USER'S MANUAL

RENESAS

μSXXXXAS17103 DEVICE FILE

PC-9800 SERIES(MS-DOS™)BASED

IBM PC/AT™(PC DOS™)BASED

VERSION V6.2

AS17103(V6)
AS17104(V6)
AS17107(V3)
AS17108(V3)
AS17103L(V1)
AS17104L(V1)
AS17107L(V1)
AS17108L(V1)

Document No. EEU-1442
(O. D. No. EEU-879)
Date Published April 1993 P
Printed in Japan

USER'S MANUAL

NEC

μSXXXXAS17103 DEVICE FILE

PC-9800 SERIES(MS-DOS™)BASED

IBM PC/AT™(PC DOS™)BASED

VERSION V6.2

AS17103(V6)
AS17104(V6)
AS17107(V3)
AS17108(V3)
AS17103L(V1)
AS17104L(V1)
AS17107L(V1)
AS17108L(V1)

PREFACE

AS17103, AS17104, AS17107, AS17108, AS17103L, AS17104L, AS17107L, and AS17108L are device files for assembling uPD1710x programs. It is used together with the AS17K assembler.

These device files contain information on the uPD1710x which is necessary in assembling, such as program memory capacity, data memory capacity, instructions, reserved symbols, and mask option information.

See AS17K Assembler User's Manual (EEU-603) for the operation of the AS17K assembler and uPD1710x device file.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 DEVICE INFORMATION

Device files (AS17103, AS17104, AS17107, AS17108, AS17103L, AS17104L, AS17107L, and AS17108L) provide the following information about the uPD1710x in assembling.

(1) Program memory

512 x 16 bits (0000H-01FFH)

(2) Data memory (RAM) capacity

16 x 4 bits

(3) Instructions

See Chapter 2.

(4) Information about read/write for port registers and system registers

See Chapter 3.

(5) Reserved symbols

See Chapter 3.

(6) Device numbers and SE board numbers

Each device file has the number of the device and the number of a SE board which is the most suitable for developing programs for the device. These numbers are also output to the ICE and PRO files by the AS17K assembler. The in-circuit emulator uses these numbers when checking the device development environment and mask products to be ordered.

Table 1-1 Correspondence between Device Files and Device and SE Board Numbers

Device file (version)	Device	Device number	SE board number	SE board
AS17103 (V6)	uPD17103	05	05	SE-17103L
AS17103L (V1)	uPD17103L	25		
AS17104 (V6)	uPD17104	06		SE-17104L
AS17104L (V1)	uPD17104L	26		
AS17107 (V3)	uPD17107	11	11	SE-17107
AS17107L (V1)	uPD17107L	12		
AS17108 (V3)	uPD17108	27	12	SE-17108
AS17108L (V1)	uPD17108L	28		

CHAPTER 2 uPD1710x INSTRUCTION SET

2.1 Overview of the Instruction Set

b_{15} / $b_{14}-b_{11}$		0		1	
BIN	HEX				
0 0 0 0	0	ADD	r, m	ADD	m, #i
0 0 0 1	1	SUB	r, m	SUB	m, #i
0 0 1 0	2	ADDC	r, m	ADDC	m, #i
0 0 1 1	3	SUBC	r, m	SUBC	m, #i
0 1 0 0	4	AND	r, m	AND	m, #i
0 1 0 1	5	XOR	r, m	XOR	m, #i
0 1 1 0	6	OR	r, m	OR	m, #i
0 1 1 1	7	RET RETSK RORC STOP HALT NOP	 r s h 		
1 0 0 0	8	LD	r, m	ST	m, r
1 0 0 1	9	SKE	m, #i	SKGE	m, #i
1 0 1 0	A				
1 0 1 1	B	SKNE	m, #i	SKLT	m, #i
1 1 0 0	C	BR	addr	CALL	addr
1 1 0 1	D			MOV	m, #i
1 1 1 0	E			SKT	m, #n
1 1 1 1	F			SKF	m, #n

