Appendix

A-1 HC boot and PC boot

(1) Overview

The PX-16 has two types of system boots: the HC boot and the PC boot. The HC boot boots up MS-DOS from the system ROM (ROM0) provided as standard equipment, and supports all PX-16 functions under operating system. However, if ROM1-3 is a boot ROM, it will boot from that ROM instead. The PC boot boots the system from a floppy disk when the system environment is equivalent to the EPSON PCe or IBM PC/XT (i.e. display of 640x200 dots minimum, and at least one FDD), and results in a system compatible with these desktop computers. When the PX-16 is PC booted, all EPSON PCe software will run normally.

The PX-16 system ROM provided as standard equipment stores BIOS, MS-DOS, device drivers, GW-BASIC and utilities. These software packages are revised and expanded versions of the software packages used on the EPSON PCe.

(2) Differences in BIOS level

The PX-16 ROM BIOS is based on the EPSON PCe ROM BIOS with specific extended functions added. This assures compatibility with the EPSON PCe, and support of all devices used with the EPSON PCe. It also supports additional devices, such as the cartridge 1 interface and the touch keyboard. However, these devices cannot be used under the PC boot system environment to maintain EPSON PCe compatibility. These extended devices are only supported at the BIOS level for HC boot.

① BIOS interrupt vectors

The PX-16 BIOS calls are compatible with the EPSON PCe for INT00H through INT1FH, but they will support the extended devices in HC boot. INT70H through INT7FH are unique PX-16 BIOS functions not found in the PCe.

The BIOS supported function unique to the HC boot system environment are listed below. For details on these BIOS calls, refer to Section 3.5.

INT00H~INT1FH (Extended support functions in HC boot)

INT05H: Hard copy to cartridge printer
INT09H: Keyboard selection of international character set
INT0BH: RS-232C transmit interrupt
INT0CH: RS-232C receive interrupt
INT10H: LCD40 and touch keyboard display
INT13H: RAM disk, ROM disk and cartridge 1
INT14H: RS-232C receive interrupt
INT17H: Support cartridge printer
INT18H: PX-16 extended functions
INT1AH: Alarm function
INT1FH: 80H~FFH character font data pointer

INT70H~INT7FH (Only enabled in HC boot)
INT70H: Generated by interrupt from slave CPU. May be from INT78H through INT 7BH depending on cause.
INT71H: UART (cartridge 1)
INT72H: IRET (cartridge 1)
INT73H: IRET (cartridge 1)
INT74H: IRET (used for BARCODE.COM)
INT75H: Touch keyboard beep control, sound timer control, cursor blink control
INT76H: IRET (RING, etc.)
INT77H: Touch keyboard
INT78H: Auto-power off counter
INT79H: IRET (alarm hook)
INT7AH: Power off through power switch
INT7BH: Power failure processing
INT7CH: Disk resume processing
INT7DH: Power control
INT7EH: Reserved (for Japanese language processing)
INT7FH: Reserved

② Battery conservation processing
The PX-16 supports a number of functions to conserve battery power, but these are only enabled in HC boot.

Function	PC boot	HC boot
Power failure processing function	.When battery low detected (4.7V or less), the power LED is flashed, and the system forcibly powered off when battery power fails .Warm start at next power on	.When low battery detected (4.7V or less), the power LED is flashed, and the system powered off if no action taken within two minutes .Next power on is resume start .Time limit until power off can be set .Uses INT 7BH
Auto power off function	———	.If there is no key input interrupt for a fixed period of time (idle time), the system automatically powers off .The next power on is a resume start .The default setting is auto power off disable. This can be changed to enable, and the time limit set .Time limit to power off is set with INT 78H
Standby mode when using TF-16	———	.The TF-16 is set to standby mode if there is no access by a BIOS call for 30 seconds .The next access will set it to active mode .Only supported when TF-16 used through cartridge 1 (when FDC is TC8566AF)

(3) Differences in MS-DOS level

① Boot type

The basic differences between HC boot and PC boot are indicated below.

PC boot	HC boot
.System (MS-DOS) is searched in order from the first FD and then the first HD, and booted from the located drive .The current drive is the boot drive	.System is searched in order ROM1 > ROM2 > ROM3 > ROM0, and booted from located drive .The current drive is the one with CONFIG.SYS, and CONFIG.SYS is searched in the following order: First FD > HD > cartridge 1 > ROM0 > ROM1 > ROM2 > ROM3 .If there is no CONFIG.SYS file drive A: is used

② Start type

The start methods supported by the HC boot and the PC boot are given below. For details on these start methods, refer to Section 3.3.

Start type	Description	PC boot	HC boot
Cold	All reset	o	o
Warm	All reset except system area and RAM disk	o	o
Resume	Return to status immediately before power off	x	o
Alarm	Start at time specified in RTC	x	o
Ring	Start to external signal, such as through modem	x	o

③ Device driver start sequence
HC boot

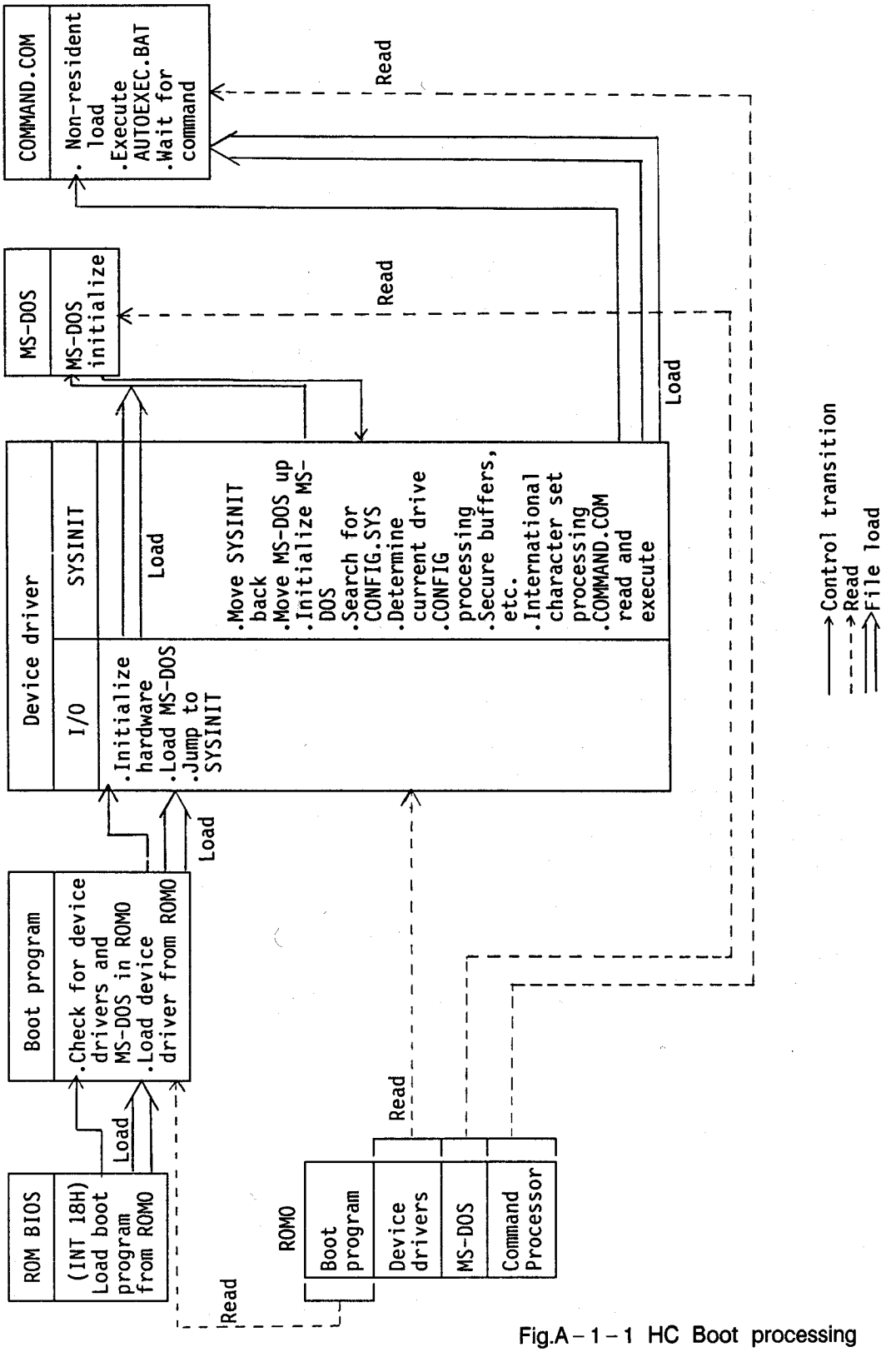


Fig.A－1－1 HC Boot processing

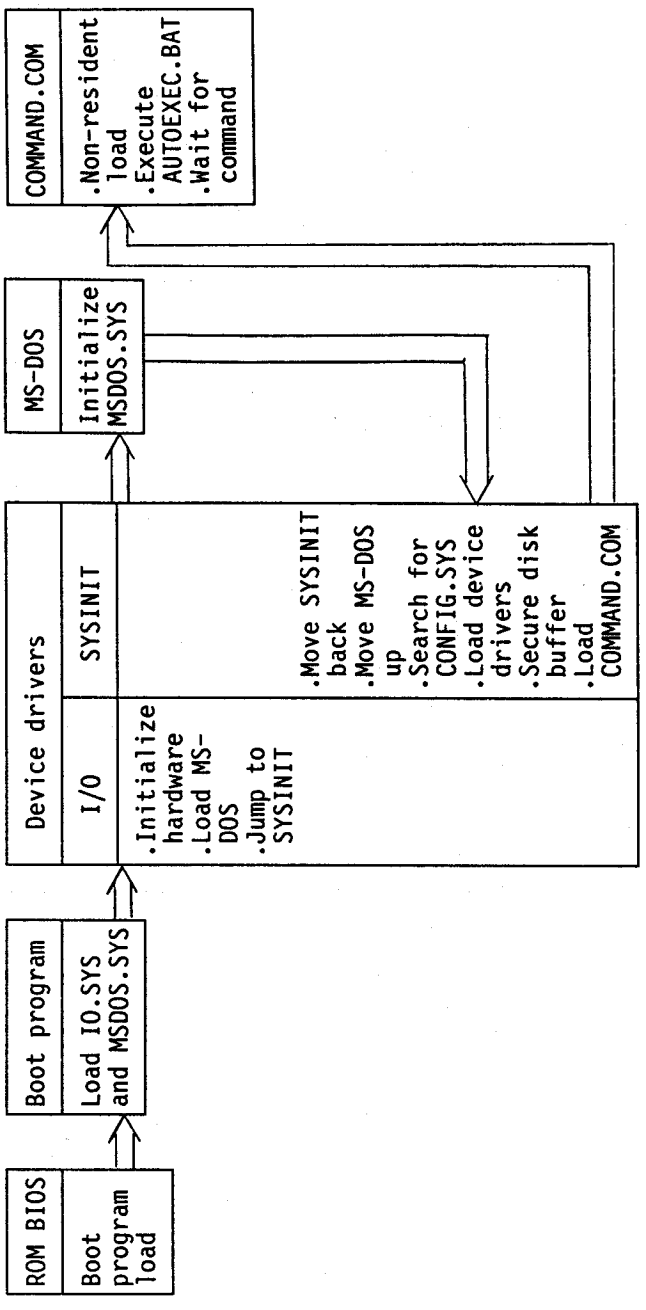


Fig.A－1－2 PC Boot processing

④ Devices

The devices supported in the HC and PC boot modes are listed below.

Character /block	Device name /drive number	Description	PC boot	HC boot
Character devices	AUX	Auxiliary I/O (serial port)	o	o
	CON	Console (keyboard,monitor)	o	o
	PRN	Printer	o	o
	NUL	Dummy I/O	o	o
	LPT1	Line printer 1	o	o
	LPT2	Line printer 2	o	o
	LPT3	Line printer 3	o	o
	LPT4	Cartridge printer	x	o
	COM1	Communications port 1	o	o
	COM2	Communications port 2	o	o
	COM3	Communications port 3	x	o
	SCRN1	Screen 1 (Cartridge 2)	x	o
	SCRN2	Screen 2 (Touch pannel)	x	o
	CLOCK$	Timer	o	o *
Block device	A to Z	FDD 1	o	o *
		FDD 2	o	o *
		HDD	o	o *
		RAM disk	x	o *
		ROM disk 1 to 4	x	o *
		Cartridge 1	x	o *

"*" : Drive numbers will very depending on whether system is FDD priority or RAM priority

⑤ International character sets

Function	PC boot	HC boot
International character set	Handled with CONFIG.SYS country command and KEYBxx command	Set with DIP switches
Japanese language	———	Supported with Japanese language driver (Optional ROM chips)

⑥ Other

a) Date set to RTC function
In PC boot mode, the date can be set through BIOS execution. In HC boot set is accomplished with the DATE and TIME commands.

b) Battery conservation processing
For a function 8 of DOS call (keyboard input without echo) in HC boot mode, if there is no keyboard input within the time limit the system will switch to HALT status. In PC boot, it will wait for input.

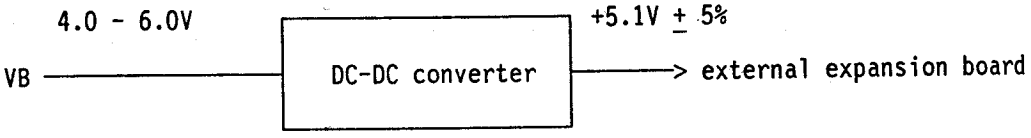
c) Line count for cartridge 2 display devices
In HC boot mode the cartridge 2 display device line count corresponds to the MS-DOS command level. In PC boot the line count is fixed.

A-2 Cautionary points regarding external expansion board design

The following points should be carefully noted during the design of an external expansion board to be connected to the expansion interface.

1. +5V current capacity (current consumption limits)
   The voltage supplied by the expansion interface +5V terminal to the external expansion board should be limited to 100μA or less. When the cartridge 2 interface and an optional external expansion board are to be used simultaneously, the board should be designed so that the current to the cartridge interface and expansion interface +5V terminals does not exceed 100μA. When a +5V current exceeding 100μA is required, the Vb terminal may be used. In this case, a CD-CD converter board should be added to the external expansion board.

   Example:



   Note 1: When printing H's on the cartridge printer, be cautious as voltage may drop as low as 4.0V.

   Ripple/spike

   Ripples/spikes are included in the +5V terminal. When setting an external expansion board which may have problems with ripples and spikes, add one of following series regulators.

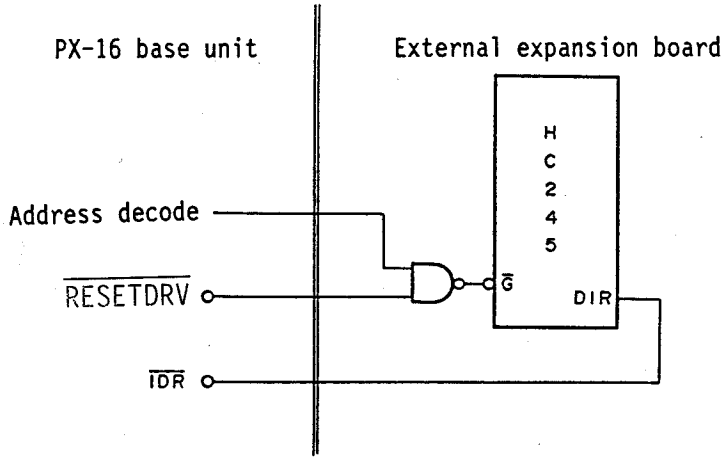
2. Vbk terminal current capacity
   Vbk terminal current capacity generally should be below 100μA. Increased Vbk consumption current decreases backup time.

3. The +7V, -7V terminals are used only by the CMOS RS-232C driver. No attempt should be made to use them for any other purpose (e.g., a bipolar RS-232C driver, or negative IC modem power, etc.).

4. Other cautionary points regarding circuit configurations

(1) When the base unit has been reset (when the $\overline{\text{RESET DRV}}$ terminal becomes active), ensure that data is not output from the data bus.

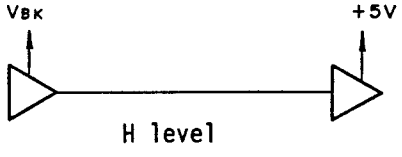
Example:



(2) As there will be situations in which IRQ and DRQ will be used by other devices, caution must be paid to ensure that no conflict occurs between signals.

IRQ2	Expansion interrupt signal
IRQ3	Expansion interface
IRQ4	RS-232C interface
IRQ5	Disk unit HDD
IRQ6	Disk unit or cartridge 2 floppy disk controller
IRQ7	Printer
DRQ2	Disk unit or cartridge 2 FDC
DRQ3	Disk unit HDD

(3) The circuit should be designed so that when the power is off, the H level output from the backed up circuit does not go to a circuit that is not backed up, as shown below.



A-3 Character Code Table

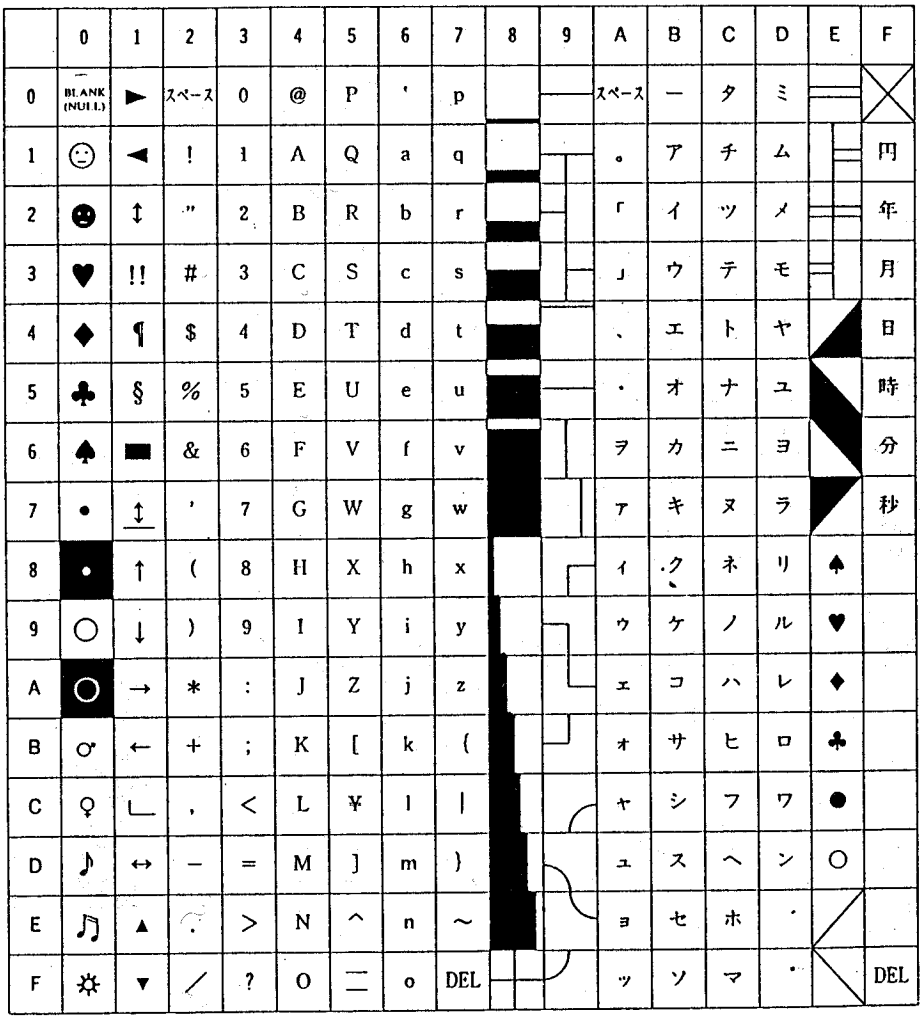
Standard (ASCII) character set

HEXA-DECIMAL VALUE	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	BLANK (NULL)	►	BLANK (SPACE)	0	@	P	‘	p	Ç	É	á	⁝⁝			α	≡
1	☺	◄	!	1	A	Q	a	q	ü	æ	í	⁝⁝			β	±
2	☻	↕	"	2	B	R	b	r	é	Æ	ó	⁝⁝			Γ	≥
3	♥	!!	#	3	C	S	c	s	â	ô	ú				π	≤
4	♦	¶	$	4	D	T	d	t	ä	ö	ñ				Σ	⌠
5	♣	§	%	5	E	U	e	u	à	ò	Ñ				σ	⌡
6	♠	▬	&	6	F	V	f	v	å	û	a				μ	÷
7	•	↨	’	7	G	W	g	w	ç	ù	o				τ	≈
8	◘	↑	(	8	H	X	h	x	ê	ÿ	¿				Φ	°
9	○	↓	)	9	I	Y	i	y	ë	Ö	⌐				Θ	•
A	◙	→	*	:	J	Z	j	z	è	Ü	¬				Ω	·
B	♂	←	+	;	K	[	k	{	ï	¢	½			■	δ	√
C	♀	∟	,	<	L	\	l	¦	î	£	¼			■	∞	n
D	♪	↔	—	=	M	]	m	}	ì	¥	¡			■	φ	2
E	♫	▲	.	>	N	^	n	~	Ä	Pt	«			■	∈	■
F	☼	▼	/	?	O	_	o	⌂	Å	ƒ	»				∩	BLANK 'FF'

