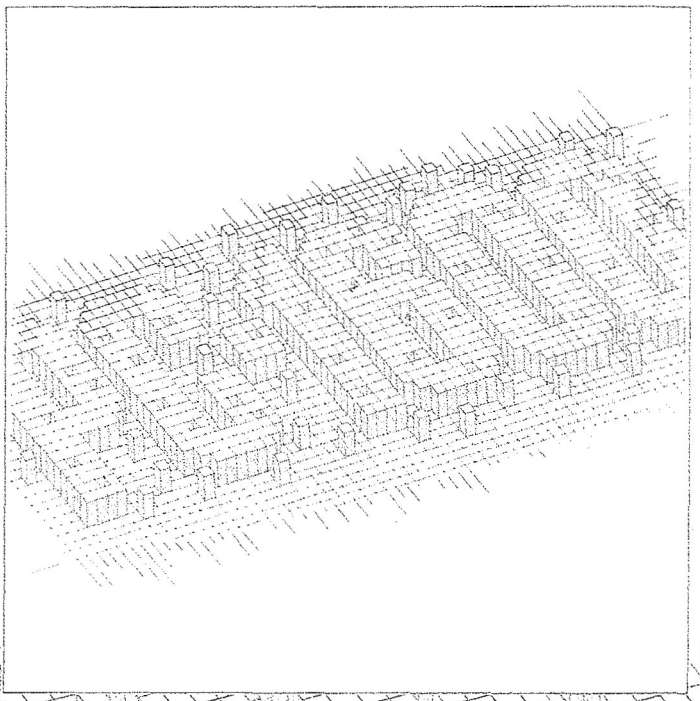
POCKET COMPUTER

PC-1600

TECHNICAL REFERENCE MANUAL



SHARP®

POCKET COMPUTER

PC-1600

TECHNICAL REFERENCE MANUAL

FORWARD

The PC-1600 Technical Reference Manual describes the specifications and usage of the IOCS (Input/Output Control System), which controls the I/O operations of the PC-1600 main unit and the peripherals, and gives information regarding the PC-1600 hardware and its interfaces (system bus, RS-232C, etc.)

The Technical Reference Manual has been compiled to provide the IOCS interface information that may be needed when advanced users and programmers write more sophisticated application programs using the machine language of the PC-1600 and to give the hardware information necessary for construction of an application hardware system using the PC-1600. This manual also contains PC-1600 programming know-how and considerations so that PC-1600 users can make the most of the PC-1600 system.

The Technical Reference Manual has been written on the assumption that the reader is already familiar with the basic knowledge of PC-1600 and the general information about computer hardware and programming (especially of Z80 CPU). Many commercial publications are available describing general computer architecture and Z80 CPU. Read them, if necessary, in addition to this manual.

We hope that PC-1600 users, software house programmers and system house engineers will use this manual to develop various kinds of application programs for the PC-1600 system and PC-1600-based application systems.

SHARP CORPORATION
Information Systems Group

CONTENTS

FORWARD

CHAPTER 1 SYSTEM CONFIGURATION 1

CHAPTER 2 Z-80 MACHINE LANGUAGE PROGRAMS AND LOAD AREA 5

2.1 Memory Map 6

2.2 BASIC Commands Related to Machine Language 7

CHAPTER 3 IOCS 11

3.1 DISPLAY 12

3.1.1 IOCS Routines for LCD 12

3.1.2 Work Area used for IOCS Routines for LCD 28

3.1.3 Character Font 29

3.2 KEY INPUT 30

3.2.1 IOCS Routines for Key Input 30

3.2.2 Work Area used for IOCS Routines for Key Input 37

3.2.3 Scanning of ON (BREAK) Key 38

3.2.4 Entry of International Characters and Symbols 38

3.2.5 Data Flow from Key Scanning to KEYGET Routine 38

3.2.6 Re-definition of Keys 39

3.3 FILES 45

3.3.1 Files Handled in BASIC 45

3.3.2 IOCS Routines for Files 48

3.3.3 Structure of Memory File 53

3.4 INTERRUPT HANDLING 58

3.4.1 Interrupt Handling 58

3.4.2 Work Area used for Interrupt handling 61

3.5 SYSTEM START-UP 62

3.5.1 Processing at Power On 62

3.5.2 Execution of Boot Program 63

3.6 RS-232C AND SIO 65

3.6.1 Handling RS-232C and SIO in BASIC 65

3.6.2 Data Format of Communications 67

3.6.3 IOCS Routines for RS-232C and SIO 67

3.7 PRINTER 75

3.7.1 IOCS Routines for Printer (1) 75

3.7.2 IOCS Routines for Printer (2) 76

3.8 DISK 94

3.8.1 Floppy Disk Format 94

3.8.2 Specifications of Floppy Disk 95

3.8.3 File Management 95

3.8.4 IOCS Routines for Floppy Disk 97

3.8.5 Processing at Power-On Time 103

3.9 TIMER/ANALOG PORT 104

3.10 BEEP 114

3.11 TAPE RECORDER 117

3.11.1 PC-1600 Mode (Mode 0) 117

3.11.2 PC-1500/PC-1500A Mode (Mode 1) 122

3.11.3 Work Area Used for Cassette Tape Recorder 125
3.12 MEMORY 127
3.12.1 Slots and Memory Modules 127
3.12.2 Work Area Used for Memory 128
3.12.3 IOCS Routines for Memory Control 130
3.13 MEMORY MODULE 135
3.13.1 Location of Memory Module 135
3.13.2 Type of Memory Module 135
3.13.3 Header Structure of Memory Module 136

CHAPTER 4 BASIC INTERPRETER 137
4.1 FUNCTIONS HANDLING AND INTERNAL EXPRESSION 138
4.1.1 Intermediate Codes of Functions 138
4.1.2 Arithmetic Registers 140
4.1.3 Internal Expression of Numeric Values and Strings 141
4.1.4 Function Operation Subroutines 143
4.2 BASIC PROGRAM TEXT HANDLING 148
4.2.1 Subroutines for Numeric Value Handling 148
4.2.2 Subroutines for ASCII Code Conversion 149
4.2.3 Subroutines for Evaluation of Expressions 150
4.2.4 Subroutines for BASIC Text 152
4.2.5 Intermediate Code Table 157

CHAPTER 5 OTHER FUNCTIONS AND PRECAUTIONS 163
5.1 AUTOMATIC LOADING AND RUNNING OF BASIC PROGRAM FILE (AUTORUN.BAS) 164
5.2 CHANGING DISPLAY CHARACTER FONT 164
5.3 EXTENDED FUNCTION OF KEYSTAT COMMAND 165
5.4 SEGMENTING ONE RAM MODULE FOR DIFFERENT USES 166
5.5 FILE FORMAT 167
5.6 DATA INPUT/OUTPUT TO FILE DEVICE 168
5.7 PRECAUTIONS FOR USE OF SERIAL PORT (RS-232C AND SIO) 171
5.8 TRANSFERRING A BASIC PROGRAM BETWEEN PC-1600 AND OTHER MACHINE 176
5.9 MERGING PROGRAM FILES 179
5.10 SAVING AND LOADING THE RESERVE AREA 180
5.11 DISABLING THE KEY INTERRUPT DUE TO ON KEY STATEMENT 181
5.12 CE-153 CONTROL UTILITY (FOR PC-1600) 182
5.13 RST COMMANDS OF SC-7852 (Z-80) 186
5.14 SC-7852 (Z-80) AND LH-5803 MICROPROCESSORS 187
5.15 COMPATIBILITY WITH PC-1500 188
5.16 PRECAUTIONS FOR APPLICATION PROGRAM DEVELOPMENT 191

CHAPTER 6 WORK AREA USED FOR BASIC 195
6.1 OVERVIEW OF WORK AREA 196
6.2 EXPANSION OF WORK AREA AND BUFFER 197
6.3 WORK AREA MAP 200

CHAPTER 7 PC-1600 HARDWARE 207
7.1 CPU 208
7.1.1 Specifications of SC-7852 208

7.1.2 Specifications of LH-5803 215
7.1.3 Specifications of LU57813P 218
7.1.4 Interface Between SC-7852 (Z-80) and LH-5803 222
7.1.5 Interface Between Sub-CPU and Main CPU 222
7.2 MEMORY 223
7.2.1 Memory Map Viewed from SC-7852 (Z-80) 223
7.2.2 Memory Chip Select Signals 225
7.2.3 Memory Map Viewed from LH-5803 225
7.3 LCD 226
7.4 KEYBOARD 227
7.5 BUZZER 227
7.6 RS-232C/SIO INTERFACE 227
7.7 POWER SUPPLY 231
7.7.1 Kinds of Supply Voltages 231
7.7.2 Kinds of Power Supplies 231
7.8 GATE ARRAY 232
7.9 CONTROL OF I/O PORT CONTROLLER 234

CHAPTER 8 HARDWARE OF PERIPHERAL DEVICES 237
8.1 CE-1600P 238
8.2 CE-1600F/CE-1650F 245
8.3 CE-1600M 248
8.4 CE-1620M/CE-1601E/PROM PROGRAMMER 251
8.5 CE-1600L/CE-1601T 260
8.6 CE-1601L ... CE-1605L 260
8.7 CE-160CA 261

CHAPTER 9 CIRCUIT DIAGRAM 263
9.1 CIRCUIT DIAGRAM OF PC-1600 264
9.2 CIRCUIT DIAGRAM OF PERIPHERAL DEVICES 268

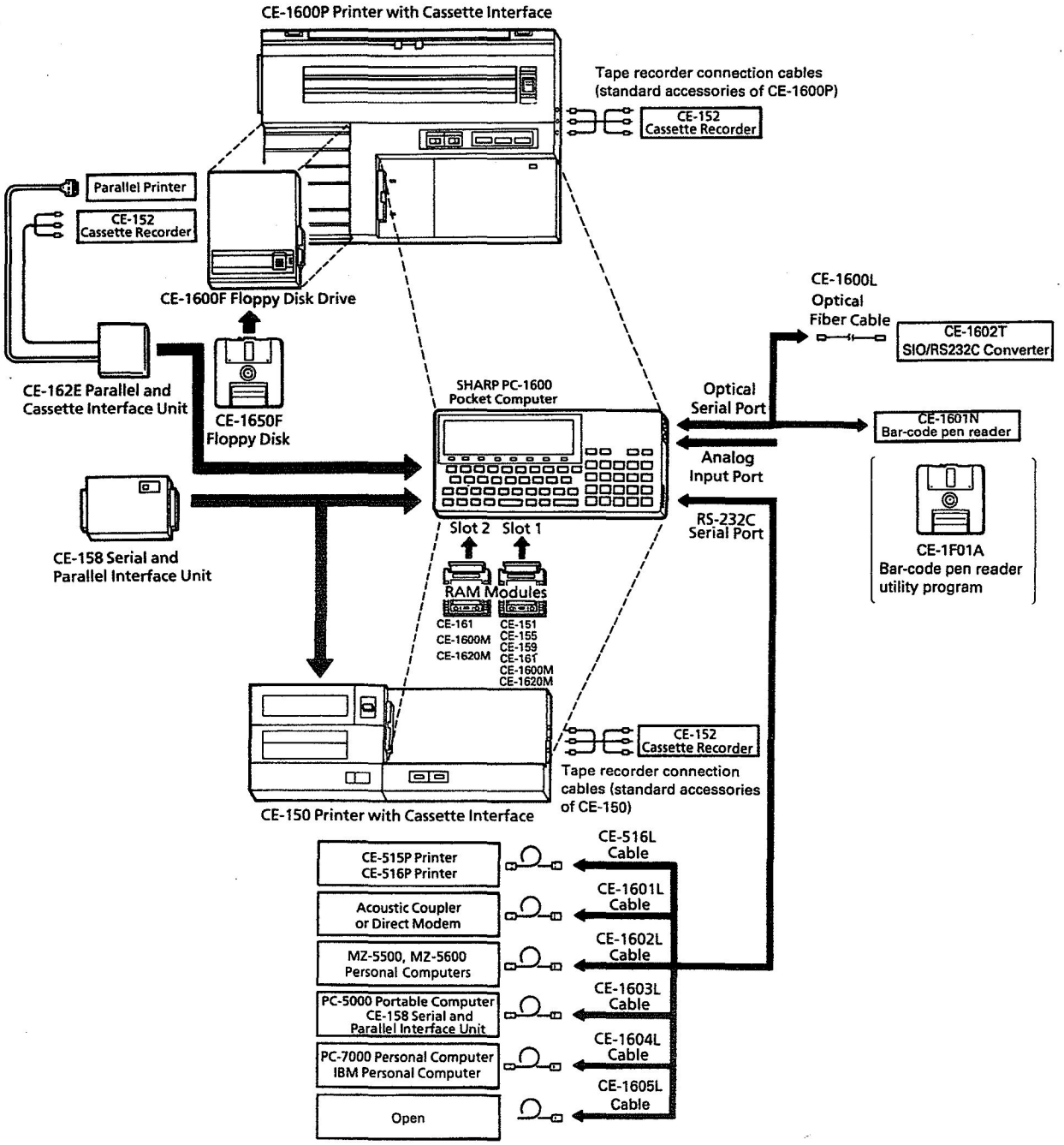
CHAPTER 10 APPENDICES 273
10.1 CHARACTER CODE TABLE 274
10.2 KEY CODE TABLE 275
10.3 CONNECTOR PIN CONFIGURATION 278
10.4 Z-80 MNEMONIC CODES 281
10.5 MNEMONIC CODES OF LH-5803 297

CHAPTER 1

SYSTEM CONFIGURATION

The PC-1600 can be connected with various kinds of optional peripheral devices. The following figure shows the system configuration of the PC-1600 and these peripherals. Since the system bus of the PC-1600 is compatible with that of the PC-1500 serial, the PC-1600 can use most of the PC-1500/PC-1500A peripheral devices.

PC-1600 System Configuration



Note: Connection of CE-1600F requires CE-1600P.
CE-158 and CE-162E cannot be connected to CE-1600P.

(1) Peripheral devices for PC-1600

CE-1600P:	A4-size 4-color plotter-printer
CE-1600F:	2.5-inch floppy disk drive
CE-1600M:	32KB RAM module
CE-1650F:	2.5-inch floppy disk package (10 disks)
CE-1602T:	SIO/RS-232C converter
CE-1600L:	Optical fiber cable
CE-1601L:	Cable for modem/acoustic coupler
CE-1602L:	RS-232C cable for MZ-5600 and MZ-5500
CE-1603L:	RS-232C cable for PC-5000 and CE-158
CE-1620M:	PROM module (32KB ROM)
CE-1601N:	Bar-code pen reader
CE-1F01A:	Bar-code pen reader utility program (floppy disk)
CE-1604L:	RS-232C cable for IBM-PC and PC-7000
CE-1605L:	RS-232C cable (open end)

(2) Peripheral devices for PC-1500

CE-150:	Color graphics printer
CE-151:	Memory module (4KB RAM)
CE-152:	Cassette tape recorder
CE-155:	Memory module (8KB RAM)
CE-158:	RS-232C/Parallel interface
CE-159:	Program module (8KB RAM)
CE-161:	Program module (16KB RAM)
CE-162E:	Parallel/Cassette interface

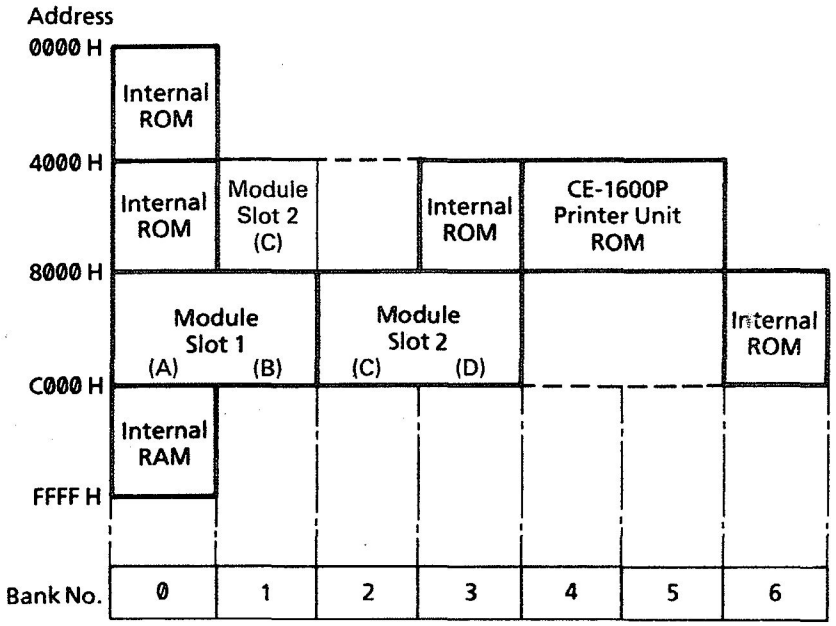
Note: CE-150 and CE-158 cannot be used together with CE-1600P.