2.2 Legend

addr:	Program memory address (11 bits, upper two bits always set to 0)
a_H:	Program memory address high (3 bits, upper two bits always set to 0)
a_M:	Program memory address middle (four bits)
a_L:	Program memory address low (four bits)
CMP:	Compare flag
CY:	Carry flag
h:	Halt release condition
i:	Immediate data
M:	Data memory address
m:	Data memory address excluding bank $m = [m_H, m_L]$
m_H:	Data memory row address (three bits)
m_L:	Data memory column address (four bits)
n:	Bit position (four bits)
PC:	Program counter
R:	General register address R = [(RP), r]
r:	General register column address
R(n):	Bit n of the general register
RP:	General register pointer (always set to 0)
s:	Stop release condition
SP:	Stack pointer
STACK:	Value of the stack pointed to by the stack pointer
[ ]:	Data memory or register address
( ):	Data memory or register value

2.3 Instruction List

Instruction set	Mnemonic	Operand	Function	Operation	Machine code			
					Op code	3 bits	4 bits	4 bits
Add	ADD	r,m	Add memory to register	(R) ← (R) + (M)	00000	m_H	m_L	r
		m,#i	Add immediate data to memory	(M) ← (M) + i	10000	m_H	m_L	i
	ADDC	r,m	Add memory to register with carry	(R) ← (R) + (M) + (CY)	00010	m_H	m_L	r
		m,#i	Add immediate data to memory with carry	(R) ← (M) + i + (CY)	10010	m_H	m_L	i
Subtract	SUB	r,m	Subtract memory from register	(R) ← (R) - (M)	00001	m_H	m_L	r
		m,#i	Subtract immediate data from memory	(M) ← (M) - i	10001	m_H	m_L	i
	SUBC	r,m	Subtract memory from register with borrow	(R) ← (R) - (M) - (CY)	00011	m_H	m_L	r
		m,#i	Subtract immediate data from memory with borrow	(M) ← (M) - i - (CY)	10011	m_H	m_L	i
Compare	SKE	m,#i	Skip if memory equal to immediate data	(M) - i, skip if zero	01001	m_H	m_L	i
	SKGE	m,#i	Skip if memory greater than or equal to immediate data	(M) - i, skip if not borrow	11001	m_H	m_L	i
	SKLT	m,#i	Skip if memory less than immediate data	(M) - i, skip if borrow	11011	m_H	m_L	i
	SKNE	m,#i	Skip if memory not equal to immediate data	(M) - i, skip if not zero	01011	m_H	m_L	i
Logical operation	AND	m,#i	Logical AND of memory and immediate data	(M) ← (M) AND i	10100	m_H	m_L	i
		r,m	Logical AND of register and memory	(R) ← (R) AND (M)	00100	m_H	m_L	r
	OR	m,#i	Logical OR of memory and immediate data	(M) ← (M) OR i	10110	m_H	m_L	i
		r,m	Logical OR of register and memory	(R) ← (R) OR (M)	00110	m_H	m_L	r
	XOR	m,#i	Logical XOR of memory and immediate data	(M) ← (M) XOR i	10101	m_H	m_L	i
		r,m	Logical XOR of register and memory	(R) ← (R) XOR (M)	00101	m_H	m_L	r

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Function	Operation	Machine code			
					Op code	3 bits	4 bits	4 bits
Transfer	LD	r,m	Load memory of register	(R) ← (M)	01000	m_H	m_L	r
	ST	m,r	Store register to memory	(M) ← (R)	11000	m_H	m_L	r
	MOV	m,#i	Move immediate data to memory	(M) ← i	11101	m_H	m_L	i
Test	SKT	m,#n	Test memory bits, then skip if all bits specified are true	CMP ← 0, skip if M_n = all "1"	11110	m_H	m_L	n
	SKF	m,#n	Test memory bits, then skip if all bits specified are false	CMP ← 0, skip if M_n = all "0"	11111	m_H	m_L	n
Branch	BR	addr	Jump to the address	(PC) ← addr	01100	a_H	a_M	a_L
Rotation	RORC	r	Rotate register right with carry	$\begin{tikzcd} (CY) \arrow[r] & R(3) \arrow[r] & R(2) \arrow[d] \\ & R(0) \arrow[ul] & R(1) \arrow[l] \end{tikzcd}$	00111	000	0111	r
Subroutine	CALL	addr	Call subroutine	SP ← (SP) - 1, (STACK) ← ((PC) + 1), (PC) ← addr	11100	a_H	a_M	a_L
	RET		Return to main routine from subroutine	(PC) ← (STACK), (SP) ← (SP) + 1	00111	000	1110	0000
	RETSK		Return to main routine from subroutine, then skip unconditionally	(PC) ← (STACK), (SP) ← (SP) + 1 and skip	00111	001	1110	0000
Others	STOP	s	Stop clock	Stop	00111	010	1111	s
	HALT	h	Halt the CPU, restart by condition h	Halt	00111	011	1111	h
	NOP		No operation	No operation	00111	100	1111	0000

