLD	READ HEXA DATA FILE	VI-86

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,WORKD↵        . . . . . 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 section 2.7.)
- (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 section 2.5.)

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↵
ADDR  FD:ICE
0100  FFF:FFC
0300  FFC:FFB
```

. . . . C6200A0H.HEX and C6200A0L.HEX files and the program memory are collated

. . . . The contents of the FD address and the memory are 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 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 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	<p>Loads the contents of the emulation memories of ICE6200 and the contents of each setting from files or save them to files.</p> <p>(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.</p> <p>(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>.</p> <p>The loaded and saved contents are as follows:</p> <ul style="list-style-type: none"> – 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). <p>These instructions are valid when power is switched off and reapplied.</p>
Examples	<pre>#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</pre>
Notes	<p>(1) Notes (1), (2), (3), (4), (5), and (6) of page 82 are applicable to these commands.</p> <p>(2) A file name of up to seven characters may be specified as <file name> for #CS,<file name>.</p>

OPTLD *READ HEXA DATA FILE*

Format #OPTLD,0,<file name>↵

Function Load melody HEX files in the EVA628X melody data memory.
 These are HEX files output by the melody assembler and have intel HEX format.

Example #OPTLD,C2810A0↵ C2810A0.HEX files are loaded in the melody data memory

2.4.5 ROM Command Group

RP	LOAD ROM PROGRAM	VI-88
VP	VERIFY ROM PROGRAM	VI-89
ROM	ROM TYPE SELECT	VI-90

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
#
```

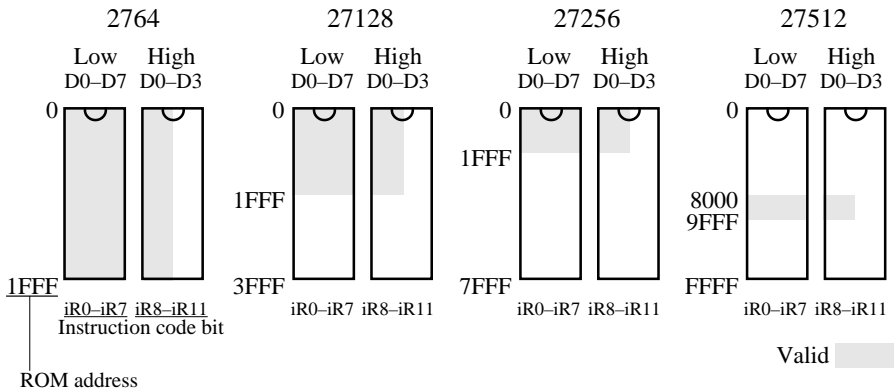
ROM ROM TYPE SELECT

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.

2.4.6 Control Command Group

I	INITIALIZE TARGET CPU	VI-92
TIM	TIME OR STEP MODE SELECTION	VI-93
OTF	ON THE FLY MODE SET	VI-94
Q	QUIT	VI-95

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

TIME OR STEP MODE SELECTION**TIM**

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	<pre>#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 #</pre>	
Note	Refer to section 2.3.10 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 section 2.3.4.

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

2.4.7 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↵ Error is generated if a value other than 1 to 8 is entered

* COMMAND ERROR *

#

Format**#HELP**,
#HELP, n

(With guidance)

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  #OPTLD
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, 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

Format

#HELP ↵

(With guidance)

#HELP, n ↵ (n=1 to 8)

Examples

#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, 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.
- (9)#OPTLD,option no.,file HEXA data load.