CHAPTER 2

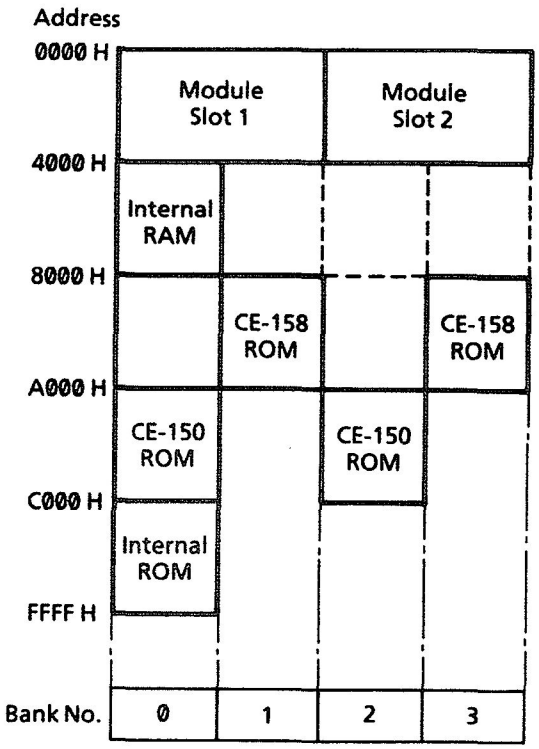
Z-80 MACHINE LANGUAGE PROGRAMS AND LOAD AREA

2.1 Memory Map

The following figure shows the PC-1600 memory map viewed from SC-7852 (Z-80). As shown in the figure, the memory space is extended by the bank switching. The bank switching is accomplished in the way: the 64KB memory space of Z-80 is segmented into four 16KB areas, and to each area is allocated one of the memory blocks (16KB/block) belonging to that area. Refer to section 3.12.3 for the bank switching procedures.

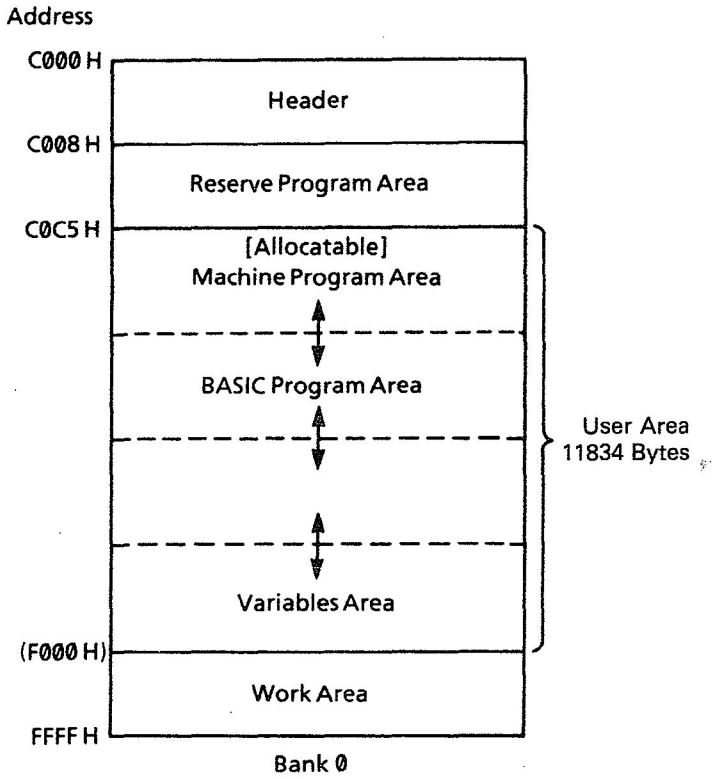


Overall Memory [Viewed from Z-80A processor]



Overall Memory [Viewed from LH-5803 Sub-Processor]

The PC-1600 can address 8 mermory banks (bank 0 to bank 7). The first four banks are allocated to RAM. Banks 4 to 6 hold internal system ROMs and peripheral memory. Bank 7 is unused, but is addressable.



Internal RAM [Viewed from Z-80A Processor]

With no modules in the expansion slots, the internal RAM has 11834 bytes of user area. The above memory map shows some of the user area allocated for machine language programs. This machine program area can be set to 0 with the NEW Command.

2.2 BASIC Commands Related to Machine Language

(1) NEW command

To use the machine language, reserve a machine language program area with NEW command.

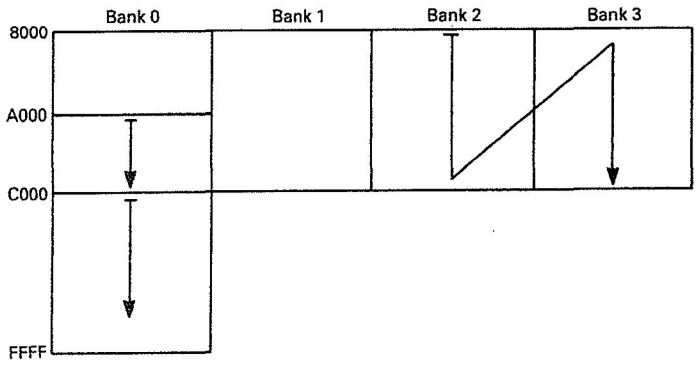
$$\text{NEW} \begin{Bmatrix} \text{"S0:"} \\ \text{"S1:"} \\ \text{"S2:"} \end{Bmatrix} \text{,<expression>}$$

where <expression> specifies a value of (the desired machine language program area size in bytes) plus C5H.

When this command is executed, the memory area from the the starting address of the memory in the slot plus C5H to the memory starting address plus <expression> minus 1 is allocated for the machine language program area.

(Example)

When CE-159 and CE-1600M are set respectively in slots 1 and 2, both as the program module.



When the following NEW commands are executed:

```
NEW "S1:",&1000
NEW "S2:",&5000
NEW "S0:",&1000
```

the following memory areas can be used for the machine language program:

Bank 0: A0C5H to AFFFH
Bank 2: 80C5H to BFFFH
Bank 3: 8000H to 8FFFH
Bank 0: C0C5H to CFFFH

(2) PEEK command

PEEK #(<bank>,<expression>)

This command reads the contents of the memory address specified by <expression> of the specified <bank>.

If the memory address you want to read is between C000H and FFFFH, you can use the following format: PEEK <expression>

(3) POKE command

POKE [#<bank>,]<expression 1>,<expression 2>,<expression 3>...

This command writes data items specified by <expression 2>, <expression 3> ... consecutively into the memory area whose bank and starting address are specified respectively by <bank> and <expression 1>.

If no bank is specified, bank 0 is selected.

(Example)

When the following command is executed:

```
POKE &C700,&01,&02,&03
```

data (01H, 02H and 03H) are written into the memory as follows.

Address	Data
C700H	01H
C701H	02H
C702H	03H

(4) CALL command

CALL [#<bank>,]<address>[,<variable>]

This command executes a machine language program that is stored in the memory area whose bank and starting address are specified by <bank> and <address>.

Control returns from the machine language program when RET command is executed.

If no bank is specified, bank 0 is selected.

If <variable> is a numeric variable (value: −32768 to 32767), the following operations are executed:

(1) transfer the value of the variable to DE register,
(2) execute the machine language program, and
(3) when returning from the machine language program, if there is a carry, transfer the contents of DE register to the specified variable.

If <variable> is a string variable, the following operations are executed:

(1) transfer the starting address of the string variable to DE register,
(2) execute the machine language program, and
(3) when returning from the machine language program, if the carry flag is 1, transfer the character string whose starting address is specified by DE register and whose length is specified by B register into the string variable.

Machine Language Programs

The PC-1600 can be programmed directly in Z-80A Assembler code by the advanced programmer.

BASIC Machine Language Related Commands

BASIC supports a number of commands to load, save, access and call machine language routines, or to control the machine I/O ports directly. Some of them address the main Z-80A processor, and some address the LH-5803 sub-processor. They are:

BLOAD, BSAVE, CALL, CLOADM, CSAVEM, INP, OUT, PEEK, POKE, XCALL, XPEEK, XPEEK#, XPOKE, XPOKE#

Memory Allocation for Machine Language Programs

Internal RAM can be allocated for machine language programs with the NEW command, which sets the lower address of the BASIC program area. The user can also access system utilities in other memory areas, or the peripheral device ROMs, but this needs a detailed knowledge of the memory allocation of the PC-1600 above the covered in this manual.

Compatibility with the PC-1500

The following table lists the PC-1600 machine language related commands which are different from the PC-1500 command set:

COMMAND NAME PC-1600	PC-1500	FUNCTION
XCALL	CALL	Runs machine language program for LH-5801/3 sub-processor.
CALL		Runs machine language program in PC-1600's main processor (Z-80A).
XPOKE	POKE	Writes data to LH-5801/3 memory space.
POKE		Writes data to main Z-80A processor memory space.
XPEEK	PEEK	Reads data from LH-5801/3 memory area.
PEEK		Reads data from main Z-80A processor memory area.
XPOKE#	POKE#	Sends data byte to LH-5801/3 machine I/O port.
XPEEK#	PEEK#	Returns data byte from LH-5801/3 machine I/O port.

CHAPTER 3

IOCS

The PC-1600 has many IOCS routines that perform various kinds of basic input and output operations of the PC-1600. The user may use these routines to efficiently develop a machine language program. The entry addresses of the IOCS routines will not be changed even when the PC-1600 BASIC interpreter is up-versioned in the future. Use the IOCS routines for access to the I/O. If you write a program that directly access the I/O (i.e., a program that accesses the I/O with OUT and IN (INP) commands without IOCS routines), the program may not run properly on a new machine that will be released as a up-graded model of PC-1600. The IOCS routines described in this manual are all for the SC-7852 (Z-80) mode.
The terms used in the explanations of the IOCS routines have the following meanings:

- **Entry address**
  Most of the IOCS routines are executed by directly calling the entry address of each routine.
- **IOCS number**
  Some of the IOCS routines are executed by calling a certain address with the particular IOCS number set in C register.
- **Function**
  Describes the operation of the IOCS routine.
- **Parameter**
  Some IOCS routines require the parameter that prescribes the operation of the routine. Set the parameter in memory or registers before calling the routine.
- **Return**
  Some IOCS routines return data when the routine is completed and control returns from the routine. These return data are set in memory or registers.
- **Affected register**
  When a routine is executed, the contents of some registers or memory locations are destroyed.

3.1 DISPLAY

3.1.1 IOCS Routines for LCD

The following table lists the names, functions and entry addresses of the IOCS routines. Use the following format to call up these IOCS routines:

CALL Entry-address

The LCD related IOCS routines are for displaying data on a single line only: the data are not displayed over more than one line.

Name	Entry address	Function
PRTANK	0100H	Display one character.
PRTASTR	00EBH	Display a string of characters.
CRSRSET	0115H	Set the cursor position.
CRSRPOS	0118H	Read the current cursor position.
CRSRSTAT	011EH	Specify the cursor type.
UPSCRL	012DH	Scroll up the screen.
DWNSCRL	0130H	Scroll down the screen.
INS1LN	0142H	Insert a blank line.
ERS1LN	0145H	Erase the contents of a line.

Name	Entry address	Function
ERSSTR	013FH	Display a specified number of spaces.
SMBLSET	013CH	Set the state of the status line symbols.
SMBLREAD	0139H	Read the state of the status line symbols.
RVSCHR	011BH	Change the display of a string of characters currently on the screen to the reverse-video mode.
SETANK	0109H	Set the display to the character mode.
DOTSET	0127H	Display a dot in the set/preset/reverse mode.
DOTREAD	012AH	Read the display state of a dot.
LINE	0121H	Draw a line.
BOX	0124H	Draw a box.
GCRSRSET	014BH	Set the graphics cursor position.
GCRSRPOS	0148H	Read the current graphics cursor position.
PRTGCHR	014EH	Display a character at the current graphics cursor position.
PRTGSTR	00EEH	Display a string of characters from the current graphics cursor position.
PRTGPTN	0154H	Display a 1 × 8 dot pattern at the current graphics cursor position.
GPTNREAD	015AH	Read the 1 × 8 dot pattern at the current graphics cursor position.
CGMODE	0133H	Change the character generator mode between the PC-1500 mode and the PC-1600 mode.
CPY1500LCD	0157H	Copy the contents of the fourth line of the screen to the PC-1500 mode LCD RAM.
CLS	0112H	Clear the screen display.
BSPCTR	00E5H	Enable/disable the LCD.
SAVELCD	015DH	Save the 156 × 8 dot pattern of the specified line to RAM.
LOADLCD	0160H	Load the 156 × 8 dot pattern from RAM to the specified line.

PRTANK

Entry Address 0100H

Function Display the character of a character code set in A register at the current cursor position, then move the cursor one column to the right. If the character is displayed at the right most column of the screen (i.e., if the X coordinate of the current cursor position is 25), then CF is set to 1, the cursor display is turned off, and the cursor remains at the same position.

Parameter A = Character code

Return CF = 1 if the character is displayed at the right most column of the screen.

Affected Register AF, CRSRX (Cursor X coordinate: F060H), CRSRST (Cursor type: F067H)

PRTASTR

Entry Address 00EBH

Function Display consecutively from the current cursor position a string of characters whose character codes are stored in consecutive memory locations. This routine displays starting from the beginning of the contents of the memory locations whose starting address is given in DE register pair until it encounters a character whose code is given in A register (this code is not included in the screen display). If the routine displays characters up to the right most column of the screen, then CF is set to 1, the cursor display is turned off, and the cursor remains at the right most position.

Parameter DE = Starting address of the memory locations that contain the character codes
A = Character code of the character (when the routine encounters this code, it terminates the current display operation, not including that code in the screen display.)

Return DE = (Address of the memory location which contains the character code of the character last displayed) + 1
CF = 1 if a character is displayed at the right most column of the screen.

Affected Register AF, DE, CRSRX (Cursor X coordinate: F060H), CRSRST (Cursor type: F067H)

Example When character codes (41H, 42H, 43H and 44H) are stored in memory locations from C000H to C003H, set

```
DE = C000H
A = 44H
```

and execute

```
CALL 00EBH
```

then "ABC" is displayed consecutively from the current cursor position on the screen. (The letter "D" (code 44H) is not displayed.)

CRSRSTAT

Entry Address 011EH

Function Specify whether or not to display the cursor, and the cursor type if displayed.

Parameter
A=00H: Turn off the cursor display.
A=01H: Display the underline cursor.
A=02H: Display the square cursor in blinking.
A=03H: Display the space cursor in blinking.

Return none

Affected Register CRSRST (Cursor type: F067H)

CRSRPOS

Entry Address 0118H

Function Read the current cursor position. (Character mode)

Parameter none

Return
D = Cursor X coordinate
E = Cursor Y coordinate

Affected Register DE

CRSRSET

Entry Address 0115H

Function Set the cursor position. (Character mode)

Parameter
D = Cursor X coordinate
E = Cursor Y coordinate

Return CF = 1 if the specified position is out of the displayable range of LCD.

Affected Register AF, CRSRX (Cursor X coordinate: F060H), CRSRY (Cursor Y coordinate: F05FH)

UPSCRL

Entry Address 012DH

Function Scroll up the screen one line. The bottom line is cleared and the cursor display is turned off.

Parameter none

Return none

Affected Register none

DWNSCRL

Entry Address 0130H

Function Scroll down the screen one line. The top line is cleared and the cursor display is turned off.

Parameter none

Return none

Affected Register none

INS1LN

Entry Address 0142H

Function Insert one blank line at the specified line position of the screen. The screen contents on that line and the below are scrolled down one line. The cursor display is turned off.

Parameter A = Line position (0 to 3)

Return none

Affected Register AF

ERS1LN

Entry Address 0145H

Function Clear the specified line of the screen. The line remains as a blank line.

Parameter A = Line position (0 to 3)

Return none

Affected Register AF

ERSSTR

Entry Address 013FH

Function Display as many space characters as specified by the number given in B register, from the specified position of the screen.

Parameter
A = 0: Character mode
D = X coordinate of the screen position
E = Y coordinate of the screen position
A = 1: Graphics mode
DE = X coordinate of the screen position
HL = Y coordinate of the screen position
B = Number of spaces

Return
CF = 0: Normal termination
1: Some of the specified number of spaces could not be displayed. B register stores the number of the spaces that could not be displayed.

Affected Register AF, BC, DE

Remarks The position and state of the cursor remain unchanged.

SMBLREAD

Entry Address 0139H

Function Read the state of the status line symbols.
The state of a set of symbols specified by B register is read into A register.

Parameter B = Symbol set number (0 to 2)

B = 00H	DEF	I	II	III	SMALL		SHIFT	BUSY
B = 01H		RUN	PRO	RES ERVE		RAD	G	DE
B = 02H	KB II				S		CTRL	Low battery

MSB LSB

Content of A register 1 = ON

Return A = State of symbols
(If a bit of A register is 1, this means the symbol corresponding to that bit is displayed on the status line of the screen. If a bit is 0, the symbol is not displayed.)

Affected Register AF

SMBLSET

Entry Address 013CH

Function Set a set of symbols specified by B register to the state specified by A register.

Parameter B = Symbol set number (0 to 2)
A = Symbol state (expressed by bit pattern)

B = 00H	DEF	I	II	III	SMALL		SHIFT	BUSY
B = 01H		RUN	PRO	RES ERVE		RAD	G	DE
B = 02H	KB II				S		CTRL	Low battery

MSB LSB

Content of A register 1 = ON

Return none

Remarks If either of KBII or S is set to 1, S symbol is displayed.

RVSCHR

Entry Address 011BH

Function Change the display of a string of characters currently on the screen to the reverse-video mode. (Character mode)

Parameter D = X coordinate of the beginning of the string
E = Y coordinate of the beginning of the string
A = Number of the characters to be displayed in reverse video

Return CF = 1 if the specified coordinates are out of the range.

Remarks The cursor display is turned off.