2.4 Macro Instructions Included in the Assembler (AS17K)

Legend

flag:	Any one from flag1 to flagn
flag1-flagn:	Reserved words for flag names
n:	Number
< >:	Optional

Mnemonics	Operand	n	Operation
SKTn	flag1, ...flagn	$1 \leqq n \leqq 4$	if (flag1)-(flagn) = all "1", then skip
SKFn	flag1, ...flagn	$1 \leqq n \leqq 4$	if (flag1)-(flagn) = all "0", then skip
SETn	flag1, ...flagn	$1 \leqq n \leqq 4$	(flag1)-(flagn) $\leftarrow$ 1
CLRn	flag1, ...flagn	$1 \leqq n \leqq 4$	(flag1)-(flagn) $\leftarrow$ 0

NOTn	flag1, ...flagn	$1 \leqq n \leqq 4$	if (flag) = "0", then (flag) $\leftarrow$ 1
			if (flag) = "1", then (flag) $\leftarrow$ 0

INITFLG	<NOT> flag1,
	...<NOT> flag4
			if description = NOT flag,
			then (flag) $\leftarrow$ 0
			if description = flag, then (flag) $\leftarrow$ 1

CHAPTER 3 RESERVED SYMBOLS

The port registers and system registers are defined with reserved symbols in device files (AS17103, AS17104, AS17107, AS17108, AS17103L, AS17104L, AS17107L, and AS17108L).

3.1 Port Registers and System Registers

Table 3-1 Reserved Symbols (for uPD17103, uPD17107, uPD17103L, and uPD17107L)

Name	Attribute	Value	Read/write	Description
P0B0	FLG	0.71H.0	Read/write	Bit 0 of port 0B
P0B1	FLG	0.71H.1	Read/write	Bit 1 of port 0B
P0B2	FLG	0.71H.2	Read/write	Bit 2 of port 0B
P0B3(*)	FLG	0.71H.3	Read	Always set to 0
P0C0	FLG	0.72H.0	Read/write	Bit 0 of port 0C
P0C1	FLG	0.72H.1	Read/write	Bit 1 of port 0C
P0C2	FLG	0.72H.2	Read/write	Bit 2 of port 0C
P0C3	FLG	0.72H.3	Read/write	Bit 3 of port 0C
P0D0	FLG	0.73H.0	Read/write	Bit 0 of port 0D
P0D1	FLG	0.73H.1	Read/write	Bit 1 of port 0D
P0D2	FLG	0.73H.2	Read/write	Bit 2 of port 0D
P0D3	FLG	0.73H.3	Read/write	Bit 3 of port 0D
BCD	FLG	0.7EH.0	Read/write	BCD arithmetic flag
PSW	MEM	0.7FH	Read/write	Program status word
Z	FLG	0.7FH.1	Read/write	Zero flag
CY	FLG	0.7FH.2	Read/write	Carry flag
CMP	FLG	0.7FH.3	Read/write	Compare flag

* Although a pin corresponding to P0B3 does not exist in the uPD17103, uPD17107, uPD17103L, and uPD17107L, it is defined as a read-only flag so that it is treated as a dummy bit when a built-in macro is used.

Table 3-2 Reserved Symbols (for uPD17104, uPD17108, uPD17104L, and uPD17108L)

Name	Attribute	Value	Read/write	Description
P0A0	FLG	0.70H.0	Read/write	Bit 0 of port 0A
P0A1	FLG	0.70H.1	Read/write	Bit 1 of port 0A
P0A2	FLG	0.70H.2	Read/write	Bit 2 of port 0A
P0A3	FLG	0.70H.3	Read/write	Bit 3 of port 0A
P0B0	FLG	0.71H.0	Read/write	Bit 0 of port 0B
P0B1	FLG	0.71H.1	Read/write	Bit 1 of port 0B
P0B2	FLG	0.71H.2	Read/write	Bit 2 of port 0B
P0B3	FLG	0.71H.3	Read/write	Bit 3 of port 0B
P0C0	FLG	0.72H.0	Read/write	Bit 0 of port 0C
P0C1	FLG	0.72H.1	Read/write	Bit 1 of port 0C
P0C2	FLG	0.72H.2	Read/write	Bit 2 of port 0C
P0C3	FLG	0.72H.3	Read/write	Bit 3 of port 0C
P0D0	FLG	0.73H.0	Read/write	Bit 0 of port 0D
P0D1	FLG	0.73H.1	Read/write	Bit 1 of port 0D
P0D2	FLG	0.73H.2	Read/write	Bit 2 of port 0D
P0D3	FLG	0.73H.3	Read/write	Bit 3 of port 0D
BCD	FLG	0.7EH.0	Read/write	BCD arithmetic flag
PSW	MEM	0.7FH	Read/write	Program status word
Z	FLG	0.7FH.1	Read/write	Zero flag
CY	FLG	0.7FH.2	Read/write	Carry flag
CMP	FLG	0.7FH.3	Read/write	Compare flag

