USER'S MANUAL

NEC

AS17145
DEVICE FILE

VERSION V1

PC-9800 SERIES (MS-DOS™) BASE
IBM PC/AT™ (PC DOS™) BASE

Model
μPD17145
μPD17147
μPD17149

Document No. EEU-1486A
(O. D. No. EEU-949A)
Date Published November 1994 P
Printed in Japan

SIMPLEHOST is a trademark of NEC Corporation.
MS-DOS is a trademark of Microsoft Corporation.
PC DOS and PC/AT are trademarks of IBM Corporation.

Main Revisions in this Edition

Page	Description
Throughout	Addition of RA17K assembler package (under development)
p. 29 through 31	Addition of device name to **Table 5-1 Items in Assembly Environment Data Area That Are Subject to Change** Addition of **Note** to program name Change of **Note** for creation date/time of files

The mark ★ shows revised points.

INTRODUCTION

Device files are files that contain data (device data) that is unique to and dependent upon 17K Series devices. This data is required when using the following 17K Series software development tools.

- AS17K assembler or RA17K assembler package (under development) ★
- SIMPLEHOST™

The following device files are bundled with the AS17145.

- μPD17145 device file (AS17145)
- μPD17147 device file (AS17147)
- μPD17149 device file (AS17149)

Each of the above device files includes files named with the extension ".DEV" or ".OPT". When using these files, both types of files must be stored in the same directory.

- File with ".DEV" extension: device file (main file)
- File with ".OPT" extension: option file

[List of files bundled with the AS17145]

Device file	Bundled file names
AS17145	D17145.DEV, D17145.OPT
AS17147	D17147.DEV, D17147.OPT
AS17149	D17149.DEV, D17149.OPT

For details of the AS17K assembler and the use of device files bundled with the μPD17145 subseries products, see the **AS17K ASSEMBLER USER'S MANUAL (EEU-1287)**.

Unless otherwise specified, the AS17145 is treated as the representative of the AS17145, 17147, and 17149 throughout this manual.

[MEMO]

CONTENTS

CHAPTER 1 DEVICE DATA 1

CHAPTER 2 μPD17145 SUBSERIES INSTRUCTION SET 3
2.1 Instruction Set Summary 3
2.2 Legend 4
2.3 Instruction List 5
2.4 Instructions Bundled with Assembler (AS17K) 7

CHAPTER 3 RESERVED SYMBOLS 9
3.1 Data Buffer (DBF) 10
3.2 System Register (SYSREG) 10
3.3 Port Register 11
3.4 Register File 12
3.5 Peripheral Hardware Register 13
3.6 Other Symbols 13
3.7 List of Reserved Words (in Alphabetical Order) 14
3.7.1 Instructions and pseudo-instructions 14
3.7.2 Registers and flags 15
3.7.3 Mask options 16

CHAPTER 4 MASK OPTIONS 17
4.1 Specification of Mask Options 17

CHAPTER 5 LOAD MODULE FILE FORMAT 19

LIST OF FIGURES

Figure No.	Title, Page
5-1	ICE File Format ... 20
5-2	PRO File Format ... 23

LIST OF TABLES

Table No.	Title, Page
1-1	Relations among Device Files, Device Numbers, and SE Board Numbers ... 1
4-1	List of Mask Option Definition Pseudo-instructions ... 18
5-1	Items in Assembly Environment Data Area That Are Subject to Change ... 26

CHAPTER 1 DEVICE DATA

When assembled, AS17145 provides the following data related to μPD17145 subseries products.

(1) Program memory (ROM) capacity

μPD17145: 1024 × 16 bits (0000H-03FFH)
μPD17147: 2048 × 16 bits (0000H-07FFH)
μPD17149: 4096 × 16 bits (0000H-0FFFH)

(2) Data memory (RAM) capacity

110 × 4 bits (BANK0)

(3) Usable instructions

See **CHAPTER 2 μPD17145 SUBSERIES INSTRUCTION SET**.

(4) Register file, port register, and peripheral register read and write data

See **CHAPTER 3 RESERVED SYMBOLS**.

(5) Reserved symbols

See **CHAPTER 3 RESERVED SYMBOLS**.

(6) Mask option data

See **CHAPTER 4 MASK OPTIONS**.

(7) Device files, device numbers, and SE board numbers

Device files contain a register of device numbers for each device and SE board numbers to indicate optimum SE boards for developing various products. These device files are also included in ICE files and PRO files output by the assembler (AS17K). These device files are used when the in-circuit emulator checks the development environment and during checking of mask orders.

Table 1-1. Relations among Device Files, Device Numbers, and SE Board Numbers

Device file (version)	Device name	Device number	SE board number	SE board
AS17145 (V1)	μPD17145	40	40	SE-17145
AS17147 (V1)	μPD17147	42		
AS17149 (V1)	μPD17149	44		

[MEMO]

CHAPTER 2 μPD17145 SUBSERIES INSTRUCTION SET

2.1 Instruction Set Summary

b_{14}–b_{11} \ b_{15}		0		1	
BIN	HEX				
0 0 0 0	0	ADD	r, m	ADD	m, #n4
0 0 0 1	1	SUB	r, m	SUB	m, #n4
0 0 1 0	2	ADDC	r, m	ADDC	m, #n4
0 0 1 1	3	SUBC	r, m	SUBC	m, #n4
0 1 0 0	4	AND	r, m	AND	m, #n4
0 1 0 1	5	XOR	r, m	XOR	m, #n4
0 1 1 0	6	OR	r, m	OR	m, #n4
0 1 1 1	7	INC INC MOVT BR CALL RET RETSK EI DI RETI PUSH POP GET PUT PEEK POKE RORC STOP HALT NOP	AR IX DBF, @AR @AR @AR AR AR DBF, p p, DBF WR, rf rf, WR r s h		
1 0 0 0	8	LD	r, m	ST	m, r
1 0 0 1	9	SKE	m, #n4	SKGE	m, #n4
1 0 1 0	A	MOV	@r, m	MOV	m, @r
1 0 1 1	B	SKNE	m, #n4	SKLT	m, #n4
1 1 0 0	C	BR	addr (page 0)	CALL	addr
1 1 0 1	D	BR	addr (page 1)	MOV	m, #n4
1 1 1 0	E			SKT	m, #n
1 1 1 1	F			SKF	m, #n