SETANK

Entry Address 0109H

Function Set the display mode to the character mode and move the cursor to the home position (0,0).

Parameter none

Return none

Affected Register CRSRX (Cursor X coordinate: F060H), CRSRY (Cursor Y coordinate: F05FH)

DOTSET

Entry Address 0127H

Function Give the same function as the PSET commands of BASIC.

Parameter DOTSOP (F096H) = 00H: Dot set
01H: Dot reset
02H: Invert the current dot state
X1POS (F08EH=low byte; F08FH=high byte)
= X coordinate (−32768 to 32767)
Y1POS (F090H=low byte; F091H=high byte)
= Y coordinate (−32768 to 32767)

Return none

Affected Register AF, LINPTN (F097H, F098H), DOTSOP (F096H)

Remarks The effective dot addresses of the screen are: $0 \leq X \leq 155$ and $0 \leq Y \leq 31$. Specifying a dot address out of these ranges causes nothing.
Specify X and Y coordinate values in two bytes each (a negative value in the complement expression). That is, 0 to 32767 is expressed as 0000H to 7FFFH, and −32768 to −1 as 8000H to FFFFH.

Example The following will give the same results as the PSET(100,11),X statement in BASIC.

```
POKE &F08E,&64,&00,&0B,&00
POKE &F096,&02
CALL &0127
```

LINE

Entry Address 0121H

Function Give the same function as the LINE command of BASIC.

Parameter
DOTSOP (F096H) = 00H: Dot set
01H: Dot reset
02H: Invert
X1POS (F08EH=low byte; F08FH=high byte)
= X coordinate of the starting point (−32768 to 32767)
Y1POS (F090H=low byte; F091H=high byte)
= Y coordinate of the starting point (−32768 to 32767)
X2POS (F092H=low byte; F093H=high byte)
= X coordinate of the end point (−32768 to 32767)
Y2POS (F094H=low byte; F095H=high byte)
= Y coordinate of the end point (−32768 to 32767)
LINPTN (F097H=low byte; F098H=high byte)
= Line pattern

Return
X1POS = Contents of X2POS
Y1POS = Contents of Y2POS
LINPTN = Line pattern to be drawn next time (the line pattern made by rotating the line pattern used this time one dot to the left)

Affected Register AF, BC, DE, HL, X1POS, Y1POS, LINPTN

Remarks The effective dot addresses of the screen are: $0 \leq X \leq 155$ and $0 \leq Y \leq 31$.
Specify X and Y coordinate values in two bytes each (a negative value in the complement expression). That is, 0 to 32767 is expressed as 0000H to 7FFFH, and −32768 to −1 as 8000H to FFFFH.
Make a line pattern (LINPTN) in the same manner as for the LINE command of BASIC.
For example, to make the following line pattern:

write ADA9H in LINPTN (i.e., write A9H in F097H and ADH in F098H.)

Example The following will give the same results as the LINE(−5,−3)−(100,50),,&ADA9 statement in BASIC.

```
POKE &F08E,&FB,&FF,&FD,&FF,&64,&00,&32,&00
POKE &F096,&00,&A9,&AD
CALL &0121
```

DOTREAD

Entry Address 012AH

Function Give the same function as the POINT command of BASIC.

Parameter X1POS (F08EH=low byte; F08FH=high byte)
= X coordinate (in graphics mode)
Y1POS (F090H=low byte; F091H=high byte)
= Y coordinate (in graphics mode)

Return A = 00H: A dot is not currently displayed at the specified point on the screen.
01H: A dot is currently displayed at the specified point on the screen.

Affected Register AF

Remarks If the specified point is out of the displayable range of the screen, the routine returns A=00H.
Specify X and Y coordinate values in two bytes each (a negative value in the complement expression). That is, 0 to 32767 is expressed as 0000H to 7FFFH, and −32768 to −1 as 8000H to FFFFH.

PRTGCHR

Entry Address 014EH

Function Display a character whose character code is given in A register at the current graphics cursor position, then move the graphics cursor 6 dots to the right (i.e., add 06H to the X coordinate of the graphics cursor.)

Parameter A = Character code
DOTSOP (F096H)
= 00H: Display the new character pattern, erasing the previously displayed 6 × 8 dot pattern.
= 01H: Display the ORed pattern of the new character pattern and the previously displayed 6 × 8 dot pattern.
= 02H: Display the XORed pattern of the new character pattern and the previously displayed 6 × 8 dot pattern.

Return none

Affected Register AF, GCRSRX (Graphics cursor X coordinate: F099H=low byte; F09AH=high byte)

SAVELCD

Entry Address 015DH

Function Save the contents (the bit image data of 156 bytes) of a screen line specified by A register, into the sequential memory locations whose starting address is specified by DE register.

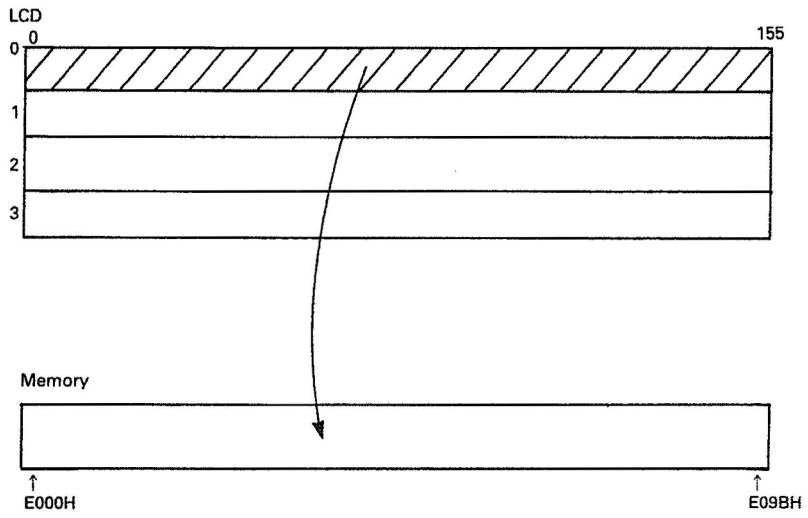
Parameter DE = Starting address of the memory locations in which the line data are saved
A = Line position on the screen (00H to 03H: 00H designates the first line of the screen.)

Return none

Affected Register AF, DE

Example The following will save the contents of the first line of the screen to the memory locations from E000H to E09BH.

```
LD DE, E000H
LD A, 00H
CALL 015DH
```



PRTGSTR

Entry Address 00EEH

Function Display consecutively from the current graphics cursor position a string of characters whose character codes are stored in consecutive memory locations. This routine displays starting from the beginning of the contents of the memory locations whose starting address is given in DE register pair until it encounters a character whose code is given in A register (this code is not included in the screen display).

Parameter DE = Starting address of the memory locations that contain the character codes
A = Character code of the character (when the routine encounters this code, it terminates the current display operation, not including that code in the screen display.)
DOTSOP (F096H)
= 00H: Display the new character patterns, erasing the previously displayed 6×8 dot patterns.
= 01H: Display the ORed patterns of the new character patterns and the previously displayed 6×8 dot patterns.
= 02H: Display the XORed patterns of the new character patterns and the previously displayed 6×8 dot patterns.

Return DE = (Address of the memory location which contains the character code of the character last displayed) + 1
GCRSRX = X coordinate of the graphics cursor for the next display position

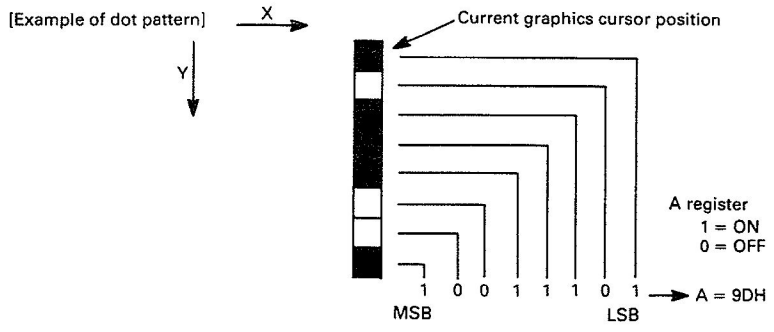
Affected Register DE, AF, GCRSRX (F099H=low byte, F09AH=high byte)

PRTGPTN

Entry Address 0154H

Function Display the content of A register as a 1 × 8 dot pattern at the current graphics cursor position, then move the graphics cursor one dot to the right (i.e., add 01H to the graphics cursor X coordinate.)

Parameter A = Dot pattern



DOTSOP (F096H)
= 00H: Display the new 1 × 8 dot pattern, erasing the previously displayed 1 × 8 dot pattern.
= 01H: Display the ORed pattern of the new 1 × 8 dot pattern and the previously displayed 1 × 8 dot pattern.
= 02H: Display the XORed pattern of the new 1 × 8 dot pattern and the previously displayed 1 × 8 dot pattern.

Return none

Affected Register AF, GCRSRX

GPTNREAD

Entry Address 015AH

Function Read the 1 × 8 dot pattern from the current graphics cursor position toward the positive direction of Y axis into A register.
The dot pattern is read into A register in the same manner as described in "Example of dot pattern" of PRTGPTN routine. If part of the 1 × 8 dot pattern to be read exceeds the screen area, those dots of the exceeded part are read to be 0.

Parameter none

Return A = One binary byte expressing the dot pattern

Affected Register AF

CGMODE

Entry Address 0133H

Function Change the character generator mode between the PC-1500 mode and the PC-1600 mode.

Parameter A = 0: PC-1600 mode
1: PC-1500 mode

Return none

Affected Register AF, LCDWK1 (F05DH)

CPY1500LCD

Entry Address 0157H

Function Copy the contents of the fourth line of the screen (dot pattern of 156 bytes) to the PC-1500 display RAM (7600H to 764FH; or 7700H to 774FH (address on LH-5803)) with the bit configuration changed for the PC-1500 format. The state of the status line symbols is also copied.

Parameter none

Return none

Affected Register none

GCRSRPOS

Entry Address 0148H

Function Read the current graphics cursor position.

Parameter none

Return DE = X coordinate ($-32768 \leq X \leq 32767$)
BC = Y coordinate ($-32768 \leq Y \leq 32767$)

Affected Register DE, BC

Remarks X and Y coordinates are returned as a two-byte value (a negative value in the complement expression). That is, 0 to 32767 is expressed as 0000H to 7FFFH, and −32768 to −1 as 8000H to FFFFH.

BOX

Entry Address 0124H

Function Draw a box with the inside filled in.

Parameter DOTSOP (F096H) = 00H: Dot set
01H: Dot reset
02H: Invert
X1POS (F08EH=low byte; F08FH=high byte)
= X coordinate of the starting point of the diagonal (−32768 to 32767)

Y1POS (F090H=low byte; F091H=high byte)
= Y coordinate of the starting point of the diagonal (−32768 to 32767)
X2POS (F092H=low byte; F093H=high byte)
= X coordinate of the end point of the diagonal (−32768 to 32767)
Y2POS (F094H=low byte; F095H=high byte)
= Y coordinate of the end point of the diagonal (−32768 to 32767)
LINPTN (F097H=low byte; F098H=high byte)
= Line pattern

Return

X1POS = Contents of X2POS
Y1POS = Contents of Y2POS
LINPTN = Line pattern to be drawn next time (the line pattern made by rotating the line pattern used this time one dot to the left)

Affected Register AF, BC, DE, HL, X1POS, Y1POS, LINPTN

Remarks

The effective dot addresses of the screen are: $0 \leq X \leq 155$ and $0 \leq Y \leq 31$.
Specify X and Y coordinate values in two bytes each (a negative value in the complement expression). That is, 0 to 32767 is expressed as 0000H to 7FFFH, and −32768 to −1 as 8000H to FFFFH.
Make a line pattern (LINPTN) in the same manner as for the LINE command of BASIC.
For example, to make the following line pattern:

write ADA9H in LINPTN (i.e., write A9H in F097H and ADH in F098H.)

Example

The following will give the same results as the LINE(−5,−3)−(100,50),,&ADA9,BF statement in BASIC.

```
POKE &F08E,&FB,&FF,&FD,&FF,&64,&00,&32,&00
POKE &F096,&00,&A9,&AD
CALL &0124
```

GCRSRSET

Entry Address 014BH

Function Set the graphics cursor position.

Parameter

DE = X coordinate ($-32768 \leq X \leq 32767$)
BC = Y coordinate ($-32768 \leq Y \leq 32767$)

Return none

Affected Register GCRSRX, GCRSRY

Remarks Specify X and Y coordinate values in two bytes each (a negative value in the complement expression). That is, 0 to 32767 is expressed as 0000H to 7FFFH, and −32768 to −1 as 8000H to FFFFH.

CLS

Entry Address 0112H

Function Clear the screen display.

Parameter none

Return none

Affected Register none

Remarks The cursor display is turned off.

DSPCTR

Entry Address 00E5H

Function Enable/disable the LCD.

Parameter A = 01H: Enable the LCD.
00H: Disable the LCD.

Return none

Affected Register AF

LOADLCD

Entry Address 0160H

Function Load 156 bytes of data from the sequential memory locations whose starting address is specified by DE register, as the 156 × 8 dot pattern to a screen line specified by A register.

Parameter DE = Starting address of the memory locations from which the 156×8 dot pattern are loaded
A = Line position on the screen (00H to 03H)

Return none

Affected Register AF, DE

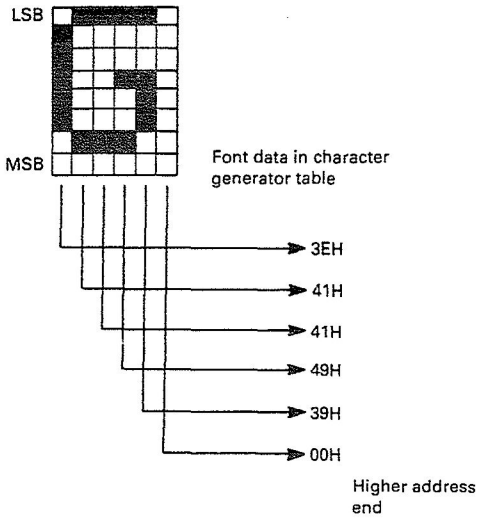
3.1.2 Work Area used for IOCS Routines for LCD

Work name	Address	Bytes	Description
LCDWK1	F05DH	1	The bits of F05DH have the following meanings. Bit 2: Specifies the selected character font set. 0 = PC-1600 font 1 = PC-1500 font Bit 3: Specifies whether or not to use the character font of the character codes 00H to 1FH. 0 = The character codes 00H to 1DH and 1FH are processed as a space, and 1EH as an insert mark "□". 1 = The character font for the codes 00H to 1FH uses the user-defined character font data that are stored in the memory locations specified by CTRCGA and CTRCGB. Bit 4: Specifies the cursor blinking speed. 0 = Normal speed 1 = Double speed
CRSRX	F060H	1	Cursor X coordinate
CRSRY	F05FH	1	Cursor Y coordinate
CTRCGA	F061H (Low byte) F062H (High byte)	2	Starting address of the memory locations which contain the character font data for the character codes 00H to 1FH (The starting address must be greater than 8000H.)
CTRCGB	F063H	1	Bank number (0 to 7) of the memory locations which contain the character font data for the character codes 00H to 1FH
UPAGGA	F064H (Low byte) F065H (High byte)	2	Starting address of the memory locations which contain the character font data for the character codes 80H to FFH (The starting address must be greater than 8000H.)
UPAGGB	F066H	1	Bank number (0 to 7) of the memory locations which contain the character font data for the character codes 80H to FFH
CRSRT	F067H	1	Cursor type 00H = Cursor display OFF 01H = Underline cursor 02H = Blinking square cursor 03H = Blinking space cursor
X1POS	F08EH (Low byte) F08FH (High byte)	2	
Y1POS	F090H (Low byte) F091H (High byte)	2	
X2POS	F092H (Low byte) F093H (High byte)	2	
Y2POS	F094H (Low byte) F095H (High byte)	2	
DOTSOP	F096H	1	00H = Dot set 01H = OR or dot reset 02H = XOR or invert

Work name	Address	Bytes	Description
LINPTN	F097H (Low byte) F098H (High byte)	2	Line pattern
GCRSRX	F099H (Low byte) F09AH (High byte)	2	Graphics cursor X coordinate
GCRSRY	F09BH (Low byte) F09CH (High byte)	2	Graphics cursor Y coordinate

3.1.3 Character Font

The character fonts used in PC-1600 are composed of 6 by 8 dots each and the character font data are stored in the character generator table in the following format: each 8-dot column of one character font is expressed in one byte and one character font is formed by arranging six column data (6 bytes) from the left end to the right end of the font. (See the figure below.)



The PC-1600 has three character generator tables:

(1) Character generator table for those characters of codes 20H to 7FH
This table is in the main ROM and the fonts of these characters cannot be changed by the user.

(2) Character generator table for those characters of codes 80H to FFH
This table is in the main ROM. The user can use different fonts for these codes by preparing user-defined fonts in RAM and changing the contents of UPAGGA and UPAGGB.

(3) Character generator table for those characters of codes 00H to 1FH
No font data exist in the main ROM. The user can define his own fonts for these codes: prepare user-defined fonts in RAM, change the contents of CTRCGA and CTRCGB, and set bit 3 of LCDWK1 to "1".
As described above, the user can change the character generator tables (2) and (3) to the user-defined tables prepared in RAM. In this case, since all character fonts of the codes assigned to each table are to be changed, the user must prepare font data for all of these codes in RAM.