3.2 Reserved Words (In Alphabetical Order)

3.2.1 Instructions and pseudo instructions

ADD	EXITR	NIBBLE6	SET3
ADDC	EXTRN	NIBBLE6V	SET4
AND	FLG	NIBBLE7	SFCOND
BELOW	GLOBAL	NIBBLE7V	SKE
BR	HALT	NIBBLE8	SKF
C14344	IF	NIBBLE8V	SKF1
C4444	IFCHAR	NOBMAC	SKF2
CALL	IFNCHAR	NOLIST	SKF3
CASE	INCLUDE	NOMAC	SKF4
CLR1	INITFLG	NOP	SKGE
CLR2	IRP	NOT1	SKLT
CLR3	LAB	NOT2	SKNE
CLR4	LBMAC	NOT3	SKT
CSEG	LD	NOT4	SKT1
DAT	LFCOND	OBMAC	SKT2
DB	LIST	OMAC	SKT3
DW	LITERAL	OPTION	SKT4
EJECT	LMAC	OR	SMAC
ELSE	MACRO	ORG	ST
END	MEM	OTHER	STOP
ENDCASE	NIBBLE	PUBLIC	SUB
ENDIF	NIBBLE1	PURGE	SUBC
ENDIFC	NIBBLE2	REPT	SUMMARY
ENDIFNC	NIBBLE2V	RET	TAG
ENDM	NIBBLE3	RETSK	TITLE
ENDOP	NIBBLE3V	RORC	XOR
ENDP	NIBBLE4	SBMAC	ZZZERROR
ENDR	NIBBLE4V	SET	ZZZMCHK
EOF	NIBBLE5	SET1	ZZZMSG
EXIT	NIBBLE5V	SET2	ZZZOPT

3.2.2 Registers and flags

(1) AS17103, AS17103L, AS17107, and AS17107L

BCD	POC2	ZZZ1	ZZZLSARG
CMP	POC3	ZZZ2	ZZZPRINT
CY	PODO	ZZZ3	ZZZSKIP
OPEN	POD1	ZZZ4	ZZZSYDOC
POBO	POD2	ZZZ5	ZZZALBMAC
POB1	POD3	ZZZ6	ZZZALMAC
POB2	PSW	ZZZ7	ZZZARGC
POBPLUP	RESPLUP	ZZZ8	ZZZLINE
POCO	Z	ZZZ9	
POC1	ZZZO	ZZZDEVID	

(2) AS17104, AS17104L, AS17108, and AS17108L

BCD	POB3	RESPLUP	ZZZ9
CMP	POBPLUP	Z	ZZZDEVID
CY	POCO	ZZZO	ZZZLSARG
OPEN	POC1	ZZZ1	ZZZPRINT
POAO	POC2	ZZZ2	ZZZSKIP
POA1	POC3	ZZZ3	ZZZSYDOC
POA2	PODO	ZZZ4	ZZZALBMAC
POA3	POD1	ZZZ5	ZZZALMAC
POBO	POD2	ZZZ6	ZZZARGC
POB1	POD3	ZZZ7	ZZZLINE
POB2	PSW	ZZZ8	

CHAPTER 4 MASK OPTION DEFINITION PSEUDO INSTRUCTIONS

To create programs for uPD1710x products, it is necessary to specify mask options in source programs to be assembled using mask option definition pseudo instructions.

The uSxxxxAS1710x contains the device file (D1710x.DEV) and option file (D1710x.OPT), which correspond to the uPD1710x product. If these device and option files are registered in the same current directory, the device and option files are automatically loaded, and assembling is performed by specifying a device file name in a sequential file in assembling or specifying a device file name when starting the assembling.