Scandinavian character set

HEXA-DECIMAL VALUE	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	BLANK (NULL)	►	BLANK (SPACE)	0	@	P	‘	p	Ç	É	á	⁝⁝			α	≡
1	☺	◄	!	1	A	Q	a	q	ü	æ	í	⁝⁝			β	±
2	☻	↕	"	2	B	R	b	r	é	Æ	ó	⁝⁝			Γ	≥
3	♥	!!	#	3	C	S	c	s	â	ô	ú				π	≤
4	♦	¶	$	4	D	T	d	t	ä	ö	ñ				Σ	⌠
5	♣	§	%	5	E	U	e	u	à	ò	Ñ				σ	⌡
6	♠	▬	&	6	F	V	f	v	å	û	õ				μ	÷
7	•	↨	’	7	G	W	g	w	ç	ù	Õ				τ	≈
8	◘	↑	(	8	H	X	h	x	ê	ÿ	¿				Φ	°
9	○	↓	)	9	I	Y	i	y	ë	Ö	ã				Θ	•
A	◙	→	*	:	J	Z	j	z	è	Ü	Ã				Ω	·
B	♂	←	+	;	K	[	k	{	ï	ø	ℓ			■	δ	√
C	♀	∟	,	<	L	\	l	¦	î	£	ń			■	∞	n
D	♪	↔	—	=	M	]	m	}	ì	Ø	¡			■	φ	2
E	♫	▲	.	>	N	^	n	~	Ä	Ł	3			■	∈	■
F	☼	▼	/	?	O	_	o	▵	Å	ł	¤			■	∩	BLANK "FF"

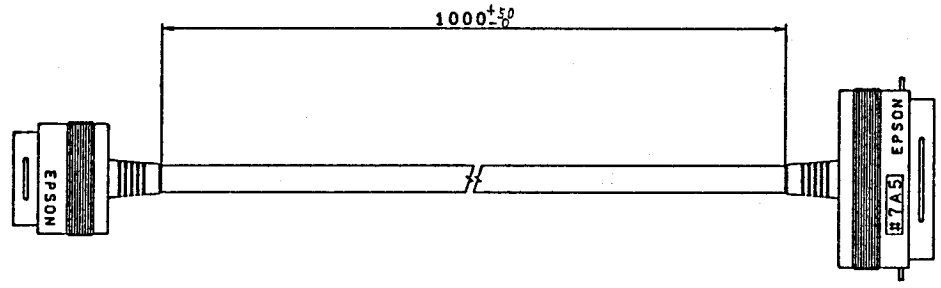
Kana (Japanese) character set



A-4 Optional Cables

Cable set #7A5

Printer I/F (Base Unit) ←--------→ Terminal Printer

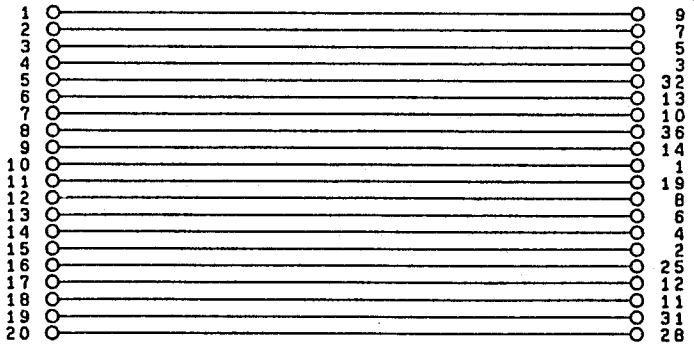


D57-20PC-X0HS
(DDK)

57JE-36P
(DDK)

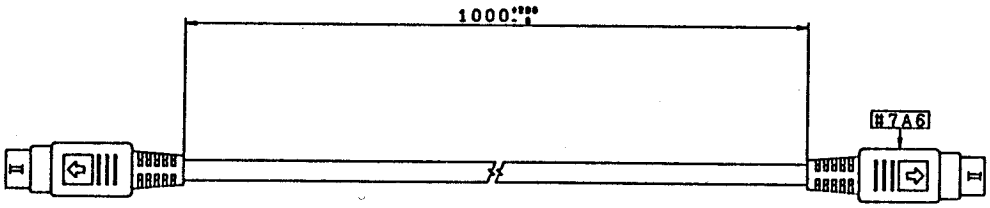
1	DATA7		11	GND	
2	DATA5		12	DATA6	
3	DATA3		13	DATA4	
4	DATA1		14	DATA2	
5	ERR		15	DATA0	
6	SLCT		16	GND	
7	ACK		17	PE	
8	SLCTIN		18	BUSY	
9	ATLF		19	INT	
10	STRB		20	GND	

1	STRB		19	GND	
2	DATA0		20		
3	DATA1		21		
4	DATA2		22		
5	DATA3		23		
6	DATA4		24		
7	DATA5		25	GND	
8	DATA6		26		
9	DATA7		27		
10	ACK		28	GND	
11	BUSY		29		
12	PE		30		
13	SLCT		31	INT	
14	ATLF		32	ERR	
15			33		
16			34		
17			35		
18			36	SLCTIN	



Cable set #7A6

RS-232C I/F (Base Unit) <------> RS-232C I/F (Base Unit)

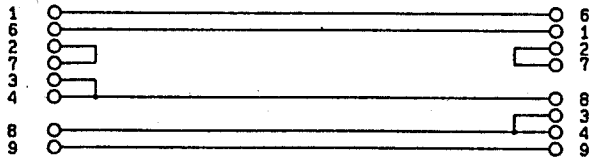


TCP8926
(Hoshi-Denki)

TCP8926
(Hoshi-Denki)

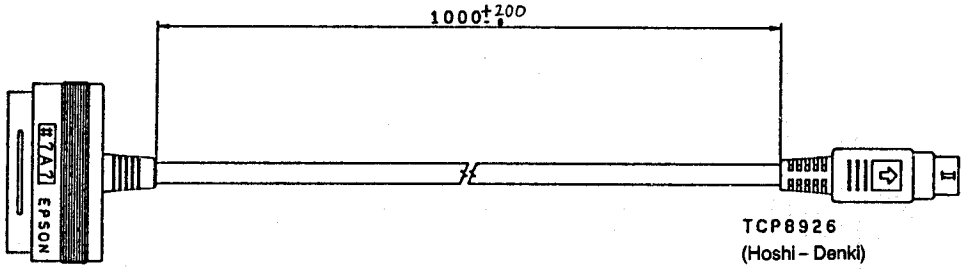
1	RXD	
2	CTS	
3	DSR	
4	DCD	
5		
6	TXD	
7	RTS	
8	DTR	
9	SG	
	CG	

1	RXD	
2	CTS	
3	DSR	
4	DCD	
5		
6	TXD	
7	RTS	
8	DTR	
9	SG	
	CG	



Cable set #7A7

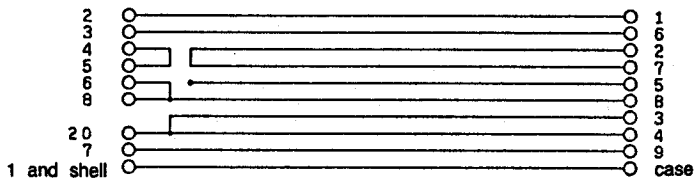
Epson PC, PC+, PCe / IBM PC, PC/XT <- - - - - -> RS-232C I/F (Base Unit)



17JE-S825-2C(9) female
(DDK)

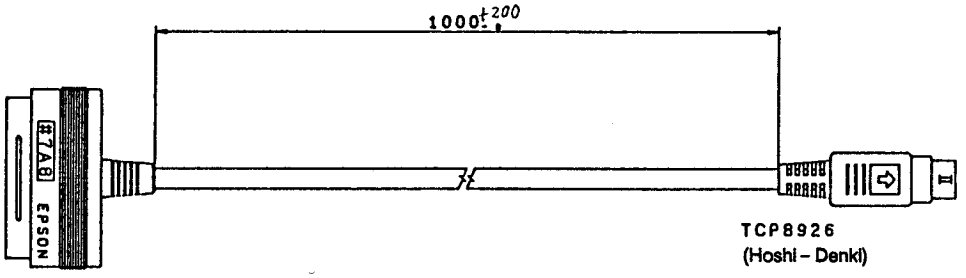
1	shield		14		
2	TXD		15		
3	RXD		16		
4	RTS		17		
5	CTS		18		
6	DSR		19		
7	SG		20	DTR	
8	DCD		21		
9			22		
10			23		
11			24		
12			25		
13					

1	RXD	
2	CTS	
3	DSR	
4	DCD	
5		
6	TXD	
7	RTS	
8	DTR	
9	SG	
	CG	



Cable set #7A8

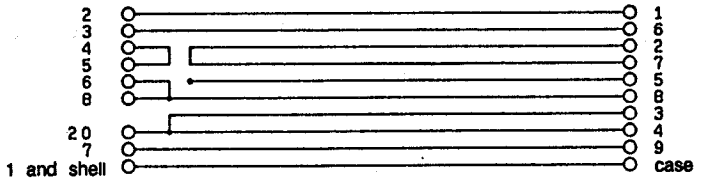
NEC PC-9801, IBM 5550 <------> RS-232C I/F (Base Unit)



17JE-PB25-2C(9) male
(DDK)

1	shield		14		
2	TXD		15		
3	RXD		16		
4	RTS		17		
5	CTS		18		
6	DSR		19		
7	SG		20	DTR	
8	DCD		21		
9			22		
10			23		
11			24		
12			25		
13					

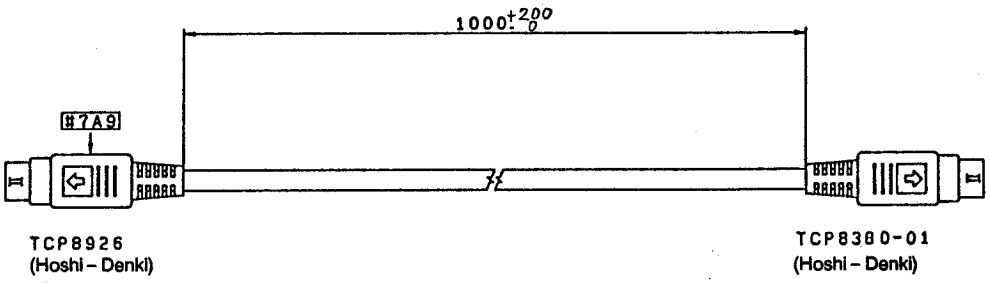
1	RXD	
2	CTS	
3	DSR	
4	DCD	
5		
6	TXD	
7	RTS	
8	DTR	
9	SG	
	CG	



Rev. B

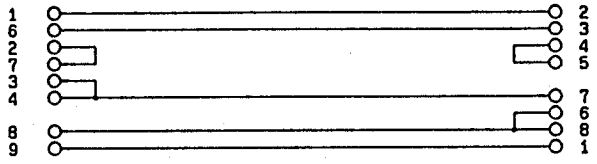
Cable set #7A9

RS – 232C I/F (Base Unit) < – – – – – > PX – 4/HC – 40 • 45,PX – 8/HC – 80 • 88
EHT – 10 • 10/2 / HC – 10 • 10II



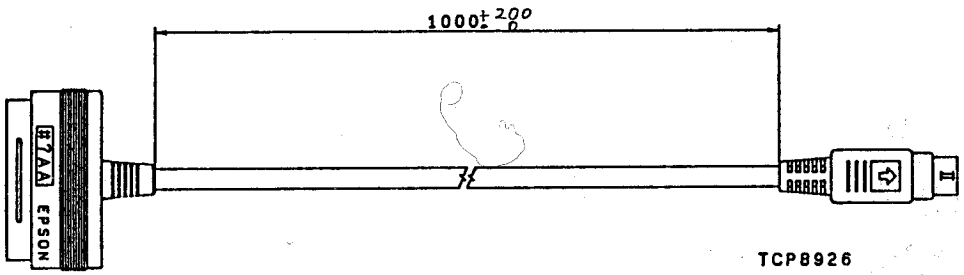
1	RXD	
2	CTS	
3	DSR	
4	DCD	
5		
6	TXD	
7	RTS	
8	DTR	
9	SG	
	CG	

1	SG	
2	TXD	
3	RXD	
4	RTS	
5	CTS	
6	DSR	
7	DTR	
8	DCD	
	CG	



Cable set #7AA

External Direct Modem <- - - - - -> RS-232C (Base Unit)

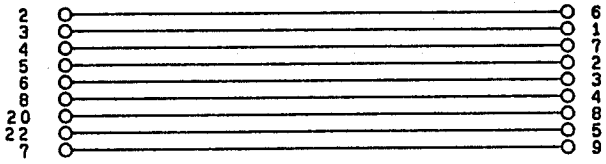


TCP8926
(Hoshi-Denki)

17JE-PB25-2C(9) male
(DDK)

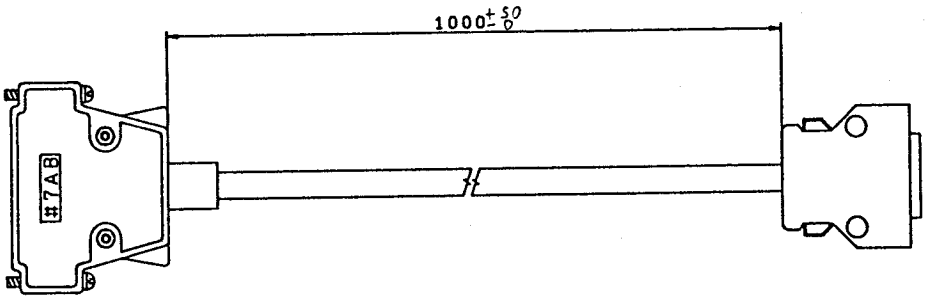
1			14		
2	TXD		15		
3	RXD		16		
4	RTS		17		
5	CTS		18		
6	DSR		19		
7	SG		20	DTR	
8	DCD		21		
9			22	CI	
10			23		
11			24		
12			25		
13					

1	RXD	
2	CTS	
3	DSR	
4	DCD	
5	CI	
6	TXD	
7	RTS	
8	DTR	
9	SG	
	CG	



Cable set #7AB

TF-16 / TF-160 <- - - - - -> FDD I/F (Option Cartridge2)



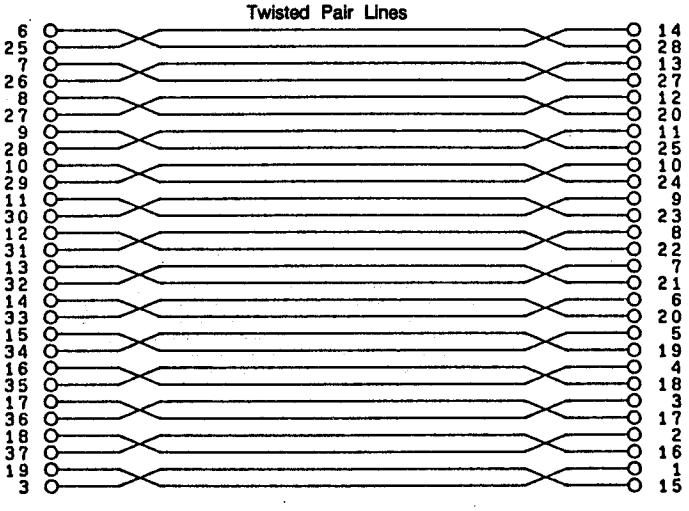
connecter : FDCD-37P (Hirose) male
cover : HDC-CTH (Hirose)

DX30-28-CV
(Hirose)

1			20		
2			21		
3	HIGH D		22		
4			23		
5			24		
6	*INDEX		25	GND	
7	*MON0		26	GND	
8	*DS1		27	GND	
9	*DS0		28	GND	
10	*MON1		29	GND	
11	*DIR		30	GND	
12	*STEP		31	GND	
13	*WDATA		32	GND	
14	*WGATE		33	GND	
15	*TRK00		34	GND	
16	*WPRT		35	GND	
17	*RDATA		36	GND	
18	*SIDE		37	GND	
19	DISK C				

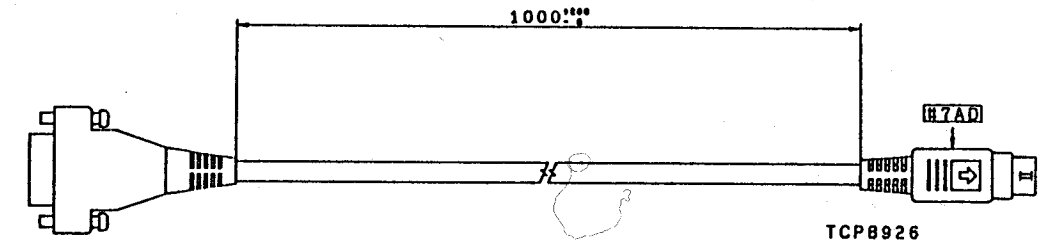
14	*INDEX		28	GND	
13	*MON0		27	GND	
12	*DS1		26	GND	
11	*DS0		25	GND	
10	*MON1		24	GND	
9	*DIR		23	GND	
8	*STEP		22	GND	
7	*WDATA		21	GND	
6	*WGATE		20	GND	
5	*TRK00		19	GND	
4	*WPRT		18	GND	
3	*RDATA		17	GND	
2	*SIDE		16	GND	
1	DISK C		15	HIGH D	

Twisted Pair Lines



Cable set #7AD

Epson PC AX, PC AX2 / IBM PC/AT <- - - - - -> RS-232C (Base Unit)

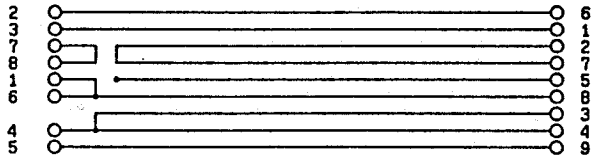


17JE-SC 9-2C(9) female
(DDK)

TCP8926
(Hoshi-Denki)

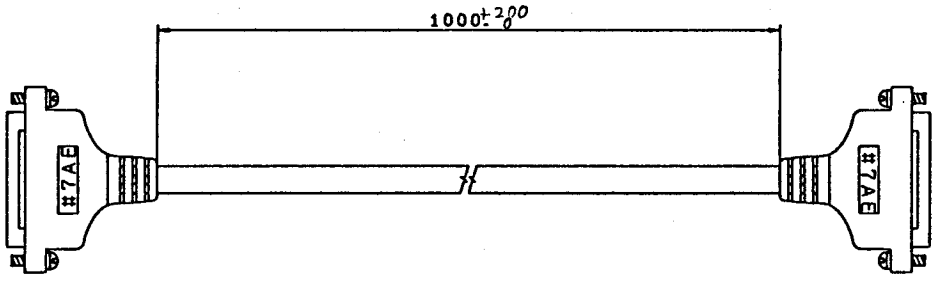
1	DCD		6	DSR	
2	RXD		7	RTS	
3	TXD		8	CTS	
4	DTR		9		
5	SG				

1	RXD	
2	CTS	
3	DSR	
4	DCD	
5		
6	TXD	
7	RTS	
8	DTR	
9	SG	
	CG	



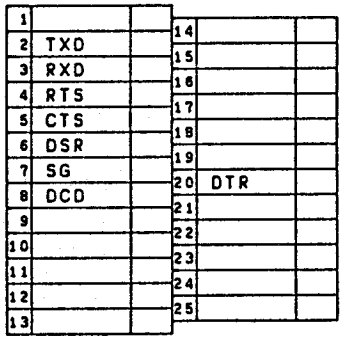
Cable set #7AE

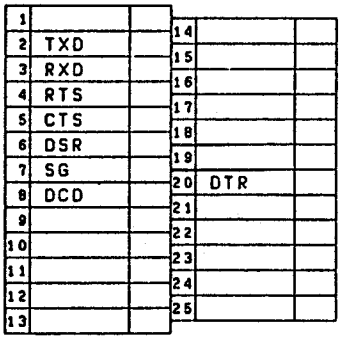
Asynchronous RS board <- - - - - -> NEC PC-9801, IBM 5550

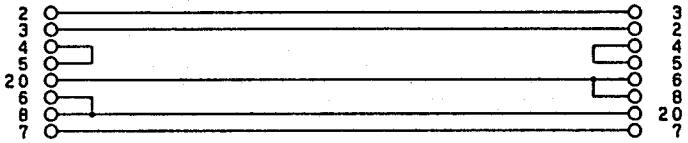


GMP-C25F female
(HONDA)

GMP-C25M male
(HONDA)

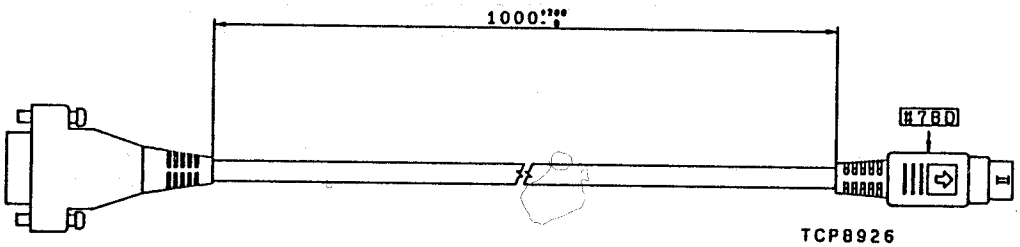






Cable set #7BD

EHT-7 / HC-7 <------> RS-232C I/F (Base Unit)

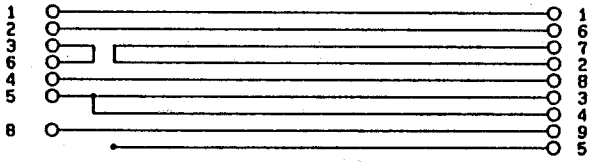


17JE-PA9-2A(7) male
(DDK)

TCP8926
(Hoshi-Denki)

1	TXD		6	CTS
2	RXD		7	
3	RTS		8	SG
4	DSR		9	
5	DTR			

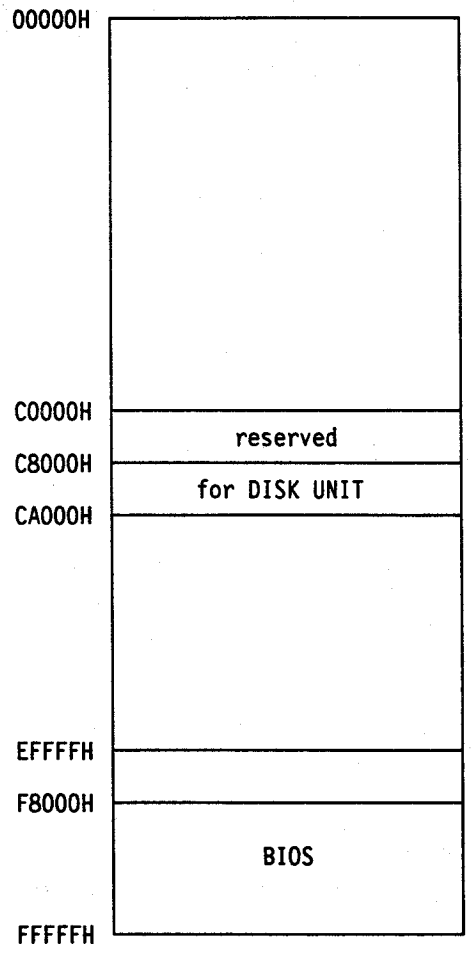
1	RXD	
2	CTS	
3	DSR	
4	DCD	
5		
6	TXD	
7	RTS	
8	DTR	
9	SG	
	CG	



A-5 Expansion ROM

Addresses C0000H - EFFFFH can be assigned to the expansion I/O ROM. At power-on, this ROM can initialize I/O, expand BIOS, and do I/O RESUME. The expansion ROM is installed on the expansion device board.