2.2 Legend

AR	: Address register
ASR	: Address stack register to indicate stack pointers
addr	: Program memory address (lower 11 bits)
BANK	: Bank register
CMP	: Compare flag
CY	: Carry flag
DBF	: Data buffer
h	: Hold cancellation conditions
INTEF	: Interrupt enable flag
INTR	: Register that is automatically saved to a stack when an interrupt occurs
INTSK	: Interrupt stack register
IX	: Index register
MP	: Data memory row address pointer
MPE	: Memory pointer enable flag
m	: Data memory address indicated by m_R and m_C
m_R	: Data memory row address (high)
m_C	: Data memory column address (low)
n	: Bit position (4 bits)
n4	: Immediate data (4 bits)
PAGE	: Page (program counter's bit 11)
PC	: Program counter
p	: Peripheral address
p_H	: Peripheral address (higher 3 bits)
p_L	: Peripheral address (lower 4 bits)
r	: General register column address
rf	: Register file address
rf_R	: Register file row address (higher 3 bits)
rfc	: Register file column address (lower 4 bits)
SP	: Stack pointer
s	: Stop cancellation conditions
WR	: Window register
(x)	: × indicates addressed contents

2.3 Instruction List

Instruction set	Mnemonic	Operand	Operation	Instruction code			
				Op code	Operand		
Add	ADD	r, m	(r) ← (r) + (m)	00000	m_R	mc	r
		m, #n4	(m) ← (m) + n4	10000	m_R	mc	n4
	ADDC	r, m	(r) ← (r) + (m) + CY	00010	m_R	mc	r
		m, #n4	(m) ← (m) + n4 + CY	10010	m_R	mc	n4
	INC	AR	AR ← AR + 1	00111	000	1001	0000
		IX	IX ← IX + 1	00111	000	1000	0000
Subtract	SUB	r, m	(r) ← (r) – (m)	00001	m_R	mc	r
		m, #n4	(m) ← (m) – n4	10001	m_R	mc	n4
	SUBC	r, m	(r) ← (r) – (m) – CY	00011	m_R	mc	r
		m, #n4	(m) ← (m) – n4 – CY	10011	m_R	mc	n4
Logical operation	OR	r, m	(r) ← (r) ∨ (m)	00110	m_R	mc	r
		m, #n4	(m) ← (m) ∨ n4	10110	m_R	mc	n4
	AND	r, m	(r) ← (r) ∧ (m)	00100	m_R	mc	r
		m, #n4	(m) ← (m) ∧ n4	10100	m_R	mc	n4
	XOR	r, m	(r) ← (r) ∀ (m)	00101	m_R	mc	r
		m, #n4	(m) ← (m) ∀ n4	10101	m_R	mc	n4
Decision	SKT	m, #n	CMP ← 0, if (m) ∧ n = n, then skip	11110	m_R	mc	n
	SKF	m, #n	CMP ← 0, if (m) ∧ n = 0, then skip	11111	m_R	mc	n
Compare	SKE	m, #n4	(m) – n4, skip if zero	01001	m_R	mc	n4
	SKNE	m, #n4	(m) – n4, skip if not zero	01011	m_R	mc	n4
	SKGE	m, #n4	(m) – n4, skip if not borrow	11001	m_R	mc	n4
	SKLT	m, #n4	(m) – n4, skip if borrow	11011	m_R	mc	n4
Rotate	RORC	r	→ CY → $(r)_{b3}$ → $(r)_{b2}$ → $(r)_{b1}$ → $(r)_{b0}$ →	00111	000	0111	r
Transfer	LD	r, m	(r) ← (m)	01000	m_R	mc	r
	ST	m, r	(m) ← (r)	11000	m_R	mc	r
	MOV	@r, m	if MPE = 1: (MP, (r)) ← (m) if MPE = 0: (BANK, m_R, (r)) ← (m)	01010	m_R	mc	r
		m, @r	if MPE = 1: (m) ← (MP, (r)) if MPE = 0: (m) ← (BANK, m_R, (r))	11010	m_R	mc	r
		m, #n4	(m) ← n4	11101	m_R	mc	n4
	MOVT	DBF, @AR	SP ← SP – 1, ASR ← PC, PC ← AR, DBF ← (PC), PC ← ASR, SP ← SP + 1	00111	000	0001	0000
	PUSH	AR	SP ← SP – 1, ASR ← AR	00111	000	1101	0000

Instruction set	Mnemonic	Operand	Operation	Instruction code			
				Op code	Operand		
Transfer	POP	AR	AR ← ASR, SP ← SP + 1	00111	000	1100	0000
	PEEK	WR, rf	WR ← (rf)	00111	rf_R	0011	rf_C
	POKE	rf, WR	(rf) ← WR	00111	rf_R	0010	rf_C
	GET	DBF, p	DBF ← (p)	00111	p_H	1011	p_L
	PUT	p, DBF	(p) ← DBF	00111	p_H	1010	p_L
Branch	BR	addr	**Note**	**Note**	addr		
		@AR	PC ← AR	00111	000	0100	0000
Subroutine	CALL	addr	SP ← SP − 1, ASR ← PC PC_{10-0} ← addr	11100	addr		
		@AR	SP ← SP − 1, ASR ← PC PC ← AR	00111	000	0101	0000
	RET		PC ← ASR, SP ← SP + 1	00111	000	1110	0000
	RETSK		PC ← ASR, SP ← SP + 1 and skip	00111	001	1110	0000
	RETI		PC ← ASR, INTR ← INTSK, SP ← SP + 1	00111	100	1110	0000
Interrupt	EI		INTEF ← 1	00111	000	1111	0000
	DI		INTEF ← 0	00111	001	1111	0000
Others	STOP	s	STOP	00111	010	1111	s
	HALT	h	HALT	00111	011	1111	h
	NOP		No operation	00111	100	1111	0000

Note The operation and op code of "BR addr" for the μPD17145, μPD17147, and μPD17149 are as follows.

(a) μPD17145 and μPD17147

Mnemonic	Operand	Operation	Op code
BR	addr	PC_{10-0} ← addr	01100

(b) μPD17149

Mnemonic	Operand	Operation	Op code
BR	addr	PC_{10-0} ← addr, PAGE ← 0	01100
		PC_{10-0} ← addr, PAGE ← 1	01101

2.4 Instructions Bundled with Assembler (AS17K)

Legend

flagn : FLG-type symbol
n : Bit No.
< > : Contents between < > symbols can be omitted.

	Mnemonic	Operand	Operation	n
Bundled macros	SKTn	flag1, ...flagn	if (flag1) to (flagn) = all "1", then skip	1 ≤ n ≤ 4
	SKFn	flag1, ...flagn	if (flag1) to (flagn) = all "0", then skip	1 ≤ n ≤ 4
	SETn	flag1, ...flagn	(flag1) to (flagn) ← 1	1 ≤ n ≤ 4
	CLRn	flag1, ...flagn	(flag1) to (flagn) ← 0	1 ≤ n ≤ 4
	NOTn	flag1, ...flagn	if (flagn) = "0", then (flagn) ← 1 if (flagn) = "1", then (flagn) ← 0	1 ≤ n ≤ 4
	INITFLG	<NOT> flag1, ... <<NOT> flagn>	if description = NOT flagn, then (flagn) ← 0 if description = flagn, then (flagn) ← 1	1 ≤ n ≤ 4
	BANKn		(BANK) ← n	n = 0

[MEMO]

CHAPTER 3 RESERVED SYMBOLS

The symbols defined by the AS17145 are described on the following pages.
These symbols are listed below.