To specify mask options, register the device file (D1710x.DEV) and option file (D1710x.OPT) in the same current directory before assembly. For uPD17103, for instance, register the D17103.DEV and D17103.OPT files in the same current directory.

Specify mask options for the following pins:

[For uPD17103, uPD17107, uPD17103L, and uPD17107L]

. Port 0B ($P0B_0$, $P0B_1$, $P0B_2$)
. $\overline{\text{RESET}}$ pin

[For uPD17104, uPD17108, uPD17104L, and uPD17108L]

. Port 0B ($P0B_0$, $P0B_1$, $P0B_2$, $P0B_3$)
. $\overline{\text{RESET}}$ pin

4.1 OPTION and ENDOP Pseudo Instructions

The block from the OPTION pseudo instruction to the ENDOP pseudo instruction is defined as the mask option definition block.

The format for the mask option definition block is shown below. Only the two pseudo instructions listed in Table 4-1 and Table 4-2 can be described in this block.

[Format]

Symbol	Mnemonic	Operand	Comment
[label:]	OPTION : ENDOP		[;comment]

4.2 Mask Option Definition Pseudo Instructions

(1) For uPD17103, uPD17107, uPD17103L, and uPD17107L

Table 4-1 lists the pseudo instructions which define the mask options for each pin.

Table 4-1 Mask Option Definition Pseudo Instructions (for uPD17103, uPD17107, uPD17103L, and uPD17107L)

Pin	Mask option pseudo instruction	Number of operands	Operand name
POB-POB$_0$	OPTPOB	3	POBPLUP (with pull-up resistor) OPEN (without pull-up resistor)
$\overline{\text{RESET}}$	OPTRES	1	RESPLUP (with pull-up resistor) OPEN (without pull-up resistor)

The OPTPOB format is shown below. Specify the mask options in the operand field starting at the first operand in the order of POB$_2$, POB$_1$, and POB$_0$.

Symbol	Mnemonic	Operand	Comment
[label]	OPTPOB	(POB$_2$),(POB$_1$),(POB$_0$)	[;comment]

The OPTRES format is shown below. Specify the $\overline{\text{RESET}}$ mask option in the operand field.

Symbol	Mnemonic	Operand	Comment
[label]	OPTRES	($\overline{\text{RESET}}$)	[;comment]

Example of specifying mask options

Specify the following mask options in a source file to be assembled for the uPD17103.

. POB_2: Pull-up, POB_1: Open, POB_0: Open
. $\overline{RESET}$ pin: Pull-up

```
Symbol                    Mnemonic     Operand                 Comment
;uPD17103
Setting mask options:     OPTION
                          OPTPOB       POBPLUP, OPEN, OPEN
                          OPTRES       RESPLUP
                          ENDOP
```

(2) For uPD17104, uPD17108, uPD17104L, and uPD17108L

Table 4-2 lists the pseudo instructions which define the mask options for each pin.

Table 4-2 Mask Option Definition Pseudo Instructions (for uPD17104, uPD17108, uPD17104L, and uPD17108L)

Pin	Mask option pseudo instruction	Number of operands	Operand name
POB_3-POB_0	OPTPOB	4	POBPLUP (with pull-up resistor) OPEN (without pull-up resistor)
$\overline{RESET}$	OPTRES	1	RESPLUP (with pull-up resistor) OPEN (without pull-up resistor)

The OPTPOB format is shown below. Specify the mask options in the operand field starting at the first operand in the order of POB_3, POB_2, POB_1, and POB_0.

Symbol	Mnemonic	Operand	Comment
[label]	OPTPOB	(POB_3),(POB_2),(POB_1),(POB_0)	[;comment]

The OPTRES format is shown below. Specify the $\overline{\text{RESET}}$ mask option in the operand field.

Symbol	Mnemonic	Operand	Comment
[label]	OPTRES	($\overline{\text{RESET}}$)	[;comment]

Example of specifying mask options

Specify the following mask options in a source file to be assembled for the uPD17104.