(1) memory map



Assign the start address of the ROM to C0000H + 800H x n (n=0,1,2,...). Do not assign the following addresses because these addresses are used by the system.

C0000 - C7FFFH	reserved
C8000 - C9FFFH	disk unit

(2) ROM configuration

The first 3 bytes of the expansion ROM must be as follows:

+0 55H
+1 AAH → header
+2 ROM size (in bytes)/512

The lower byte of the arithmetic sum of all bytes in the ROM must be 0.

(3) Operation at power-on

Control is passed to the first 3 bytes of the ROM when the power is turned on (by cold start or warm start). Therefore, the initialization program must be allocated to addresses following these three bytes. The initialization program must return control to the system with the RET instruction.

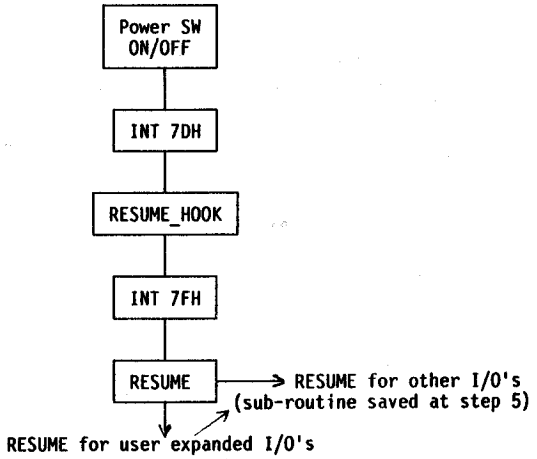
(Example)

```
        DB 55H              ;HEADER
        DB 0AAH             ;HEADER
        DB 10H              ;ROM SIZE (8 kbyte)
        JMP INIT

INIT:   (                   ;INITIALIZE ROUTINE
        )

        RETF                ;RETURN TO SYSTEM
```

(4) RESUME

RESUME is performed by replacing INT 7DH. However, if INT 7DH is replaced directly, the system will not operate normally when the power is turned on after the expansion ROM board has been removed during power-off. Perform the following procedures in the INIT routine described above.

Step 1: Jump to step 6 when the INT 7FH vector contents are not 0000:0000 (that is, when INT 7FH has been expanded).
Step 2: Save the INT 7DH vector.
Step 3: Copy the INT 7DH vector contents to INT 7FH vector.
Step 4: Set the RESUME_HOOK routine to the INT 7DH vector.
Step 5: Load the RESUME_HOOK routine into the area following A800:0058.
Step 6: Save the INT 7FH vector.
Step 7: Set the RESUME routine to the INT 7FH vector.



At step 4, the RESUME_HOOK routine is loaded into the area following A800:0058.

```
RESUME_HOOK:    CMP     AL,1
                JBE     R_HOOK
                MOV     CS: SAVE_SS, SS
                MOV     CS: SAVE_SP, SP
                MOV     SS, CS: INT_7F_SS
                MOV     SP, CS: INT_7F_SP
                INT     7FH
                MOV     SS, CS:SAVE_SS
                MOV     SP, CS:SAVE_SP
                IRET
R_HOOK:         DB      0EAH            ;FAR JUMP OP
ENTRY_INT_7D:   DW      ?               ;INT 7DH VECTOR
                DW      ?               ;  SAVED BY STEP 2
INT_7F_SP:      DW      0               ;SP IN INT 7FH
INT_7F_SS:      DW      0A868H          ;SS IN INT 7FH
SAVE_SP:        DW      ?               ;SP SAVE AREA
SAVE_SS:        DW      ?               ;SS SAVE AREA
```

The RESUME routine (INT 7FH) is provided for the user, and an expansion I/O RESUME can also be added.
The input parameters of INT 7FH are the same as INT 7DH but INT 7FH cannot be called when AL=0 or 1. (See "RESUME_HOOK routine.")
To add AX=0102H to RESUME, use the following routine.

```
RESUME:                 CMP     AX, 0102H
                        JNE     DEFAULT_RESUME
                        PUSH    AX

                        (User RESUME routine)

                        POP     AX
DEFAULT_RESUME:         DB      0EAH            ;FAR KUMP OP
                        DW      OFFSET H_HOOK
                        DW      OA800H
```

Place the H_HOOK routine in a free space in segment A800H.

```
H_HOOK:         DB      0EAH            ;FAR JUMP OP
ENTRY_INT_7F:   DW      ?               ;INT 7FH VECTOR
                DW      ?               ;  SAVED BY STEP6
```

(5) Sample program

```
                        page    54,132
                        ;===============================================================================
                        ;       << SAMPLE PROGRAM FOR EXPANSION ROM USAGE>>
                        ;       < NOTE >
                        ;               Address         Contents
                        ;               -------         ---------------------------------
                        ;                  0            Header 55h
                        ;                  1            Header 0AAh
                        ;                  2            ROM size (bytes/512)
                        ;                  3            Initialize routine start here
                        ;
                        ;               You must make check sum 00h for ROM data
                        ;===============================================================================

0000                    ZSEG    SEGMENT AT      0

01F4                            org     7Dh*4
01F4    02 [            INT_7D_VECTOR   dw      2 dup(?)
            ????
                ]

01FC                            org     7Fh*4
01FC    02 [            INT_7F_VECTOR   dw      2 dup(?)
            ????
                ]

0200                    ZSEG    ENDS

0000                    XSEG    SEGMENT AT      0a800h

0058                            org     58h
0058    26 [            RESUME_HOOK     db      26h dup(?)      ; INT 7D -> INT 7F
            ??
                ]

007E    02 [            ENTRY_INT_7D    dw      2 dup(?)        ; INT 7Dh vector save area
            ????
                ]

0082    ????            INT_7F_SP       dw      ?               ; SP for INT 7Fh
0084    ????            INT_7F_SS       dw      ?               ; SS for INT 7Fh
0086    ????            SAVE_SP         dw      ?               ; SP save area
0088    ????            SAVE_SS         dw      ?               ; SS save area

01EA                            org     1eah
01EA    ??              H_HOOK          db      ?               ; Free area for example
01EB    02 [            ENTRY_INT_7F    dw      2 dup(?)        ; Stored far jump op.
            ????                                                ; INT 7Fh vector save area
                ]

01EF                    XSEG    ENDS

0000                    SSEG    SEGMENT AT      0a868h

0000                            org     0
0000    5C [            STACK_INT_7F    dw      92 dup(?)       ; Stack area for INT 7Fh
            ????
                ]

00B8                    SSEG    ENDS

0000                    CODE    SEGMENT
                                ASSUME  CS:CODE

0000                            org     0
0000                    START:
0000    55                      db      55h                     ; HEADER 55
0001    AA                      db      0aah                    ; HEADER AA
0002    10                      db      10h                     ; SIZE (ex. 16*512 byte = 8 kbyte)
0003    EB 0B 90                jmp     INIT

                        ;       *********************************
                        ;               Resume  I/O port
                        ;       *********************************

0006                    RESUME:
0006    3D 0102                 cmp     ax,0102h
0009    75 00                   jne     DEFAULT_INT_7F          ; If AX = 0102h, resume user I/O

                        ;       ****************************************
                        ;               User resume routine here
                        ;       ****************************************


                        ;       < ON ENTRY >
                        ;               AH : 0 for time to power off
                        ;                    1 for time to power on
                        ;               AL : function or device for resume (refer to INT 7Dh spec)
                        ;       < ON EXIT >
                        ;               All registers must be preserved,and goto entry INT 7Fh
                        ;               routine,or exit INT 7Fh.
                        ;               Ex. (AX)=0102h = resume I/O port when power on

000B                    DEFAULT_INT_7F:
000B    EA                      db      0eah                    ; Far jmp op.
000C    01EA R                  dw      offset H_HOOK           ; Offset
000E    ---- R                  dw      XSEG                    ; Segment
```

```
                        ;       ***************************************************
                        ;               Initialize expand I/O and set RESUME
                        ;       ***************************************************
                        ;
                        ;       < NOTE >
                        ;            1. Make resume routine
                        ;                 Step 1 - If INT 7Fh vector isn't 0000:0000 (INT 7Fh already
                        ;                          extended), go to step 6.
                        ;                 Step 2 - Save INT 7Dh vector. (use in RESUME_HOOK routine)
                        ;                 Step 3 - Copy INT 7Dh vector to INT 7Fh vector.
                        ;                 Step 4 - Set up INT 7Dh vector to RESUME_HOOK routine.
                        ;                 Step 5 - Load RESUME_HOOK routine in XSEG segment
                        ;                 Step 6 - Save INT 7Fh vector. (use in RESUME routine)
                        ;                 Step 7 - Set up INT 7Fh vector to user routine RESUME.
                        ;
                        ;            2. Initialize I/O
0010                    INIT:
                        ;       ***************************************************
                        ;               Make RESUME routine
                        ;       ***************************************************
0010  FA                        cli
0011  FC                        cld
0012  B8 ---- R                mov     ax,ZSEG                 ;
0015  8E D8                     mov     ds,ax                   ;
0017  B8 ---- R                mov     ax,XSEG                 ;
001A  8E C0                     mov     es,ax                   ;
                                assume  ds:ZSEG,es:XSEG         ;
001C  A1 01FC R                 mov     ax,INT_7F_VECTOR        ;
001F  0B 06 01FE R              or      ax,INT_7F_VECTOR+2      ; Test INT 7Fh
0023  75 2E                     jne     SET_INT_7F              ; If already set INT 7Fh,
0025  BF 0058 R                 mov     di,offset RESUME_HOOK   ;
0028  A1 01F4 R                 mov     ax,INT_7D_VECTOR        ;
002B  8B 16 01F6 R              mov     dx,INT_7D_VECTOR+2      ; Save INT 7Dh
002F  A3 01FC R                 mov     INT_7F_VECTOR,ax        ;
0032  89 16 01FE R              mov     INT_7F_VECTOR+2,dx      ; Copy INT 7Dh to INT 7Fh
0036  89 3E 01F4 R              mov     INT_7D_VECTOR,di        ;
003A  8C 06 01F6 R              mov     INT_7D_VECTOR+2,es      ; Set INT 7Dh
003E  BE 006A R                 mov     si,offset RESUME_HOOK_DATA ;
0041  B9 0026                   mov     cx,offset size RESUME_HOOK ;
0044  F3/ 2E: A4        rep     movs    byte ptr es:[di],byte ptr cs:[si]
0047  AB                       stosw                           ;
0048  8B C2                     mov     ax,dx                   ; INT 7Dh offset on entry
004A  AB                       stosw                           ;
004B  B8 0088                   mov     ax,offset size STACK_INT_7F ; INT 7Dh segment on entry
004E  AB                       stosw                           ;
004F  B8 ---- R                mov     ax,SSEG                 ; SP for INT 7Fh
0052  AB                       stosw                           ; SS for INT 7Fh
0053
0053  BE 01FC R        SET_INT_7F:
0056  BF 01EA R                 mov     si,offset INT_7F_VECTOR ;
0059  B0 EA                     mov     di,offset H_HOOK        ;
005B  AA                       mov     al,0eah                 ;
005C  A5                       stosb                           ; Far jump op.
005D  A5                       movsw                           ; INT 7Fh offset on entry
005E  C7 06 01FC R 0006 R      movsw                           ; INT 7Fh segment on entry
0064  8C 0E 01FE R             mov     INT_7F_VECTOR,offset RESUME ;
0068  FB                       mov     INT_7F_VECTOR+2,cs      ;
                               sti                             ; Set new INT 7F
                        ;       ***************************************************
                        ;               Initialize user expansion I/O here
                        ;       ***************************************************
0069
0069  CB                DUMMY   PROC    FAR
                                ret
006A                    DUMMY   ENDP                            ; Must be far return
006A
006A                    RESUME_HOOK_DATA:
006A  3C 01                     cmp     al,1                    ;
006C  76 21                     jbe     R_HOOK                  ; If AL = 0, 1, goto old INT 7Dh
                                assume  cs:XSEG                 ;
006E  2E: 8C 16 0088 R          mov     cs:SAVE_SS,ss           ; Save SS
0073  2E: 89 26 0086 R          mov     cs:SAVE_SP,sp           ; Save SP
0078  2E: 8E 16 0084 R          mov     ss,cs:INT_7F_SS         ; Get SS for INT 7Fh
007D  2E: 8B 26 0082 R          mov     sp,cs:INT_7F_SP         ; Get SP for INT 7Fh
0082  CD 7F                     int     7fh                     ; INT 7Fh - RESUME routine
0084  2E: 8E 16 0088 R          mov     ss,cs:SAVE_SS           ; Restore SS
0089  2E: 8B 26 0086 R          mov     sp,cs:SAVE_SP           ; Restore SP
008E  CF                        iret                            ;
                                assume  cs:CODE                 ;
008F  EA                R_HOOK: db      0eah                    ; Far jmp op.
0090                    CODE    ENDS
                                END     START
```

A-6 Differences between EPSON PCe and PX-16 with PC boot

This section summarizes differences in operation between the PX- 16 (when started by PC boot) and the EPSON PCe.

1. Hardware differences

(1) Wait cycles

	PX-16	PCe
Built-in ROM byte access (10 MHz)	3 [7]	0 [4]

[x] = Total CLK cycles per bus cycle

However, the PX-16 can be switched between 0 waits and 3 waits. Due to limitations involving access speed of 4 Mbit mask ROM, 3 waits is used in the current version.

(2) CMOS RAM/RTC

	PX-16	PCe
CMOS RAM	Not used	Used
RTC	Not used	Used

The PX-16 uses time management by the slave CPU (µPD75106) instead of RTC.

The CMOS RAM is included in the RTC, and is not used in the PX- 16. Instead, the PX-16 is equipped with SRAM (2 KB).

(3) DIP switches

	PX-16	PCe
Number of DIP switches	10 bits	4 + 8 bits

Functions of individual DIP switch bits are not the same. For details, see the relevant parts of the DIP switch explanations.

(4) I/O ports

	PX-16	PCe
I/O ports 70-71H	x	o
I/O ports 11B4-11B5H	x	o
I/O ports 11D0-11FFH	o	x

o - valid, x - not valid

P70-P71H are registers for CMOS RAM access.
P11B4-P11B5H are registers for RTC access.
P11D0-P11FFH are registers for PX-16 expansion.

(5) Numeric coprocessor

A numeric coprocessor can be installed in the PCe, but cannot be installed in the PX-16.

(6) Parity check

The PX-16 does not have a parity check function for RAM. Instead, it is equipped with a read check function for expansion RAM (the RAM disk).

(7) Floppy disk drives

	PX-16	PCe
Number of drives	Up to 2	Up to 4

Although both a disk unit and TF-16 can be installed in the PX- 16 (for a total of four drives), only two of the drives can actually be used.

(8) Light pen

A light pen can be used with the PCe, but not with the PX-16.

(9) Keyboard

	PX-16	PCe
Number of keys	78/79 keys	101/102 keys

However, the PX-16 supports the functions of all 101/102 keys by using other keys in combination with the Fn key.

2. Software Differences

(1) INT 02H

	PX-16	PCe
INT 02H causes	NMI upon reading expansion RAM	NMI upon RAM battery check

(2) INT 10H

	PX-16	PCe
Input (AH) = 04H	Dummy	Lightpen

(3) INT 11H

		PX-16	PCe
Output (AX)	bits 3,2	RAM type	Not used
	bit 1	0	Numeric coprocessor

(4) INT 15H

	PX-16	PCe
Input (AH) = 84H	Dummy	Joystick

(5) INT 16H

	PX-16	PCe
Input (AH) = 03H	x	o

(6) INT 18H

Function of this interrupt is completely different. With the PX- 16, it is used as the HC boot entry.

(7) INT 1AH

	PX-16	PCe
Input (AH) = 03H (Daylight savings time code)	Not used	Used
Input (AH) = 06H, 07H (Alarm)	Used	Not used