#

#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**Format**

#HELP↵

(With guidance)

#HELP, n↵ (n=1 to 8)

Examples

```

#
#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.

#

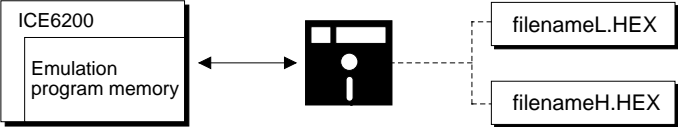
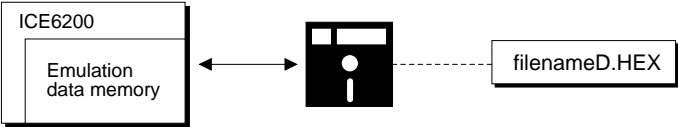
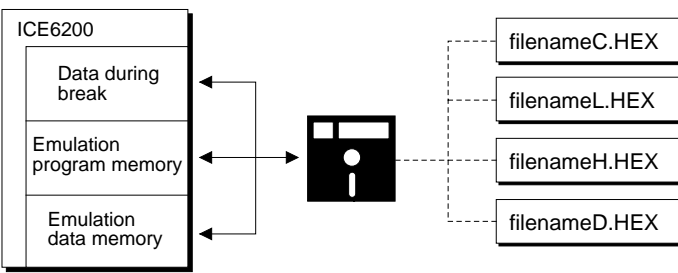
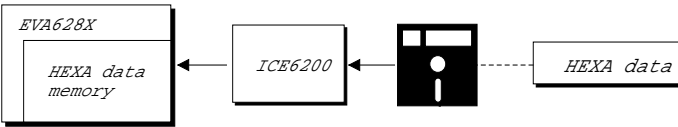
```

2.5 Error Message Summary

<i>Error message</i>	* COMMUNICATION ERROR OR ICE NOT READY *
<i>Meaning</i>	ICE6200 is disconnected or power is OFF.
<i>Recovery procedure</i>	Switch OFF the host power supply, connect cable, and reapply power. Or switch ON power to ICE6200.
<i>Error message</i>	* TARGET DOWN(1) *
<i>Meaning</i>	Evaluation board is disconnected. (Check at power ON)
<i>Recovery procedure</i>	Switch OFF power to ICE, and connect the evaluation board. Then, apply power to ICE6200.
<i>Error message</i>	* TARGET DOWN(2) *
<i>Meaning</i>	Evaluation board disconnected. (Check at command execution)
<i>Recovery procedure</i>	Switch OFF power to ICE, and connect the evaluation board. Then, apply power to ICE6200.
<i>Error message</i>	* UNDEFINED PROGRAM CODE EXIST *
<i>Meaning</i>	Undefined code is detected in the program loaded from ROM or FD. (ICE program terminates)
<i>Recovery procedure</i>	Convert ROM and FD data with the E0C62XX/62*XX cross assembler, then restart the ICE6200.
<i>Error message</i>	* COMMAND ERROR *
<i>Meaning</i>	A miss occurs by command input.
<i>Recovery procedure</i>	Reenter the proper command.
<i>Error</i>	No response after power on.
<i>Meaning</i>	The ICE-to-HOST cable is disconnected on the host side.
<i>Recovery procedure</i>	Connect the cable.

2.6 FD File Configuration

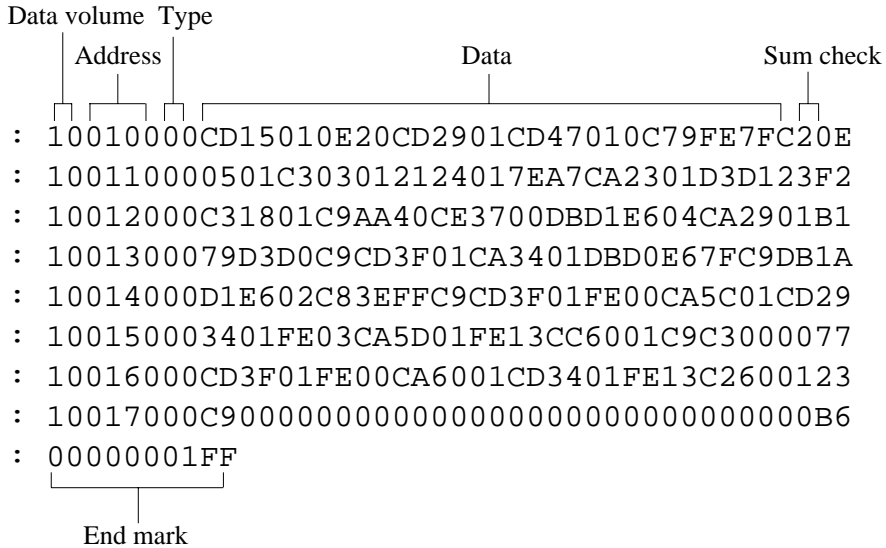
The ICE6200 uses the types of FD files listed below. All are in hexadecimal file format. For more details on hex file format, refer to section 2.7.