- Data buffer (DBF)
- System register (SYSREG)
- Port register
- Register file
- Peripheral hardware register
- Others

3.1 Data Buffer (DBF)

Symbol name	Attribute	Value	Read/ Write	Description
DBF3	MEM	0.0CH	R/W	DBF bits b15 to b12
DBF2	MEM	0.0DH	R/W	DBF bits b11 to b8
DBF1	MEM	0.0EH	R/W	DBF bits b7 to b4
DBF0	MEM	0.0FH	R/W	DBF bits b3 to b0

3.2 System Register (SYSREG)

Symbol name	Attribute	Value	Read/ Write	Description
AR3	MEM	0.74H	R	Address register bits b15-b12
AR2	MEM	0.75H	R/W	Address register bits b11-b8
AR1	MEM	0.76H	R/W	Address register bits b7-b4
AR0	MEM	0.77H	R/W	Address register bits b3-b0
WR	MEM	0.78H	R/W	Window register
BANK	MEM	0.79H	R/W	Bank register
IXH	MEM	0.7AH	R/W	Index register (high)
MPH	MEM	0.7AH	R/W	Data memory row address pointer (high)
MPE	FLG	0.7AH.3	R/W	Memory pointer enable flag
IXM	MEM	0.7BH	R/W	Index register (middle)
MPL	MEM	0.7BH	R/W	Data memory row address pointer (low)
IXL	MEM	0.7CH	R/W	Index register (low)
RPH	MEM	0.7DH	R/W	General register pointer (high)
RPL	MEM	0.7EH	R/W	General register pointer (low)
PSW	MEM	0.7FH	R/W	Program status word
BCD	FLG	0.7EH.0	R/W	BCD flag
CMP	FLG	0.7FH.3	R/W	Compare flag
CY	FLG	0.7FH.2	R/W	Carry flag
Z	FLG	0.7FH.1	R/W	Zero flag
IXE	FLG	0.7FH.0	R/W	Index enable flag

3.3 Port Register

Symbol name	Attribute	Value	Read/ Write	Description
P0A3	FLG	0.70H.3	R/W	Port 0A bit b3
P0A2	FLG	0.70H.2	R/W	Port 0A bit b2
P0A1	FLG	0.70H.1	R/W	Port 0A bit b1
P0A0	FLG	0.70H.0	R/W	Port 0A bit b0
P0B3	FLG	0.71H.3	R/W	Port 0B bit b3
P0B2	FLG	0.71H.2	R/W	Port 0B bit b2
P0B1	FLG	0.71H.1	R/W	Port 0B bit b1
P0B0	FLG	0.71H.0	R/W	Port 0B bit b0
P0C3	FLG	0.72H.3	R/W	Port 0C bit b3
P0C2	FLG	0.72H.2	R/W	Port 0C bit b2
P0C1	FLG	0.72H.1	R/W	Port 0C bit b1
P0C0	FLG	0.72H.0	R/W	Port 0C bit b0
P0D3	FLG	0.73H.3	R/W	Port 0D bit b3
P0D2	FLG	0.73H.2	R/W	Port 0D bit b2
P0D1	FLG	0.73H.1	R/W	Port 0D bit b1
P0D0	FLG	0.73H.0	R/W	Port 0D bit b0
P0E3	FLG	0.6EH.3	R/W	Port 0E bit b3
P0E2	FLG	0.6EH.2	R/W	Port 0E bit b2
P0E1	FLG	0.6EH.1	R/W	Port 0E bit b1
P0E0	FLG	0.6EH.0	R/W	Port 0E bit b0
P0F1	FLG	0.6FH.1	R	Port 0F bit b1
P0F0	FLG	0.6FH.0	R	Port 0F bit b0

3.4 Register File

Symbol name	Attribute	Value	Read/ Write	Description
SP	MEM	0.81H	R/W	Stack pointer
SIOTS	FLG	0.82H.3	R/W	Serial interface (SIO) start flag
SIOHIZ	FLG	0.82H.2	R/W	$P0D_1$/SO pin function selector flag
SIOCK1	FLG	0.82H.1	R/W	SIO source clock select flag bit 1
SIOCK0	FLG	0.82H.0	R/W	SIO source clock select flag bit 0
WDTRES	FLG	0.83H.3	R/W	Watchdog timer reset flag
WDTEN	FLG	0.83H.0	R/W	Watchdog timer enable flag
TM1OSEL	FLG	0.8BH.3	R/W	$P0D_3$/$\overline{TM1OUT}$ pin function selector flag
SIOEN	FLG	0.8BH.0	R/W	SIO enable flag
P0EGPU	FLG	0.8CH.2	R/W	P0E group pull-up select flag (pull-up = 1)
P0BGPU	FLG	0.8CH.1	R/W	P0B group pull-up select flag (pull-up = 1)
P0AGPU	FLG	0.8CH.0	R/W	P0A group pull-up select flag (pull-up = 1)
P0DBPU3	FLG	0.8DH.3	R/W	$P0D_3$ pull-up select flag (pull-up = 1)
P0DBPU2	FLG	0.8DH.2	R/W	$P0D_2$ pull-up select flag (pull-up = 1)
P0DBPU1	FLG	0.8DH.1	R/W	$P0D_1$ pull-up select flag (pull-up = 1)
P0DBPU0	FLG	0.8DH.0	R/W	$P0D_0$ pull-up select flag (pull-up = 1)
INT	FLG	0.8FH.0	R	INT pin status flag
TM0EN	FLG	0.91H.3	R/W	Timer 0 enable flag
TM0RES	FLG	0.91H.2	R/W	Timer 0 reset flag
TM0CK1	FLG	0.91H.1	R/W	Timer 0 source clock select flag bit 1
TM0CK0	FLG	0.91H.0	R/W	Timer 0 source clock select flag bit 0
TM1EN	FLG	0.92H.3	R/W	Timer 1 enable flag
TM1RES	FLG	0.92H.2	R/W	Timer 1 reset flag
TM1CK1	FLG	0.92H.1	R/W	Timer 1 source clock select flag bit 1
TM1CK0	FLG	0.92H.0	R/W	Timer 1 source clock select flag bit 0
BTMISEL	FLG	0.93H.3	R/W	BTM interrupt request clock selector flag
BTMRES	FLG	0.93H.2	R/W	BTM reset flag
BTMCK1	FLG	0.93H.1	R/W	BTM source clock select flag bit 1
BTMCK0	FLG	0.93H.0	R/W	BTM source clock select flag bit 0
P0C3IDI	FLG	0.9BH.3	R/W	$P0C_3$ input port inhibit flag (ADC_3/$P0C_3$ selection)
P0C2IDI	FLG	0.9BH.2	R/W	$P0C_2$ input port inhibit flag (ADC_2/$P0C_2$ selection)
P0C1IDI	FLG	0.9BH.1	R/W	$P0C_1$ input port inhibit flag (ADC_1/$P0C_1$ selection)
P0C0IDI	FLG	0.9BH.0	R/W	$P0C_0$ input port inhibit flag (ADC_0/$P0C_0$ selection)