A-7 Sample program lists

Subroutines for sample programs

```
                                page    56,132
                        ;       ================================================================================
                        ;                               << UTILITY SUBROUTINES >>
                        ;       < NOTE >
                        ;               1. dump         dump CX bytes from DS:BX
                        ;               2. dsphexw      display a word data in hexadecimal form
                        ;               3. dsphexb      display a byte data in hexadecimal form
                        ;               4. dspcrlf      print CR+LF
                        ;               5. dspchar      display a character
                        ;               6. dspmsg       display message
                        ;               7. set_cursor   set a cursor position
                        ;       ================================================================================
0000                    code    segment byte public
                                assume  cs:code,ds:code,es:code,ss:code
                        ;       <> public define <>
                                public  dump                    ; dump CX byts form DS:BX
                                public  dsphexw                 ; display a word data in hexadecimal form
                                public  dsphexb                 ; display a byte data in hexadecimal form
                                public  dspcrlf                 ; move a cursor to top of next line
                                public  dspchar                 ; display a character
                                public  dspmsg                  ; display message
                                public  set_cursor              ; set a cursor position
                        ;       ****************************************
                        ;               dump CX bytes from DS:BX
                        ;       ****************************************
                        ;       < NOTE>
                        ;               dump CX bytes from DS:BX
                        ;       < ON ENTRY >
                        ;               CX    : counts
                        ;               DS:DX : start address
0000                    dump            proc    near
0000  50                               push    ax              ; save registers
0001  53                               push    bx              ;
0002  51                               push    cx              ;
                        ;       < display a line >
0003                    linelp:                                 ; < 1st loop start >
0003  51                               push    cx              ; save 1st loop counter
0004  B9 0010                          mov     cx,10h          ; set 2nd loop counter
0007  53                               push    bx              ; save pointer
                        ;       < display data in hexadecimal form (16 bytes) >
0008                    hexlp:                                  ; < 2nd loop start >
0008  8A 07                            mov     al,[bx]         ; get a character
000A  E8 004D R                        call    dsphexb         ; display a data in hexadecimal form
000D  B0 20                            mov     al,' '          ; display a space
000F  E8 0079 R                        call    dspchar         ;
0012  43                               inc     bx              ; increase a pointer
0013  E2 F3                            loop    hexlp           ; < 2nd loop end >
0015  B0 20                            mov     al,' '          ; display a space
0017  E8 0079 R                        call    dspchar         ;
001A  B0 20                            mov     al,' '          ; display a space
001C  E8 0079 R                        call    dspchar         ;
001F  5B                               pop     bx              ; recover a pointer
0020  B9 0010                          mov     cx,10h          ; set 3rd loop counter
                        ;       < display data in character form (16 bytes) >
0023                    chrlp:
0023  8A 07                            mov     al,[bx]         ; get data.
0025  3C 20                            cmp     al,' '          ; if the data <,space or data > 7fh
0027  72 04                            jb      badchr          ; then display '.'
0029  3C 7F                            cmp     al,7fh          ; else print the data.
002B  72 02                            jb      goodchr         ;
002D                    badchr:
002D  B0 2E                            mov     al,'.'
002F                    goodchr:
002F  E8 0079 R                        call    dspchar         ; print the data.
0032  43                               inc     bx              ; increase a pointer
0033  E2 EE                            loop    chrlp           ; < 3rd loop end >
0035  E8 006C R                        call    dspcrlf         ; print CR+LF
0038  59                               pop     cx              ; recover 1st loop counter
0039  83 E9 10                        sub     cx,10h          ; update 1st loop counter
003C  77 C5                            ja      linelp          ; < 1st loop end >
003E  59                               pop     cx              ; Recover registers
003F  5B                               pop     bx              ;
0040  58                               pop     ax              ;
0041  C3                               ret                     ;
0042                    dump            endp
```

```
                        ;       ***************************************************
                        ;               print a word data in hexadecimal form
                        ;       ***************************************************
                        ;       < NOTE >
                        ;               print a word data in hexadecimal form
                        ;       < ON ENTRY >
                        ;               AX : data
0042                    dsphexw         proc    near
0042  50                        push    ax                      ;
0043  8A C4                     mov     al,ah                   ;
0045  E8 004D R                 call    dsphexb                 ; print out high byte
0048  58                        pop     ax                      ;
0049  E8 004D R                 call    dsphexb                 ; print out low byte
004C  C3                        ret
004D                    dsphexw         endp

                        ;       ***************************************************
                        ;               print a byte data in hexadecimal form
                        ;       ***************************************************
                        ;       < NOTE >
                        ;               print a byte data in hexadecimal form
                        ;       < ON ENTRY >
                        ;               AL : data
004D                    dsphexb         proc    near
004D  50                        push    ax                      ;
004E  51                        push    cx                      ;
004F  B1 04                     mov     cl,4                    ; shift 4 bits to right
0051  D2 C8                     ror     al,cl                   ;
0053  E8 005C R                 call    dsphexn                 ; print high nibble
0056  59                        pop     cx                      ;
0057  58                        pop     ax                      ;
0058  E8 005C R                 call    dsphexn                 ; print low nibble
005B  C3                        ret                             ;
005C                    dsphexb         endp

                        ;       ***************************************************
                        ;               print a nibble data in hexadecimal form
                        ;       ***************************************************
                        ;       < NOTE >
                        ;               print a nibble data in hexadecimal form
                        ;       < ON ENTRY >
                        ;               AL : data(low 4 bit only)
005C                    dsphexn         proc    near
005C  50                        push    ax                      ;
005D  24 0F                     and     al,0fh                  ; mask out low 4 bits.
005F  04 30                     add     al,30h                  ; change to ASCII code '0' to '9'
0061  3C 3A                     cmp     al,3ah                  ; if code > '9'
0063  72 02                     jb      dsphexn_10              ;
0065  04 07                     add     al,7                    ; then change to ASCII code 'A' to 'F'
0067                    dsphexn_10:
0067  E8 0079 R                 call    dspchar                 ; print out.
006A  58                        pop     ax                      ;
006B  C3                        ret                             ;
006C                    dsphexn         endp

                        ;       ****************************
                        ;               print CR + LF
                        ;       ****************************
                        ;       < NOTE >
                        ;               print (display) CR + LF
                        ;       < ON ENTRY >
                        ;               NON
006C                    dspcrlf         proc    near
006C  50                        push    ax                      ;
006D  B0 0D                     mov     al,0dh                  ;
006F  E8 0079 R                 call    dspchar                 ; print CR code
0072  B0 0A                     mov     al,0ah                  ;
0074  E8 0079 R                 call    dspchar                 ; print LF code
0077  58                        pop     ax                      ;
0078  C3                        ret                             ;
0079                    dspcrlf         endp
```

```
                        ;       *********************************
                        ;               print a character
                        ;       *********************************
                        ;
                        ;       < NOTE >
                        ;               print a character
                        ;
                        ;       < ON ENTRY >
                        ;               AL : data
0079                    dspchar         proc    near

0079  50                        push    ax              ; save registers
007A  53                        push    bx              ;
007B  B4 0E                     mov     ah,0eh          ; set Int 10h function code (write tty)
007D  B3 07                     mov     bl,07h          ; set attribute normal
007F  CD 10                     int     10h             ; print a character
0081  5B                        pop     bx              ; recover registers
0082  58                        pop     ax              ;
0083  C3                        ret                     ;

0084                    dspchar         endp

                        ;       *********************************
                        ;               display message
                        ;       *********************************
                        ;
                        ;       < NOTE >
                        ;               Display message
                        ;
                        ;       < ON ENTRY >
                        ;               DS:SI : message string (00h is terminater)
                        ;
0084                    dspmsg          proc    near

0084  50                        push    ax              ; save registers
0085  53                        push    bx              ;

0086                    dspmsg_loop:
0086  8A 04                     mov     al,[si]         ; get a character
0088  46                        inc     si              ; pinter update
0089  3C 00                     cmp     al,0            ; terminate character ?
008B  74 08                     jz      dspmsg_exit     ; end if Yes
008D  B4 0E                     mov     ah,0eh          ; set write tty function code
008F  B3 07                     mov     bl,7            ; set attribute normal
0091  CD 10                     int     10h             ; print a character
0093  EB F1                     jmp     dspmsg_loop     ; ### LOOP ###

0095                    dspmsg_exit:
0095  5B                        pop     bx              ; recover registers
0096  58                        pop     ax              ;
0097  C3                        ret

0098                    dspmsg          endp


                        ;       *********************************
                        ;               set a cursor posirion
                        ;       *********************************
                        ;
                        ;       < NOTE >
                        ;               set a cursor position
                        ;
                        ;       < ON ENTRY >
                        ;               DH : row position
                        ;               DL : column position
0098                    set_cursor      proc    near

0098  50                        push    ax              ; save registers
0099  53                        push    bx              ;
009A  B3 00                     mov     bl,0            ; set attribute
009C  B4 02                     mov     ah,2            ; set function code
009E  CD 10                     int     10h             ; set cursor position
00A0  5B                        pop     bx              ; recover registers
00A1  58                        pop     ax              ;
00A2  C3                        ret

00A3                    set_cursor      endp

00A3                    code    ends

                                end
```

(2) INT 10H / Cartridge2 video access

```
                                page    54,132
                        ;       ================================================================================
                        ;
                        ;               << SAMPLE PROGRAM FOR INT 10H (cartridge 2 video) >>
                        ;
                        ;       < NOTE >
                        ;               This program emulates WRITE-TELETYPE function with attribute.
                        ;               This program suports only 80 * 25 size screen. If you want to
                        ;               execute this program on other size screen(40 * 25, 40 * 10),
                        ;               you need to change COLUMN and ROW numbers.
                        ;
                        ;       ================================================================================

0000                    code    segment byte public
                                assume  cs:code,ds:code,es:code,ss:code

= 0050                  COLUMN  equ     80                      ; screen column size
= 0019                  ROW     equ     25                      ; screen row size

0100                            org     100h

                        ;       ****************************
                        ;               Main program
                        ;       ****************************
                        ;
                        ;       < NOTE >
                        ;               This porgram displays messages with each attribute set.

0100                    int10h          proc    near

0100  BB 7007                   mov     bx,7007h                ; set initial attribute(BH for reverse)
0103  BE 0132 R                 mov     si,offset msg1          ; set message 1 pointer
0106  BF 014E R                 mov     di,offset msg2          ; set message 2 pointer

0109                    int10h_loop1:
0109  56                       push    si
010A  57                       push    di                      ; save massage pointer

010B                    int10h_loop2:
010B  8A 04                     mov     al,[si]                 ; get a character
010D  46                        inc     si                      ; increase a pointer
010E  3C 00                     cmp     al,0                    ; check a terminater
0110  74 05                     jz      int10h_next             ; if terminater is found, stay in this loop
0112  E8 015B R                 call    wrt_tty                 ; write one character & attribute
0115  EB F4                     jmp     intT0h_loop2            ; ### loop ###

0117                    int10h_next:

0117  5E                        pop     si                      ; pop up a message pointer & exchange
0118  5F                        pop     di
0119  80 FB 10                  cmp     bl,10h                  ; if attribute is reversed ?
011C  72 04                     jb      int10h_next_10          ; No
011E  81 EB 0110                sub     bx,110h                 ; update attribute

0122                    int10h_next_10:
0122  8A E3                     mov     ah,bl                   ; change forgrand attribute
0124  8A DF                     mov     bl,bh                   ;                  with background
0126  8A FC                     mov     bh,ah                   ;
0128  83 FB 00                  cmp     bx,0                    ; if attribute is 0, then end
012B  75 DC                     jnz     int10h_loop1            ;

012D                    int10h_exit:
012D  B8 4C00                   mov     ax,4c00h                ; terminate this program
0130  CD 21                     int     21h                     ;

0132                    int10h          endp

0132  42 49 4F 53 20 49 msg1    db      'BIOS INT10H test.(Normal) :',0
      4E 54 31 30 48 20
      74 65 73 74 2E 28
      4E 6F 72 6D 61 6C
      29 20 3A 00
014E  20 28 52 65 76 65 msg2    db      ' (Reverse)',0ah,0dh,0
      72 73 65 29 0A 0D
      00

                        ;       ********************************************
                        ;               Write teletype with attribute
                        ;       ********************************************
                        ;
                        ;       < NOTE >
                        ;               Write teletype to an active screen
                        ;               Attribute is valid on every screen mode
                        ;       < ON ENTRY >
                        ;               AL : character to write
                        ;               BL : attribute

015B                    wrt_tty         proc    near

015B  52                        push    dx                      ; save registers
015C  51                        push    cx
015D  53                        push    bx
015E  50                        push    ax

015F  50                        push    ax                      ; save character & attibute
```

```
0160  53                              push    bx                      ; get current page number
0161  E8 01E7 R                       call    get_scr_mode            ; save current page number
0164  8A E7                           mov     ah,bh                   ; recover attribute
0166  5B                              pop     bx                      ; set current page number
0167  8A FC                           mov     bh,ah                   ; get current cursor position in DX
0169  E8 01AE R                       call    get_cursor              ;
016C  58                              pop     ax                      ;

016D  3C 08                           cmp     al,8                    ; Backspace?
016F  74 22                           je      tty_bs                  ;
0171  3C 0A                           cmp     al,10                   ; Line-Feed?
0173  74 2A                           je      tty_lf                  ;
0175  3C 0D                           cmp     al,13                   ; Carriage-Return?
0177  74 22                           je      tty_cr                  ;

0179  E8 01DA R                       call    wrt_char                ; write one character & attribute

017C  FE C2                           inc     dl                      ; increase column number
017E  80 FA 50                        cmp     dl,COLUMN               ;
0181  75 23                           jne     tty_cur                 ; jump if new cursor position is not
0183  32 D2                           xor     dl,dl                   ;   on a right side
0185  FE C6                           inc     dh                      ; increase row number
0187  80 FE 19                        cmp     dh,ROW                  ;
018A  75 1A                           jne     tty_cur                 ; jump if new cursor position is not
                                                                      ;   on the bottom of screen
018C                          tty_srl:
018C  FE CE                           dec     dh                      ;
018E  E8 01B8 R                       call    scrollup                ; scroll one line up
0191  EB 13                           jmp     short tty_cur           ;

0193                          tty_bs:
0193  0A D2                           or      dl,dl                   ; check column
0195  74 12                           je      tty_ext                 ; if left corner, then ignore
0197  FE CA                           dec     dl                      ; else backspace
0199  EB 0B                           jmp     short tty_cur           ; set a new cursor position

019B                          tty_cr:
019B  32 D2                           xor     dl,dl                   ; reset column
019D  EB 07                           jmp     short tty_cur           ; set a new cursor position

019F                          tty_lf:
019F  FE C6                           inc     dh                      ;
01A1  80 FE 19                        cmp     dh,ROW                  ; row overflow?
01A4  74 E6                           je      tty_srl                 ; if Yes, then to scroll

01A6                          tty_cur:
01A6  E8 01B3 R                       call    set_cursor              ; set a new cursor position

01A9                          tty_ext:
01A9  58                              pop     ax                      ; recover registers
01AA  5B                              pop     bx                      ;
01AB  59                              pop     cx                      ;
01AC  5A                              pop     dx                      ;
01AD  C3                              ret                             ;

01AE                          wrt_tty         endp

                              ;       ********************************************************
                              ;               Get cursor position(INT 10H Function 03h)
                              ;       ********************************************************
                              ;
                              ;       < NOTE >
                              ;               Get a cursor position
                              ;       < ON ENTRY >
                              ;               BH : display page number
                              ;       < ON EXIT >
                              ;               DH : current row position
                              ;               DL : current colmn position
                              ;               CX : current cursor type

01AE                          get_cursor      proc    near

01AE  B4 03                           mov     ah,03h                  ; get cursor function
01B0  CD 10                           int     10h                     ;
01B2  C3                              ret

01B3                          get_cursor      endp

                              ;       ********************************************************
                              ;               Set cursor position(INT 10H Function 02h)
                              ;       ********************************************************
                              ;
                              ;       < NOTE >
                              ;               Set a cursor position
                              ;       < ON ENTRY >
                              ;               BH : display page number
                              ;               DH : new row position
                              ;               DL : new colmn position

01B3                          set_cursor      proc    near

01B3  B4 02                           mov     ah,02h                  ; set cursor function
01B5  CD 10                           int     10h                     ;
01B7  C3                              ret

01B8                          set_cursor      endp
```

```
                        ;       ****************************
                        ;               Scroll up
                        ;       ****************************
                        ;
                        ;       < NOTE >
                        ;               Scroll one line up on a whole screen
                        ;       < ON ENTRY >
                        ;               None
01B8                    scrollup        proc    near

01B8  52                        push    dx                      ; save registers
01B9  51                        push    cx
01BA  53                        push    bx                      ;

01BB  E8 01E2 R                 call    read_char               ; read attribute of current cursor position
01BE  50                        push    ax
01BF  E8 01E7 R                 call    get_scr_mode            ; get screen mode
01C2  3C 03                     cmp     al,3                    ; graphics mode ?
01C4  58                        pop     ax
01C5  B7 00                     mov     bh,0                    ; set attribute (for graphics mode)
01C7  77 02                     ja      scrollup_10             ; jump, if graphics mode
01C9  8A FC                     mov     bh,ah                   ; set current attribute

01CB                    scrollup_10:
01CB  33 C9                     xor     cx,cx                   ; upper left corner (0,0)
01CD  B6 18                     mov     dh,ROW-1
01CF  B2 4F                     mov     dl,COLUMN-1             ; bottom right coner
01D1  B8 0601                   mov     ax,0601h                ; scroll up one line
01D4  CD 10                     int     10h                     ;

01D6  5B                        pop     bx                      ; recover registers
01D7  59                        pop     cx
01D8  5A                        pop     dx                      ;
01D9  C3                        ret

01DA                    scrollup        endp

                        ;       ********************************************
                        ;               Write a character & attribute
                        ;       ********************************************
                        ;
                        ;       < NOTE >
                        ;               Write one character & attribute
                        ;       < ON ENTRY >
                        ;               AL : character to write
                        ;               BL : attribute
                        ;               BH : display page number
01DA                    wrt_char        proc    near

01DA  B9 0001                   mov     cx,1                    ; only a character
01DD  B4 09                     mov     ah,09h                  ; write character function
01DF  CD 10                     int     10h
01E1  C3                        ret

01E2                    wrt_char        endp

                        ;       ******************************************
                        ;               Read a character & attribute
                        ;       ******************************************
                        ;
                        ;       < NOTE >
                        ;               Read one character & attribute
                        ;       < ON ENTRY >
                        ;               BH : display page number
                        ;       < ON EXIT >
                        ;               AL : character
                        ;               BL : attribute
01E2                    read_char       proc    near

01E2  B4 08                     mov     ah,08h                  ; write character function
01E4  CD 10                     int     10h
01E6  C3                        ret

01E7                    read_char       endp

                        ;       ********************************
                        ;               Get screen mode
                        ;       ********************************
                        ;
                        ;       < NOTE >
                        ;               Get a current screen mode
                        ;       < ON ENTRY >
                        ;               none
                        ;       < ON EXIT >
                        ;               AL : current screen mode
                        ;               AH : number of character columns on a screen
                        ;               BH : current active display page number
01E7                    get_scr_mode    proc    near

01E7  B4 0F                     mov     ah,15           ; get screen mode
01E9  CD 10                     int     10h
01EB  C3                        ret

01EC                    get_scr_mode    endp

01EC                    code    ends

                                end     int10h
```

(3) INT 10H / Touch keyboard LCD access

```
                                        page    54,132
                                ;       ===============================================================================
                                ;               << SAMPLE PROGRAM FOR INT 10H (touch panel screen) >>
                                ;       < NOTE >
                                ;               This program displays message to a touch panel screen.
                                ;       ===============================================================================

0000                            code    segment byte public
                                        assume  cs:code,ds:code,es:code,ss:code

0100                                    org     100h

                                ;       *****************************
                                ;               Main program
                                ;       *****************************

                                ;       < NOTE >
                                ;               Display message with BOX attribute

0100                            int10ht         proc    near

                                                                        ; Clear screen (for screen mode 0,1,4,5)
0100  B9 0000                           mov     cx,0000h                ; from top left corner
0103  BA 0819                           mov     dx,0819h                ; to bottom right coner
0106  B7 00                             mov     bh,0                    ; attribute set to 0
0108  B8 4600                           mov     ax,4600h                ; clear screen
010B  CD 10                             int     10h                     ;

010D  BA 0300                           mov     dx,0300h                ; set cursor position, (3,0)
0110  B7 00                             mov     bh,0                    ; page 0
0112  B4 42                             mov     ah,42h                  ;
0114  CD 10                             int     10h                     ;

0116  BE 013D R                         mov     si,offset msg           ; set message 1 pointer

0119  B0 20                             mov     al,' '                  ; set dummy character
011B  B3 45                             mov     bl,01000101b            ; write left side attributes '['
011D  B4 4E                             mov     ah,4eh                  ; write one character & attribute
011F  CD 10                             int     10h                     ;

0121                            int10ht_loop:
0121  8A 04                             mov     al,[si]                 ; get one character
0123  46                                inc     si                      ; increase a pointer
0124  3C 00                             cmp     al,0                    ; check terminater
0126  74 08                             jz      int10ht_next            ; if terminater is found, stay in this loop
0128  B3 0F                             mov     bl,00001111b            ; set attributes (vertical ruler lines)
012A  B4 4E                             mov     ah,4eh                  ; write one character & attribute
012C  CD 10                             int     10h                     ;
012E  EB F1                             jmp     int10ht_loop            ; ### loop ###

0130                            int10ht_next:
0130  B0 20                             mov     al,' '                  ; dummy character
0132  B3 4A                             mov     bl,01001010b            ; write left side attributes ']'
0134  B4 4E                             mov     ah,4eh                  ; write a character & attribute (write teletype)
0136  CD 10                             int     10h                     ;

0138  B8 4C00                           mov     ax,4c00h                ; terminate this program
013B  CD 21                             int     21h                     ;

013D                            int10ht         endp

013D  53 45 49 4B 4F 20         msg     db      'SEIKO EPSON CORPORATION',0
      45 50 53 4F 4E 20
      43 4F 52 50 4F 52
      41 54 49 4F 4E 00


0155                            code    ends
                                        end     int10ht
```

(4) INT 13H / Read FDD

```
                        page    54,132
                        ;===============================================================================
                        ;                       << SAMPLE PROGRAM FOR INT 13H >>
                        ;
                        ; < NOTE >
                        ;       Read floppy disk sequentially and display the read data.
                        ;       This program supports only 360KB FDD. IF you want to execute
                        ;       this program for athor disk size, you must change initial
                        ;       values of MAXTRACK, MAXHEAD, MAXSWCTOR and DRV.
                        ;===============================================================================
0000                    code    segment byte public
                                assume  cs:code,ds:code,es:code,ss:code

                        ;       <> constant define <>

= 0027                  MAXTRACK        equ     39              ; max. track number
= 0001                  MAXHEAD         equ     1               ; max. head number
= 0009                  MAXSECTOR       equ     9               ; max. sector number

= 0001                  SECN            equ     1               ; sector count number
= 0000                  DRV             equ     0               ; drive number

                        ;       <> public define <>

                                extrn   dump:near               ; dump CX byts form DS:BX
                                extrn   dsphexb:near            ; print a byte data in hexadecimal form
                                extrn   dspcrlf:near            ; move cursor to top of next line
                                extrn   dspchar:near            ; print a character

0100                            org     100h

                        ;       ****************************
                        ;             main routine
                        ;       ****************************
                        ;
                        ;       < NOTE >
                        ;               Read floppy disk sequentially and display the read data

0100                    int13h          proc    near

0100                    loop1:
0100  B4 02                             mov     ah,2                    ; read sector function code
0102  B0 01                             mov     al,SECN                 ; read sector counts
0104  B2 00                             mov     dl,DRV                  ; set drive number
0106  8A 36 0188 R                      mov     dh,byte ptr head        ; set head number
010A  8A 2E 0187 R                      mov     ch,byte ptr track       ; set track number
010E  8A 0E 0189 R                      mov     cl,byte ptr sector      ; set sector number
0112  BB 018A R                         mov     bx,offset buffer        ; ES:BX buffer address
0115  CD 13                             int     13h                     ; ### read sector ###

0117  E8 0152 R                         call    dspsec                  ; display read data
011A  B4 00                             mov     ah,0                    ; get key
011C  CD 16                             int     16h                     ; if ESC then exit
011E  3C 1B                             cmp     al,1bh
0120  74 2B                             jz      int13h_end              ;

0122  FE 06 0189 R                      inc     byte ptr sector         ; increase sector number
0126  80 3E 0189 R 09                   cmp     byte ptr sector,MAXSECTOR ; sector number overflowed ?
012B  76 D3                             jbe     loop1                   ; No
012D  C6 06 0189 R 01                   mov     byte ptr sector,1       ; reset sector number

0132  FE 06 0188 R                      inc     byte ptr head           ; increase head number
0136  80 3E 0188 R 01                   cmp     byte ptr head,MAXHEAD   ; head number over flowed ?
013B  76 C3                             jbe     loop1                   ; No
013D  C6 06 0188 R 00                   mov     byte ptr head,0         ; reset head number

0142  FE 06 0187 R                      inc     byte ptr track          ; increase track number
0146  80 3E 0187 R 27                   cmp     byte ptr track,MAXTRACK ; track number overflowed ?
014B  76 B3                             jbe     loop1                   ; No

014D                    int13h_end:
014D  B8 4C00                           mov     ax,4c00h                ; terminate this program
0150  CD 21                             int     21h                     ;

0152                    int13h          endp
```

```
                        ;       ********************************
                        ;               display a sector
                        ;       ********************************
                        ;
                        ;       < NOTE >
                        ;               display data (512 bytes)
                        ;       < ON ENTRY >
                        ;               None
0152                    dspsec          proc    near

0152  BB 018A R                        mov     bx,offset buffer        ; set buffer's top address
0155  B9 0020                          mov     cx,20h                  ; set loop counter 20H * 10H byte
0158                    dspsec_loop:
0158  51                               push    cx                      ; save counter
0159  A0 0187 R                        mov     al,byte ptr track       ; get track number
015C  E8 0000 E                        call    dsphexb                 ; display
015F  B0 2D                            mov     al,'-'                  ; display separator
0161  E8 0000 E                        call    dspchar                 ;

0164  A0 0188 R                        mov     al,byte ptr head        ; get head number
0167  E8 0000 E                        call    dsphexb                 ; display
016A  B0 2D                            mov     al,'-'                  ; display separator
016C  E8 0000 E                        call    dspchar                 ;

016F  A0 0189 R                        mov     al,byte ptr sector      ; get sector number
0172  E8 0000 E                        call    dsphexb                 ; display
0175  B0 3A                            mov     al,':'                  ; display separator
0177  E8 0000 E                        call    dspchar                 ;

017A  B9 0010                          mov     cx,10h                  ; set counter (10h bytes)
017D  E8 0000 E                        call    dump                    ; display 10 byte
0180  83 C3 10                         add     bx,10h                  ; pointer update
0183  59                               pop     cx                      ; recover loop counter
0184  E2 D2                            loop    dspsec_loop             ; # loop #
0186  C3                               ret                             ;

0187                    dspsec          endp



                        ;       <> work area <>

0187  00                track   db      0               ; track number
0188  00                head    db      0               ; head number
0189  01                sector  db      1               ; sector number


018A                    buffer  label byte

018A  0200 [                            db      512 dup(0)
              00
                 ]


038A                    code    ends
                                end     int13h
```