Command	File
WF , <filename>↵ RF , <filename>↵ VF , <filename>↵	<p>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.</p> 
WFD , <filename>↵ RFD , <filename>↵ VFD , <filename>↵	<p>The contents of data RAM (4 bits. high order 4 bits are meaningless) are output (or input, or compared) to filenameD.HEX.</p> 
CS , <filename>↵	<p>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.</p> 
OPTLD , n, <filename>↵	<p>HEXA data is loaded in the EVA62XX HEXA data memory with the ICE6200 using this command.</p> 

2.7 Appendix HEX File Format

Description of HEX file format

Example:



- 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.

VII. E0C6281MASK DATA CHECKER MANUAL

PREFACE

This manual explains how to operate the MDC6281 Mask Data Checker for 4-bit single-chip E0C6281 microcomputer.

Refer to "E0C6281 Technical Hardware Manual", "E0C6281 Technical Software Manual", and "E0C6281 Development Tool Manuals" for details about the E0C6281. Refer to "E0C62 Family Technical Guide" for details about the development procedure.

CONTENTS

1. INTRODUCTION	VII-1
1.1 Outline of the Mask Data Checker	VII-1
1.2 Execution Flow and Input/Output Files	VII-2
2. MASK DATA CHECKER OPERATION	VII-3
2.1 Creating a Work Disk	VII- 3
2.2 Copying the Data File	VII-3
2.3 Execution of MDC6281	VII-4
2.3.1 Starting MDC6281	VII-4
2.3.2 Packing of data	VII-5
2.3.3 Unpacking of data	VII-6
3. ERROR MESSAGES	VII-7
3.1 Data Error	VII-7
3.1.1 Program data error	VII-7
3.1.2 Function option data error	VII-7
3.1.3 Segment option data error	VII-8
3.2 File Error	VII-8
3.3 System Error	VII-8
4. PACK FILE CONFIGURATION	VII- 9
4.1 Program Data, Melody ROM Data and Scale ROM Data	VII-10
4.2 Segment Data	VII-11

1 INTRODUCTION

1.1 Outline of the Mask Data Checker

The Mask Data Checker MDC6281 is a software tool which checks the program data (C281yyyH.HEX and C281yyyL.HEX), option data (C281yyyF.DOC and C281yyyS.DOC), and melody data (C281yyyA.DOC) created by the user and creates the data file (C6281yyy.PAn) for generating mask patterns.

The user must send the file generated through this software tool to Seiko Epson.

Moreover, MDC6281 has the capability to restore the generated data file (C6281yyy.PAn) to the original file format (C281yyyH.HEX, C281yyyL.HEX, C281yyyF.DOC, C281yyyS.DOC, and C281yyyA.DOC).

Two MDC6281 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:

MDC6281.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 MDC6281 is shown in Figure 1.2.