. POB_3: Pull-up, POB_2: Pull-up, POB_1: Open, POB_0: Open
. $\overline{\text{RESET}}$ pin: Pull-up

Symbol	Mnemonic	Operand	Comment
;uPD17104			
Setting mask options:	OPTION		
	OPTPOB	POBPLUP, POBPLUP, OPEN, OPEN	
	OPTRES	RESPLUP	
	ENDOP		

CHAPTER 5 FORMATS OF LOAD MODULE FILES

Hexadecimal load module files to be output by the AS17K assembler are classified into two output format types: an ICE file and a PRO file.

The ICE and PRO files must be used according to their applications. These files contain their user program areas, assembling environment information areas, and in-circuit emulator operating environment information areas.

(1) Format of a hexadecimal load module file

The assembler outputs data in hexadecimal load module files in the following sample format:

[Example of the format of a hexadecimal load module file]

```
: 10 0002 00 2B41000BFC80F ··· 3A20 EC
|  |   |   |        |               |
① ②   ③   ④       ⑤              ⑥
:  00 0000 01 FF
|  |   |   |  |
① ②   ③   ④ ⑥
```

① Record mark

Start of a record

② Number of codes (two digits)

Number of codes (data items in bytes) stored in a record. The number is represented in hexadecimal up to 10H (corresponding to 16 codes). The number is 00H for the last record.

③ Address (four digits)

Start address of codes in a record. 0000H, which is used for the last record, is not related to the address.

④ Record type (two digits)

Record type 00H indicates that the record is a data record. Record type 01H indicates that the record is the last record.

⑤ Code (up to 32 digits (16 bytes))

Data of up to 16 bytes is output to this field byte by byte.

⑥ Checksum (two digits)

The byte data is output to field ⑥ so that the lowest-order byte of the sum of the data items in ②, ③, ④, ⑤, and ⑥ in bytes is 00H (even parity).

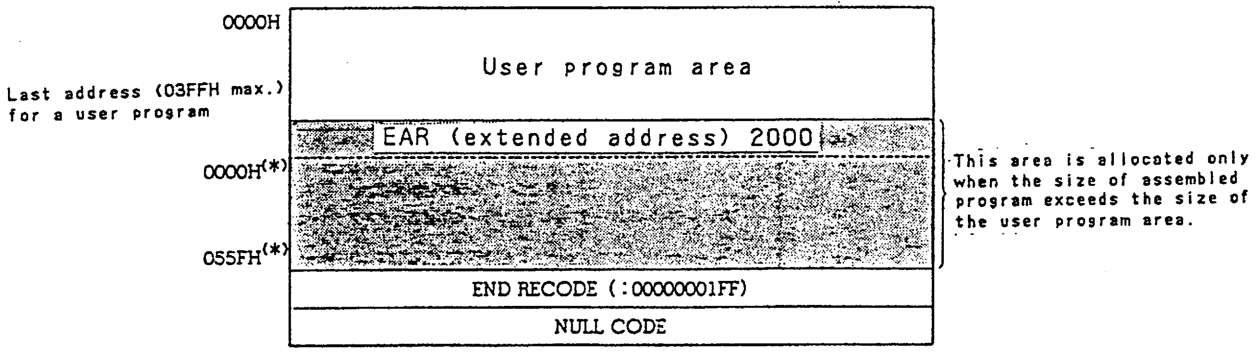
(2) ICE file

The ICE file contains hexadecimal data to be output by the AS17K assembler. The data is used for an in-circuit emulator (IE-17K or IE-17K-ET) only. Figure 5-1 shows the output format of the file data assembled using the uSxxxxAS17103.

This file consists of two subfiles. The first subfile contains a program area which consists of a user program area and patch area. The patch area is allocated only when patching is performed in the in-circuit emulator. The second subfile contains an in-circuit emulator operating environment information area, an assembling environment information area, and an SE board environment information area. Various data items specifying the operation of the in-circuit emulator are contained in these areas.

Fig. 5-1 Format of the ICE File

First subfile: Program area


0000H
User program area
Last address (03FFH max.) for a user program
EAR (extended address) 2000
0000H(*)
055FH(*)
This area is allocated only when the size of assembled program exceeds the size of the user program area.
END RECODE (:00000001FF)
NULL CODE

* 8000 to 855FH for the in-circuit emulator

(to be continued)

Fig. 5-1 Format of the ICE File (Cont'd)

Second subfile: In-circuit emulator operating environment information area and assembling environment information area