(5) INT 14H / RS-232C communication

```
                        page    54,132
                        ;===============================================================================
                        ;                       << SAMPLE PROGRAM FOR INT 14H >>
                        ;       < NOTE >
                        ;               This program receives one character data form a serial port and
                        ;               displays it, and sends one character data input form keyboard
                        ;               to serial port.
                        ;===============================================================================
0000                    code    segment byte public
                                assume  cs:code,ds:code,es:code,ss:code
                        ;       <> external define <>
                                extrn   dspchar:near            ; display one character
                        ;       < > constant define <>
= 0100                  BUF_SIZE        equ     256             ; receive-buffer size
= 0000                  RS_PORT         equ     0               ; selial port number
= 0013                  DUMMY_XON       equ     13h             ; XON code
= 0012                  DUMMY_XOFF      equ     12h             ; XOFF code

0100                            org     100h
                        ;       ****************************
                        ;               Mian program
                        ;       ****************************
                        ;       < NOTE >
                        ;               This program enables to communicate via RS-232c port 0 by
                        ;               interrupt mode.
                        ;               If CTRL+'C' is pressed, it terminates this program.
0100                    int14h          proc    near

0100  E8 0130 R                call    init_rs                 ; initialize RS-232C port
0103  E8 013C R                call    open_rs                 ; open RS-232C by interrupt mode
0106  B8 0000                  mov     ax,0                    ; clear the sent data & status
0109                    int14h_loop:
0109  50                        push    ax                      ; save key status
010A  E8 0161 R                 call    get_rs                  ; get a data from RS-232C
010D  75 03                     jnz     int14h_10               ; jump if, nothing received
010F  E8 0000 E                 call    dspchar                 ; display received data
0112
0112  58                int14h_10:
0113  80 FC 00                 pop     ax                      ; recover send data & status
0116  75 0B                     cmp     ah,0                    ; pending data exist ?
0118  E8 017C R                 jnz     int14h_20               ; jump if Yes
011B  B4 00                     call    get_kb                  ; if No, then get data from keyboard
011D  74 EA                     mov     ah,0                    ; clear send data status
011F  3C 03                     jz      int14h_loop             ; jump if key is not input
0121  74 05                     cmp     al,3                    ; if terminate character (ctrl+'C')
                                jz      int14h_exit             ; jump if Yes
0123
0123  E8 0157 R         int14h_20:
                                call    put_rs                  ; Send data to RS-232C
0126  EB E1                     jmp     int14h_loop             ; if cannot send, return value AH <> 0
                                                                ; loop
0128                    int14h_exit:

0128  E8 0150 R                 call    close_rs                ; close RS-232C PORT
012B  B8 4C00                   mov     ax,4c00h                ; terminate this program
012E  CD 21                     int     21h                     ;
0130                    int14h          endp

                        ;       ********************************
                        ;               initialize RS-232C
                        ;       ********************************
                        ;       < NOTE >
                        ;               Initialize RS-232C port 0
                        ;               (4800bps, 8bit data, non parity, 1 stop bit)
                        ;       < ON ENTRY >
                        ;               None
                        ;       < ON EXIT >
                        ;               AH : line status
                        ;               AL : modem status
0130                    init_rs         proc    near

0130  B4 40                     mov     ah,40h                  ; set function code
0132  B0 07                     mov     al,07h                  ; set 4800bps
0134  B7 03                     mov     bh,00000011b            ; set 8bit data, non parity, 1 stop bit
0136  BA 0000                   mov     dx,RS_PORT              ; set PORT number
0139  CD 14                     int     14h                     ;
013B  C3                        ret     ;
013C                    init_rs         endp
```

```
                        ;       ****************************
                        ;               open RS-232C
                        ;       ****************************
                        ;
                        ;       < NOTE >
                        ;               Open RS-232C port 0 by interrupt mode
                        ;               Buffer is controlled by DTR/DSR line
                        ;       < ON ENTRY >
                        ;               None
                        ;       < ON EXIT >
                        ;               AH : line status
                        ;               AL : modem status
013C                    open_rs         proc    near

013C  B4 41                     mov     ah,41h                  ; set function code
013E  B0 07                     mov     al,00000111b            ; DTR,RTS active & DTR/DSR cntroll ON
0140  B7 13                     mov     bh,DUMMY_XON            ; set dummy XON code
0142  B3 12                     mov     bl,DUMMY_XOFF           ; set dummy XOFF code
0144  B9 0100                   mov     cx,BUF_SIZE             ; set buffer size
0147  BF 0189 R                 mov     di,offset buffer        ; set buffer address(ES:SI)
014A  BA 0000                   mov     dx,RS_PORT              ; set RS-232C port
014D  CD 14                     int     14h                     ;
014F  C3                        ret                             ;

0150                    open_rs         endp

                        ;       ****************************
                        ;               Close RS-232C
                        ;       ****************************
                        ;
                        ;       < NOTE >
                        ;               Close RS-232C port 0
                        ;       < ON ENTRY >
                        ;               None
0150                    close_rs        proc    near

0150  B4 42                     mov     ah,42h                  ; set function code
0152  BA 0000                   mov     dx,RS_PORT              ; set RS-232C port
0155  CD 14                     int     14h                     ;

0157                    close_rs        endp

                        ;       ************************
                        ;               Send data
                        ;       ************************
                        ;
                        ;       < NOTE >
                        ;               Send a character data to RS-232C port 0
                        ;       < ON ENTRY >
                        ;               AL : character data
                        ;       < ON EXIT >
                        ;               AH : sending status (when cannot send, AH is not zero)
                        ;               Z-flag : set when completed
0157                    put_rs          proc    near

0157  B4 44                     mov     ah,44h                  ; set function code
0159  BA 0000                   mov     dx,RS_PORT              ; set RS-232C port
015C  CD 14                     int     14h                     ;
015E  0A E4                     or      ah,ah                   ;
0160  C3                        ret

0161                    put_rs          endp



                        ;       ****************************
                        ;               receive data
                        ;       ****************************
                        ;
                        ;       < NOTE >
                        ;               Receive a character data form receive-buffer.
                        ;       < ON ENTRY >
                        ;               None
                        ;       < ON EXIT >
                        ;               AL : received data
                        ;               AH : receiving status (when cannot receive. AH is not zero)
                        ;               Z-flag : set when completed
0161                    get_rs          proc    near

0161  50                        push    ax                      ; save data
0162  B4 43                     mov     ah,43h                  ; get RS_232C status
0164  BA 0000                   mov     dx,RS_PORT              ;
0167  CD 14                     int     14h                     ;
0169  8A D0                     mov     dl,al                   ; save receiving status
016B  58                        pop     ax                      ; recover data
016C  8A E2                     mov     ah,dl                   ;
016E  0A E4                     or      ah,ah                   ;
0170  75 09                     jnz     get_rs_exit             ; reseive data exist & error does not occuerd ?
                                                                ; jump if No
0172  B4 45                     mov     ah,45h                  ;
0174  BA 0000                   mov     dx,RS_PORT              ;
0177  CD 14                     int     14h                     ; reserve data
0179  0A E4                     or      ah,ah                   ; set Z-flag

017B                    get_rs_exit:
017B  C3                        ret

017C                    get_rs          endp
```

```
                        ;       ***********************************************
                        ;               Get a character form keyboard
                        ;       ***********************************************
                        ;       < NOTE >
                        ;               Get one character data from keyboard
                        ;       < ON ENTRY >
                        ;               None
                        ;       < ON EXIT >
                        ;               Z-flag = 0 : code is available
                        ;                        AL : character code
                        ;                        AH : scan code
                        ;               Z-flag = 1 : no code available
017C                    get_kb          proc    near

017C  B4 01                     mov     ah,1                    ; check input key existence
017E  CD 16                     int     16h                     ;
0180  74 06                     jz      get_kb_exit             ; jump if it does not exist
0182  B4 00                     mov     ah,0                    ;
0184  CD 16                     int     16h                     ; get data
0186  0B C0                     or      ax,ax                   ; clear Z-flag

0188                    get_kb_exit:
0188  C3                       ret

0189                    get_kb          endp


                        ;       << communication buffer >>

0189                    buffer  label   byte
0189  0100 [            db      256 dup(?)
              ??
                 ]


0289                    code    ends
                        end     int14h
```

(6) INT 15H / Keyboard interrupt

```
                                page    54,132
                        ;===============================================================================
                        ;                       << SAMPLE PROGRAM FOR INT 15H >>
                        ;       < NOTE >
                        ;               This program expands INT 15H fook process on function 91h,
                        ;               interrupt complete. This is an example of keyboard interrupt.
                        ;===============================================================================
0000                    code    segment byte public
                                assume  cs:code,ds:code,es:code,ss:code

                        ;       <> constant define <>

= 0054                  INT15H_ENT      equ     15h*4           ; INT 15H vercter address
= 0020                  HOOK_SIZE       equ     20h             ; hook program size as 200h bytes

0100                            org     100h

                        ;       ****************************************
                        ;               INT 15H install program
                        ;       ****************************************
                        ;       < NOTE >
                        ;               This program installs INT 15H expanded program
0100                    int15h          proc    near

0100  FA                        cli                             ; disable interrupts
0101  B8 0000                  mov     ax,0                    ;
0104  8E D8                     mov     ds,ax                   ; set segment 0
0106  BB 0054                  mov     bx,INT15H_ENT           ; Int 15h vector offset address

0109  B8 011E R                 mov     ax,offset hook          ; set Int 15h verctor
010C  89 07                     mov     word ptr [bx],ax        ; ( set offset)
010E  83 C3 02                  add     bx,2                    ;
0111  8C C8                     mov     ax,cs                   ;
0113  89 07                     mov     word ptr [bx],ax        ; (set segment)
0115  FB                        sti                             ;

0116  B8 3100                   mov     ax,3100H                ; terminate this program
0119  BA 0020                   mov     dx,HOOK_SIZE            ; set resident program size
011C  CD 21                     int     21h                     ;

011E                    int15h          endp

                        ;       ************************************
                        ;               INT 15H hook program
                        ;       ************************************
                        ;       < NOTE >
                        ;               This program is a hook process for INT 15H.
                        ;               This program is called by some BIOS interrupt process completion,
                        ;               but nothing executen except called by keybord interrupt with
                        ;               a key click realization.
                        ;       < ON ENTRY >
                        ;               AH : function code
                        ;                       91H : called when interrupt completed
                        ;               AL : case
                        ;                       02H : by keyboard interrupt
011E                    hook            proc    far

011E  9C                        pushf                           ; Save flag register
011F  3D 9102                   cmp     ax,9102h                ; if key interrupt complete ?
0122  75 1A                     jnz     hook_exit               ; jump if No
0124  50                        push    ax                      ; save registers
0125  51                        push    cx                      ;
0126  52                        push    dx                      ;

0127  B4 09                     mov     ah,9                    ; buzzer OFF
0129  B9 0000                   mov     cx,0                    ;
012C  BA 0000                   mov     dx,0                    ;
012F  CD 18                     int     18h                     ;

0131  B4 09                     mov     ah,9                    ; Buzzer ON as 1000Hz 50msec.
0133  B9 03E8                   mov     cx,1000                 ;
0136  BA 0032                   mov     dx,50                   ;
0139  CD 18                     int     18h                     ;

013B  5A                        pop     dx                      ; restore registers
013C  59                        pop     cx                      ;
013D  58                        pop     ax                      ;

013E                    hook_exit:
013E  9D                        popf                            ; rstore flag register
013F  CF                        iret                            ;

0140                    hook            endp
0140                    code    ends

                                end     int15h
```

(7) INT 16H / Keyboard scan

```
                                ;       page    54,132
                                ;       ===============================================================================
                                ;                       << SAMPLE PROGRAM FOR INT 16H >>
                                ;       < NOTE >
                                ;               This program displays keybord status, scan code and character
                                ;               code.
                                ;       ===============================================================================
0000                            code    segment byte public
                                        assume  cs:code,ds:code,es:code,ss:code
                                ;       <> external define <>
                                        extrn   dsphexb:near            ; print a word data in hexadecimal form
                                        extrn   dsphexw:near            ; print a word data in hexadecimal form
                                        extrn   dspchar:near            ; print a character
                                        extrn   dspmsg:near             ; print a message
                                        extrn   set_cursor:near         ; set a cursor position
0100                                    org     100h
                                ;       ****************************
                                ;               MAIN PROGRAM
                                ;       ****************************
                                ;       < NOTE >
                                ;               This program displays keybord status, scan code and character
                                ;               code. If CTRL+'C' is pressed, this program will be terminated.
0100                            int16h          proc    near
0100  B8 0003                          mov     ax,0003h                ; clear screen
0103  CD 10                            int     10h                     ;
0105  BE 017C R                        mov     si,offset msg           ; print message
0108  E8 0000 E                        call    dspmsg                  ;
010B                            int16h_loop:
010B  B4 01                             mov     ah,1                    ; check key buffer
010D  CD 16                             int     16h                     ;
010F  75 12                             jnz     int16h_in               ; jump if key exist
0111  B4 02                             mov     ah,2                    ; check keyboard shift status
0113  CD 16                             int     16h                     ;
0115  3B 06 0172 R                     cmp     ax,oldst                ; check old status
0119  74 F0                             jz      int16h_loop             ; jump if not deffered
011B  A3 0172 R                         mov     oldst,ax                ; save new status
011E  E8 0135 R                         call    dsp_keys                ; display shift status
0121  EB E8                             jmp     int16h_loop             ; ### LOOP ###
0123                            int16h_in:
0123  B4 00                             mov     ah,0                    ; Get input key
0125  CD 16                             int     16h                     ;
0127  50                                push    ax                      ;
0128  E8 0154 R                         call    dsp_keyd                ; display caracter code & scan code
012B  58                                pop     ax                      ;
012C  3C 03                             cmp     al,3                    ; Ctrl + 'C' ?
012E  75 DB                             jnz     int16h_loop             ; if yes, then exit
0130  B8 4C00                           mov     ax,4c00h                ; terminate this program
0133  CD 21                             int     21h
0135                            int16h          endp
                                ;       ****************************************
                                ;               display key shift status
                                ;       ****************************************
                                ;       < NOTE >
                                ;               display key shift status
                                ;       < ON ENTRY >
                                ;               None
0135                            dsp_keys        proc    near
0135  BE 0174 R                        mov     si,offset on_msg        ; message 'ON'
0138  BF 0178 R                        mov     di,offset off_msg       ; message 'OFF'
013B  BA 0212                           mov     dx,0212h                ; set a cursor position
013E  B9 0008                           mov     cx,8                    ; set a counter
0141                            dsp_keys_loop:
0141  E8 0000 E                        call    set_cursor              ; set a cursor position
0144  56                                push    si                      ; save message pointer
0145  D0 C0                             rol     al,1                    ; key status --> Carry
0147  72 02                             jc      dsp_keys_10             ; Jump if this status is active
0149  8B F7                             mov     si,di                   ; else change massage pointer to 'OFF'
014B                            dsp_keys_10:
014B  E8 0000 E                        call    dspmsg                  ; display 'ON' or 'OFF'
014E  5E                                pop     si                      ; recover message pointer
014F  FE C6                             inc     dh                      ; change a cursor row poition
0151  E2 EE                             loop    dsp_keys_loop           ; LOOP
0153  C3                                ret
0154                            dsp_keys        endp
```

```
                        ;       ***************************************************
                        ;               display character code & scan code
                        ;       ***************************************************
                        ;
                        ;       < NOTE >
                        ;               display a character code & a scan code
                        ;       < ON ENTRY >
                        ;               AH : scan code
                        ;               AL : character code
0154                    dsp_keyd        proc    near

0154  BA 0012                           mov     dx,0012h                ;
0157  E8 0000 E                         call    set_cursor              ; set a cursor position
015A  E8 0000 E                         call    dsphexb                 ; display a character code (hexadecimal)

015D  BA 0015                           mov     dx,0015h                ;
0160  E8 0000 E                         call    set_cursor              ; set a cursor position
0163  E8 0000 E                         call    dspchar                 ; display a character code

0166  BA 0112                           mov     dx,0112h                ;
0169  E8 0000 E                         call    set_cursor              ; set a cursor position
016C  8A C4                             mov     al,ah                   ;
016E  E8 0000 E                         call    dsphexb                 ; display a scan code (hexadecimal)
0171  C3                                ret

0172                    dsp_keyd        endp

0172  FFFF             oldst   dw      0ffffh                  ; old key status


                        ;       <> message define <>

0174  4F 4E 20 00       on_msg  db      'ON ',0
0178  4F 46 46 00       off_msg db      'OFF',0

017C                    msg     label byte
017C  63 68 61 72 61 63         db      'character code  :   ( )',0dh,0ah
      74 65 72 20 63 6F
      64 65 20 20 3A 20
      20 20 28 20 29 0D
      0A

0195  73 63 61 6E 20 63         db      'scan code       : ',0dh,0ah
      6F 64 65 20 20 20
      20 20 20 20 3A 20
      0D 0A
01A9  20 20 20 20 20 49 6E      db      '    Ins Lock    : ',0dh,0ah
      73 20 4C 6F 63 6B
      20 20 20 20 3A 20
      0D 0A
01BD  20 20 20 20 20 43 61      db      '    Caps Lock   : ',0dh,0ah
      70 73 20 4C 6F 63
      6B 20 20 20 3A 20
      0D 0A
01D1  20 20 20 20 20 4E 75      db      '    Num Lock    : ',0dh,0ah
      6D 20 4C 6F 63 6B
      20 20 20 20 3A 20
      0D 0A
01E5  20 20 20 20 53 63         db      '    Scroll Lock : ',0dh,0ah
      72 6F 6C 6C 20 4C
      6F 63 6B 20 3A 20
      0D 0A
01F9  20 20 20 20 41 6C         db      '    Alt Lock    : ',0dh,0ah
      74 20 4C 6F 63 6B
      20 20 20 20 3A 20
      0D 0A
020D  20 20 20 20 43 74         db      '    Ctrl        : ',0dh,0ah
      72 6C 20 20 20 20
      20 20 20 20 3A 20
      0D 0A
0221  20 20 20 20 53 68         db      '    Shift(L)    : ',0dh,0ah
      69 66 74 28 4C 29
      20 20 20 20 3A 20
      0D 0A
0235  20 20 20 20 53 68         db      '    Shift(R)    : '
      69 66 74 28 52 29
      20 20 20 20 3A 20
0247  00                        db      0

0248                    code    ends
                                end     int16h
```