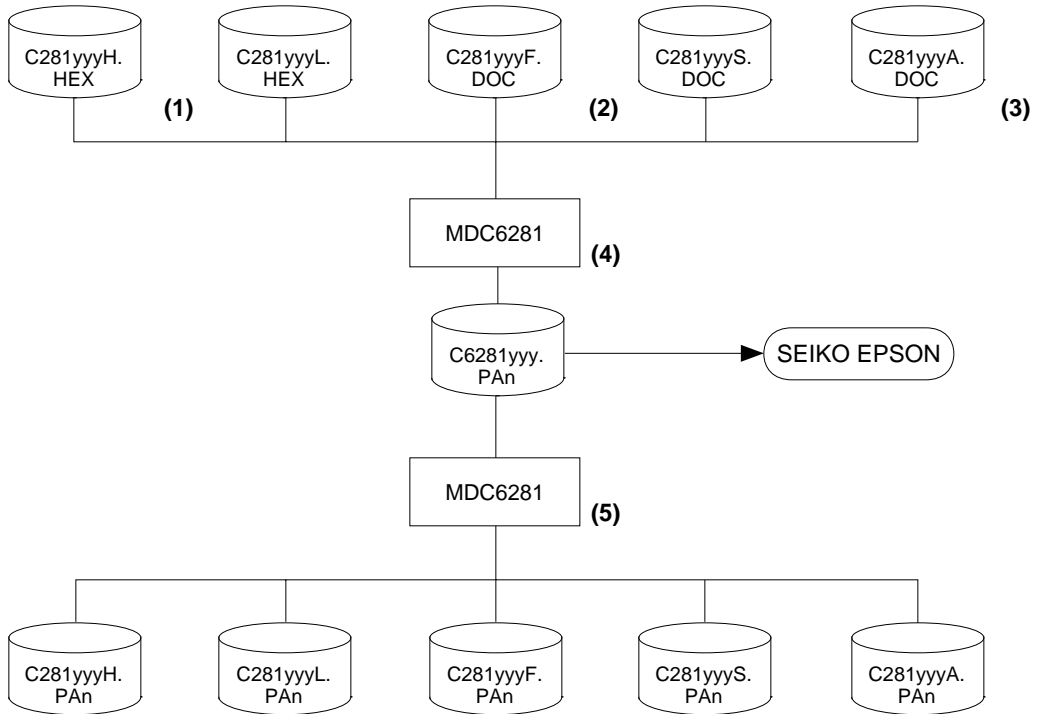


Fig. 1.2 MDC6281 Execution Flow

(1) Preparation of program data files (C281yyyH.HEX and C281yyyL.HEX)

Prepare the program data files generated from the Cross Assembler (ASM6281).

(2) Preparation of option data files (C281yyyF.DOC and C281yyyS.DOC)

Prepare the function option data file (function option) generated from the Function Option Generator (FOG6281) and segment option data file (segment option) generated from the Segment Option Generator (SOG6281).

(3) Preparation of melody data file (C281yyyA.DOC)

Prepare the melody data file generated from the Melody Assembler (MLA6281).

(4) Packing of Data

Using the Mask Data Checker (MDC6281), compile the program data, option data, and melody data in one mask data file (C6281yyy.PAn). This file must be sent to Seiko Epson.

(5) Unpacking of Data

Furthermore, the mask data file (C6281yyy.PAn) may be restored to the original program data, option data, and melody data files using the Mask Data Checker (MDC6281).

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 "MDC6281.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 (ASM6281), Function Option Generator (FOG6281), Segment Option Generator (SOG6281) and Melody Assembler (MLA6281).

Be sure to assign the following file names (the yyy portion of the file name should be as designated by Seiko Epson):

- Program data (HIGH side) : C281yyyH.HEX
(LOW side) : C281yyyL.HEX
- Option data (function option) : C281yyyF.DOC
(segment option) : C281yyyS.DOC
- Melody data (melody ROM, scale ROM, melody option) : C281yyyA.DOC

2.3 Execution of MDC6281

2.3.1 Starting MDC6281

To start MDC6281, 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>MDC6281 ↵
```

*:↵ means press the RETURN key.

When MDC6281 is started, the following message is displayed:

```

*** E0C6281 PACK / UNPACK PROGRAM Ver 1.00 ***

EEEEEEEEEE PPPPPPPP SSSSSSS OOOOOOOO NNN NNN
EEEEEEEEEE PPPPPPPPP SSS SSS OOO OOO NNNN NNN
EEE PPP PPP SSS SSS OOO OOO NNNNN NNN
EEE PPP PPP SSS OOO OOO NNNNNN NNN
EEEEEEEEEE PPPPPPPPP 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 SSSSSS OOOOOOO 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 (C281yyyH.HEX, C281yyyL.HEX, C281yyyF.DOC, C281yyyS.DOC and C281yyyA.DOC), select "2".

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

2.3.2 Packing of data

When generating data for submission to Seiko Epson, selecting "1" in the above section 2.3.1 will prompt for the name of the file to be generated as follows:

```

C281XXXH.HEX -----+
                    |
C281XXXL.HEX -----+
                    |
C281XXXA.DOC -----+----- C6281XXX.PA0 (PACK FILE)
                    |
C281XXXF.DOC -----+
                    |
C281XXXS.DOC -----+

PLEASE INPUT PACK FILE NAME (C6281XXX.PAn) ? C62810A0.PA0 [J]

```

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-submitting data after "C62810A0.PA0" has been submitted, the pack file name should be entered as "C62810A0.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.

```

C2810A0H.HEX -----+
                    |
C2810A0L.HEX -----+
                    |
C2810A0A.DOC -----+----- C2810A0.PA0
                    |
C2810A0F.DOC -----+
                    |
C2810A0S.DOC -----+

```

With this, the mask file (C6281yyy.PAn) is generated. Submit this file to Seiko Epson.

Note: Don't use the data generated with the -N option of the Cross Assembler (ASM6281) 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.
```

2.3.3 Unpacking of data

In the process of restoring the packed data to the original file, when "2" is selected in the step described in Section 2.3.1, the user is prompted for the input file name as follows:

```
PLEASE INPUT PACKED FILE NAME (C6281XXX.PAn) ? C62810A0.PA0 [↓]
```

When the file name has been entered, the unpacking process is executed and the corresponding file names are displayed.

```

                                     +----- C2810A0H.PA0
                                     |
                                     +----- C2810A0L.PA0
                                     |
C62810A0.PA0 -----+----- C2810A0A.PA0
                                     |
                                     +----- C2810A0F.PA0
                                     |
                                     +----- C2810A0S.PA0

```

With this, the mask data file (C6281yyy.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) : C281yyyH.PAn
- (LOW side) : C281yyyL.PAn
- Option data (function option) : C281yyyF.PAn
- (segment option) : C281yyyS.PAn
- Melody data (melody ROM, scale ROM, melody option) : C281yyyA.PAn

3 ERROR MESSAGES

3.1 Data Error

The program data file, option data file and melody 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.

3.1.1 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, melody and scale 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.

3.1.2 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.

3.1.3 Segment option data error

Error Message	Explanation
1. SEGMENT DATA ERROR : START MARK.	The start mark is not "¥SEGMENT". (during unpacking) *
2. SEGMENT DATA ERROR : DATA.	The segment data is not correct.
3. SEGMENT DATA ERROR : SEGMENT NUMBER.	The SEG No. is not correct.
4. SEGMENT DATA ERROR : SPEC.	The output specification of the SEG terminal is not correct.
5. SEGMENT 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.

4 PACK FILE CONFIGURATION

The pack file is configured according to the following format:

```

*
* SMC6281 MASK DATA VER 1.00
*
Program Data Header  — ¥ROM1
Model Name          — SMC6281yyy PROGRAM ROM
                    :100000000.....
Program Data        — :100010000.....
High Side (Intel Hexa Format) — : : : : : : : :
                    :00000001FF
Program Data        — :100000000.....
Low Side (Intel Hexa Format) — :100010000.....
                    : : : : : : : :
                    :00000001FF
End Mark           — ¥END
Melody ROM Header  — ¥ROM2
Model Name          — SMC6281yyy MELODY ROM
                    :100000000.....
Melody ROM Data    — :100000000.....
High Side (Intel Hexa Format) — : : : : : : : :
                    :00000001FF
Melody ROM Data    — :100000000.....
Low Side (Intel Hexa Format) — : : : : : : : :
                    :00000001FF
End Mark           — ¥END
Melody Scale ROM Header — ¥ROM3
Model Name          — SMC6281yyy SCALE ROM
                    :100000000.....
Melody Scale ROM Data (Intel Hexa Format) — :00000001FF
End Mark           — ¥END
Melody Option Data Header — ¥OPTION1
*
* OCTAVE CIRCUIT
* 32kHz ----- SELECTED
Melody Option Data — OPT2001 01
                    : : : : : : : :
                    OPT2104 04
End Mark           — ¥END
Function Option Header — ¥OPTION2
* SMC6281 FUNCTION OPTION DOCUMENT V 3.00
*
* FILE NAME C281yyyF.DOC
* USER'S NAME SEIKO EPSON CORP.
* INPUT DATE 89/11/03
*
Function Option Data — * OPTION NO.1
* < DEVICE TYPE >
* SMC6281 ( NORMAL TYPE ) ----- SELECTED
                    OPT0101 01
                    : : : : : : : :
                    OPT1602 01
End Mark           — ¥END
Segment Option Header — ¥SEGMENT
* SMC6281 SEGMENT OPTION DOCUMENT Ver 3.00
*
* FILE NAME C281yyyS.DOC
* USER'S NAME SEIKO-EPSON CORP.
* INPUT DATE 89/09/20
* COMMENT TOKYO DESIGN CENTER
* 390-4 HINO HINO-SHI TOKYO 191 JAPAN
* TEL 0425-83-7313
* FAX 0425-83-7413
*
Segment Option Data — * OPTION NO.22
*
* < LCD SEGMENT DECODE TABLE >
*
* SEG COM0 COM1 COM2 COM3
*
* 0 ... .. S
* 1 ... .. C
* : : : : :
End Mark           — ¥END