Symbol name	Attribute	Value	Read/ Write	Description
P0CBIO3	FLG	0.9CH.3	R/W	$P0C_3$ input/output select flag (1 = output port)
P0CBIO2	FLG	0.9CH.2	R/W	$P0C_2$ input/output select flag (1 = output port)
P0CBIO1	FLG	0.9CH.1	R/W	$P0C_1$ input/output select flag (1 = output port)
P0CBIO0	FLG	0.9CH.0	R/W	$P0C_0$ input/output select flag (1 = output port)
IEGMD1	FLG	0.9FH.1	R/W	INT pin edge detect selector flag bit 1
IEGMD0	FLG	0.9FH.0	R/W	INT pin edge detect selector flag bit 0

3.5 Peripheral Hardware Register

Symbol name	Attribute	Value	Read/ Write	Description
SIOSFR	DAT	01H	R/W	Shift register's peripheral address
TM0M	DAT	02H	W	Timer 0 modulo register's peripheral address
TM1M	DAT	03H	W	Timer 1 modulo register's peripheral address
ADCR	DAT	04H	R/W	A/D converter data register's peripheral address
TM0TM1C	DAT	45H	R	Timer 0 timer 1 count register's peripheral address
AR	DAT	40H	R/W	Address register's peripheral address for GET/PUT/PUSH/CALL/BR/MOVT /INC instructions

3.6 Other Symbols

Symbol name	Attribute	Value	Description
DBF	DAT	0FH	Fixed operand value for PUT, GET, and MOVT instructions
IX	DAT	01H	Fixed operand value for INC instruction

3.7 List of Reserved Words (in Alphabetical Order)

3.7.1 Instructions and pseudo-instructions

ADD	EXTRN	NIBBLE6	SET1
ADDC	FLG	NIBBLE6V	SET2
AND	GET	NIBBLE7	SET3
BANK0	GLOBAL	NIBBLE7V	SET4
BELOW	HALT	NIBBLE8	SFCOND
BR	IF	NIBBLE8V	SKE
C14344	IFCHAR	NOBMAC	SKF
C4444	IFNCHAR	NOLIST	SKF1
CALL	INC	NOMAC	SKF2
CASE	INCLUDE	NOP	SKF3
CLR1	INITFLG	NOT1	SKF4
CLR2	IRP	NOT2	SKGE
CLR3	LAB	NOT3	SKLT
CLR4	LBMAC	NOT4	SKNE
CSEG	LD	OBMAC	SKT
DAT	LFCOND	OMAC	SKT1
DB	LIST	OR	SKT2
DI	LITERAL	ORG	SKT3
DW	LMAC	OTHER	SKT4
EI	MACRO	PEEK	SMAC
EJECT	MEM	POKE	ST
ELSE	MOV	POP	STOP
END	MOVT	PUBLIC	SUB
ENDCASE	NIBBLE	PURGE	SUBC
ENDIF	NIBBLE1	PUSH	SUMMARY
ENDIFC	NIBBLE2	PUT	TAG
ENDIFNC	NIBBLE2V	REPT	TITLE
ENDM	NIBBLE3	RET	XOR
ENDP	NIBBLE3V	RETI	ZZZERROR
ENDR	NIBBLE4	RETSK	ZZZMCHK
EOF	NIBBLE4V	RORC	ZZZMSG
EXIT	NIBBLE5	SBMAC	ZZZOPT
EXITR	NIBBLE5V	SET	

3.7.2 Registers and flags

ADCCH0
ADCCH1
ADCCH2
ADCCH3
ADCCMP
ADCEND
ADCR
ADCSOFT
ADCSTRT
AR
AR0
AR1
AR2
AR3
AR_EPA0
AR_EPA1
BANK
BCD
BTMCK0
BTMCK1
BTMISEL
BTMRES
CMP
CY
DBF
DBF0
DBF1
DBF2
DBF3
IEGMD0
IEGMD1
INT
IP
IPBTM
IPSI0
IPTM0
IPTM1
IRQ
IRQBTM
IRQSIO
IRQTM0
IRQTM1
IX
IXE
IXH
IXL
IXM
MPE
MPH
MPL
P0A0
P0A1
P0A2
P0A3
P0AGIO
P0AGPU
P0B0
P0B1
P0B2
P0B3
P0BGIO
P0BGPU
P0C0
P0C0IDI
P0C1
P0C1IDI
P0C2
P0C2IDI
P0C3
P0C3IDI
P0CBIO0
P0CBIO1
P0CBIO2
P0CBIO3
P0D0
P0D1
P0D2
P0D3
P0DBIO0
P0DBIO1
P0DBIO2
P0DBIO3
P0DBPU0
P0DBPU1
P0DBPU2
P0DBPU3
P0E0
P0E1
P0E2
P0E3
P0EGIO
P0EGPU
P0F0
P0F1
PSW
RPH
RPL
SIOCK0
SIOCK1
SIOEN
SIOHIZ
SIOSFR
SIOTS
SP
TM0CK0
TM0CK1
TM0EN
TM0M
TM0OSEL
TM0RES
TM0TM1C
TM1CK0
TM1CK1
TM1EN
TM1M
TM1OSEL
TM1RES
WDTEN
WDTRES
WR
Z
ZZZ0
ZZZ1
ZZZ2
ZZZ3
ZZZ4
ZZZ5
ZZZ6
ZZZ7
ZZZ8
ZZZ9
ZZZDEVID
ZZZEPA
ZZZLSARG
ZZZPRINT
ZZZSKIP
ZZZSYDOC
ZZZALBMAC
ZZZALMAC
ZZZARGC
ZZZLINE

3.7.3 Mask options

ENDOP
NOUSE
OPEN
OPTION
OPTRES
OPTP0F
OPTINT
OPTPOC
PULLUP
USE

CHAPTER 4 MASK OPTIONS

The μPD17145 subseries products include the following mask options.