3.2 KEY INPUT

3.2.1 IOCS Routines for Key Input

The following table lists the names, entry addresses and functions of the IOCS routines for key input. Refer to section 10.2 for the key codes and the codes handled in the key buffer.

Name	Entry address	Function
KEYGET	0166H	Read one character from the key buffer. If the key buffer is empty, the routine waits for a key input.
KEYGETR	0169H	Same function as KEYGET except the routine does not wait for a key input even if the key buffer is empty.
KBUFSET	016CH	Load data into the key buffer or clear the key buffer.
BREAKCHK	016FH	Read the state of the BREAK key.
CURUDCHK	0172H	Read the state of the [↓] or [↑] key.
KEYDIRECT	0175H	Scan the keys and read the key code of the key that has been pressed when the routine is called.
KEYSTRB	0178H	Scan one row of keys to identify a particular one of more than one key pressed at the same time.
KEYAUX	017BH	Specify the key input device (main keyboard or RS-232C).
KEYSTATSET	017EH	Specify the key repeat function and the key click function.
KEYSTATREAD	0181H	Read the settings of the key repeat and click functions and the current key input device.
OFFCHK	0184H	Read the state of the OFF key.
KEYGETND	0187H	Read the first character in the key buffer without changing the contents of the key buffer.
BREAKRESET	018AH	Clear the latch of the BREAK key.

KEYGET

Entry Address 0166H

Function Read one character from the key buffer (64 bytes). If the key buffer is empty, the routine waits for a key input. When the auto power-off function is enabled, if no key input occurs for about 10 minutes, the power is automatically turned off. In this state, if the BREAK[ON] key is pressed, the routine is restarted.

Parameter none

Return CF = 0: Normal termination A = Key code
1: The routine has resulted in a timeout error. (That is, when this routine was called, the key buffer was empty and no key input occurred within 10 minutes. If this happens, the key wait abort bit (bit 4 of KEYWK2 (F07AH)) is set to 1.)

Affected Register AF

KEYGETR

Entry Address 0169H

Function Same as the KEYGET routine except the routine does not wait for a key input even if the key buffer is empty

Parameter none

Return CF = 1 if the key buffer is empty.

Affected Register AF

KBUFSET

Entry Address 016CH

Function Clear the key buffer, then load as many data as specified by A register, from the memory locations whose starting address is specified by DE register into the key buffer.
While the routine is being executed, the key scan interrupt is disabled.

Parameter DE = Starting address of the memory locations where data are stored
A = Number of data to be loaded ($1 \leq A \leq 63$; If A=0, then the key buffer is cleared.)

Return none

Affected Register DE

Remarks Data to be loaded to the key buffer must be within only one bank.

BREAKCHK

Entry Address 016FH

Function Read the state of the BREAK key. If the BREAK key has been pressed, the key buffer is cleared.

Parameter none

Return CF = 1: The BREAK key has been pressed.
0: The BREAK key has not been pressed.

Affected Register AF

CURUDCHK

Entry Address 0172H

Function Check whether or not the [↑] or [↓] key has been pressed.

Parameter none

Return CF = 0: Neither [↑] nor [↓] key has been pressed.
1: The [↑] and/or [↓] key has been pressed.
A = Kind of the key(s) pressed
(If bit 7 is 1, the [↓] key has been pressed, and if bit 6 is 1, the [↑] key has been pressed.)

Affected Register AF

KEYDIRECT

Entry Address 0175H

Function Read the code of the key that has been pressed when the routine is called. While the routine is being executed, the key scan interrupt is disabled. The key buffer is also cleared.

Parameter none

Return A = Key code (A = 00H if no key has been pressed.)

Affected Register AF

KEYSTRB

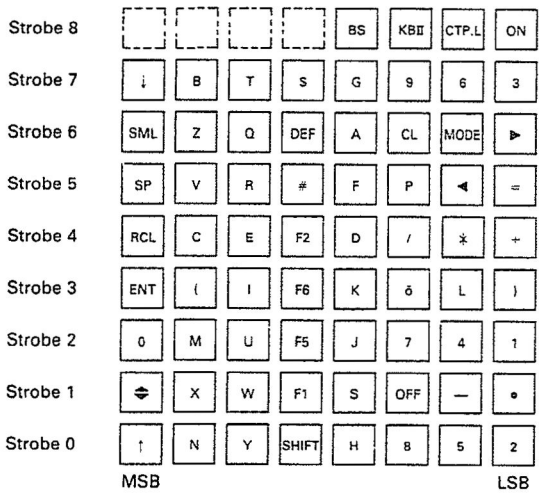
Entry Address 0178H

Function Read the state of the keys corresponding to the specified key strobe. Those bits corresponding to the keys pressed are read to be 0. While the routine is being executed, the key scan interrupt is disabled.

Parameter A = Strobe number (00H to 08H)

Return A = State of the keys

Affected Register AF



Key state data

0 = The key has been pressed.
1 = The key has not been pressed.

KEYAUX

Entry Address 017BH

Function Specify the key input device (main keyboard or RS-232C).

Parameter A = 00H: Main keyboard
02H: RS-232C

Return CF = 1: Parameter specification error

Affected Register AF

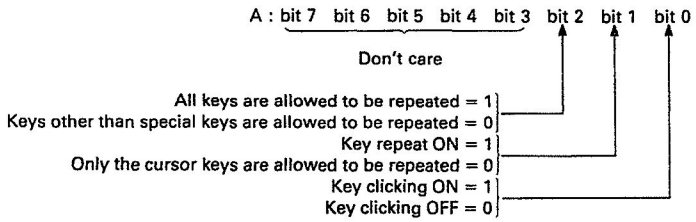
Remarks If RS-232C is selected as the key input device, KEYGET, KEYGETR and KEYDIRECT routines are executed to RS-232C.

KEYSTATSET

Entry Address 017EH

Function Specify the key repeat function and the key click function.

Parameter A = Function



Return none

Affected Register none

Remarks The special keys described in "Parameter" are: CTRL, SHIFT, SML, ⇔, function keys, KBII, RCL, MODE, ON, and OFF.

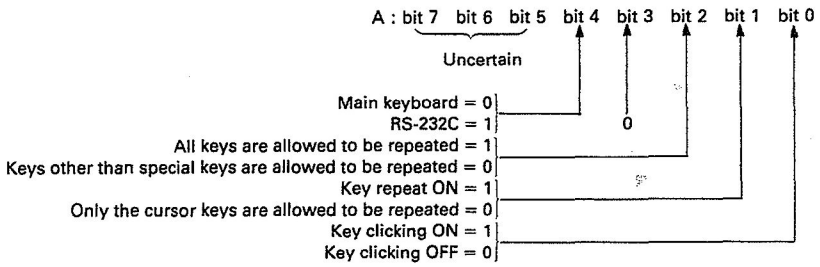
KEYSTATREAD

Entry Address 0181H

Function Read the settings of the key repeat and click functions and the current key input device.

Parameter none

Return A = Settings of the functions



Affected Register AF

Remarks The special keys described in "Return" are: CTRL, SHIFT, function keys, KBII, RCL, MODE, SML, ⇔, ON, and OFF.

OFFCHK

Entry Address 0184H

Function Read the state of the OFF key.

Parameter none

Return CF = 0: The OFF key has not been pressed.
1: The OFF key has been pressed.

Affected Register AF

KEYGETND

Entry Address 0187H

Function Read one byte of key code from the beginning of the key buffer without changing the contents of the key buffer.

Parameter none

Return CF = 0: A key code was read. A = Key code
1: The key buffer was empty.

Affected Register none

BREAKRESET

Entry Address 018AH

Function Whether or not the BREAK key has been pressed can be known by checking the state of bit 1 of the contents of the I/O address 1BH: if bit 1 is "0", the BREAK key has not been pressed, and if bit 1 is "1", the BREAK key has been pressed.
This routine resets the content of that bit.
The key buffer is cleared when the routine is called.

Parameter none

Return none

Affected Register AF

3.2.2 Work Area used for IOCS Routines for Key Input

The following table shows the work area map for key input.

Address	Contents
F079H	Key function flag Bit 0: Disable the key scan interrupt. Bit 1: Enable the key clicking. Bit 2: Key repeat Bit 3: Range of keys to be repeated 0: All keys other than the special keys 1: All keys including the special keys Bit 4: Delay before starting the key repeat 0: Long 1: Short Bit 5: Conditions for key repeat 0: If the same key code is generated consecutively, only one key code is accepted into the key buffer. 1: Even the same key code is repeated. Bit 6: Repeat pitch 0: Slow 1: Fast Bit 7: Disable the key code conversion upon execution of the KEYGET routine.
F07BH	Key function flag Bit 1: Disable the auto power-off function. Bit 2: Disable the OFF key.
F07FH	Key buffer write pointer This pointer specifies the position in the key buffer to which the next key data is to be written. If the key buffer is full, MSB is set to "1".
F080H	Key buffer read pointer This pointer specifies the position in the key buffer from which the next key data is to be read. If the read pointer has the same value as the write pointer, this means there is no data to be read.
F083H	Key code last returned
F084H	Bank number of the key code conversion table for SHIFT mode
F085H ~ F086H	Address of the key code conversion table for SHIFT mode This specifies the starting address of the key code conversion table used in the SHIFT mode upon execution of the KEYGET routine.
F087H	Bank number of the key code conversion table for KBII mode
F088H ~ F089H	Address of the key code conversion table for KBII mode This specifies the starting address of the key code conversion table used in the [S] mode upon execution of the KEYGET routine.
F08AH	Bank number of the key code conversion table for SHIFT-KBII mode
F08BH ~ F08CH	Address of the key code conversion table for SHIFT-KBII mode This specifies the starting address of the key code conversion table used in the SHIFT-[S] mode upon execution of the KEYGET routine.
F0DFH ~ F11EH	Key buffer
F11FH	Bank number of the key code table
F120H ~ F121H	Address of the key code table This specifies the starting address of the key matrix to key code conversion table used in the key scan routine.

3.2.3 Scanning of ON (BREAK) Key

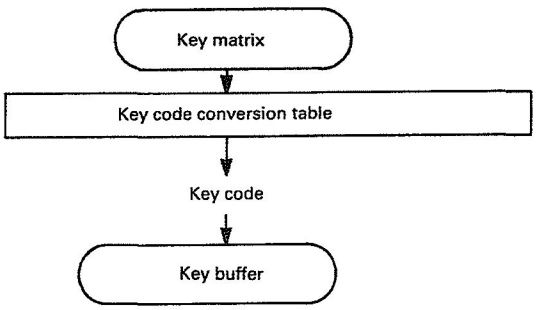
Whether or not the BREAK key has been pressed can be known by checking the state of bit 1 of the contents of the I/O address 1BH: if bit 1 is "0", the BREAK key has not been pressed, and if bit 1 is "1", the BREAK key has been pressed. This state of bit 1 is latched. To reset it, execute BREAKRESET routine (entry address 018AH). The key buffer is cleared when the routine is executed.

3.2.4 Entry of International Characters and Symbols

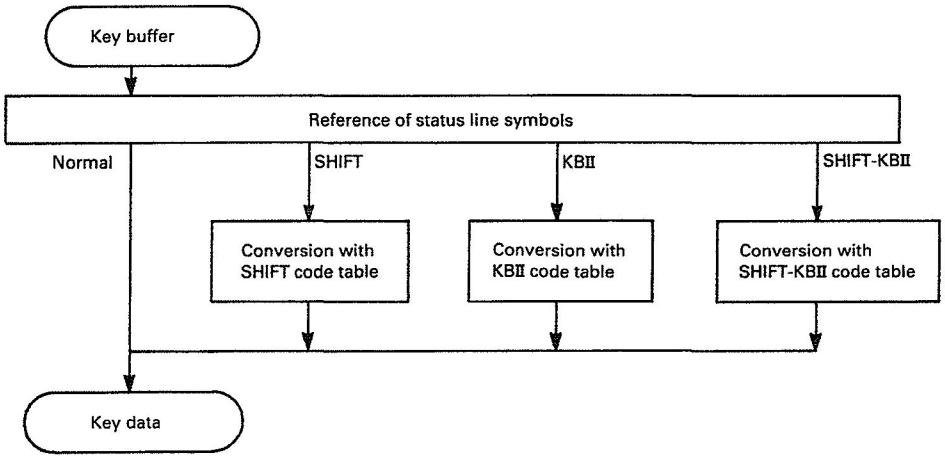
When the KBII and/or SHIFT symbol is on, execution of KEYGET or KEYGETR key input IOCS routine allows to enter the international characters and symbols.

3.2.5 Data Flow from Key Scanning to KEYGET Routine

(1) 1/64-sec interrupt handling routine



(2) KEYGET routine



Key data are made as described above. The key code table and the other tables are assigned by specifying the starting address of each table in RAM with a pointer. Therefore, the user can make his own key layout by changing the contents of these pointers so as to point the user-defined tables. (See also section 3.2.6.)

3.2.6 Re-definition of Keys

(1) Key tables

The PC-1600 has four kinds of standard key tables in ROM. Since the PC-1600 uses these tables by specifying the starting address of each table with a pointer, the user can use his own tables by changing the contents of these pointers.

Table	Function	Pointer			
		Name	Address	Size (in bytes)	Contents
Key code table	Convert key matrix to key code.	KEYCDA	F120H	2	Starting address of the key code table (in the order of the low and the high bytes)
		KEYCDB	F11FH	1	Bank number of the key code table
SHIFT code table	Convert key code to SHIFT code.	SFTCDA	F084H	2	Starting address of the SHIFT code table (in the order of the low and the high bytes)
		SFTCDB	F086H	1	Bank number of the SHIFT code table
KBII code table	Convert key code to KBII code.	KNCD1A	F087H	2	Starting address of the KBII code table (in the order of the low and the high bytes)
		KNCD1B	F089H	1	Bank number of the KBII code table
SHIFT-KBII code table	Convert key code to SHIFT-KBII code.	KNCD2A	F08AH	2	Starting address of the SHIFT-KBII code table (in the order of the low and the high bytes)
		KNCD2B	F089H	1	Bank number of the SHIFT-KBII code table

(2) Structure of key table

The structure of each table is shown in the following source program list. To make a user-defined table, use the same structure as that of these standard tables provided in ROM.

The label names used in the source program list are related to the tables as follows:

Table	Label name
Key code table	KYCDTB
SHIFT code table	SFTCDT
KBII code table	KNCDT1
SHIFT-KBII code table	KNCDT2