(8) INT 17H / Print out

```
                                page    54,132
                        ;       ================================================================================
                        ;                               << SAMPLE PROGRAM FOR INT 17H >>
                        ;       < NOTE >
                        ;               This program prints out messages to a priner. It displays cause
                        ;               of errors if any errors are detected.
                        ;               This program prints out to only printer port NO 0. If you wont
                        ;               to print out to the other ports, you need to change PRN_NO
                        ;               (ex. : 3 for a cartridge printer H)
                        ;       ================================================================================
0000                    code    segment byte public
                                assume  cs:code,ds:code,es:code,ss:code
                        ;       <> exteral define <>
                                extrn   dspmsg:near     ; display message
                        ;       <> constant define <>
= 0000                  PRN_NO  equ     0               ; printer code
0100                            org     100h
                        ;       ******************************
                        ;               Main program
                        ;       ******************************
                        ;       < NOTE >
                        ;               This program prints out message to a priner and displays cause
                        ;               of errors if any error is detected.
0100                    int17h          proc    near

0100  BE 0175 R                 mov     si,offset msg   ; set printout message pointer
0103  BA 0000                   mov     dx,PRN_NO       ; set printer number

0106                    int17h_loop:
0106  E8 0123 R                 call    lst_stat        ; check a printer status
0109  72 10                     jc      int17h_err      ; jump if error
010B  D0 C4                     rol     ah,1            ; check a busy bit
010D  73 F7                     jnc     int17h_loop     ;

010F  8A 04                     mov     al,[si]         ; get a character
0111  46                        inc     si              ; increase a pointer
0112  3C 00                     cmp     al,0            ; terminate character ?
0114  74 08                     jz      int17h_exit     ; jump if Yes

0116  E8 0133 R                 call    lst_out         ; print out a character
0119  73 EB                     jnc     int17h_loop     ; ### loop ###

011B                    int17h_err:
011B  E8 0143 R                 call    lst_err         ; call error handler

011E                    int17h_exit:
011E  B8 4C00                   mov     ax,4c00h        ; terminate this program
0121  CD 21                     int     21h             ;

0123                    int17h          endp

                        ;       ************************************
                        ;               check printer status
                        ;       ************************************
                        ;       < NOTE >
                        ;               check a printer status
                        ;       < ON ENTRY >
                        ;               None
                        ;       < ON EXIT >
                        ;               AH              : printer status
                        ;               C-flag ON       : error detect
                        ;               C-Flag ON       : printer available
0123                    lst_stat        proc    near

0123  B4 02                     mov     ah,2            ;
0125  CD 17                     int     17h             ; get printer status
0127  F6 C4 10                 test    ah,00010000b    ; printer selected ?
012A  74 05                     jz      lst_stat_err    ; jump if No
012C  F6 C4 29                  test    ah,00101001b    ; paper out, printer error or time out ?
012F  74 01                     jz      lst_stat_exit   ; jump if No

0131                    lst_stat_err:
0131  F9                        stc                     ; set carry-flag on an error

0132                    lst_stat_exit:
0132  C3                       ret

0133                    lst_stat        endp
```

```
                        ;       ************************************
                        ;               print out a character
                        ;       ************************************
                        ;       < NOTE >
                        ;               print out one character
                        ;       < ON ENTRY >
                        ;               AL : character data
                        ;       < ON EXIT >
                        ;               AH          : printer status (bit 7 is a ready status)
                        ;               C-flag ON   : error detect
                        ;               C-Flag ON   : printer available

0133                                    lst_out         proc    near

0133  B4 00                                     mov     ah,0            ; print out a character.
0135  CD 17                                     int     17h             ;
0137  F6 C4 10                                  test    ah,00010000b    ; printer selected ?
013A  74 05                                     jz      lst_out_err     ; if No then error
013C  F6 C4 29                                  test    ah,00101001b    ; paper out, printer error or time out?
013F  74 01                                     jz      lst_out_exit    ; jump if No

0141                                    lst_out_err:
0141  F9                                        stc                     ; set carry-flag on an error return

0142                                    lst_out_exit:
0142  C3                                       ret

0143                                    lst_out         endp


                        ;       ****************************
                        ;               error handler
                        ;       ****************************
                        ;       < NOTE >
                        ;               display error messages
                        ;       < ON ENTRY >
                        ;               AH : printer status
                        ;       < ON EXIT >
                        ;               None

0143                                    lst_err         proc    near

0143  BE 0191 R                                 mov     si,offset err_timeout   ; set time out error message
0146  F6 C4 01                                  test    ah,00000001b            ; time out error ?
0149  74 03                                     jz      lst_err_10              ; jump if No
014B  E8 0000 E                                 call    dspmsg                  ; if Yes then display the message
014E                                    lst_err_10:
014E  83 FA 03                                  cmp     dx,3                    ; if cartridge printer is selected
0151  74 21                                     jz      lst_err_40              ; then skip following checks

0153  BE 01AA R                                 mov     si,offset err_prterr    ; set printer error message
0156  F6 C4 08                                  test    ah,00001000b            ; printer error ?
0159  74 03                                     jz      lst_err_20              ; jump if No
015B  E8 0000 E                                 call    dspmsg                  ; if Yes, then display the message

015E                                    lst_err_20:
015E  BE 01C2 R                                 mov     si,offset err_select    ; set select error message
0161  F6 C4 10                                  test    ah,00010000b            ; select error ?
0164  75 03                                     jnz     lst_err_30              ; jump if No
0166  E8 0000 E                                 call    dspmsg                  ; if Yes, then display the message

0169                                    lst_err_30:
0169  BE 01E4 R                                 mov     si,offset err_nopaper   ; set paper out error message
016C  F6 C4 20                                  test    ah,00100000b            ; paper out ?
016F  74 03                                     jz      lst_err_40              ; jump if No
0171  E8 0000 E                                 call    dspmsg                  ; if Yes, then display the message

0174                                    lst_err_40:
0174  C3                                        ret


0175                                    lst_err         endp

                        ;       <> print out message define >

0175  42 49 4F 53 20 49 msg             db      'BIOS INT 17H TEST PROGRAM',0dh,0ah,0
      4E 54 20 31 37 48
      20 54 45 53 54 20
      50 52 4F 47 52 41
      4D 0D 0A 00


                        ;       <> error messages define <>

0191  21 21 21 20 74 69 err_timeout     db      '!!! time out error !!!',0dh,0ah,0
      6D 65 20 6F 75 74
      20 65 72 72 6F 72
      20 21 21 21 0D 0A
      00
01AA  21 21 21 20 70 72 err_prterr      db      '!!! printer error !!!',0dh,0ah,0
      69 6E 74 65 72 20
      65 72 72 6F 72 20
      21 21 21 0D 0A 00
01C2  21 21 21 20 70 72 err_select      db      '!!! printer does not select !!!',0dh,0ah,0
      69 6E 74 65 72 20
      64 6F 65 73 20 6E
      6F 74 20 73 65 6C
      65 63 74 20 21 21
      21 0D 0A 00
01E4  21 21 21 20 70 61 err_nopaper     db      '!!! paper  out !!!',0dh,0ah,0
      70 65 72 20 20 6F
      75 74 20 21 21 21
      0D 0A 00

01F9                                    code    ends
                                        end     int17h
```

(9) INT 18H / Read system values

```
                                page    54,132
                        ;       ================================================================================
                        ;
                        ;               << SAMPLE PROGRAM FOR INT 18H (sub function 00) >>
                        ;
                        ;       < NOTE >
                        ;               This program reads and displays system values.
                        ;
                        ;       ================================================================================
0000                            code    segment byte public
                                assume  cs:code,ds:code,es:code,ss:code
                        ;       <> extrenal define <>
                                extrn   dspmsg:near     ; display message
                                extrn   dsphexb:near    ; display a byte data in hexadecimal form

0100                            org     100h
                        ;       ****************************
                        ;               Main program
                        ;       ****************************
                        ;       < NOTE >
                        ;               This program reads and displays system values.
0100                    int18h00        proc    near
                        ;       < Auto power off time >
0100  BE 01D1 R                 mov     si,offset fn00          ; dispaly message
0103  E8 0000 E                 call    dspmsg                  ; read auto power off time
0106  B8 0000                   mov     ax,0                    ;
0109  CD 18                     int     18h                     ; display the time read
010B  E8 0000 E                 call    dsphexb                 ; wait for key-in
010E  B4 00                     mov     ah,0                    ;
0110  CD 16                     int     16h                     ;
                        ;       < Auto back light off time >
0112  BE 01F3 R                 mov     si,offset fn01          ; display message
0115  E8 0000 E                 call    dspmsg                  ; read auto back light off time
0118  B8 0001                   mov     ax,1                    ;
011B  CD 18                     int     18h                     ;
011D  E8 0000 E                 call    dsphexb                 ; display the time read
0120  B4 00                     mov     ah,0                    ; wait for key-in
0122  CD 16                     int     16h                     ;
                        ;       < Auto cartridge printer power off time >
0124  BE 021B R                 mov     si,offset fn02          ; display message
0127  E8 0000 E                 call    dspmsg                  ;
012A  B8 0002                   mov     ax,2                    ; read auto cartridge printer off time
012D  CD 18                     int     18h                     ;
012F  E8 0000 E                 call    dsphexb                 ; display the time read
0132  B4 00                     mov     ah,0                    ; wait for key-in
0134  CD 16                     int     16h                     ;
                        ;       < Power ON start Mode >
0136  BE 024A R                 mov     si,offset fn03          ; display message
0139  E8 0000 E                 call    dspmsg                  ;
013C  B8 0003                   mov     ax,3                    ; read current Power ON start mode
013F  CD 18                     int     18h                     ;
0141  BE 02A1 R                 mov     si,offset fn03_00       ; set message pointer for warm-start
0144  3C 00                     cmp     al,0                    ; worm-start ?
0146  74 03                     jz      int1800_3x              ; jump if Yes
0148  BE 02AC R                 mov     si,offset fn03_01       ; set message pointer for resume-start
014B                    int1800_3x:
014B  E8 0000 E                 call    dspmsg                  ; display current mode
014E  B4 00                     mov     ah,0                    ; wait for key-in
0150  CD 16                     int     16h                     ;
                        ;       <Keyboard country >
0152  BE 0263 R                 mov     si,offset fn04          ; display message
0155  E8 0000 E                 call    dspmsg                  ;
0158  B8 0004                   mov     ax,4                    ; read current keyboard mode
015B  CD 18                     int     18h                     ;
015D  BE 02B9 R                 mov     si,offset fn04_00       ; set message pointer for ASCII
0160  3C 00                     cmp     al,0                    ; ASCII ?
0162  74 11                     jz      int1800_4x              ; jump if Yes
0164  BE 02BF R                 mov     si,offset fn04_01       ; set message pointer for EUROPE I
0167  FE C8                     dec     al                      ; EUROPE I(except for GERMAN & FRANCE) ?
0169  74 0A                     jz      int1800_4x              ; jump if Yes
016B  BE 02CD R                 mov     si,offset fn04_02       ; set message pointer for EUROPE II
016E  FE C8                     dec     al                      ; EUROPE II (GERMAN & FRANCE) ?
0170  74 03                     jz      int1800_4x              ; jump if Yes
0172  BE 02D6 R                 mov     si,offset fn04_03       ; set message pointer for KANA
0175                    int1800_4x:
0175  E8 0000 E                 call    dspmsg                  ; display country
0178  B4 00                     mov     ah,0                    ;
017A  CD 16                     int     16h                     ;
                        ;       < Character set >
017C  BE 0279 R                 mov     si,offset fn05          ; display massage
017F  E8 0000 E                 call    dspmsg                  ;
0182  B8 0005                   mov     ax,5                    ; read current character set
0185  CD 18                     int     18h                     ;
0187  BE 02DB R                 mov     si,offset fn05_00       ; set message pointer for ASCII light
018A  3C 00                     cmp     al,0                    ; ASCII light ?
018C  74 11                     jz      int1800_5x              ; jump if Yes
018E  BE 02E7 R                 mov     si,offset fn05_01       ; set message pointer for ASCII bold
0191  FE C8                     dec     al                      ; ASCII bold ?
0193  74 0A                     jz      int1800_5x              ; jump if Yes
0195  BE 02F2 R                 mov     si,offset fn05_02       ; set message pointer for Scandinavian
0198  FE C8                     dec     al                      ; Scandinavian ?
019A  74 03                     jz      int1800_5x              ; jump if Yes
019C  BE 02FF R                 mov     si,offset fn05_03       ; set message pointer for KANA
019F                    int1800_5x:
019F  E8 0000 E                 call    dspmsg                  ; display current character set
01A2  B4 00                     mov     ah,0                    ;
01A4  CD 16                     int     16h                     ; wait for key-in
```

```
                        < Printer Port >
01A5  BE 028C R         mov     si,offset fn06          ; display message
01A9  E8 0000 E         call    dspmsg                  ;
01AC  B8 0006           mov     ax,6                    ; read current printer port
01AF  CD 18             int     18h                     ;
01B1  BE 0304 R         mov     si,offset fn06_00       ; set message pointer for port 0
01B4  3C 00             cmp     al,0                    ; port 0 ?
01B6  74 11             jz      int1800_6x              ; jump if Yes
01B8  BE 0315 R         mov     si,offset fn06_01       ; set message pointer for port 1
01BB  FE C8             dec     al                      ; Port 1 ?
01BD  74 0A             jz      int1800_6x              ; jump if Yes
01BF  BE 0321 R         mov     si,offset fn06_02       ; set message pointer for port 2
01C2  FE C8             dec     al                      ; Port 2 ?
01C4  74 03             jz      int1800_6x              ; jump if Yes
01C6  BE 032D R         mov     si,offset fn06_03       ; set message pointer for port 3

01C9                    int1800_6x:
01C9  E8 0000 E         call    dspmsg                  ; display current printer port

01CC  B8 4C00           mov     ax,4c00h                ; terminate this porgram
01CF  CD 21             int     21h                     ;

01D1                    int18h00        endp

                        ;       <> message define <>
01D1  0D 0A 41 75 74 6F fn00    db      0dh,0ah,'Auto powr off time.(minutes) : ',0
      20 70 6F 77 72 20
      6F 66 66 20 74 69
      6D 65 2E 28 6D 69
      6E 75 74 65 73 29
      20 3A 20 00
01F3  0D 0A 41 75 74 6F fn01    db      0dh,0ah,'Auto back light off time.(minutes) : ',0
      20 62 61 63 6B 20
      6C 69 67 68 74 20
      6F 66 66 20 74 69
      6D 65 2E 28 6D 69
      6E 75 74 65 73 29
      20 3A 20 00
021B  0D 0A 41 75 74 6F fn02    db      0dh,0ah,'Auto cartridge printer off time.(minutes) : ',0
      20 63 61 72 74 72
      69 64 67 65 20 70
      72 69 6E 74 65 72
      20 6F 66 66 20 74
      69 6D 65 2E 28 6D
      69 6E 75 74 65 73
      29 20 3A 20 00
024A  0D 0A 50 6F 77 65 fn03    db      0dh,0ah,'Power on start mode : ',0
      72 20 6F 6E 20 73
      74 61 72 74 20 6D
      6F 64 65 20 3A 20
      00
0263  0D 0A 6B 65 79 62 fn04    db      0dh,0ah,'keyboard country : ',0
      6F 61 72 64 20 63
      6F 75 6E 74 72 79
      20 3A 20 00
0279  0D 0A 43 68 61 72 fn05    db      0dh,0ah,'Character set : ',0
      61 63 74 65 72 20
      73 65 74 20 3A 20
      00
028C  0D 0A 44 65 66 61 fn06    db      0dh,0ah,'Default printer : ',0
      75 6C 74 20 70 72
      69 6E 74 65 72 20
      3A 20 00

02A1  57 6F 72 6D 20 73 fn03_00 db      'Worm start',0
      74 61 72 74 00
02AC  52 65 73 75 6D 65 fn03_01 db      'Resume start',0
      20 73 74 61 72 74
      00

02B9  41 53 43 49 49 00 fn04_00 db      'ASCII',0
02BF  47 45 52 4D 41 4E fn04_01 db      'GERMAN,FRENCH',0
      2C 46 52 45 4E 43
      48 00
02CD  45 55 52 4F 50 45 fn04_02 db      'EUROPEAN',0
      41 4E 00
02D6  4B 41 4E 41 00    fn04_03 db      'KANA',0

02DB  41 53 43 49 49 20 fn05_00 db      'ASCII light',0
      6C 69 67 68 74 00
02E7  41 53 43 49 49 20 fn05_01 db      'ASCII bold',0
      62 6F 6C 64 00
02F2  53 63 61 6E 64 69 fn05_02 db      'Scandinavian',0
      6E 61 76 69 61 6E
      00
02FF  4B 41 4E 41 00    fn05_03 db      'KANA',0

0304  42 75 69 6C 74 20 fn06_00 db      'Built in printer',0
      69 6E 20 70 72 69
      6E 74 65 72 00
0315  72 65 73 65 72 76 fn06_01 db      'reserve (1)',0
      65 20 28 31 29 00
0321  72 65 73 65 72 76 fn06_02 db      'reserve (2)',0
      65 20 28 32 29 00
032D  43 61 72 74 72 69 fn06_03 db      'Cartridge printer',0
      64 67 65 20 70 72
      69 6E 74 65 72 00


033F                    code    ends
                                end     int18h00
```

(10) INT 18H / Define touch – keys

```
                                ;       page    54,132
                                ;===============================================================================
                                ;               << SAMPLE PROGRAM FOR INT 18H (TOUCH KEY DEFINE) >>
                                ;       < NOTE >
                                ;                       This program defines and displays touch keyboard.
                                ;===============================================================================
0000                            code    segment byte public
                                        assume  cs:code,ds:code,es:code,ss:code

0100                                    org     100h

                                ;       ****************************
                                ;               main program
                                ;       ****************************
                                ;       < NOTE >
                                ;                       This program defines and displays touch keys on a touch keyboard.
0100                            int18h10        proc    near

0100  B8 4000                           mov     ax,4000h                ; Clear touch keyboard LCD (set mode 0)
0103  CD 10                             int     10h                     ;

0105  B8 1002                           mov     ax,1002h                ; initialize touch key (all disable)
0108  BA 0000                           mov     dx,0                    ;
010B  CD 18                             int     18h                     ;

010D  B4 14                             mov     ah,14h                  ; set a function code
010F  B0 20                             mov     al,20h                  ; set a key position (top left side)
0111  B3 4F                             mov     bl,4fh                  ; set display attribite (box)
0113  B6 01                             mov     dh,01h                  ; set key attribute (click ON)
0115  BF 012E R                         mov     di,offset keydsp        ; set key display pointer
0118  B9 0060                           mov     cx,8*12                 ; set key block number (loop counter)

011B                            int18h10_loop:
011B  51                                push    cx                      ;
011C  8A 15                             mov     dl,[di]                 ; set a key code (same as display data).
011E  B9 0101                           mov     cx,0101h                ; set a key number (one key only)
0121  CD 18                             int     18h                     ; define touch key & display
0123  FE C0                             inc     al                      ; increase key position
0125  47                                inc     di                      ; increase key display pointer
0126  59                                pop     cx                      ; if all defined
0127  E2 F2                             loop    int18h10_loop           ; if not then loop

0129  B8 4C00                           mov     ax,4c00h                ; terminate this porgram
012C  CD 21                             int     21h                     ;

012E                            int18h10        endp

                                ;       <> display data define <>
012E                            keydsp  label   byte
012E  20 21 22 23 24 25                 db      20h,21h,22h,23h,24h,25h,26h,27h,28h,29h,2ah,2bh,2ch,2dh,2eh,2fh
      26 27 28 29 2A 2B
      2C 2D 2E 2F
013E  30 31 32 33 34 35                 db      30h,31h,32h,33h,34h,35h,36h,37h,38h,39h,3ah,3bh,3ch,3dh,3eh,3fh
      36 37 38 39 3A 3B
      3C 3D 3E 3F
014E  40 41 42 43 44 45                 db      40h,41h,42h,43h,44h,45h,46h,47h,48h,49h,4ah,4bh,4ch,4dh,4eh,4fh
      46 47 48 49 4A 4B
      4C 4D 4E 4F
015E  50 51 52 53 54 55                 db      50h,51h,52h,53h,54h,55h,56h,57h,58h,59h,5ah,5bh,5ch,5dh,5eh,5fh
      56 57 58 59 5A 5B
      5C 5D 5E 5F
016E  60 61 62 63 64 65                 db      60h,61h,62h,63h,64h,65h,66h,67h,68h,69h,6ah,6bh,6ch,6dh,6eh,6fh
      66 67 68 69 6A 6B
      6C 6D 6E 6F
017E  70 71 72 73 74 75                 db      70h,71h,72h,73h,74h,75h,76h,77h,78h,79h,7ah,7bh,7ch,7dh,7eh,7fh
      76 77 78 79 7A 7B
      7C 7D 7E 7F

018E                            code    ends

                                        end     int18h10
```