- On-chip pull-up resistor for $\overline{\text{RESET}}$ pin
- On-chip pull-up resistor for $P0F_1$ and $P0F_0$ pins
- On-chip pull-up resistor for INT pin
- On-chip POC circuit

When creating programs, use mask option definition pseudo-instructions to make all of the above mask option specifications as required for source programs.

Device files include files named with the extension ".DEV" or ".OPT".

- File with ".DEV" extension: device file (main file)
- File with ".OPT" extension: option file

Option files are required when specifying mask options.
When using these files, both types of files must be stored in the same directory.

Remark If the full path name of the device file (.DEV file) is specified for the sequence file (.SEQ file), the assembler searches an option file (.OPT file) from the same directory as the device file.

4.1 Specification of Mask Options

Use the following pseudo-instructions to specify mask options in assembler source programs.

- OPTION pseudo-instruction, ENDOP pseudo-instruction
- Mask option definition pseudo-instructions

(1) OPTION pseudo-instruction, ENDOP pseudo-instruction

These pseudo-instructions specify the range for specifying mask options (mask option definition block). Use mask option definition pseudo-instructions to specify mask options within the range set by the OPTION and ENDOP pseudo-instructions.

Coding format

Symbol column	Mnemonic column	Operand column	Comment column
[Label:]	OPTION		[; comments]
	⋮		
	ENDOP		

(2) Mask option definition pseudo-instructions

Table 4-1. List of Mask Option Definition Pseudo-instructions

Option	Definition pseudo-instruction and format	Operand	Meaning of definition
On-chip pull-up resistor for $\overline{\text{RESET}}$ pin	OPTRES <operand>	OPEN	Absent
		PULLUP	Present
On-chip pull-up resistor for $P0F_1$ and $P0F_0$ pins	OPTP0F <operand 1>, <operand 2> Note	OPEN	Absent
		PULLUP	Present
On-chip pull-up resistor for INT pin	OPTINT <operand>	OPEN	Absent
		PULLUP	Present
On-chip POC circuit	OPTPOC <operand>	NOUSE	Not used
		USE	Used

Note <operand 1> specifies the mask option for the $P0F_1$ pin and <operand 2> specifies the mask option for the $P0F_0$ pin.

(3) Mask option coding examples

; Mask option coding example for the μPD17145

```
MASK_OPTION:

    OPTION                  ; Start of mask option definition block
    OPTRES   PULLUP         ; On-chip pull-up resistor is present for RESET pin
    OPTP0F   PULLUP, OPEN   ; P0F1 pin has on-chip pull-up resistor and the P0F0 pin
                              is open (external pull-up).
    OPTINT   PULLUP         ; On-chip pull-up resistor is present for INT pin
    OPTPOC   NOUSE          ; Does not use on-chip POC circuit
    ENDOP                   ; End of mask option definition block
```

Cautions 1. Enter the specified number of parameters for operands. If there are too many or too few parameters, an error will occur. "Operand count error"

2. Use symbols defined as reserved words for operands. If a symbol having a format or value other than the specified symbol is entered, an error will result. "Invalid value for XXXXX terminal"

3. Option files (.OPT files) must be stored in the same directory as device files (.DEV files). If they are not in the same directory, an error will result and the mask option will not be set. "D17145.OPT : LIB : No such file" When this error occurs, an error will also occur during linking due to the lack of mask option specification. "No option definition block"

CHAPTER 5 LOAD MODULE FILE FORMAT

HEX-format load module files output by the assembler (AS17K) have two types of output file formats: ICE files and PRO files.

These two types of files must be used according to the application being used. Besides having a user program area, they also have an assembly environment data area, an in-circuit emulator operating environment data area, and other areas.

(1) HEX-format load module file format

The data contained in HEX-format load module files output by the assembler is output in the following format.

[Example of HEX-format load module file format]

```
: 10 0002 00 2B41000BFC80F...3A20 EC
| |  |  |                 |      |
(a) (b) (c) (d)           (e)    (f)

: 00 0000 01 FF
| |  |  | |
(a) (b) (c) (d) (f)
```

(a) Record mark
This indicates the start of a record.

(b) Code amount (2 digits)
This indicates the amount of code (byte data) stored in a record. This value is expressed as a hexadecimal number, with a maximum value of 10H (16 units). The value for the end record is 00H.

(c) Address (4 digits)
This indicates the start address of the code shown in a record. The value for the end record is 0000H, and has no relation to the address.

(d) Record type (2 digits)
A value of 00H indicates "data record" as the record type and a value of 01H indicates "end record."

(e) Code (up to 32 digits (16 bytes))
Code is output to this field one byte at a time, up to 16 bytes.

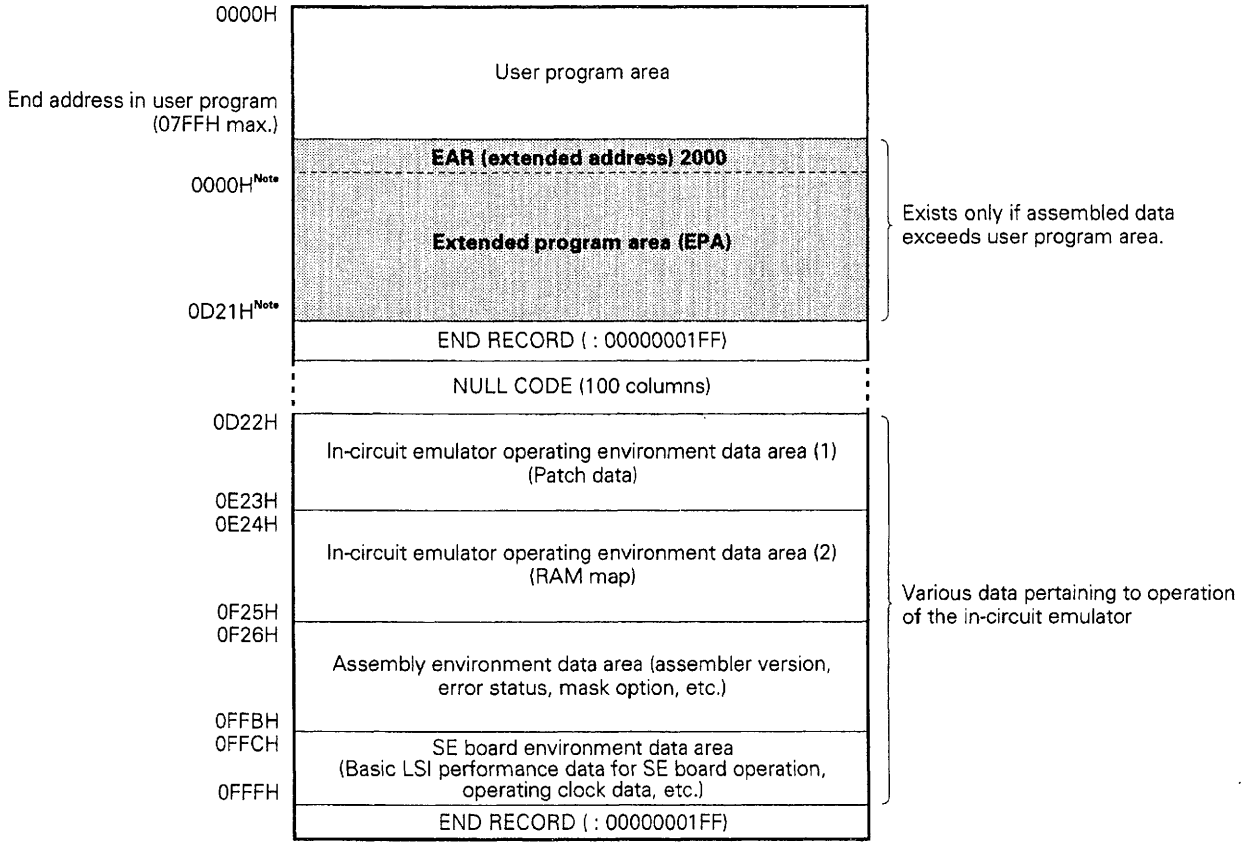
(f) Check sum (2 digits)
Data from fields (b), (c), (d), (e), and (f) is output to this field (with even parity) as byte data with a least significant bit value of 00H based on byte-unit sums.