Each table must be made within one bank.

```
;-----------------------------------------------------------------------
KYCDTB:
;                                       --- PB7 ---
        DEFB    ØEH                             ;on
;                                       --- PB6 ---
        DEFB    Ø3H                             ;ctrl
        DEFB    Ø4H                             ;2ndF
        DEFB    Ø5H                             ;BS
;                                       --- PA7 ---
        DEFB    33H                             ;3
        DEFB    36H                             ;6
        DEFB    39H                             ;9
        DEFB    47H                             ;G
        DEFB    14H                             ;F4
        DEFB    54H                             ;T
        DEFB    42H                             ;B
        DEFB    ØAH                             ;curosr down
;                                       --- PA6 ---
        DEFB    ØCH                             ;cursor right
        DEFB    1FH                             ;mode
        DEFB    18H                             ;cl
        DEFB    41H                             ;A
        DEFB    1BH                             ;def
        DEFB    51H                             ;Q
        DEFB    5AH                             ;Z
        DEFB    Ø2H                             ;small key
;                                       --- PA5 ---
        DEFB    3DH                             ;=
        DEFB    Ø8H                             ;cursor left
        DEFB    50H                             ;P
        DEFB    46H                             ;F
        DEFB    13H                             ;F3
        DEFB    52H                             ;R
        DEFB    56H                             ;V
        DEFB    20H                             ;space
;                                       --- PA4 ---
        DEFB    2BH                             ;+
        DEFB    2AH                             ;*
        DEFB    2FH                             ;/
        DEFB    44H                             ;D
        DEFB    12H                             ;F2
        DEFB    45H                             ;E
        DEFB    43H                             ;C
        DEFB    19H                             ;rcl
;                                       --- PA3 ---
        DEFB    29H                             ;)
        DEFB    4CH                             ;L
        DEFB    4FH                             ;O
        DEFB    4BH                             ;K
        DEFB    16H                             ;F6
        DEFB    49H                             ;I
        DEFB    28H                             ;(
        DEFB    ØDH                             ;enter
;                                       --- PA2 ---
        DEFB    31H                             ;1
        DEFB    34H                             ;4
        DEFB    37H                             ;7
        DEFB    4AH                             ;J
        DEFB    15H                             ;F5
        DEFB    55H                             ;U
        DEFB    4DH                             ;M
        DEFB    30H                             ;Ø
;                                       --- PA1 ---
        DEFB    2EH                             ;.
        DEFB    2DH                             ;-
        DEFB    ØFH                             ;off
        DEFB    53H                             ;S
        DEFB    11H                             ;F1
        DEFB    57H                             ;W
        DEFB    58H                             ;X
        DEFB    Ø9H                             ;rotary key
;                                       --- PAØ ---
        DEFB    32H                             ;2
        DEFB    35H                             ;5
        DEFB    38H                             ;8
        DEFB    48H                             ;H
        DEFB    Ø1H                             ;shift
        DEFB    59H                             ;Y
        DEFB    4EH                             ;N
        DEFB    ØBH                             ;curosor up
```

```
;
;-------------------------------------------------------------------------
SFTCDT:                         ;shift key code
;                       -- 08H --
        DEFB    1DH             ;cursol left
        DEFB    09H             ;rotary
        DEFB    0AH             ;curosor down
        DEFB    0BH             ;cursor up
        DEFB    1CH             ;cursor right
        DEFB    0DH             ;enter
        DEFB    0EH             ;ON
        DEFB    0FH             ;OFF
;                       -- 10H --
        DEFB    0               ;no key
        DEFB    21H             ;F1
        DEFB    22H             ;F2
        DEFB    23H             ;F3
        DEFB    24H             ;F4
        DEFB    25H             ;F5
        DEFB    26H             ;F6
        DEFB    0               ;no key
        DEFB    1AH             ;CL
        DEFB    19H             ;RCL
        DEFB    0               ;no key
        DEFB    1BH             ;DEF
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    1FH             ;MODE
;                       -- 20H --
        DEFB    5EH             ;space
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    3CH             ;(
        DEFB    3EH             ;)
        DEFB    3AH             ;*
        DEFB    3BH             ;+
        DEFB    0               ;no key
        DEFB    2CH             ;-
        DEFB    5FH             ;.
        DEFB    3FH             ;/
;                       -- 30H --
        DEFB    7CH             ;0
        DEFB    27H             ;1
        DEFB    5BH             ;2
        DEFB    5DH             ;3
        DEFB    60H             ;4
        DEFB    7BH             ;5
        DEFB    7DH             ;6
        DEFB    5CH             ;7
        DEFB    7EH             ;8
        DEFB    39H             ;9
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    40H             ;=
        DEFB    0               ;no key
        DEFB    0               ;no key
;                       --- 40H ---
        DEFB    0               ;no key
        DEFB    61H             ;A
        DEFB    62H             ;B
        DEFB    63H             ;C
        DEFB    64H             ;D
        DEFB    65H             ;E
        DEFB    66H             ;F
        DEFB    67H             ;G
        DEFB    68H             ;H
        DEFB    69H             ;I
        DEFB    6AH             ;J
        DEFB    6BH             ;K
        DEFB    6CH             ;L
        DEFB    6DH             ;M
        DEFB    6EH             ;N
        DEFB    6FH             ;O
```

```
;                       --- 50H ---
        DEFB    70H             ;P
        DEFB    71H             ;Q
        DEFB    72H             ;R
        DEFB    73H             ;S
        DEFB    74H             ;T
        DEFB    75H             ;U
        DEFB    76H             ;V
        DEFB    77H             ;W
        DEFB    78H             ;X
        DEFB    79H             ;Y
        DEFB    7AH             ;Z
;
;-----------------------------------------------------------------------
KNCDT1:                         ;KB2 key code table
                                ;
;                       -- 08H --
        DEFB    08H             ;cursol left
        DEFB    09H             ;rotary
        DEFB    0AH             ;curosor down
        DEFB    0BH             ;cursor up
        DEFB    0CH             ;cursor right
        DEFB    0DH             ;enter
        DEFB    0EH             ;ON
        DEFB    0FH             ;OFF
;                       -- 10H --
        DEFB    0               ;no key
        DEFB    11H             ;F1
        DEFB    12H             ;F2
        DEFB    13H             ;F3
        DEFB    14H             ;F4
        DEFB    15H             ;F5
        DEFB    16H             ;F6
        DEFB    0               ;no key
        DEFB    18H             ;CL
        DEFB    0BBH            ;RCL
        DEFB    0               ;no key
        DEFB    1BH             ;DEF
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    1FH             ;MODE
;                       -- 20H --
        DEFB    20H             ;space
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    09EH            ;(
        DEFB    09FH            ;)
        DEFB    2AH             ;*
        DEFB    2BH             ;+
        DEFB    0               ;no key
        DEFB    2DH             ;-
        DEFB    2EH             ;.
        DEFB    2FH             ;/
;                       -- 30H --
        DEFB    30H             ;0
        DEFB    31H             ;1
        DEFB    32H             ;2
        DEFB    33H             ;3
        DEFB    34H             ;4
        DEFB    35H             ;5
        DEFB    36H             ;6
        DEFB    37H             ;7
        DEFB    38H             ;8
        DEFB    39H             ;9
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    3DH             ;=
        DEFB    0               ;no key
        DEFB    0               ;no key
```

```
;                       ---- 40H ---
        DEFB    0                       ;no key
        DEFB    0A0H                    ;A
        DEFB    097H                    ;B
        DEFB    08DH                    ;C
        DEFB    0A1H                    ;D
        DEFB    08BH                    ;E
        DEFB    0A2H                    ;F
        DEFB    0A3H                    ;G
        DEFB    0ADH                    ;H
        DEFB    08FH                    ;I
        DEFB    0A8H                    ;J
        DEFB    0A6H                    ;K
        DEFB    0A7H                    ;L
        DEFB    09BH                    ;M
        DEFB    09CH                    ;N


        DEFB    0A5H                    ;O
;                       --- 50H ---
        DEFB    080H                    ;P
        DEFB    08EH                    ;Q
        DEFB    099H                    ;R
        DEFB    090H                    ;S
        DEFB    09AH                    ;T
        DEFB    092H                    ;U
        DEFB    095H                    ;V
        DEFB    089H                    ;W
        DEFB    08AH                    ;X
        DEFB    098H                    ;Y
        DEFB    085H                    ;Z
;
;-------------------------------------------------------------------------------
KNCDT2:                                 ;shift KB2 key code table
                                        ;
;                       -- 08H --
        DEFB    1DH                     ;cursol left
        DEFB    09H                     ;rotary
        DEFB    0AH                     ;curosor down
        DEFB    0BH                     ;cursor up
        DEFB    1CH                     ;cursor right
        DEFB    0DH                     ;enter
        DEFB    0EH                     ;ON
        DEFB    0FH                     ;OFF
;                       -- 10H --
        DEFB    0                       ;no key
        DEFB    21H                     ;F1
        DEFB    22H                     ;F2
        DEFB    23H                     ;F3
        DEFB    24H                     ;F4
        DEFB    25H                     ;F5
        DEFB    26H                     ;F6
        DEFB    0                       ;no key
        DEFB    1AH                     ;CL
        DEFB    19H                     ;RCL
        DEFB    0                       ;no key
        DEFB    1BH                     ;DEF
        DEFB    0                       ;no key
        DEFB    0                       ;no key
        DEFB    0                       ;no key
        DEFB    1FH                     ;MODE
```

```
;                       -- 20H --
        DEFB    5EH             ;space
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    0AEH            ;(
        DEFB    0AFH            ;)
        DEFB    3AH             ;*
        DEFB    3BH             ;+
        DEFB    0               ;no key
        DEFB    2CH             ;-
        DEFB    5FH             ;.
        DEFB    3FH             ;/
;                       -- 30H --
        DEFB    7CH             ;0
        DEFB    27H             ;1
        DEFB    5BH             ;2
        DEFB    5DH             ;3
        DEFB    60H             ;4
        DEFB    7BH             ;5
        DEFB    7DH             ;6
        DEFB    5CH             ;7
        DEFB    7EH             ;8
        DEFB    39H             ;9
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    0               ;no key
        DEFB    40H             ;=
        DEFB    0               ;no key
        DEFB    0               ;no key
;                       --- 40H ---
        DEFB    0               ;no key
        DEFB    0A0H            ;A
        DEFB    096H            ;B
        DEFB    08CH            ;C
        DEFB    0A1H            ;D
        DEFB    08BH            ;E
        DEFB    0A2H            ;F
        DEFB    0A3H            ;G
        DEFB    0ABH            ;H
        DEFB    086H            ;I
        DEFB    0ACH            ;J
        DEFB    0A9H            ;K
        DEFB    0AAH            ;L
        DEFB    09BH            ;M
        DEFB    09DH            ;N
        DEFB    0A4H            ;O
;                       --- 50H ---
        DEFB    087H            ;P
        DEFB    084H            ;Q
        DEFB    094H            ;R
        DEFB    082H            ;S
        DEFB    081H            ;T
        DEFB    091H            ;U
        DEFB    093H            ;V
        DEFB    089H            ;W
        DEFB    088H            ;X
        DEFB    098H            ;Y
        DEFB    083H            ;Z
;
```

3.3 FILES

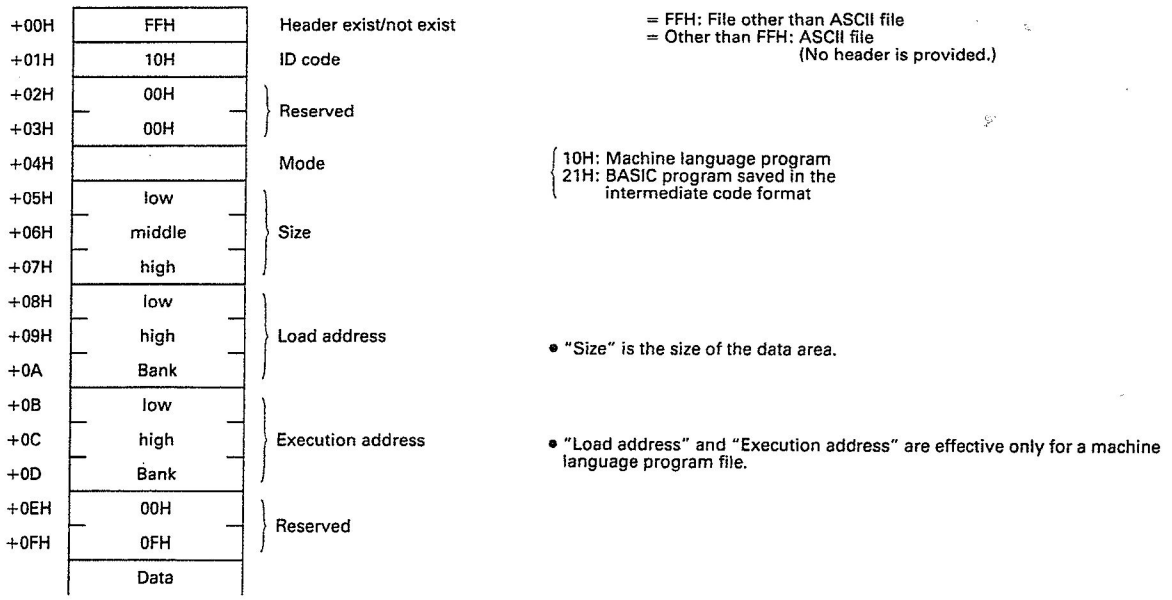
3.3.1 Files Handled in BASIC

(1) Type of Files

The PC-1600 BASIC can handle the following three types files.

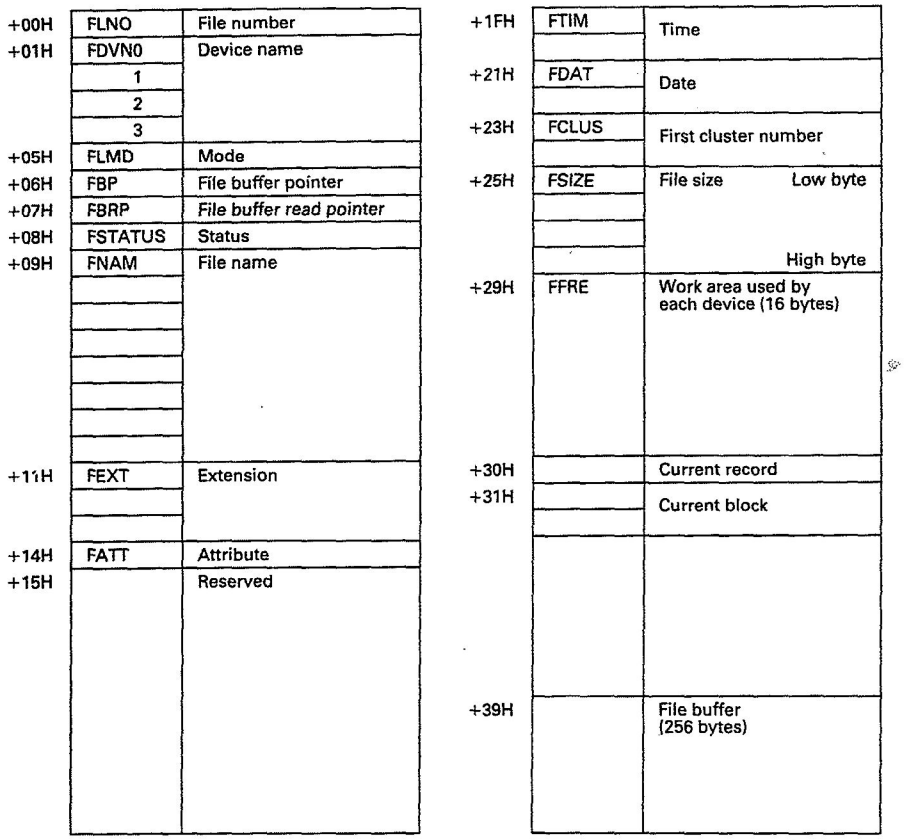
1. ASCII file
2. BASIC program file saved in the intermediate code format
3. Machine language program file

A file other than an ASCII file has the 16-byte header at the beginning of the file. The header is structured as shown in the following figure, but the details are different depending on the type of file.



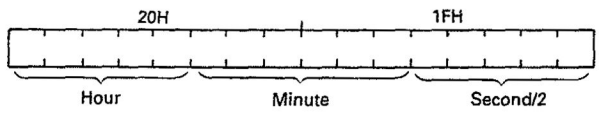
(2) File management in BASIC

In BASIC, files are managed by the 57-byte file control block (FCB) and the subsequent 256-byte file buffer. This 313 (=57+256) byte area is prepared as many as specified by MAXFILES command of BASIC.

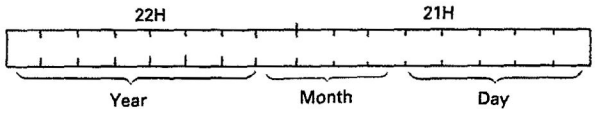


1) File number
   File number specified in an OPEN statement
2) Device name
   Name of the device opened. If the device name is less than four characters long, the space is filled with zero. When file IOCS routines (described in section 3.3.2) are called, a device name needs to be set here.
3) Mode
   File mode specified when the file is opened (input mode, output mode, append mode)
4) File buffer pointer
   Length (bytes) of the effective data in the file buffer
5) File buffer read pointer
   Address of the location in the file buffer from which data are read next when the file is opened in the input mode (The address is specified by the offset from the beginning of the file buffer)
6) Status
   While the file is opened in the input mode, if EOF is detected upon reading data from the file buffer, this byte is set to "1".
7) File name
   File name of the file opened
8) Extension
   Extension of the file opened. If no extension is given, this area is filled with space.

9) Attribute
   Attribute of the file
10) Time
   Time when the file is created



11) Date
   Date when the file is created



   Note: The year is expressed as the offset from 1980.
12) First cluster number
   Cluster number of the first one among the clusters allocated for the file
13) File size
   Size (in bytes) of the file
14) Current record, Current block (128 records/block)
   Specifies the record to be accessed next.
   The current record is set to 00H when OPEN or CREATE routine is called. Then, the current record is incremented by one each time SEQUENTIAL RD or SEQUENTIAL WR routine is called. (Current record: 00H to 7FH)
15) File buffer
   File buffer used for BASIC

3.3.2 IOCS Routines for Files

The following IOCS routines read, write or search for files. Call these routines as follows:

(1) set appropriate parameters in FCB and registers,
(2) set the IOCS number of the desired routine in C register, and
(3) call 01DEH.

When control returns from the called routine, the contents of all registers are destroyed, except that the following information is stored in A register.

A = 00H : Normal termination
= Error code : When a error (errors) has occurred
The individual bits of the the error code given in A register designate the following errors.

Bit 7: The specified device does not exist.
Bit 6: The specified device could not accessed properly.
Bit 5: No media is set in the specified device.
Bit 4: This IOCS routine is not supported by I/O.
Bit 3: Other
Bit 2: There is no data left to be read.
Bit 1: The media does not have free space for writing.
Bit 0: The file could not found or the directory area has no free space.

Routine name	Function	IOCS NO.
OPEN FILE	Open a file.	0FH
CLOSE FILE	Close a file.	10H
SEARCH FIRST	Search for the first file.	11H
SEARCH NEXT	Search for the next file.	12H
DELETE FILE	Delete a file.	13H
SEQUENTIAL RD	Read data sequentially from a file.	14H
SEQUENTIAL WR	Write data sequentially to a file.	15H
CREATE FILE	Create a file.	16H
RENAME FILE	Rename a file.	17H
SET DMA	Set a transfer address.	1AH
GET ALLOC	Get the file device information such as the number of empty clusters.	1BH
SET ATTRB	Set file attributes.	1EH

OPEN FILE

IOCS Number 0FH

Function Open the file specified by the drive name, file name and extension given in FCB.

Parameter DE = Starting address of FCB of the file to be opened

CLOSE FILE

IOCS Number 10H

Function Close the file specified by the drive name, file name and extension given in FCB.

Parameter DE = Starting address of FCB of the file to be closed

SEARCH FIRST

IOCS Number 11H

Function Search for the directory of the file having the file name given in FCB, and transfer the contents of the directory to the memory locations specified by the disk transfer address (specified by SETDMA routine).

Parameter DE = Starting address of FCB

Remarks If the device name is COM or CAS, the routine results in an error. For specification of a file name, the wildcards "?" and "*" can be used.