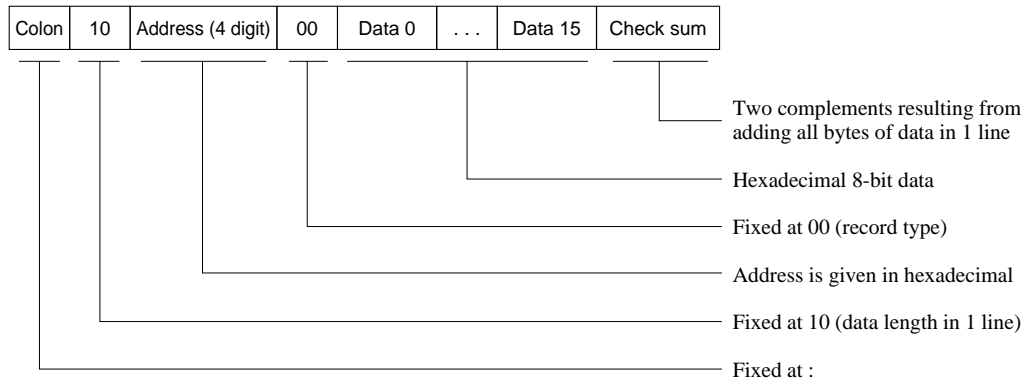
```

* ¥ sometimes appears as \, depending on the personal computer being used.

4.1 Program Data, Melody ROM Data and Scale ROM Data

The program data, melody ROM data and scale ROM data are expressed as follows, using Intel hexa format:

(1) Data Line

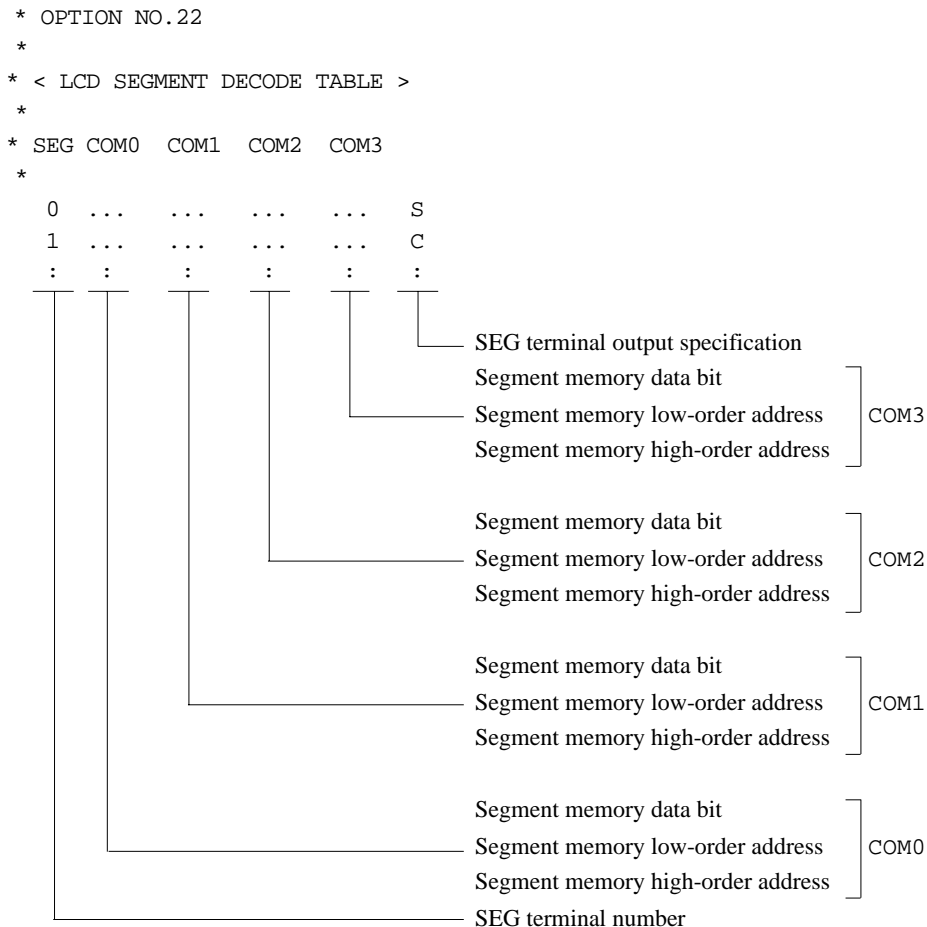


(2) End mark

: 00000001FF

4.2 Segment Data

Segment data is configured according to the following format:



AMERICA

S-MOS SYSTEMS, INC.

150 River Oaks Parkway
San Jose, CA 95134, U.S.A.
Phone: +1-408-922-0200 Fax: +1-408-922-0238
Telex: 176079 SMOS SNJUD

S-MOS SYSTEMS, INC.

EASTERN AREA SALES AND TECHNOLOGY CENTER
301 Edgewater Place, Suite 120
Wakefield, MA 01880, U.S.A.
Phone: +1-617-246-3600 Fax: +1-617-246-5443

S-MOS SYSTEMS, INC.

SOUTH EASTERN AREA SALES AND TECHNOLOGY CENTER
4300 Six Forks Road, Suite 430
Raleigh, NC 27609, U.S.A.
Phone: +1-919-781-7667 Fax: +1-919-781-6778

S-MOS SYSTEMS, INC.

CENTRAL AREA SALES AND TECHNOLOGY CENTER
1450 E.American Lane, Suite 1550
Schaumburg, IL 60173, U.S.A.
Phone: +1-847-517-7667 Fax: +1-847-517-7601

EUROPE

- HEADQUARTERS -

EPSON EUROPE ELECTRONICS GmbH

Riesstrasse 15
80992 Muenchen, GERMANY
Phone : +49-(0)89-14005-0 Fax : +49-(0)89-14005-110

- GERMANY -

EPSON EUROPE ELECTRONICS GmbH SALES OFFICE

Breidenbachstrasse 46
D-51373 Leverkusen, GERMANY
Phone : +49-(0)214-83070-0 Fax : +49-(0)214-83070-10

- UNITED KINGDOM -

EPSON EUROPE ELECTRONICS GmbH UK BRANCH OFFICE

G6 Doncastle House, Doncastle Road
Bracknell, Berkshire RG12 8PE, ENGLAND
Phone: +44-(0)1344-381700 Fax: +44-(0)1344-381701

- FRANCE -

EPSON EUROPE ELECTRONICS GmbH FRENCH BRANCH OFFICE

1 Avenue de l'Atlantique, LP 915 Les Conquerants
Z.A. de Courtaboeuf 2, F-91976 Les Ulis Cedex, FRANCE
Phone: +33-(0)1-64862350 Fax: +33-(0)1-64862355

ASIA

- HONG KONG, CHINA -

EPSON HONG KONG LTD.