(2) ICE files

ICE files are output as HEX-format files exclusive to the in-circuit emulator (IE-17K or IE-17K-ET) output by the AS17K assembler. Figure 5-1 shows the output formats when the source file is assembled by using the AS17145, 17147, or 17149.

Figure 5-1. ICE File Format (1/3)

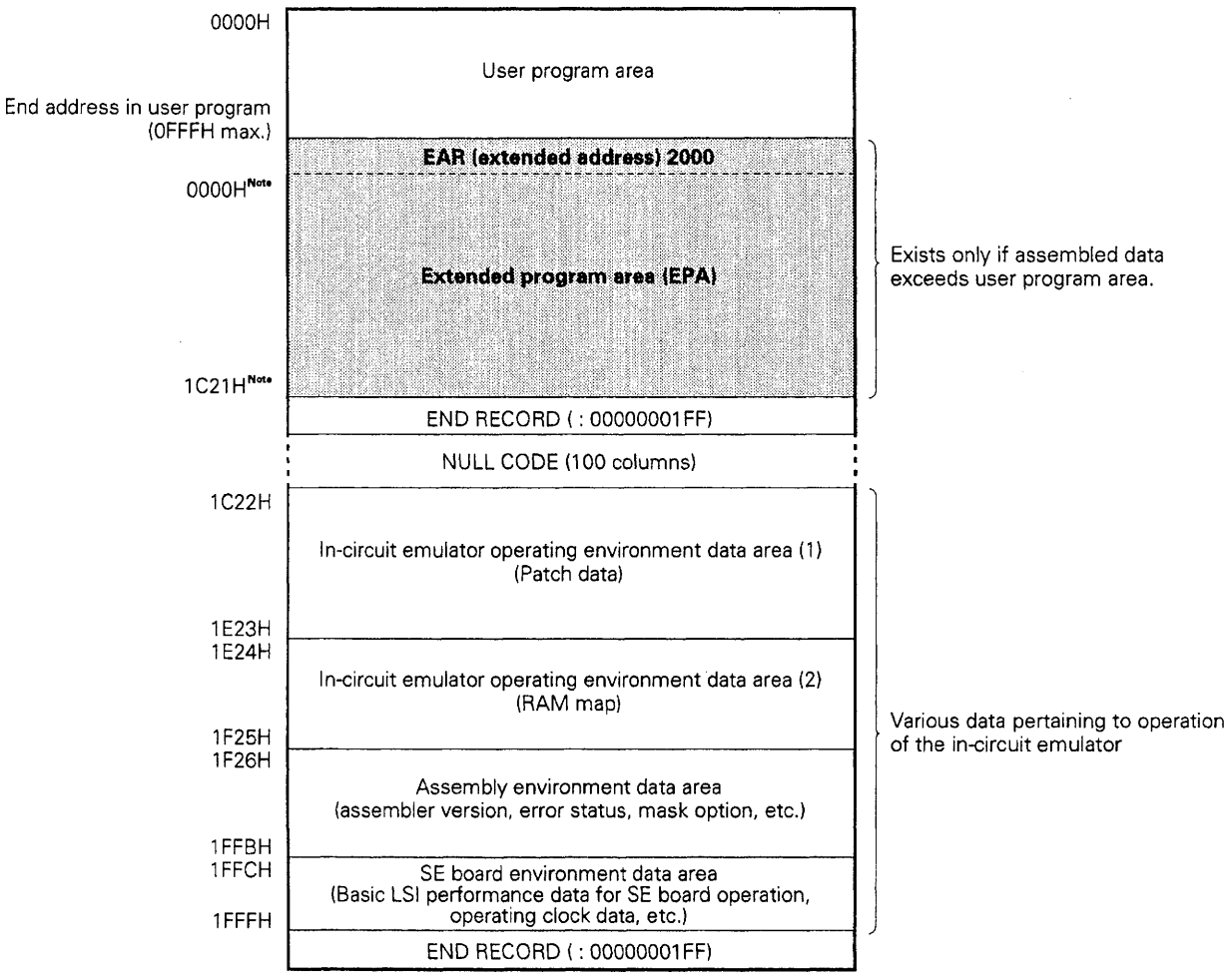
(a) When using AS17145



Note Range for in-circuit emulator is 8000H to 8D21H.