SEARCH NEXT

IOCS Number 12H

Function Search for the directory for the next file of among the files other than the one that was found by the previous SEARCH FIRST routine, and do the same thing as SEARCH FIRST routine.

DELETE FILE

IOCS Number 13H

Function Delete the file specified by FCB from the directory, and release the data area and directory entry used for that file.

Paramete DE = Starting address of FCB

Remarks If the device name is COM or CAS, the routine results in an error. For specification of a file name, the wildcards "?" and "*" can be used.

SEQUENTIAL RD

IOCS Number 14H

Function Read one record (256 bytes) of data from the opened file into the memory locations specified by the disk transfer address (specified by SETDMA routine). The subsequent records are transferred record by record each time the routine is called.

Parameter DE = Starting address of FCB

SEQUENTIAL WR

IOCS Number 15H

Function Write one record (256 bytes) of data from the memory locations specified by the disk transfer address (specified by SETDMA routine) into the opened file. The subsequent data in the memory are written record by record into the file each time the routine is called.

Paramete DE = Starting address of FCB

CREATE FILE

IOCS Number 16H

Function Create a file specified by the device name, file name and extension given in FCB. If a file with the same name already exists, the existing file is deleted, then the new file is created.

Parameter DE = Starting address of FCB

Remarks When the routine is terminated normally, the current time and date are written in the time and the date areas of FCB, and "00H" is written in the file buffer pointer, first cluster number, file size, current record, and current block.

RENAME FILE

IOCS Number 17H

Function Change the file name (and the extension) of the file specified by the contents of +09H to +13H in FCB (addresses: DE+09H to DE+13H) to the new file name (and the extension) given in +29H to +33H (addresses: DE+29H to DE+33H).

Parameter DE = Starting address of FCB

Remarks If the device name is COM or CAS or if the new file name (and the extension) is the same as that of an existing file, the routine results in an error.

SETDMA

IOCS Number 1AH

Function Set a disk transfer address.

Parameter DE = Disk transfer address

GET ALLOC

IOCS Number 1BH

Function Get the file device information of the device specified by FCB specified by DE register, as follows:
E = Number of sectors per cluster
BC = Sector size
HL = Number of empty clusters

Parameter DE = Starting address of FCB

Remarks If the device name is COM or CAS, the routine results in an error.

SET ATTRB

IOCS Number 1EH

Function Set attributes to the file specified by FCB specified by DE register. The attributes to be set must be prepared in the +14H location of FCB (address: DE+14H).

Parameter DE = Starting address of FCB

3.3.3 Structure of Memory File

(1) Physical structure

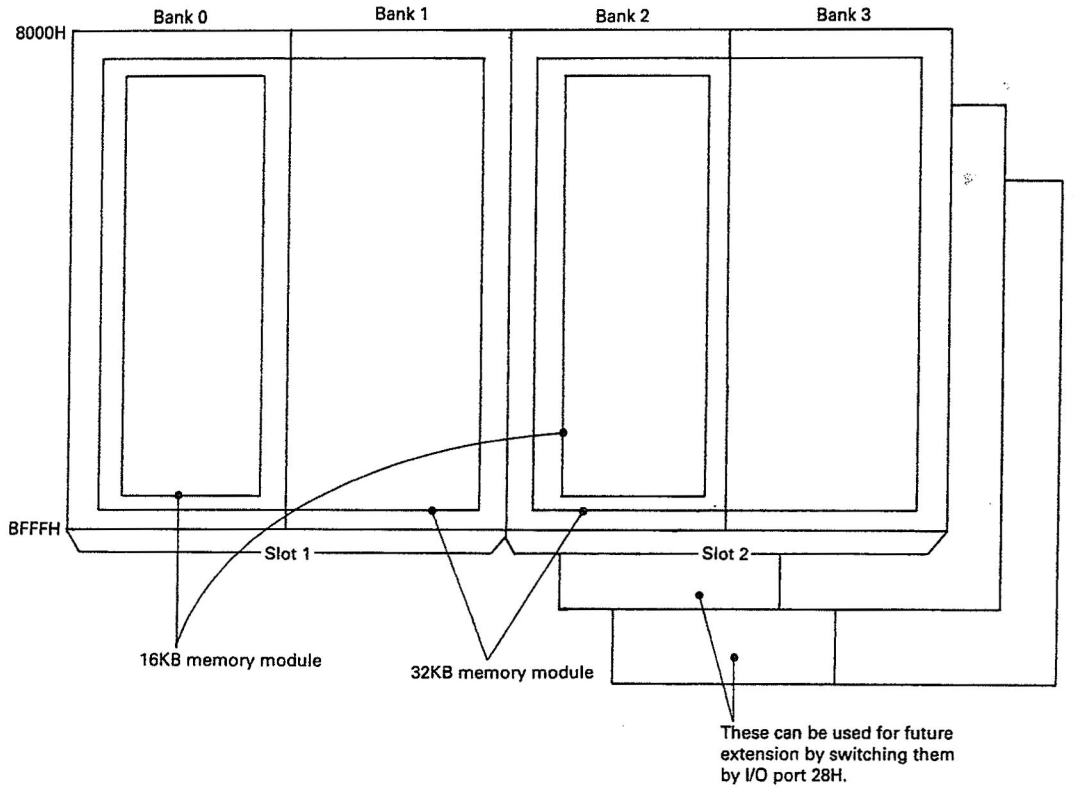
A memory file consists of 16KB segments. The following memory modules can be used as a memory file.

CE-161 : 16 KB

CE-1600M, CE-1620M: 32 KB

(CE-1620M is a 32KB EPROM module and can be used in the same way as CE-1600M except that CE-1620M is a ROM module.)

These memory modules for memory file are allocated in slots 1 and 2 as follows.



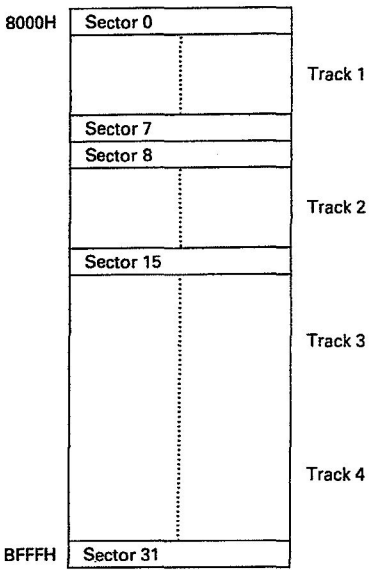
0 : Section 1

1 : Section 2

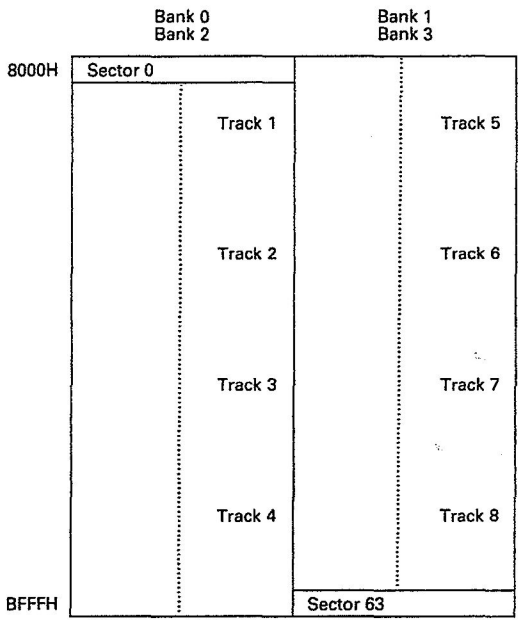
ld A, &01 : out (&28), A gives Section 2.

Each module is divided into 4K-byte units called tracks, and each track is divided into 512-byte units called sectors.
The following shows the track/sector structure of each memory module.

(1) CE-161



(2) CE-1600M/CE-1620M

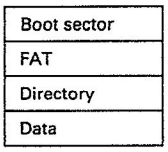


A memory file is managed using the concept of the track and sector, so that it can be handled in the same logical method as the 2.5-inch floppy disk. As described in the figure above, all sectors are managed with the serial number called the logical sector number.

(2) Logical structure

The logical sectors are classified into the following four areas and allocated as described in the figure below.

Logical sectors
- Boot sector
- File allocation table (FAT)
- Directory
- Data



(a) Boot sector

If a boot program loader written in a certain format is written in the boot sector, the contents of this sector are executed either after being loaded in the program memory or directly without the loading when the PC-1600 is powered on.
The following shows the format of the boot sector.

	Contents
00H	55H (File module header ID code)
01H	80H
02H	Check sum
03H	Specifies whether or not to boot. If the content is either C3H or 1BH, the program is booted.
04H	(Low byte) Jump address
05H	(High byte)
06H	00H
08H	Media ID (See the table below.)
09H	Sector size (Bytes/sector) (Low byte)
0AH	(See the table below.) (High byte)
0BH	(sector size / 32) - 1 (See the table below.)
0CH	DIRSFT (See the table below.)
0DH	(number of sectors per cluster) - 1 (See the table below.)
0EH	CLSSFT (See the table below.)
0FH	First logical sector number (Low byte)
10H	of FAT area (See the table below.) (High byte)
11H	FATCNT (See the table below.)
12H	MAXDIR (See the table below.)
13H	First logical sector number (Low byte)
14H	of data area (See the table below.) (High byte)
15H	MAXCLS (Low byte)
16H	(See the table below.) (High byte)
17H	FATSIZ (See the table below.)
18H	First logical sector number (Low byte)
19H	of directory area (See the table below.) (High byte)
1AH	Number of sectors per track (See the table below.)
1BH	00H
1CH	00H
1DH	Starting address for loading (Low byte)
1EH	the boot program (High byte)
1FH	00H = The boot program is loaded in memory and executed. FFH = The boot program is directly executed without being loaded.
20H ~ FFH	Boot loader

Slot 1		Slot 2		Memory module size (KB)	Media ID	Sector size	Sector size/32-1	DIRSFT	Number of sectors per cluster - 1	CLSSFT	First logical sector No. of FAT area	FATCNT	MAXDIR	First logical sector No. of data area	MAXCLS	FATSIZ	First logical sector No. of directory area	Number of sectors per track
ROM module	RAM module	ROM module	RAM module															
O	O	O	O	16(KB)	F0H	0200H	0FH	04H	00H	01H	0001	1	20H	0004H	001DH	01H	0002H	08H
O	O	O	O	32	F1H								30H	0005H	003CH			
		O	O	64	F2H							2		0006H	007BH		0003H	
		O	O	96	F3H								60H	0007H	00B8H			
		O	O	128	F4H								80H	000BH	00F6H			
		O	O	160	F5H				01H	02H			90H	000CH	009BH			
		O	O	192	F6H										00BBH			
		O	O	224	F7H								FOH	0012H	00D8H			
		O	O	256	F8H								FEH	0014H	00F7H			
		O		320	F9H				03H	03H			DOH	0010H	009DH			
		O		384	FAH										00BDH			
		O		512	FBH								FEH	0014H	00FCH			
		O		640	FCH				07H	04H			DOH	0010H	009FH			
		O		768	FDH										00BFH			
		O		896	FEH								FEH	0018H	00DEH			
		O		1024	FFH										00FEH			

(b) FAT

In the file management, the data area is managed by being segmented into units called clusters. Each cluster of the entire data area is managed by one byte of management data, and the set of these data is called FAT (File Allocation Table). The data area is managed in up to 254 clusters.

In the first byte of the FAT area is written the code that identifies the format of the memory file. This code has a value corresponding to the capacity of the module as described in the table above.

In the second and the following bytes of the FAT area are written one byte of cluster information regarding each cluster (from cluster 1 to cluster 254). The cluster information is expressed as follows.

00H : This means that the cluster is not used.

01H to FEH : This means that the cluster is used and the value designates the cluster No. of the next cluster that should come after the current cluster.

FFH : This means that it is the last cluster of the file.

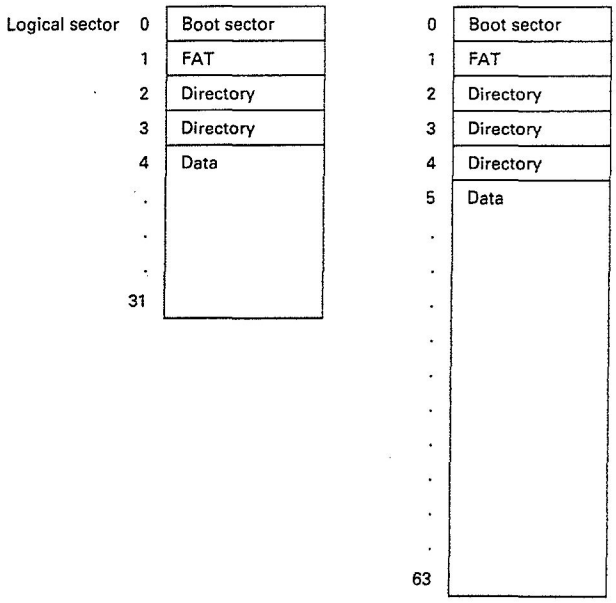
(c) Directory

See section 3.8.3 Floppy Disk Specifications, item (4).

(3) Logical format of module

The format is determined according to the capacity of each module as follows.

	CE-161	CE-1600M, 1620M
No. of tracks	4	8
Sectors/track	8	8
Bytes/sector	512	512
Sectors/1 FAT	1	1
No. of FATs	1	1
Sectors/cluster	1	1
No. of files	32(28) For the data area of 28 clusters only	48



3.4 INTERRUPT HANDLING

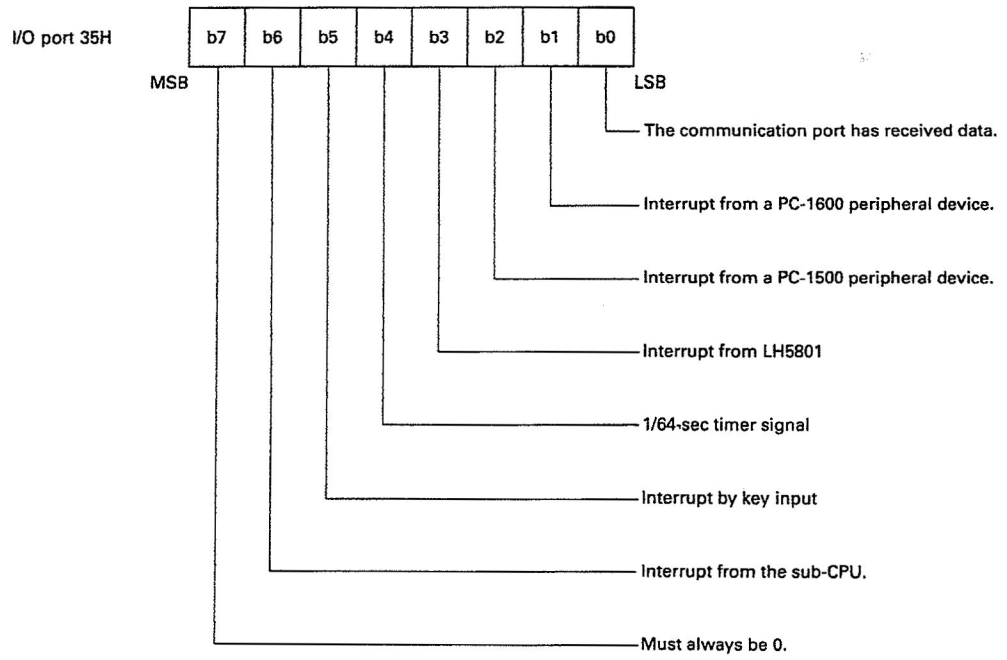
3.4.1 Interrupt Handling

(1) Interrupt cause

The PC-1600 processes many kinds of operations by using the interrupt handling function of SC-7852 (Z80). Z80 accepts the following interrupt causes.

(a) The communication port has received data.
(b) Interrupt request from a PC-1600 peripheral device.
(c) Interrupt request from a PC-1500 peripheral device.
(d) 1/64-sec timer interrupt
(e) Interrupt request from the sub-CPU.

The interrupt for these interrupt causes may be enabled or disabled individually by changing the settings of the individual bits of I/O port of 35H: when the bit is set to 1, the corresponding interrupt is enabled; when set to 0, it is disabled.



* The I/O port 35H can be both read and written.

The interrupt request state can be known by reading the contents of the I/O port 32H. The meaning of the individual bits of 32H port is the same as that of 35H port. If the bit is 1, this means that the interrupt request has been issued. If the bit is 0, this means that the interrupt request has not been issued.

(2) Interrupt handling

The interrupt requests are handled as follows.

(a) The communication port has received data.
Store the received data into the buffer, and send XON or XOFF signal if necessary.

(b) Interrupt request from a PC-1600 peripheral device.
Perform paper feed to CE-1600P, etc.

(c) Interrupt request from a PC-1500 peripheral device.
Perform paper feed to CE-150, etc.

(d) 1/64-sec timer interrupt
- Scan the keys.
- Blink the cursor.

(e) Interrupt request from the sub-CPU.
- 0.5-sec timer
  Low battery check; Analog input interrupt check; CI signal interrupt check; Auto power-off processing; RS-232C time-out
- Wakeup timer
- Alarm1 timer
- Alarm2 timer

(3) Interrupt mode

The PC-1600 handles interruption by using the mode-2 interrupt of Z80. The interrupt vector pointing the interrupt routine entry point is stored in the location of the PC-1600 system reset routine entry address minus 2, in the order of the low and the high bytes. The location of this vector is specified by I register (high byte) and 39H port (low byte). The location of the vector can be known from the entry point address of the system reset routine which is stored in 0003H and 0004H in the order of the low and the high bytes.