(11) INT 18H / Read Barcode (UART)

```
                                        page    54,132
                                ;       ===============================================================================
                                ;
                                ;               << SAMPLE PROGRAM FOR INT 18H (UART:barcode) >>
                                ;
                                ;       < NOTE >
                                ;               Open UART for barcode reader, read a barcode and display it
                                ;               Barcode reader used for this program is asynchronous type
                                ;               with a communication parameter of 9600 bps, EVEN parity,
                                ;               7 bit/char, 1 stop bit, MARK=1.
                                ;
                                ;       ===============================================================================

0000                            code    segment
                                        assume  cs:code,ds:code,es:code

                                ;       <> extrenal define <>
                                        extrn   dspmsg:near     ; display message.

0100                                    org     100h

                                ;       ************************************************
                                ;               Asynchronous type Barcode reader
                                ;       ************************************************
                                ;
                                ;       < NOTE >
                                ;               Open UART for barcode reader, read a barcode and then display
                                ;               the barcode read.

0100                            start:
0100  B8 2002                          mov     ax,20h*256+2            ; MARK=1
0103  BB 01C7                          mov     bx,1*256+11000111b      ; timeout=1sec,
                                                                        ; 9600bps, EVENparity, 7bit/char, 1stop bit
0106  B9 0100                          mov     cx,256                  ; set Rx buffer size
0109  BF 014A R                        mov     di,offset queu          ; set Rx buffer offset address
010C  CD 18                             int     18h                     ; initialize UART
010E  B4 21                             mov     ah,21h
0110  B1 50                             mov     cl,'P'                  ; command to disable barcode type ouput
0112  CD 18                             int     18h                     ; send a command to BCR
0114  72 19                             jb      error                   ; if timeout detected,
0116                            wait:
0116  B4 22                             mov     ah,22h
0118  CD 16                             int     18h                     ; receive a character from UART
011A  72 13                             jb      error                   ; if error detected,
011C  74 F8                             je      wait                    ; if data unavailable,
011E  B3 07                             mov     bl,7                    ; set attribute to display
0120  B4 0E                             mov     ah,0eh
0122  CD 10                             int     10h                     ; display a character
0124  3C 0D                             cmp     al,0dh                  ;
0126  75 EE                             jne     wait                    ;
0128  B8 0E0A                           mov     ax,0e0ah                ;
012B  CD 10                             int     10h                     ; if CR is received, append LF
012D  EB E7                             jmp     wait
012F                            error:
012F  BE 013F R                         mov     si,offset errmsg        ;
0132  E8 0000 E                         call    dspmsg                  ;
0135  B8 2000                           mov     ax,20h*256+0            ;
0138  CD 18                             int     18h                     ; disable UART
013A  B8 4C00                           mov     ax,4c00h                ;
013D  CD 21                             int     21h                     ; return to DOS


                                ;       <> error message <>

013F  0D 0A 45 72 72 6F         errmsg: db      0dh,0ah,'Error!',0dh,0ah,0
      72 21 0D 0A 00

                                ;       <> buffer define <>

014A  0100 [                    queu:   db      256 dup(?)
              ??
                 ]


024A                            code    ends
                                        end     start
```

(12) INT 18H / Read Barcode

```
                                ;       page    54,132
                                ;       ===============================================================================
                                ;               << SAMPLE PROGRAM FOR INT 18H (barcode reader) >>
                                ;       < NOTE >
                                ;                This program opens barcode reader, reads a barcode and displays.
                                ;       !!! CAUTION !!!
                                ;                BARCODE.COM must be installed before this routine is executed.
                                ;       ===============================================================================
0000                            code    segment
                                        assume  cs:code,ds:code,es:code
                                ;       <> extrenal define <>
                                        extrn   dspmsg:near     ; display message
0100                                    org     100h
                                ;       ***********************************************
                                ;               Barcode reader sample program
                                ;       ***********************************************
                                ;       < NOTE >
                                ;        This program opens barcode reader, reads a barcode and displays data.
0100                            start:
0100  B8 3000                          mov     ax,30h*256+0            ; set multi-read
0103  BB 0000                          mov     bx,0                    ; set default parameter
0106  CD 18                            int     18h                     ; initialize ICF interface
0108  F6 C4 80                        test    ah,80h                  ;
010B  74 17                            je      error                   ; error if BARCODE.COM is not installed
010D                            wait:
010D  B4 32                            mov     ah,32h                  ;
010F  CD 18                            int     18h                     ; read a barcode
0111  74 FA                            je      wait                    ; waite if data unavailable
0113  B3 07                            mov     bl,7                    ; set attribute to display
0115  B4 0E                            mov     ah,0eh                  ;
0117  CD 10                            int     10h                     ; display a character
0119  3C 0D                            cmp     al,0dh                  ;
011B  75 F0                            jne     wait                    ;
011D  B8 0E0A                          mov     ax,0e0ah                ;
0120  CD 10                            int     10h                     ; if CR is received, append LF
0122  EB E9                            jmp     wait                    ;
0124                            error:
0124  BE 012F R                        mov     si,offset errmsg        ;
0127  E8 0000 E                        call    dspmsg                  ;
012A  B8 4C00                          mov     ax,4c00h                ;
012D  CD 21                            int     21h                     ; Return to DOS
                                ;       <> error message <>
012F  0D 0A 42 41 52 43 errmsg: db      0dh,0ah,'BARCODE.COM not installed!',0dh,0ah,0
      4F 44 45 2E 43 4F
      4D 20 6E 6F 74 20
      69 6E 73 74 61 6C
      6C 65 64 21 0D 0A
      00
014E                            code    ends
                                        end     start
```

(13) INT 1AH : 79H / Alarm interrupt

```
                                page    54,132
                        ;       ================================================================================
                        ;
                        ;               << SAMPLE PROGRAM FOR INT 1AH, INT 79H >>
                        ;
                        ;       < NOTE >
                        ;               This program shows how to use alarm interrupt.
                        ;               This program use BIOS INT 18H function 43H to occur alarm
                        ;               interrupt and expand INT 79H hook to display alarm message.
                        ;
                        ;       ================================================================================

0000                    code    segment byte public
                                assume  cs:code,ds:code,es:code,ss:code


                        ;       <> constant value <>

= 01E4                  INT79H_ENT      equ     79h*4           ; INT 15H vercter address
= 0030                  HOOK_SIZE       equ     30h             ; hook program size 300h bytes

0100                            org     100h

                        ;       ****************************************
                        ;               INT 79H install program
                        ;       ****************************************
                        ;
                        ;       < NOTE >
                        ;               Set alarm time evry 1 minute and expand INT 79H hook to
                        ;               display messages.
                        ;
0100                    int1ah          proc    near

0100  B4 43                             mov     ah,43h                  ; set an alarm time
0102  BB FFFF                           mov     bx,0ffffh               ; set a 1 minute interval
0105  B9 FFFF                           mov     cx,0ffffh               ;
0108  BA 00FF                           mov     dx,00ffh                ;
010B  CD 1A                             int     1ah

010D  FA                                cli                             ; disable interrupts
010E  B8 0000                           mov     ax,0                    ;
0111  8E D8                             mov     ds,ax                   ; set segment 0
0113  BB 01E4                           mov     bx,INT79H_ENT           ; set Int 79h vector offset address

0116  B8 012B R                         mov     ax,offset int79h        ; set Int 79h verctor
0119  89 07                             mov     word ptr [bx],ax        ; (set offset)
011B  83 C3 02                          add     bx,2                    ;
011E  8C C8                             mov     ax,cs                   ;
0120  89 07                             mov     word ptr [bx],ax        ; (set segment)
0122  FB                                sti                             ;

0123  B8 3100                           mov     ax,3100H                ; terminate this program
0126  BA 0030                           mov     dx,HOOK_SIZE            ; set resident program size
0129  CD 21                             int     21h                     ;

012B                    int1ah          endp

                        ;       ************************************
                        ;               INT 79H hook program
                        ;       ************************************
                        ;
                        ;       < NOTE >
                        ;               This is an expanded alarm interrupt process.
                        ;               This program changes screen page 1, displays message
                        ;               and sounds a speeker. When ESC key is pressed, it
                        ;               recovers screen page 0 and exits.
                        ;
012B                    int79h          proc    far

012B  9C                                pushf                           ; save flag register
012C  50                                push    ax                      ; save registers
012D  53                                push    bx                      ;
012E  51                                push    cx                      ;
012F  52                                push    dx                      ;
0130  56                                push    si                      ;
0131  1E                                push    ds                      ;

0132  8C C8                             mov     ax,cs                   ; set DS
0134  8E D8                             mov     ds,ax                   ;

0136  80 3E 01AF R 00                   cmp     byte ptr int79h_flg,0   ; alarm display now ?
013B  75 6A                             jnz     int79h_exit             ; if Yes, then exit
013D  C6 06 01AF R 01                   mov     byte ptr int79h_flg,1   ;

0142  B8 0501                           mov     ax,0501h                ; set screen page 1
0145  CD 10                             int     10h                     ;

0147  B8 0600                           mov     ax,0600h                ; clear screen
014A  B9 0000                           mov     cx,0
014D  BA 1949                           mov     dx,1949h
0150  B7 07                             mov     bh,7
0152  CD 10                             int     10h

0154  B4 02                             mov     ah,02h                  ; set cursor position
0156  B7 01                             mov     bh,1
0158  BA 020A                           mov     dx,020ah
```

```
015B  CD 10                             int     10h                     ;
015D  BE 01B0 R                         mov     si,offset alarmmsg      ; set message pointer
0160                                int79h_10:
0160  8A 04                             mov     al,[si]                 ; display alarm message
0162  46                                inc     si                      ;
0163  3C 00                             cmp     al,0                    ;
0165  74 08                             jz      int79h_20               ;
0167  B4 0E                             mov     ah,0eh                  ;
0169  B3 07                             mov     bl,7                    ;
016B  CD 10                             int     10h                     ;
016D  EB F1                             jmp     int79h_10               ;
016F                                int79h_20:
016F  B4 09                             mov     ah,9                    ; set buzzer to OFF
0171  B9 0000                           mov     cx,0                    ;
0174  BA 0000                           mov     dx,0                    ;
0177  CD 18                             int     18h                     ;
0179  B9 0003                           mov     cx,3                    ; set loop counter
017C                                int79h_25:
017C  51                                push    cx                      ;
017D  B4 09                             mov     ah,9                    ; Buzzer ON 1000Hz 120msec
017F  B9 03E8                           mov     cx,1000                 ;
0182  BA 0078                           mov     dx,120                  ;
0185  CD 18                             int     18h                     ;
0187  B4 09                             mov     ah,9                    ; Buzzer ON 1500Hz 120msec
0189  B9 05DC                           mov     cx,1500                 ;
018C  BA 0078                           mov     dx,120                  ;
018F  CD 18                             int     18h                     ;
0191  59                                pop     cx                      ;
0192  E2 E8                             loop    int79h_25               ;
0194                                int79h_30:                          ; wait intill ESC is pressed
0194  B8 0000                           mov     ax,0                    ;
0197  CD 16                             int     16h                     ;
0199  3C 1B                             cmp     al,1bh                  ;
019B  75 F7                             jnz     int79h_30               ;
019D  B8 0500                           mov     ax,0500h                ; set page 0
01A0  CD 10                             int     10h                     ;
01A2  C6 06 01AF R 00                   mov     byte ptr int79h_flg,0   ;
01A7                                int79h_exit:
01A7  1F                                pop     ds                      ;
01A8  5E                                pop     si                      ;
01A9  5A                                pop     dx                      ; restore registers
01AA  59                                pop     cx                      ;
01AB  5B                                pop     bx                      ;
01AC  58                                pop     ax                      ;
01AD  9D                                popf                            ; rstore flag register
01AE  CF                                iret                            ;
01AF                                int79h          endp

01AF  00                            int79h_flg      db      0

                                    ;       <> alarm message <>
01B0  21 21 21 20 41 4C             alarmmsg        db      '!!! ALARM TIME !!!',0
      41 52 4D 20 54 49
      4D 45 20 21 21 21
      00

01C3                                code    ends
                                            end     int1ah
```

(14) RAM disk access

```
                                ;       page    54,132
                                ;===============================================================================
                                ;
                                ;               << SAMPLE PROGRAM FOR RAM BANK USAGE >>
                                ;
                                ;       < NOTE >
                                ;               This porgram shows how to access expanded RAM DISK.
                                ;
                                ;===============================================================================


0000                            code    segment public byte
                                        assume  cs:code,ds:code

                                ;       <> extrnal define <>

                                        extrn   dump:near               ; dump CX byte form DS:BX
                                        extrn   dsphexb:near            ; display a byte data in hexadecimal form
                                        extrn   dsphexw:near            ; display a word data in hexadecimal form
                                        extrn   dspmsg:near             ; display message
                                        extrn   dspchar:near            ; display 1 character

0100                                    org     100h

                                ;       ****************************************
                                ;               Display expansion RAM data
                                ;       ****************************************

                                ;       < NOTE >
                                ;               display expand RAM data
                                ;                       RAM BANK : C4000h
                                ;                       ADDRESS  : START_ADDR
                                ;                                  (ex. 0h  --- RAM DISK FAT & DIRECTORY)
                                ;                       SIZE     : DUMP_SIZE  (ex. 1000H --- 4096 BYTES)

0100                            main:
0100  BE 016A R                         mov     si,offset message       ; display start message
0103  E8 0000 E                         call    dspmsg                  ;

0106  B0 00                             mov     al,0                    ; RAM BANK : C4000h
0108  BA 0103                           mov     dx,259                  ;   port 0259 bit7 = 0
010B  EE                                out     dx,al                   ;   port 4259 bit7 = 0
010C  B0 00                             mov     al,0                    ;   port 8259 bit7 = 0
010E  BA 4259                           mov     dx,4259h                ;
0111  EE                                out     dx,al                   ;
0112  B0 00                             mov     al,0                    ;
0114  BA 8259                           mov     dx,8259h                ;
0117  EE                                out     dx,al                   ;

0118  C4 1E 0164 R                      les     bx,start_addr           ; ES <== upper BX <== lower
011C  8C C0                             mov     ax,es                   ;
011E  8A E0                             mov     ah,al                   ; AH <== A20-A16
0120  8A C7                             mov     al,bh                   ; AL <== A15-A8
0122  B1 06                             mov     cl,6                    ;
0124  D3 E8                             shr     ax,cl                   ; AX <== A20-A14
0126  0C 80                             or      al,80h                  ; set RAM ENABLE bit
0128  BA 4258                           mov     dx,4258h                ; BANK PORT <== 4258h
                                                                        ;  (Because A15,A14 of C8000h is 01b)
012B  EE                                out     dx,al                   ; set BANK REGISTER
012C  8B F3                             mov     si,bx                   ;
012E  81 E3 3FFF                        and     bx,3fffh                ; SI <== A13-A0

0132  8B 0E 0168 R                      mov     cx,dump_size            ; dump size (line counter)
0136  B8 C400                           mov     ax,0c400h               ; DS <== address of ROM BANK
0139  8E D8                             mov     ds,ax                   ;

013B                            repeat:
013B  51                                push    cx                      ; save loop counter
013C  8C C0                             mov     ax,es                   ; display upper address
013E  E8 0000 E                         call    dsphexb                 ;
0141  8B C6                             mov     ax,si                   ; display lower address
0143  E8 0000 E                         call    dsphexw                 ;
0146  B0 20                             mov     al,' '                  ; display two spaces
0148  E8 0000 E                         call    dspchar                 ;
014B  E8 0000 E                         call    dspchar                 ;
014E  B9 0010                           mov     cx,16                   ; display length <== 16 bytes
0151  E8 0000 E                         call    dump                    ; display ROM data
0154  83 C3 10                          add     bx,16                   ; update offset & counter
0157  83 C6 10                          add     si,16                   ;
015A  59                                pop     cx                      ;
015B  E2 DE                             loop    repeat                  ; loop

015D  B8 4C00                           mov     ax,4c00h                ; terminate this program
0160  CD 21                             int     21h                     ;


                                ;       <> data define <>

0162  C400                      bank_addr       dw      0c400h          ; bank address
0164  00 00 00 00               start_addr      dd      0h              ; start address for dump
0168  0100                      dump_size       DW      100h            ; dump size (1000h bytes)

                                ;       <> message define <>

016A  0A 0D                     message db      0ah,0dh
016C  20 20 20 20 20 20                 db                      DUMP OF RAM"
      20 20 20 20 20 20
      20 20 20 20 20 44
      55 4D 50 20 4F 46
      20 52 41 4D
0188  0A 0D 0A 0D 00                    db      0ah,0dh,0ah,0dh,0

018D                            code    ends
                                        end     main
```

(15) ROM disk access

```
                        page    54,132
                        ;===============================================================================
                        ;               << SAMPLE PROGRAM FOR ROM BANK USAGE >>
                        ; < NOTE >
                        ;               This porgram shows how to access ROM bank.
                        ;===============================================================================

0000                    code    segment public byte
                                assume  cs:code,ds:code

                        ;       <> extrnal define <>

                                extrn   dump:near               ; dump CX byte form DS:BX
                                extrn   dsphexb:near            ; display a byte data in hexadecimal form
                                extrn   dsphexw:near            ; display a word data in hexadecimal form
                                extrn   dspmsg:near             ; display message
                                extrn   dspchar:near            ; display one character
0100                            org     100h

                        ;       ************************************
                        ;               Display ROM bank data
                        ;       ************************************
                        ;       < NOTE >
                        ;               display expand ROM bank data.
                        ;                       ADDRESS : START_ADDR (ex. 78000h --- BIOS)
                        ;                       SIZE    : DUMP_SIZE  (ex. 1000H  --- 4096 BYTES)

0100                    main:
0100  BE 0158 R                 mov     si,offset message       ; display start message
0103  E8 0000 E                 call    dspmsg

0106  C4 1E 0152 R              les     bx,start_addr           ; ES <== upper BX <== lower
010A  8C C0                     mov     ax,es
010C  8A E0                     mov     ah,al                   ; AH <== A20-A16
010E  8A C7                     mov     al,bh                   ; AL <== A15-A8
0110  B1 07                     mov     cl,7
0112  D3 E8                     shr     ax,cl                   ; AX <== A20-A15
0114  0C 80                     or      al,80h                  ; set ROM ENABLE bit
0116  0C 40                     or      al,40h                  ; set RAM ENABLE bit for SYSTEM
0118  BA 11E7                   mov     dx,11e7h
011B  EE                        out     dx,al                   ; set BANK REGISTER
011C  8B F3                     mov     si,bx
011E  81 E3 7FFF                and     bx,7fffh                ; SI <== A14-A0

0122  8B 0E 0156 R              mov     cx,dump_size            ; display size (line counter)
0126  B8 F000                   mov     ax,0f000h               ; DS <== address of ROM BANK
0129  8E D8                     mov     ds,ax

012B                    repeat:
012B  51                        push    cx
012C  8C C0                     mov     ax,es                   ; display upper address
012E  E8 0000 E                 call    dsphexb
0131  8B C6                     mov     ax,si                   ; display lower address
0133  E8 0000 E                 call    dsphexw

0136  B0 20                     mov     al,' '                  ; display two space
0138  E8 0000 E                 call    dspchar
013B  E8 0000 E                 call    dspchar

013E  B9 0010                   mov     cx,16                   ; display length <== 16 bytes
0141  E8 0000 E                 call    dump                    ; display ROM data
0144  83 C3 10                  add     bx,16                   ; update offset & counter
0147  83 C6 10                  add     si,16
014A  59                        pop     cx                      ; recover counter
014B  E2 DE                     loop    repeat                  ; loop

014D  B8 4C00                   mov     ax,4c00h                ; terminate this program.
0150  CD 21                     int     21h

0152  00 80 07 00       start_addr      dd      78000h          ; start address for dump
0156  0100              dump_size       dw      100h            ; dump size (1000 bytes)

0158  0A 0D             message db      0ah,0dh
015A  20 20 20 20 20 20 20              db                              "                DUMP OF ROM"
      20 20 20 20 20 20 20 20
      20 20 20 20 20 20 44
      55 4D 50 20 4F 46
      20 52 4F 4D
0176  0A 0D 0A 0D 00                   db      0ah,0dh,0ah,0dh,0

017B                    code    ends
                                end     main
```