Figure 5-1. ICE File Format (2/3)

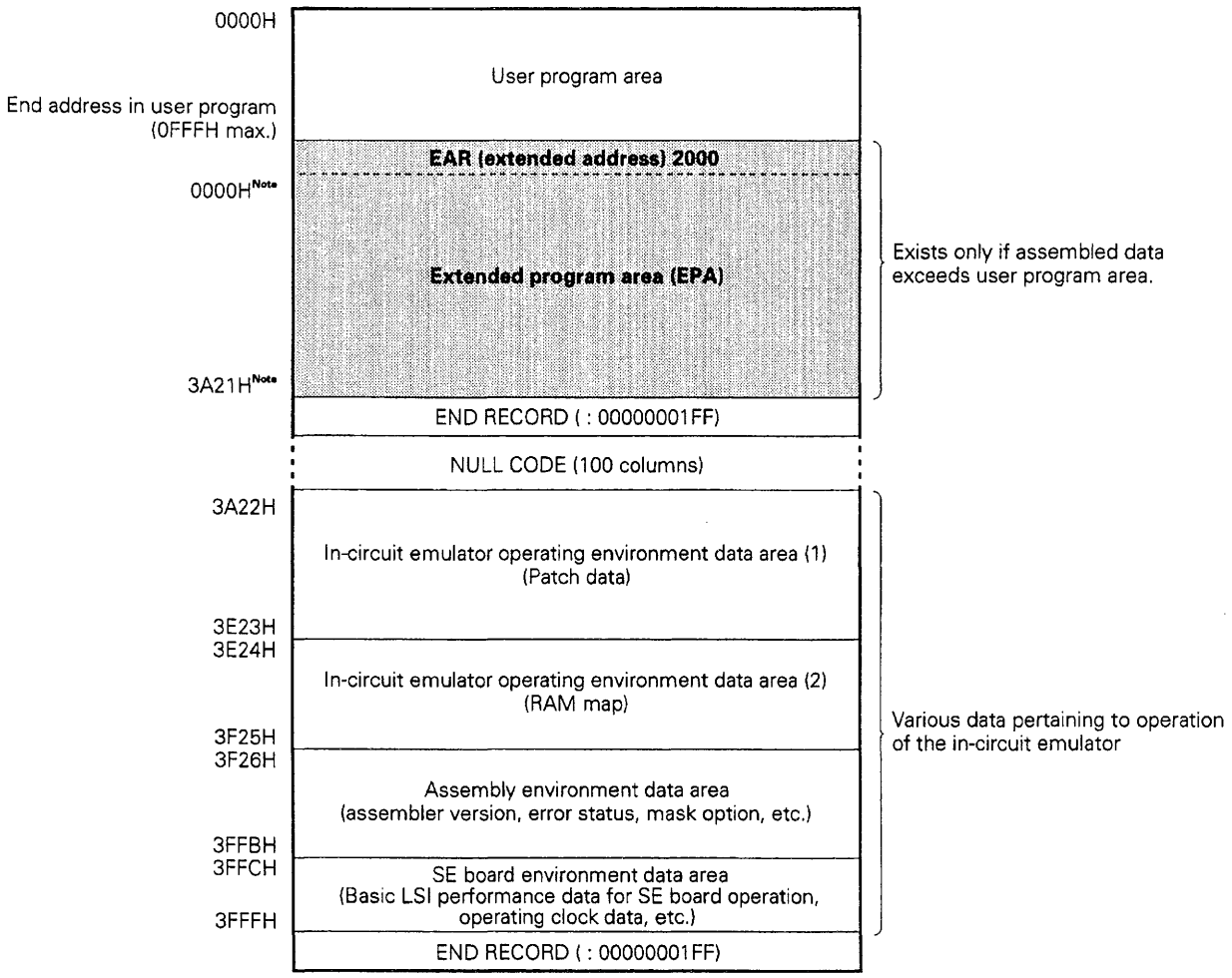
(b) When using AS17147



Note Range for in-circuit emulator is 8000H to 9C21H.

Figure 5-1. ICE File Format (3/3)

(c) When using AS17149



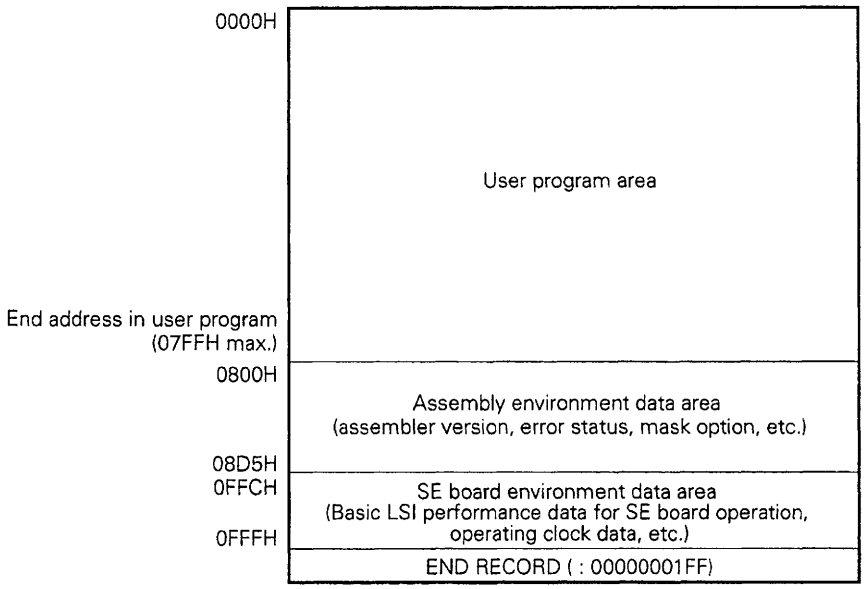
Note Range for in-circuit emulator is 8000H to BA21H.

(3) PRO files

PRO files are output as HEX-format files exclusive to the mask orders and PROM and single-write PROM products (μPD17P149) for stand-alone SE board evaluations that are output by the AS17K assembler. These are output when "/PRO" is specified as an assembly option during assembly. Figure 5-2 shows the output formats when the source file is assembled by using the AS17145, 17147, or 17149.

Figure 5-2. PRO File Format (1/3)

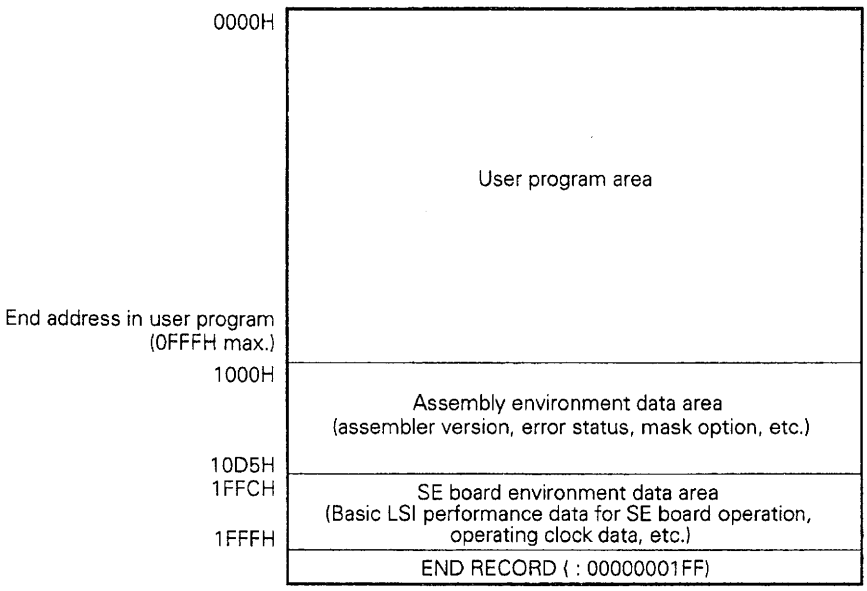
(a) When using AS17145



Remarks
1. The assembly environment data area also includes the mask option data used for creating LSI masks. Consequently, when making a mask order, mask option-related information is not required.
2. The range of 08D6H to 0FFBH does not exist in PRO files.