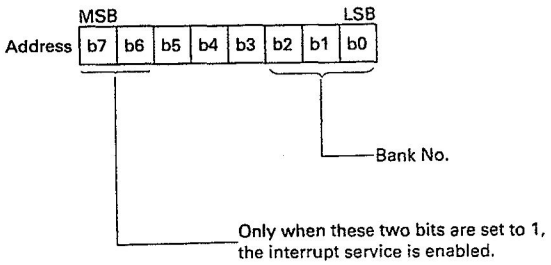
(4) Interrupts available to the user

(a) 1/64-sec timer
(b) 0.5-sec timer
(c) Alarm1 timer
(d) Alarm2 timer
(e) Wakeup timer

For these interrupts, the PC-1600 has a function to provide an interrupt service to a user-defined interrupt handling routine. There is a work area to store the entry bank number and address of a user routine which receives the interrupt service for a particular interrupt. By setting a bank number and an address in this work area, the user can define a routine call from the interrupt routine to the desired user routine.

The work area has the following structure.

- Work that specifies the bank number



Address

1/64-sec timer F0BCH

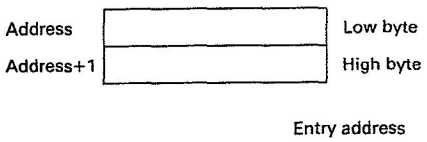
0.5-sec timer F0BFH

Alarm1 timer F0C2H

Alarm2 timer F0C5H

Wakeup timer F0C8H

- Work that specifies the address



Address

1/64-sec timer F0BDH

0.5-sec timer F0C0H

Alarm1 timer F0C3H

Alarm2 timer F0C6H

Wakeup timer F0C9H

Besides the setting of the bank number and the address, for the 1/64-sec timer and the 0.5-sec timer, the most significant bit of the following work must be set to 1.

1/64-sec timer: F0B7H

0.5-sec timer: F0B5H

3.4.2 Work Area used for Interrupt handling

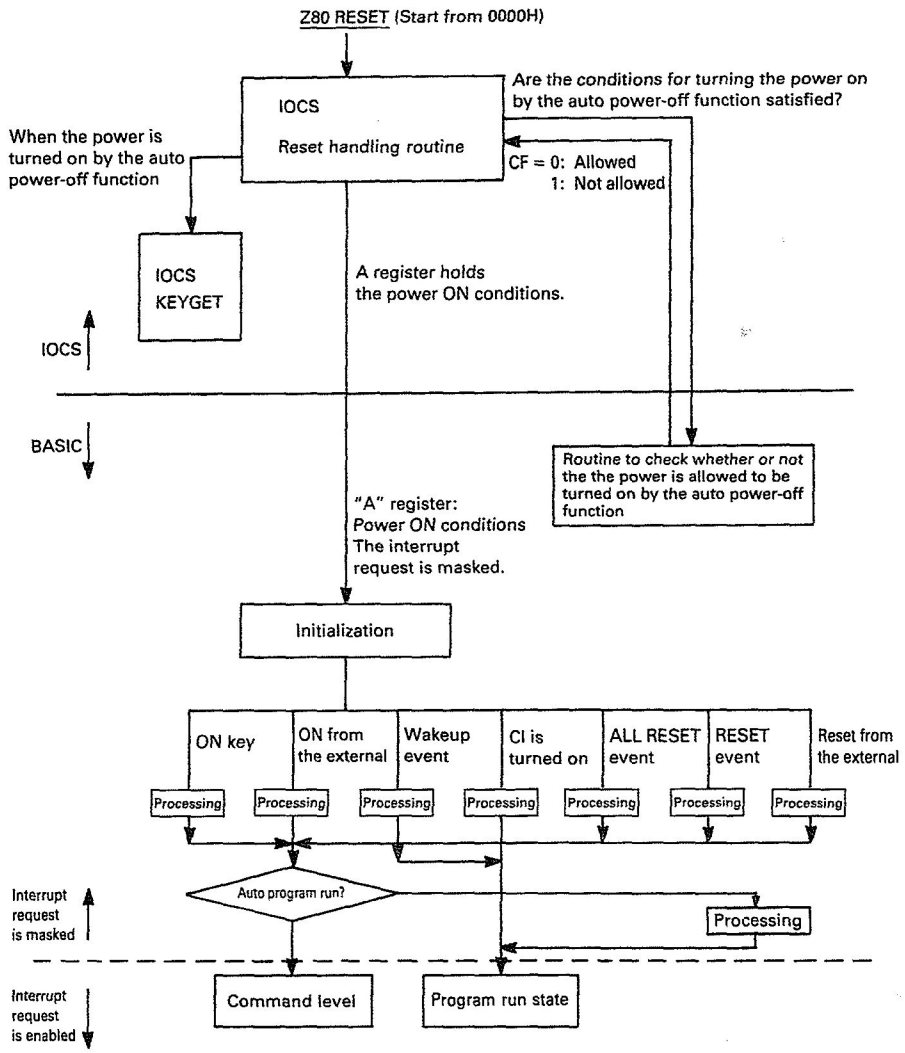
The PC-1600 uses the following work area for the interrupt handling, in addition to the work area described in section 3.4.1 Interrupt Handling, item (4) Interrupts available to the user.

Address	Contents
F05EH	Work for interrupt handling regarding display Bit 0: Cursor blinking state. If bit 0 is 1, this means the cursor is in the ON state during the blinking. Bit 1: A flag to disable the access to the LCD driver during the interrupt handling routine. If this flag is 1, no display is allowed in the interrupt handling routine.
F068H	Cursor blinking counter
F069H ~ F078H	Locations to which the dot pattern at the cursor position is saved
F07AH	Work for interrupt handling regarding key input Bit 0: Key bounce flag This flag is set to 1 during the key bounce processing Bit 2: KEYGET IOCS routine execution flag This flag is set to 1 during execution of KEYGET routine. This is used for the auto power-off processing. Bit 3: A flag to disable the access to the key port during the interrupt handling routine. If this flag is 1, no access to the key port is allowed in the interrupt handling routine. Bit 4: A flag to request a compulsive return from KEYGET IOCS routine. If this flag is set to 1, control is forced to return from KEYGET routine.

3.5 SYSTEM START-UP

3.5.1 Processing at Power On

The following diagram outlines the processing when the PC-1600 is powered on.



3.5.2 Execution of Boot Program

The PC-1600 can start a program other than the BASIC interpreter (a boot program) at power-on time. The PC-1600 searches for a boot program in the following order:

(1) a boot program stored in a file device
(2) a program stored in the system software module
(3) a program pointed by the application entry pointer

If a boot program exists in a file device such as a memory file or a 2.5-inch floppy disk, the PC-1600 loads it into memory and execute it. (A boot program must be written in the format particular to that file device.)

If a boot program does not exist in any file devices, then the PC-1600 searches for the system software module. If the system software module exists, the PC-1600 jumps to the entry address of the program written in that module.

If the system software module does not exist, then the PC-1600 jumps to the memory location specified by the application entry pointer. The PC-1600 BASIC interpreter is started in this method.

(1) Execution of boot program

(a) Execution of the boot program in a file device

A device that can handle files has a function to execute a boot program. The format of the boot program differs depending on the device in which the boot program is stored. See the explanation of a particular device. When the PC-1600 is powered on, the PC-1600 checks whether or not there is a boot program in the file devices. If there is a file device that has a boot program, the PC-1600 makes the device boot the program: the PC-1600 loads the boot program into memory and jumps to the start address (written in that program).

(b) Execution of the program in the system software module

If the system software module is installed in a slot, the PC-1600 jumps to the entry address written in the header of that module. See section 3.13.3 Structure of Memory Module for the *structure of the header.*

(c) Execution of the program specified by the application entry pointer

At the end of the system start-up processing, the PC-1600 jumps from the IOCS to the entry address (and bank) of the application software. This action is performed if a boot program does not exist in the system software module or in the file devices. The PC-1600 BASIC interpreter is started in this method: the entry address and bank of the BASIC interpreter are set in the particular work area when the PC-1600 is reset or all-reset.

■ When control enters this entry point, the system has the following state:

1) The contents of the registers and flags other than the A, I and SP registers and the IRQ mask flag are uncertain.
2) The SP register is in the reset state.
3) The I register and the interrupt mode (mode 2) are in the set state, and the IRQ mask flag of Z-80 is in the DI (0) state.
4) One of the eight bits of the A register is set to 1, which indicates how the system was started up.

Bit 0: System started up by an ALL RESET event.
Bit 1: System started up by a RESET event.
Bit 2: System started up by a RESET event from the external bus.
Bit 3: Be always 0.
Bit 4: System started up by a power-on event by the ON key.
Bit 5: System started up by a power-on event from the external bus.
Bit 6: System started up by a power-on event by the wakeup function.
Bit 7: System started up by a power-on event by the CI signal entry of SIO.

5) The I/O ports are in the reset state.
6) The LCD display is cleared, but the status line symbols are preserved.

■ Application entry pointer

- F0DCH: Bank number where the application entry point exists (0 to 7)
- F0DDH ~ F0DEH : Address of the application entry point (in the order of the low and the high bytes)

■ The IOCS checks, for the application software such as the BASIC interpreter, whether it is allowed or not to turn on the power from the auto power-off state. Because of this, the application software must have a subroutine to check this and return the result in the carry flag (CF).

CF = 0: It is allowed to turn on the power from the auto power-off state.
CF = 1: It is not allowed to turn on the power from the auto power-off state.

Based on the result of this subroutine, the IOCS determines whether or not to turn on the power from the auto power-off state.

In the initialization routine provided in the application software, the entry address and bank of that subroutine must be set in the auto power-off ON check routine pointer.

- Auto power-off ON check routine pointer

  F0CBH: Bank number where the auto power-off ON check routine entry point exists (0 to 7)

  F0CCH ~ F0CDH : Address of the entry point (in the order of the low and the high bytes)

3.6 RS-232C AND SIO

3.6.1 Handling RS-232C and SIO in BASIC

(1) General

The PC-1600 has two communications connectors (for RS-232C and for SIO) while it has only one serial/parallel conversion LSI. Because of this, RS-232C and SIO cannot be used at the same time. They can be switched by SETDEV command of BASIC:

SETDEV "COM1:" ..... RS-232C is selected.
SETDEV "COM2:" ..... SIO is selected.

Selection of RS-232C or SIO can also be done in OPEN, SAVE, LOAD, BSAVE, or BLOAD command, by explicitly specifying "COM1:" or "COM2:" in the statement of these commands.
The default communication device at the power-on time is SIO (that is, the same state as when SETDEV "COM2:" statement is executed.)

(2) Communication control of RS-232C

(Transmission control)
With the default setting, the CS signal of RS-232C is used for transmission control. When a send command is executed in BASIC, the PC-1600 does not start the transmission action immediately. It waits until the CS signal of RS-232C goes high, then starts the transmission action. This transmission control can be changed by SNDSTAT command:

SNDSTAT "COM1:",63 (The transmission starts regardless of any control signals.)
SNDSTAT "COM1:",55 (The CD signal is used for transmission control.)
SNDSTAT "COM1:",47 (The DR signal is used for transmission control.)

With the default setting, the PC-1600 would suspend the transmission action for ever if the transmission control signal did not go high. This period of time for which the PC-1600 waits for the transmission to be enabled, or the timeout value, can also be changed. If a finite time is specified as the timeout value, when the transmission is not enabled for the specified time, the transmission session results in an error.
SNDSTAT "COM1:",,20 (The timeout value is set to be 10 seconds.)
The timeout value to be specified in a SNDSTAT statement is in units of 0.5 second. When the timeout value is set to be 0, the PC-1600 keeps waiting until the transmission is enabled.

(Reception control)
With the default setting, the PC-1600 starts the reception action regardless of any control signals of RS-232C. The following RS-232C signals can be used for reception control.

RCVSTAT "COM1:",59 (The data reception action is enabled when the CS signal goes high.)
RCVSTAT "COM1:",55 (The data reception action is enabled when the CD signal goes high.)
RCVSTAT "COM1:",47 (The data reception action is enabled when the DR signal goes high.)

If data are sent to the PC-1600 while the specified control signal is low, the data are aborted.
Like the transmission control, a timeout value can also be set for the reception control. When a finite time is set as the timeout value, if no data are sent to the PC-1600 for that period of time, the reception session results in an error.

RCVSTAT "COM1:",,20 (If no data are received for more than 10 seconds, the reception session results in an error.)

The timeout value to be specified in a RCVSTAT statement is in units of 0.5 second. When the timeout value is set to be 0, the PC-1600 keeps waiting until data are sent to the PC-1600.

(3) Auto handshake mode

While "COM1:" is selected, if an OUTSTAT "COM1:" statement is executed, the PC-1600 is set in the auto handshake mode. In the auto handshake mode, when the empty space in the receive

buffer becomes 8 bytes or less, the RS signal goes low. When the received data in the receive buffer are read into the system and the remaining data in the received buffer becomes 8 bytes or less, the RS signal goes high.
While the PC-1600 is in the auto handshake mode, if an INPUT, LPRINT or LLIST command is executed to RS-232C or SIO, the RS and the ER signals go high.

(4) Carriage-return (CR) code

When a LLIST, LFILES or LPRINT command is executed to RS-232C or SIO, every CR code included in the transmission data can be converted to CR (no conversion), LF, or CR + LF by a PCONSOLE "COMn:" statement. However, when an OPEN "COMn:" statement is executed, a CR code in a PRINT# output or in an ASCII save operation is always converted to CR + LF codes.

(5) Communications by SETDEV command

By using SETDEV command, INPUT, LPRINT and LFILES commands can be executed to a communication device.

SETDEV "COMn:",KI (The specified communication device is used as the input device for an INPUT command.)

SETDEV "COMn:",PO (The specified communication device is used as the output device for an LPRINT, LLIST or LFILES command.)

SETDEV "COMn:",KI,PO (The specified communication device is used as the input device for an INPUT command and as the output device for an LPRINT, LLIST or LFILES command.)

SETDEV "COMn:" (The input and the output devices for INPUT, LPRINT, LLIST and LFILES commands are reset to the default standard devices: the keyboard and the printer.)

Communication procedure

1. Set a baud rate with SETCOM command.
2. Specify a communication device and KI and/or PO with SETDEV command.
3. Execute INPUT or LPRINT command.
4. If necessary, release KI and/or PO with SETDEV command.

(6) Communications by OPEN command

When you input and output data to and from RS-232C or SIO with an OPEN "COMn:" statement and an INPUT# or PRINT# command, the data input/output action is performed in units of 256 bytes. If the last block of data to be transmitted is less than 256 bytes, the block of data is transmitted when the communication device is closed.

Communication procedure

1. Set a baud rate with SETCOM command.
2. Open a communication device with OPEN command.
3. Execute INPUT# or PRINT# command.
4. Close the communication device with CLOSE command.

(7) ON COMn GOSUB command

When the interrupt from a communication device has been declared by ON COMn GOSUB command, reception of data at the specified communication device (RS-232C or SIO) causes an interrupt. This interrupt is issued even when the communication device receives a control code such as an XON, XOFF, SIN or SOUT code. If the subroutine to which control is jumped by occurrence of an interrupt needs to know whether there is any data to be read in the receive buffer, this may be checked by using RXD$ function at the beginning of the subroutine.

3.6.2 Data Format of Communications

See section 5.7 "Precautions for Use of Serial Ports (RS-232C and SIO)".

3.6.3 IOCS Routines for RS-232C and SIO

The IOCS routines described below must be called in the following procedure.

(1) Set the IOCS number of the routine to be called in the C register.

(2) If a channel number needs to be specified, set it in the D register.
Value to be set in D register (channel no.) ..... Channel name
00H ..... "COM:"
01H ..... "COM1:"
02H ..... "COM2:"

(3) Execute the following command:
CALL 01D8H
The table below lists the names, functions and IOCS Nos. of the IOCS routines related to the serial ports.

IOCS Name	Function	IOCS NO.
CWCOM	Set communication parameters.	01H
CRCOM	Read the current communication parameters.	02H
CSNDA	Transmit one byte of data.	03H
CRCVA	Receive one byte of data (if there is no data to read, wait until data are sent in.)	04H
CRCV1	Receive one byte of data (do not wait even if there is no data to read.)	07H
CSETHS	Set RS and ER signals to high in the auto handshake mode.	0EH
CRESHS	Set RS and ER signals to low in the auto handshake mode.	0FH
CWOUTS	Set the state of the outgoing control signals (RS and ER).	10H
CRCTRL	Read the state of the control signals.	11H
CWDEV	Select a channel and set the input and output device selection parameters for that channel.	12H
CRDEV	Read the currently selected channel number and the input and output device selection parameters set by CWDEV routine.	13H
CESND	Enable the transmission only when the specified incoming control signal (or signals) is high.	14H
CERCV	Enable the reception only when the specified incoming control signal (or signals) is low.	15H
CSBRK	Send the specified number of break characters	16H
CSRCVB	Reserve a receive buffer in memory.	17H
CCLRSB	Clear the work area for transmission.	1BH
CCLRRB	Clear the receive buffer.	1CH

CWCOM

IOCS Number 01H

Function Set communication parameters.