20/F., Harbour Centre, 25 Harbour Road
Wanchai, HONG KONG
Phone: +852-2585-4600 Fax: +852-2827-4346
Telex: 65542 EPSCO HX

- CHINA -

SHANGHAI EPSON ELECTRONICS CO., LTD.

4F, Bldg., 27, No. 69, Gui Jing Road
Caohejing, Shanghai, CHINA
Phone: 21-6485-5552 Fax: 21-6485-0775

- TAIWAN, R.O.C. -

EPSON TAIWAN TECHNOLOGY & TRADING LTD.

10F, No. 287, Nanking East Road, Sec. 3
Taipei, TAIWAN, R.O.C.
Phone: 02-2717-7360 Fax: 02-2712-9164
Telex: 24444 EPSONTB

EPSON TAIWAN TECHNOLOGY & TRADING LTD. HSINCHU OFFICE

13F-3, No. 295, Kuang-Fu Road, Sec. 2
HsinChu 300, TAIWAN, R.O.C.
Phone: 03-573-9900 Fax: 03-573-9169

- SINGAPORE -

EPSON SINGAPORE PTE., LTD.

No. 1 Temasek Avenue, #36-00
Millenia Tower, SINGAPORE 039192
Phone: +65-337-7911 Fax: +65-334-2716

- KOREA -

SEIKO EPSON CORPORATION KOREA OFFICE

10F, KLI 63 Bldg., 60 Yoido-Dong
Youngdeungpo-Ku, Seoul, 150-010, KOREA
Phone: 02-784-6027 Fax: 02-767-3677

- JAPAN -

SEIKO EPSON CORPORATION ELECTRONIC DEVICES MARKETING DIVISION

Electronic Device Marketing Department IC Marketing & Engineering Group

421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-(0)42-587-5816 Fax: +81-(0)42-587-5624

ED International Marketing Department I (Europe & U.S.A.)

421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-(0)42-587-5812 Fax: +81-(0)42-587-5564

ED International Marketing Department II (Asia)

421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-(0)42-587-5814 Fax: +81-(0)42-587-5110



In pursuit of **“Saving” Technology**, Epson electronic devices.
Our lineup of semiconductors, liquid crystal displays and quartz devices
assists in creating the products of our customers’ dreams.
Epson IS energy savings.

EPSON

SEIKO EPSON CORPORATION
ELECTRONIC DEVICES MARKETING DIVISION

■ Electronic devices information on the Epson WWW server

<http://www.epson.co.jp>

Issue APRIL 1998 Printed in Japan  A