Figure 5-2. PRO File Format (2/3)

(b) When using AS17147

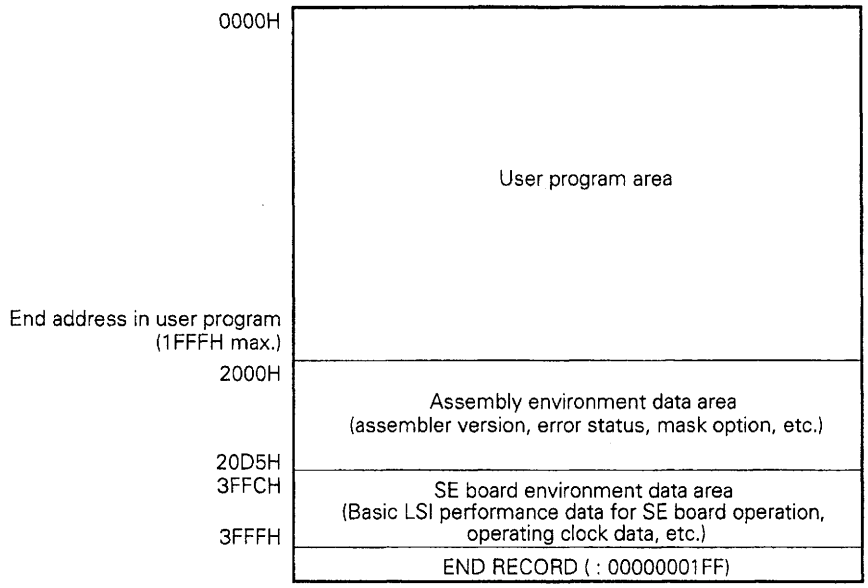


Remarks

1. The assembly environment data area also includes the mask option data used for creating LSI masks. Consequently, when making a mask order, mask option-related information is not required.
2. The range of 10D6H to 1FFBH does not exist in PRO files.

Figure 5-2. PRO File Format (3/3)

(c) When using AS17149



Remarks
1. The assembly environment data area also includes the mask option data used for creating LSI masks. Consequently, when making a mask order, mask option-related information is not required.
2. The range of 20D6H to 3FFBH does not exist in PRO files.

(4) Changes in load module files

Even when there are no changes in the source file, reassembling them may result in changes in part (i.e., the assembly environment data area) of the load module file. This is because the data in the assembly environment data area includes the source file creation date.

Table 5-1. Items in Assembly Environment Data Area That Are Subject to Change (1/3)

(a) When using AS17145

Item	Address	
	ICE file	PRO file
Program name Note 1 (character string of up to 64 bytes, specified by assembly option /'PROG='.)	0F26H-0F65H	0800H-083FH
Mask option data	0F66H-0F69H	0840H-0843H
SIMPLEHOST data	0FADH	0887H
Error or warning status	0FB0H	088AH
File creation date (YY/MM/DD) and time Note 2	0FBEH-0FC7H	0898H-08A1H
★ Device name	0FC8H-0FD7H	08A2H-08B1H
Device file version	0FDCH, 0FDDH	08B6H, 08B7H
Assembler version	0FDEH-0FE1H	08B8H-08BBH

★ **Notes** 1. When using the RA17K (under development), the program name area consists of the following 32 bytes:

ICE file: 0F26H-0F45H

PRO file: 0800H-081FH

The remaining 32 bytes are used by the RA17K a system reserved area.

★ 2. The most latest data of creation date/time among source files or sequence files are written.

Caution Do not make direct changes to load module files.

Load module files do not match with other file histories, and can cause bugs to occur. Always reassemble when revising load module files.

Table 5-1. Items in Assembly Environment Data Area That Are Subject to Change (2/3)

(b) When using AS17147

Item	Address	
	ICE file	PRO file
Program name [Note 1] (character string of up to 64 bytes, specified by assembly option /'PROG='.)	1F26H-1F65H	1000H-103FH
Mask option data	1F66H-1F69H	1040H-1043H
SIMPLEHOST data	1FADH	1087H
Error or warning status	1FB0H	108AH
File creation date (YY/MM/DD) and time [Note 2]	1FBEH-1FC7H	1098H-10A1H
Device name	1FC8H-1FD7H	10A2H-10B1H ★
Device file version	1FDCH, 1FDDH	10B6H, 10B7H
Assembler version	1FDEH-1FE1H	10B8H-10BBH

Notes 1. When using the RA17K (under development), the program name area consists of the following 32 bytes: ★
ICE file: 1F26H-1F45H
PRO file: 1000H-101FH
The remaining 32 bytes are used by the RA17K a system reserved area.

2. The most latest data of creation date/time among source files or sequence files are written. ★

Caution Do not make direct changes to load module files.
Load module files do not match with other file histories, and can cause bugs to occur. Always reassemble when revising load module files.

Table 5-1. Items in Assembly Environment Data Area That Are Subject to Change (3/3)

(c) When using AS17149

Item	Address	
	ICE file	PRO file
Program name [Note 1] (character string of up to 64 bytes, specified by assembly option /'PROG='.)	3F26H-3F65H	2000H-203FH
Mask option data	3F66H-3F69H	2040H-2043H
SIMPLEHOST data	3FADH	2087H
Error or warning status	3FB0H	208AH
File creation date (YY/MM/DD) and time [Note 2]	3FBEH-3FC7H	2098H-20A1H
★ Device name	3FC8H-3FD7H	20A2H-20B1H
Device file version	3FDCH, 3FDDH	20B6H, 20B7H
Assembler version	3FDEH-3FE1H	20B8H-20BBH

★ **Notes** **1.** When using the RA17K (under development), the program name area consists of the following 32 bytes:

ICE file: 3F26H-3F45H

PRO file: 2000H-201FH

The remaining 32 bytes are used by the RA17K a system reserved area.

★ **2.** The most latest data of creation date/time among source files or sequence files are written.

Caution Do not make direct changes to load module files.
Load module files do not match with other file histories, and can cause bugs to occur. Always reassemble when revising load module files.