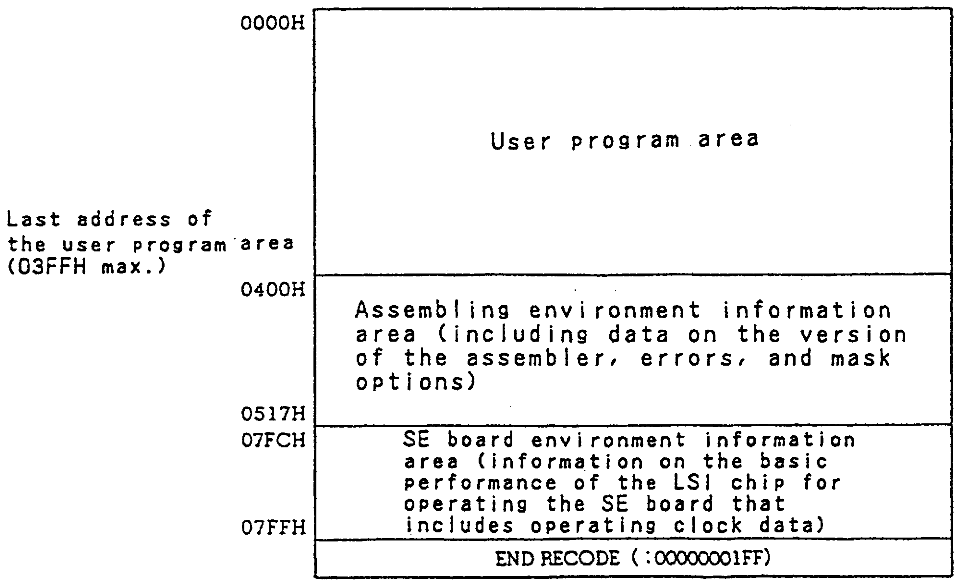
Address	Contents
0560H–05E1H	In-circuit emulator operating environment information area ① (Patch information)
05E2H–06E3H	In-circuit emulator operating environment information area ② (RAM map)
06E4H–07FBH	Assembling environment information area (including data on the version of the assembler, errors, and mask options)
07FCH–07FFH	SE board environment information area (information on the basic performance of the LSI chip for operating the SE board that includes operating clock data)
	END RECODE (:00000001FF)

(3) PRO file

The PRO file contains hexadecimal data to be output by the AS17K assembler. The data is used only for mask products to be ordered, PROM products to be evaluated with a single SE board, and one-time PROM products (uPD17P10x). To output the PRO file data, /PRO must be specified with an assemble option during assembly. Figure 5-2 shows the output format of the file data assembled using the uSxxxxAS17103.

The PRO file contains only one file. It consists of a user program area, an assembling environment information area, and an SE board environment information area.

Fig. 5-2 Format of the PRO File


0000H
User program area
Last address of the user program area (03FFH max.)
0400H
Assembling environment information area (including data on the version of the assembler, errors, and mask options)
0517H
07FCH
SE board environment information area (information on the basic performance of the LSI chip for operating the SE board that includes operating clock data)
07FFH
END RECODE (:00000001FF)

Remarks 1. The assembling environment information area also contains mask option information for LSI chip masking. Documents on mask options are unnecessary when mask products are ordered.

2. Addresses 0518H to 07FBH are not assigned to the PRO file.

(4) Comparison between load module files: ICE and PRO files

Even when no source files are changed, the ICE and PRO files may differ in the assembler output results; namely, the contents of the assembling environment information areas in these files may be different. This is because these areas contain information on the dates when their source files were created.

Table 5-1 Data Items Which May Differ between the ICE and PRO Files Even If the Source File Does Not Change

Data item	Address	
	ICE file	PRO file
Program name (character string of up to 64K bytes which is specified with an assemble option (/' PROG='))	06E4H - 0723H	0400H - 043FH
Data on mask options	0724H, 0725H	0440H, 0441H
Information for a simple host	07ADH	04C9H
Information of whether an error has occurred or whether a warning has been issued	07B0H	04CCH
Time, day, month, and year source file was created(*)	07BEH - 07C7H	04DAH - 04E3H
Version of a device file	07DCH, 07DDH	04F8H, 04F9H
Version of the assembler	07DEH, 07E1H	04FAH, 04FDH

* If a source file is divided into multiple modules and the modules are updated, time, day, month, and year are also updated.

Caution: Do not change a load module directly. To change a load module, change and reassemble the source file. If a load module is directly changed, the history of the load module file does not agree with those of other files, causing a software bug.