(16) LCD80 VRAM access

```
                                page    54,132
                        ;===============================================================================
                        ;                       << SAMPLE PROGRAM FOR VRAM DIRECT ACCESS >>
                        ;       < NOTE >
                        ;               This program shows how to access VRAM for LCD80 directly.
                        ;===============================================================================

0000                    code    segment
                                assume  cs:code,ds:code

0100                            org     100h
                        ;       ************************************
                        ;               Draw lines directly
                        ;       ************************************
                        ;       < NOTE >
                        ;               Draw a line on LCD80.

0100                    start:
0100  B8 0006                   mov     ax,6
0103  CD 10                     int     10h                     ; set screen mode to 6 (graphics mode)
                        ;       < draw lines (0,199)-(x,0) >
0105  C7 06 020A R 0000         mov     word ptr x1,0           ; set a screen position
010B  C7 06 020C R 00C7         mov     word ptr y1,199         ;   (x1,y1)=(0,199)
0111  C7 06 0210 R 0000         mov     word ptr y2,0
0117  B8 0004                   mov     ax,4
011A                    draw1:
011A  A3 020E R                 mov     word ptr x2,ax          ;   (x2,y2)=(ax,0)
011D  E8 014D R                 call    line
0120  05 000F                   add     ax,15
0123  3D 027F                   cmp     ax,639
0126  76 F2                     jbe     draw1
                        ;       < line (639,199)-(x,0) >
0128  C7 06 020A R 027F         mov     word ptr x1,639
012E  C7 06 020C R 00C7         mov     word ptr y1,199         ;   (x1,y1)=(639,199)
0134  C7 06 0210 R 0000         mov     word ptr y2,0
013A  B8 027A                   mov     ax,634
013D                    draw2:
013D  A3 020E R                 mov     word ptr x2,ax          ;   (x2,y2)=(ax,0)
0140  E8 014D R                 call    line
0143  2D 000F                   sub     ax,15
0146  73 F5                     jnb     draw2
0148  B8 4C00                   mov     ax,4c00h
014B  CD 21                     int     21h
                        ;       ****************************
                        ;               draw a line
                        ;       ****************************
                        ;       < NOTE >
                        ;               Draw a line at (x1,y1)-(x2,y2)
                        ;       < ON ENTRY >
                        ;               X1,Y1 = (x,y) coordinate for point 1
                        ;               X2,Y2 = (x,y) coordinate for point 2
014D                    line:
014D  50                        push    ax
014E  53                        push    bx
014F  51                        push    cx
0150  52                        push    dx
0151  8B 0E 020A R              mov     cx,word ptr x1          ; Get x1
0155  8B 16 020C R              mov     dx,word ptr y1          ; Get y1
0159  A1 020E R                 mov     ax,word ptr x2          ; Get x2
015C  8B 1E 0210 R              mov     bx,word ptr y2          ; Get y2
0160  2B DA                     sub     bx,dx
0162  BF 01EE R                 mov     di,offset down          ; If Y2>Y1, move DOWN
0165  73 05                     jnb     line1
0167  BF 01DE R                 mov     di,offset up            ; If Y2<Y1, move UP
016A  F7 DB                     neg     bx                      ; ABS(Y2-Y1)
016C                    line1:
016C  2B C1                     sub     ax,cx
016E  BE 0204 R                 mov     si,offset right         ; If X2>X1, move to RIGHT
0171  73 05                     jnb     line2
0173  BE 01FE R                 mov     si,offset left          ; If X2<X1, move to LEFT
0176  F7 D8                     neg     ax                      ; ABS(X2-X1)
0178                    line2:
0178  3B D8                     cmp     bx,ax                   ; ABS(Y2-Y1) - ABS(X2-X1) ?
017A  73 03                     jnb     line3                   ; If ABS(Y2-Y1) > ABS(X2-X1)
017C  93                        xchg    ax,bx
017D  87 F7                     xchg    si,di
017F                    line3:
017F  A3 0212 R                 mov     word ptr min,ax         ; min(abs(X2-X1),abs(Y2-Y1))
0182  89 1E 0214 R              mov     word ptr max,bx         ; max(abs(X2-X1),abs(Y2-Y1))
0186  53                        push    bx
0187  E8 01AC R                 call    mapxy                   ; Map (x,y) to VRAM address BX, AL
018A  5A                        pop     dx
018B  8B CA                     mov     cx,dx
018D  41                        inc     cx                      ; CX = number of points
018E  D1 EA                     shr     dx,1
0190                    line4:
0190  E8 01D3 R                 call    pset                    ; set the point
0193  03 16 0212 R              add     dx,word ptr min
0197  3B 16 0214 R              cmp     dx,word ptr max
019B  72 06                     jb      line5
019D  2B 16 0214 R              sub     dx,word ptr max
01A1  FF D6                     call    si                      ; increase x or y (min)
01A3                    line5:
01A3  FF D7                     call    di                      ; increase x or y (max)
01A5  E2 E9                     loop    line4
01A7  5A                        pop     dx
01A8  59                        pop     cx
01A9  5B                        pop     bx
01AA  58                        pop     ax
01AB  C3                        ret
```

```
                        ;       ************************************
                        ;               Calculate VRAM address
                        ;       ************************************
                        ;
                        ;       < NOTE >
                        ;               Calculate VRAM address for (x,y)
                        ;       < ON ENTRY >
                        ;               CX : x coordinate
                        ;               DX : y coordinate
                        ;       < ON EXIT >
                        ;               ES : VRAM segment address
                        ;               BX : VRAM offset address
                        ;               AH : VRAM mask address
                        ;
01AC                    mapxy:
01AC  B8 B800                   mov     ax,0b800h               ; ES = VRAM segment address
01AF  8E C0                     mov     es,ax
01B1  D1 EA                     shr     dx,1                    ; DX = y/2
01B3  9C                        pushf                           ; Cy_flag=0 for even scans, 1 for odd scans
01B4  B8 0050                   mov     ax,80                   ; Horizontal bytes per line
01B7  F7 E2                     mul     dx                      ; AX = (y/2) * 80
01B9  9D                        popf
01BA  73 03                     jnb     mapxy1                  ; Is it even scans or odd scans?
01BC  05 2000                   add     ax,2000h                ; If odd scans, add 2000h
01BF                    mapxy1:
01BF  8B D8                     mov     bx,ax                   ;
01C1  8B C1                     mov     ax,cx                   ;
01C3  D1 E8                     shr     ax,1                    ;
01C5  D1 E8                     shr     ax,1                    ;
01C7  D1 E8                     shr     ax,1                    ;
01C9  03 D8                     add     bx,ax                   ; BX = offset address
01CB  B4 80                     mov     ah,80h                  ;
01CD  80 E1 07                  and     cl,7                    ; AH = mask address
01D0  D2 EC                     shr     ah,cl                   ;
01D2  C3                        ret
                        ;
                        ;       ******************************
                        ;               Set the point
                        ;       ******************************
                        ;
                        ;       < NOTE >
                        ;               Set the point
                        ;       < ON ENTRY >
                        ;               ES : VRAM segment address
                        ;               BX : VRAM offset address
                        ;               AH : VRAM mask address
                        ;
01D3                    pset:
01D3  26: 8A 07                 mov     al,es:[bx]              ;
01D6  34 FF                     xor     al,0ffh                 ; point set (point reset, if 0)
01D8  22 C4                     and     al,ah                   ; mask
01DA  26: 30 07                 xor     es:[bx],al              ;
01DD  C3                        ret
                        ;
                        ;       ****************************************
                        ;               move one pixcel subroutines
                        ;       ****************************************
                        ;
                        ;       < NOTE >
                        ;               Move up, down, to left, and to right by one pixel.
                        ;
                        ;       < ON ENTRY >
                        ;               BX : VRAM offset address
                        ;               AH : VRAM mask address
                        ;       < ON EXIT >
                        ;               BX,AH updated
                        ;
                        ;       <>  Move up 1 pixel <>
                        ;
01DE                    up:
01DE  81 FB 2000                cmp     bx,2000h                ; Even scan or odd scan?
01E2  72 05                     jb      up1                     ; (N)th line (odd) to (N-1)th line (even)
01E4  81 EB 2000                sub     bx,2000h                ;
01E8  C3                        ret
01E9                    up1:
01E9  81 C3 1FB0                add     bx,1fb0h                ; (N)th line (even) to (N-1)th line (odd)
01ED  C3                        ret
                        ;
                        ;       <> Move down 1 pixel <>
                        ;
01EE                    down:
01EE  81 FB 2000                cmp     bx,2000h                ; Even scan or odd scan?
01F2  72 05                     jb      down1                   ; (N)th line (odd) to (N+1)th line (even)
01F4  81 EB 1FB0                sub     bx,1fb0h                ;
01F8  C3                        ret
01F9                    down1:
01F9  81 C3 2000                add     bx,2000h                ; (N)th line (even) to (N+1)th line (odd)
01FD  C3                        ret
                        ;
                        ;       <> Move left 1 pixel <>
                        ;
01FE                    left:
01FE  D0 C4                     rol     ah,1                    ; Move one dot to left
0200  73 01                     jnb     left1                   ; Decrease VRAM address
0202  4B                        dec     bx
0203                    left1:
0203  C3                        ret
                        ;
                        ;       <> Move right 1 pixel <>
                        ;
0204                    right:
0204  D0 CC                     ror     ah,1                    ; Move one dot to right
0206  73 01                     jnb     right1                  ; Increase VRAM address
0208  43                        inc     bx
0209                    right1:
0209  C3                        ret
                        ;
                        ;       <> work area <>
                        ;
020A  ????              x1:     dw      ?
020C  ????              y1:     dw      ?
020E  ????              x2:     dw      ?
0210  ????              y2:     dw      ?
0212  ????              min:    dw      ?
0214  ????              max:    dw      ?
                        ;
0216                    code    ends
                                end     start
```

(17) Communication with slave CPU

```
                                page    54,132
                                ;===============================================================================
                                ;                       << SAMPLE PROGRAM FOR SLAVE I/F >>
                                ;       < NOTE >
                                ;               This program shows how to connumicate with slave CPU.
                                ;===============================================================================
0000                            code    segment byte public
                                        assume  cs:code,ds:code,es:code,ss:code

                                ;       <> constant define <>

= 11E9                          STAT_REG        equ     11e9h           ; slave CPU I/F status register
= 11E9                          CTRL_REG        equ     11e9h           ; slave CPU I/F cotroll register
= 11E8                          DATA_REG        equ     11e8h           ; slave CPU I/F data register


                                ;       <> external define <>

                                        extrn   dump:near               ; display data in hexadecimal form

0100                                    org     100h

                                ;       ****************************
                                ;               Main Porgram
                                ;       ****************************
                                ;       < NOTE >
                                ;               This program sets alarm time & alarm interrupt ON
0100                            main    proc    near

0100  BE 012D R                         mov     si,offset set_alarm     ; set send-text packet (DS:SI)
0103  BF 0123 R                         mov     di,offset buffer        ; set recieve-text packet (ES:SI)
0106  E8 0138 R                         call    slave                   ; set alarm time
0109  BE 0136 R                         mov     si,offset alarm_on      ; set send-text packet
010C  E8 0138 R                         call    slave                   ; enable alarm interrupt
010F  BE 0137 R                         mov     si,offset read_alarm    ; set send-text packet
0112  E8 0138 R                         call    slave                   ; read alarm time

0115  BB 0123 R                         mov     bx,offset buffer        ; set receive-text pachet
0118  B9 000A                           mov     cx,10                   ; display alarm time
011B  E8 0000 E                         call    dump                    ;

011E  B8 4C00                           mov     ax,4c00h                ; terminate this program
0121  CD 21                             int     21h                     ;
0123                            main    endp


                                ;       <> deta define <>

0123     0A [                   buffer          db      10 dup(0)
                00
              ]

012D  D7 19 88 02 09 10         set_alarm       db      0d7h,19h,88h,02h,09h,10h,30h,20h,02h
      30 20 02
0136  D5                        alarm_on        db      0d5h
0137  D6                        read_alarm      db      0d6h



                                ;       ************************************
                                ;               slave I/F subroutine
                                ;       ************************************
                                ;       < NOTE >
                                ;               Send a commad text to slave CPU & receive a responce text.
                                ;               Length of a text is automatically set by a command code.
                                ;       < ON ENTRY >
                                ;               DS:SI : command text packet address
                                ;               DS:DI : buffer address for response text
                                ;       < ON EXIT >
                                ;               C-flag OFF : complete
                                ;                       (DS:DI) : response text
                                ;               C-flag ON  : command error
0138                            slave           proc    near

                                        push    ax                      ; save registers
0138  50                                push    bx                      ;
0139  53                                push    cx                      ;
013A  51                                push    dx                      ;
013B  52                                push    si                      ;
013C  56                                push    di                      ;
013D  57
                                        call    chk_txt                 ; check command text & get command length
013E  E8 0189 R                         jc      slave_exit              ; jump if command error
0141  72 3E
                                        cli                             ; disable all interrupts
0143  FA                                push    cx                      ; save length (CH:receive, CL:command)
0144  51                                mov     ch,0                    ;
0145  B5 00
```

```
0147                    snd_txt:
0147  BA 11E9                   mov     dx,STAT_REG             ; RDYSIO active (ready) ?
014A  EC                        in      al,dx                   ;
014B  A8 02                     test    al,00000010b            ;
014D  74 F8                     jz      snd_txt                 ; jump if No

014F  8A 04                     mov     al,[si]                 ; Get command text one byte
0151  46                        inc     si                      ;
0152  BA 11E8                   mov     dx,DATA_REG             ; output it
0155  EE                        out     dx,al                   ;

0156  BA 11E9                   mov     dx,CTRL_REG             ;
0159  EC                        in      al,dx                   ;
015A  0C 02                     or      al,00000010b            ; reset RDYSIO (command transmit to slave)
015C  EE                        out     dx,al                   ;
015D  E2 E8                     loop    snd_txt                 ; LOOP untill command text ends

015F  59                        pop     cx                      ; recover receive data length
0160  80 FD 00                  cmp     ch,0                    ; if receive data is not required
0163  74 1C                     jz      slave_exit              ; then end

0165                    rcv_txt:
0165  BA 11E9                   mov     dx,STAT_REG             ; RDUSIO active (ready)
0168  EC                        in      al,dx                   ;
0169  A8 02                     test    al,00000010b            ;
016B  74 F8                     jz      rcv_txt                 ; jump if No (wait)

016D  BA 11E8                   mov     dx,DATA_REG             ; get recive data
0170  EC                        in      al,dx                   ;
0171  88 05                     mov     [di],al                 ; set to buffer
0173  47                        inc     di                      ; increase pointer
0174  FE CD                     dec     ch                      ; decrease counter
0176  74 09                     jz      slave_exit              ; LOOP untill receive data ends

0178  BA 11E9                   mov     dx,CTRL_REG             ; RESET RDYSIO (transmit next text to host)
017B  EC                        in      al,dx                   ;
017C  0C 02                     or      al,00000010b            ;
017E  EE                        out     dx,al                   ;
017F  EB E4                     jmp     rcv_txt                 ; LOOP

0181                    slave_exit:
0181  FB                        sti                             ; enable inerrupts
0182  5F                        pop     di                      ; recover registers
0183  5E                        pop     si                      ;
0184  5A                        pop     dx                      ;
0185  59                        pop     cx                      ;
0186  5B                        pop     bx                      ;
0187  58                        pop     ax                      ;

0188  C3                        ret

0189                    slave           endp


                        ;       **********************************
                        ;               Check command text
                        ;       **********************************
                        ;
                        ;       < NOTE >
                        ;               Check a commad text to slave CPU.
                        ;               Length of a text is automatically set by a command code.
                        ;
                        ;       < ON ENTRY >
                        ;               DS:SI : command text packet address
                        ;       < ON EXIT >
                        ;               C-flag OFF : complete
                        ;               C-flag ON  : command error
                        ;
0189                    chk_txt         proc    near

0189  56                        push    si                      ; save registers
018A  53                        push    bx                      ;

018B  8A 1C                     mov     bl,[si]                 ; get a command code
018D  B7 00                     mov     bh,0                    ;
018F  81 EB 00A0                sub     bx,0a0h                 ; if command < 0a0h
0193  72 25                     jc      chk_txt_err             ; then error

0195  03 DB                     add     bx,bx                   ; BX : command table offset
0197  81 C3 01C2 R              add     bx,offset command_tbl   ;
019B  8A 0F                     mov     cl,[bx]                 ; get command text length
019D  43                        inc     bx                      ;
019E  8A 2F                     mov     ch,[bx]                 ; get receive text length

01A0  80 F9 00                  cmp     cl,0                    ; if command text length is 0
01A3  74 15                     jz      chk_txt_err             ; it is unrecognized command code

01A5  51                        push    cx                      ; save text length

01A6                    chk_txt_loop:
01A6  FE C9                     dec     cl                      ; decrease command text length
01A8  80 F9 00                  cmp     cl,0                    ;
01AB  74 10                     jz      chk_txt_ok              ; exit if all texts are checked

01AD  46                        inc     si                      ; increase pointer
01AE  8A 04                     mov     al,[si]                 ; get one byte of command text
01B0  24 F0                     and     al,0f0h                 ; if high 4 bit = fh or high 4 bit < ah
```

```
01B2  3C F0                           cmp     al,0f0h                 ; then okay
01B4  74 F0                           jz      chk_txt_loop            ; else error
01B6  3C A0                           cmp     al,0a0h                 ;
01B8  72 EC                           jc      chk_txt_loop            ;

01BA                            chk_txt_err:
01BA  F9                              stc                             ; error return
01BB  EB 02                           jmp     short chk_txt_exit

01BD                            chk_txt_ok:
01BD  59                              pop     cx                      ; normal return
01BE  F8                              clc

01BF                            chk_txt_exit:
01BF  5B                              pop     bx                      ; recover registers
01C0  5E                              pop     si                      ;
01C1  C3                              ret
01C2                            chk_txt         endp


                                ;       <> command code decision table <>

01C2                            command_tbl     label   byte
                                ;                       0   1   2   3   4   5   6   7   8   9   a   b   c   d   e   f
01C2  01 00 01 01 01 01               db      1,0,1,1,1,1,4,1,6,0,5,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
      04 01 06 00 05 00
      06 00 00 00 00 00
      00 00 00 00 00 00
      00 00 00 00 00 00
      00 00
01E2  01 00 01 00 02 00               db      1,0,1,0,2,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
      01 01 00 00 00 00
      00 00 00 00 00 00
      00 00 00 00 00 00
      00 00 00 00 00 00
      00 00
0202  01 00 01 00 02 00               db      1,0,1,0,2,0,2,0,1,0,1,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
      02 00 01 00 01 00
      02 00 00 00 00 00
      00 00 00 00 00 00
      00 00 00 00 00 00
      00 00
0222  01 00 01 00 01 08               db      1,0,1,0,1,8,9,0,1,0,1,0,1,8,9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
      09 00 01 00 01 00
      01 08 09 00 00 00
      00 00 00 00 00 00
      00 00 00 00 00 00
      00 00
0242  00 00 01 00 01 01               db      0,0,1,0,1,1,1,1,2,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
      01 01 02 00 01 00
      01 00 01 00 00 00
      00 00 00 00 00 00
      00 00 00 00 00 00
      00 00
0262  00 00 00 00 00 00               db      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
      00 00 00 00 00 00
      00 00 00 00 00 00
      00 00 00 00 00 00
      00 00 00 00 00 00
      00 00

0282                            code    ends
                                        end     main
```

(18) ROM BIOS version check

```
                                ;       page    54,132
                                ;       ================================================================================
                                ;
                                ;               << SAMPLE PROGRAM FOR ROM BIOS VERSION CHECK >>
                                ;
                                ;       < NOTE >
                                ;               This program shows how to check ROM BIOS versio.
                                ;
                                ;       ================================================================================
0000                            code    segment byte public
                                        assume  cs:code,ds:code,es:code,ss:code

                                ;       <> external define <>

                                        extrn   dspmsg:near

= FC00                          VER_SEG         equ     0fc00h          ; address of message segment
= 0003                          VER_OFFS        equ     00003h          ; address of message bigins (offset)
= 000B                          VER_OFFN        equ     0000bh          ; address of version number

= FFFF                          MTP_SEG         equ     0ffffh          ; machine type segment address
= 000D                          MTP_OFF         equ     0000dh          ; machine type offset address

= FB03                          PX16_ID         equ     0fb03h          ; PX-16, HC-160 machine code

0100                                    org     100h

                                ;       ****************************
                                ;               MAIN PROGRAM
                                ;       ****************************
                                ;
                                ;       < NOTE >
                                ;               Display ROM BIOS version number

0100                            main            proc    near

0100  E8 0117 R                        call    vers                    ; check ROM BIOS version
0103  72 07                             jc      err_exit                ; jump if machine is not PX-16

0105  06                                push    es                      ; ES:SI sige on message address
0106  1F                                pop     ds
0107  E8 0000 E                         call    dspmsg                  ; display sign on message
010A  EB 06                             jmp     short main_exit         ; EXIT

010C                            err_exit:
010C  BE 0148 R                         mov     si,offset err_msg       ; diplay error message
010F  E8 0000 E                         call    dspmsg

0112                            main_exit:
0112  B8 4C00                           mov     ax,4c00h                ; terminate this program
0115  CD 21                             int     21h

0117                            main    endp

                                ;       ************************************
                                ;               ROM BIOS VERSION CHECK
                                ;       ************************************
                                ;
                                ;       < NOTE >
                                ;               Get ROM BIOS version number
                                ;       < ON ENTRY >
                                ;               None
                                ;       < ON EXIT >
                                ;               C-flag OFF and AH = 00H : this machine is PX-16
                                ;                       AL      : major version
                                ;                       BX      : minor version
                                ;                                 ( ex. : vers. 1.23 --> AL=30h,BX=3132h )
                                ;                       ES:SI   : sign on message
                                ;               C-flag ON  and AH = FFH : this machine is not PX-16

0117                            vers    proc    near

0117  1E                                push    ds                      ; save DS register
0118  B8 FFFF                           mov     ax,MTP_SEG              ; set DS machine ID segment address
011B  8E D8                             mov     ds,ax
011D  BE 000D                           mov     si,MTP_OFF              ; set SI machine ID offset address

0120  8B 04                             mov     ax,[si]                 ; this machine is PX-16 ?
0122  3D FB03                           cmp     ax,PX16_ID
0125  75 1A                             jnz     vers_err                ; jump if No

0127  B8 FC00                           mov     ax,VER_SEG              ; set sign on message segment address
012A  8E D8                             mov     ds,ax

012C  BE 000B                           mov     si,VER_OFFN
012F  8A 04                             mov     al,[si]                 ; AL : major version
0131  83 C6 02                          add     si,2
0134  8A 3C                             mov     bh,[si]
0136  46                                inc     si                      ; BX : minor version
0137  8A 1C                             mov     bl,[si]                 ;      (BH:high,BL:low)
0139  BE 0003                           mov     si,VER_OFFS             ; set sige on message top address
013C  B4 00                             mov     ah,0                    ; Set complete value & flag
013E  F8                                clc
013F  EB 03                             jmp     short vers_exit         ; exit

0141                            vers_err:
0141  B4 FF                             mov     ah,0ffh                 ; set error value & flag
0143  F9                                stc

0144                            vers_exit:
0144  1E                                push    ds
0145  07                                pop     es                      ; ES : sign on message segment address
0146  1F                                pop     ds                      ; recover DS
0147  C3                                ret

0148                            vers    endp

0148  54 68 69 73 20 6D        err_msg db      'This machine is not PX-16.',0
      61 63 68 69 6E 65
      20 69 73 20 6E 6F
      74 20 50 58 2D 31
      36 2E 00

0163                            code    ends

                                        end     main
```