Parameter HL = Starting address of the memory locations in which communication parameters are stored
D = Channel number

Affected Register AF, AF'

Remarks Communication parameters stored in memory

Address	Contents
HL	Baud rate data low byte
HL + 1	Baud rate data high byte

Value of 76800/baud rate

HL + 2 Parameter data

MSB	b7	b6	b5	b4	b3	b2	b1	b0	LSB

- b1 = 0
- b0 — Stop bit length {0: One stop bit, 1: Two stop bits}

b2	0	1	0	1
b3	0	0	1	1
Character length	5	6	7	8

- b4 — Parity check {0: Disable, 1: Enable}
- b5 — Parity {0: Odd, 1: Even}
- b6 — XON/XOFF control {0: Disable, 1: Enable}
- b7 — SIN/SOUT control {0: Disable, 1: Enable}

CRCOM

IOCS Number 02H

Function Read into DE register the starting address of the memory locations in which the communication parameters are stored.

Parameter D = Channel number

Return DE = Starting address

Affected Register AF, AF', DE

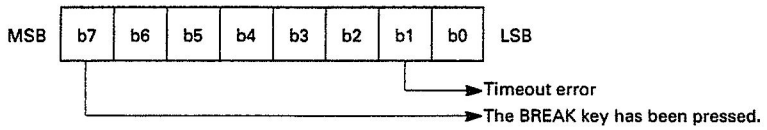
CSNDA

IOCS Number 03H

Function Transmit one byte of data in A register to the specified channel.

Parameter A = Data to be transmitted

Return If an error occurs, the error data is written in A register with the error bit set to "1".



Affected Register AF, AF'

CRCVA

IOCS Number 04H

Function Read one byte of data from the receive buffer into A register. If there is no data to be read from the receive buffer, the routine waits until data come in.

Parameter none

Return A = Received data
CF = 1 if an error has occurred.

Affected Register AF, AF'

CRCV1

IOCS Number 07H

Function Read one byte of data from the receive buffer into A register. If there is no data to be read from the receive buffer, ZF is set to "1" and the routine returns.

Parameter none

Return A = Received data
ZF = 1 if there is no data in the receive buffer.
CF = 1 if an error has occurred.

Affected Register AF, AF'

CSETHS

IOCS Number 0EH

Function Set RS and ER signals to high in the auto handshake mode.

Parameter none

Return none

Affected Register AF, AF'

CRESHS

IOCS Number 0FH

Function Set RS and ER signals to low in the auto handshake mode.

Parameter none

Return none

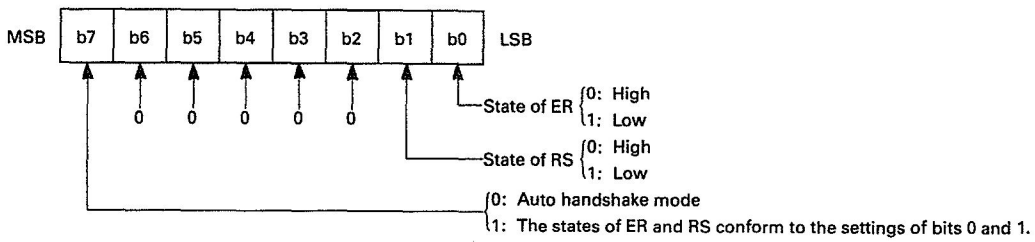
Affected Register AF, AF'

CWOUTS

IOCS Number 10H

Function Set the state of the outgoing control signals (RS and ER) of the specified serial port.

Parameter D = Channel number (effective only for RS-232C)
E = State of RS and ER control signals



Return none

Affected Register AF, AF', DE

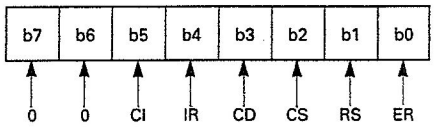
CRCTRL

IOCS Number 11H

Function Read the state of the control signals of the specified channel.

Parameter D = Channel number (effective only for RS-232C)

Return A = State of the control signals (expressed in the same format as for INSTAT function of BASIC)



The data bits in A register correspond to the control signals as described above. When a bit is "0", this means that the corresponding control signal is high. When a bit is "1", the signal is low.

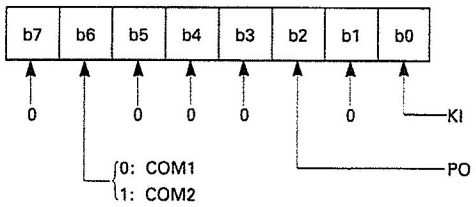
Affected Register AF, AF'

CWDEV

IOCS Number 12H

Function Select a channel and set the input and output device selection parameters for that channel.

Parameter Prepare the following data in A register.



Return none

Affected Register AF, AF'

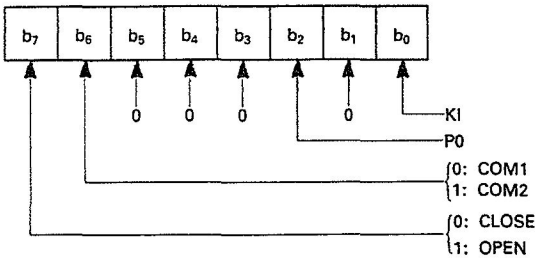
CRDEV

IOCS Number 13H

Function Read the currently selected channel number and the input and output device selection parameters set by CWDEV routine.

Parameter none

Return The following data is returned into A register.

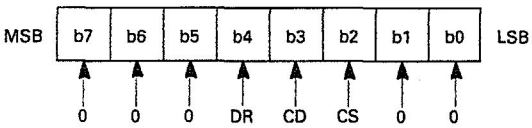


CESND

IOCS Number 14H

Function Enable the transmission only when the specified incoming control signals (DR, CD and/or CS) of the specified channel are high.

Parameter D = Channel number (effective only for RS-232C)
E = Specification of the control signals to be used for transmission control



(Specify the desired control signals by setting the corresponding bits to "0".)

For instance, if this routine is called with E = 08H, the transmission is enabled when DR and CS are both high.

B = Timeout value (0 to 255) (Unit: 0.5 sec.)
If 0 is specified, the waiting time is set as infinite.

Return none

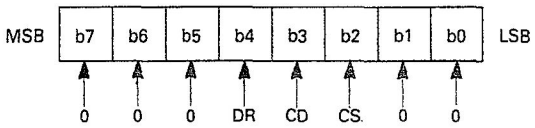
Affected Register AF, AF'

CERCV

IOCS Number 15H

Function Enable the reception only when the specified incoming control signals (DR, CD and/or CS) of the specified channel are high.

Parameter D = Channel number (effective only for RS-232C)
E = Specification of the control signals to be used for reception control



(Specify the desired control signals by setting the corresponding bits to "0".)

For instance, if this routine is called with E = 08H, the reception is enabled when DR and CS are both high.

B = Timeout value (0 to 255) (Unit: 0.5 sec.)
If 0 is specified, the waiting time is set as infinite.

Return none

AffectedRegister AF, AF'

CSBRK

IOCS Number 16H

Function Send the specified number of break characters.

Parameter B = Number of break characters

Return A = 00H: Normal termination
01H: The BREAK key was pressed during the execution of this routine.

Affected Register AF, AF', BC

CSRCVB

IOCS Number 17H

Function Reserve a receive buffer in memory.

Parameter HL = Buffer size (0000H, or 0050H to 3FFFH) (Unit: bytes)
If HL = 0000H is specified, a receive buffer of 40 bytes is reserved in *memory.*

Return

A = 00H:	Normal termination
Other than 00H:	An error has occurred.

Affected Register AF, AF'

Remarks When this routine is executed, the transmission and reception buffer and the error flags are cleared.

CCLRSB

IOCS Number 1BH

Function Initialize the work area for transmission.

Parameter none

Return none

Affected Register AF, AF'

CCLRRB

IOCS Number 1CH

Function Clear the reception buffer and the error flags.

Parameter none

Return none

Affected Register AF, AF'

3.7 PRINTER

This section describes the IOCS routines related to the printer. First, the IOCS routines which are executed by calling their entry address are explained, then the IOCS routines which are executed by calling a specific address with the IOCS number set in C register are explained.
When one of these printer IOCS routines is executed, the following two addresses must always be called immediately after the routine.

- Bank 4, Address 4029H
- Bank 4, Address 6777H

These two routines perform the post-processing of printer operation (such as turning off the printer motor power and interrupt control). If they are not executed, the printer motor power may be left ON and the key input from the keyboard may not be accepted.

3.7.1 IOCS Routines for Printer (1)

The IOCS routines described in this section can be executed as follows:

CALL <entry address>

The table below lists the names, entry addresses, and functions of the IOCS routines.

Name	Entry address	Function
PCHEK	Bank 4 4020H	Check whether the printer is ready.
POUT	Bank 4 4023H	Send one byte of character code to the printer.
PKOUT	Bank 4 4026H	Send one byte of character code to the printer.

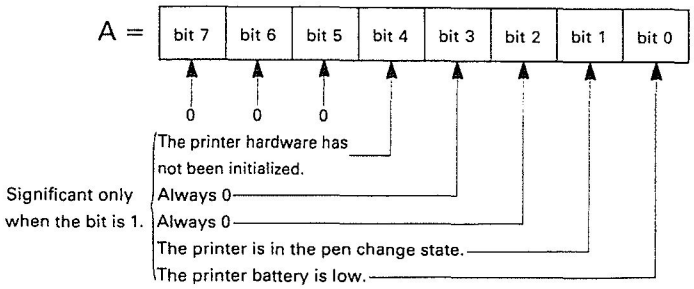
PCHEK

Entry Address Bank 4, 4020H

Function Check whether the printer is ready.

Parameter none

Return A =



Affected Register All registers

POUT

Entry Address Bank 4, 4023H

Function Send one byte of character code to the printer.

Parameter E = Character code

Return
CF = 0: Normal termination
CF = 1: An error has occurred or the BREAK key has been pressed.
A = 00H: The BREAK key has been pressed.
= Other than 00H: Error code (same as that of BASIC)

Affected Register All registers

3.7.2 IOCS Routines for Printer (2)

The IOCS routines described in this section can be executed by calling address 4008H of bank 4 with the IOCS number set in C register. When these routines are executed, the contents of all registers are destroyed.
The table below lists the names, IOCS number, and functions of the IOCS routines. Those routines indicated by Ⓖ in the table are effective only in the graphics mode. The error code is the same as that of BASIC.

Name	Function	IOCS NO.
PINIT	Initialize the printer.	00H
PTEXT	Set the printer in the text mode.	01H
PGRAPH	Set the printer in the graphics mode.	02H
PCSIZE	Set the character size.	03H
PCOLOR	Set the print color.	04H
PWIDTH	Set the line width (characters/line).	06H
PLEFTM	Set the left margin.	07H
PPITCH	Set the character pitch and the line height.	08H
PPAPER	Set the paper type.	09H
PSCRL	Set the printing area on paper in the Y (vertical) direction.	0AH
PEOL	Define the printer action for a CR code (0DH).	0BH
PZONE	Set the print zone length for LPRINT command.	0CH
PPENUP	Lift up or push down the pen.	0DH
PROTATE Ⓖ	Set the direction of the print characters.	0EH

*Those routines indicated by Ⓖ are effective only in the graphics mode.

Name	Function	IOCS NO.
PLTYPE Ⓖ	Set the line type.	0FH
PHOME Ⓖ	Move the pen to the origin with the pen up.	10H
PSORGN Ⓖ	Define the current pen position as the origin.	11H
PAMVUP Ⓖ	Move the pen in the absolute coordinate mode with the pen up.	12H
PRMVUP Ⓖ	Move the pen in the relative coordinate mode with the pen up.	13H
PAMVDN Ⓖ	Move the pen in the absolute coordinate mode with the pen down.	14H
PRMVDN Ⓖ	Move the pen in the relative coordinate mode with the pen down.	15H
PTEST	Perform the printing test.	16H
PTAB	Move the pen to the specified tab position.	17H
ALLOFF	Turn off the printer motor power.	18H
PCUP	Move the pen upward in character units with the pen up.	1AH
PCDOWN	Move the pen downward in character units with the pen up.	1BH
PCLEFT	Move the pen to the left in character units with the pen up.	1CH
PCRIGHT	Move the pen to the right in character units with the pen up.	1DH
PGUP	Move the pen upward in graphics units with the pen up.	1EH
PGDOWN	Move the pen downward in graphics units with the pen up.	1FH
PGLEFT	Move the pen to the left in graphics units with the pen up.	20H
PGRIGHT	Move the pen to the right in graphics units with the pen up.	21H
PCHGPEN	Change the pen.	22H
PRESET	Initialize the work area for the printer IOCS routines to the all-reset state.	28H
PCR	Move the pen to the left end (carriage-return action).	29H
PDIRC	Set the character printing direction.	2AH
PCRLF	Move the pen to the left end of the next line (carriage-return and line-feed action).	2BH
PMYFD	Check how many lines the pen can be moved to the −Y direction.	2CH
PPYFD	Check how many lines the pen can be moved to the +Y direction.	2DH
PBOXA	Draw a box in the absolute coordinate mode.	2FH
PBOXR	Draw a box in the relative coordinate mode.	30H
HARESET	Initialize the printer hardware.	31H

*Those routines indicated by Ⓖ are effective only in the graphics mode.

PINIT

IOCS Number 00H

Function Initialize the printer work area.

Parameter none

Return CF = 1 if an error has occurred.
A = Error code

PTEXT

IOCS Number 01H

Function Set the printer in the text mode.

Parameter none

Return CF = 1 if an error has occurred.
A = Error code

Remarks When this routine is called, the following actions are also performed.
(1) If the pen is not at the left end, the pen is moved to the left end.
(2) CSIZE2 is set.

PGRAPH

IOCS Number 02H

Function Set the printer in the graphics mode.

Parameter none

Return CF = 1 if an error has occurred.
A = Error code

Remarks When this routine is called, the following actions are also performed.
(1) If the pen is not at the left end, the pen is moved to the left end.
(2) CSIZE2 is set.
(3) The printer is set in the same state as when ROTATE 0 is executed.
(4) The pen position when this routine has been completed is set as the origin.

PCSIZE

IOCS Number 03H

Function Set the character size.

Parameter A = Character size (01H to 09H)

Return CF = 1 if an error has occurred.
A = Error code

PCOLOR

IOCS Number 04H

Function Set the print color.

Parameter A = Print color (00H to 03H)

Return CF = 1 if an error has occurred.
A = Error code

PWIDTH

IOCS Number 06H

Function Set the line width (characters/line).

Parameter A = Number of characters per line (10H to FFH)

Return CF = 1 if an error has occurred.
A = Error code

Remarks This routine is effective only in the text mode. The pen must be at the left end before the routine is executed.

PLEFTM

IOCS Number 07H

Function Set the left margin.

Parameter A = Number of characters for the left margin.

Return CF = 1 if an error has occurred.
A = Error code

Remarks This routine is effective only in the text mode. When this routine is executed, the pen is moved to the specified printing position.

PPITCH

IOCS Number 08H

Function Set the character pitch and the line height.

Parameter HL = Starting address of the memory locations in which the character pitch and line height data are stored

Data that need to be stored in memory

Address	Data
HL	Character pitch value (04H to FFH) If the value is 00H, the default character pitch is set.
HL+1	Line height value (04H to FFH) If the value is 00H, the default line height is set.

Return CF = 1 if an error has occurred.
A = Error code

PPAPER

IOCS Number 09H

Function Set the paper type.

Parameter A = 00H: Cut sheet
= Other than 00H: Roll paper

Return CF = 1 if an error has occurred.
A = Error code (If the BREAK key has been pressed, then A = 00H is returned.)

PSCRL

IOCS Number 0AH

Function Set the printing area on paper in the Y (vertical) direction.

Parameter HL = Starting address of the memory locations in which the printing area data are stored

Data that need to be stored in memory

Address	Data
HL HL+1	Value of −Y direction (low byte) Value of −Y direction (high byte) } If the value is 0000H, the default value is used.
HL+2 HL+3	Value of +Y direction (low byte) Value of +Y direction (high byte) } If the value is 0000H, the default value is used.

Return CF = 1 if an error has occurred.
A = Error code

Remarks The default values are:
−Y direction: Cut sheet 30
Roll paper 1354
+Y direction: Cut sheet 999
Roll paper 999

PSCRL

IOCS Number 0BH

Function Define the printer action for a CR code (0DH).

Parameter A = 0: CR action when a code (0DH) is received.
= 1: LF action when a code (0DH) is received.
= 2: CR+LF action when a code (0DH) is received.

Return CF = 1 if an error has occurred.
A = Error code

PZONE

IOCS Number 0CH

Function Set the print zone length to be used when a comma is used as the data delimiter in LPRINT statement.

Parameter A = Print zone length (in characters) (08H to 50H)

Return CF = 1 if an error has occurred.
A = Error code

PPENUP

IOCS Number 0DH

Function Lift up or push down the pen.

Parameter A = FFH: Push down the pen.
= Other than FFH: Lift up the pen.

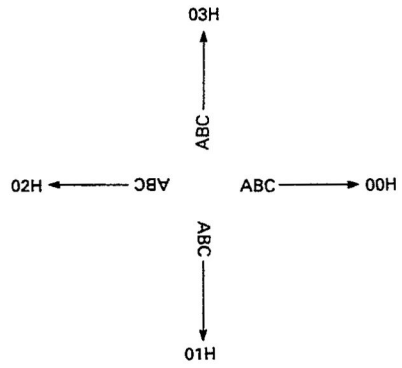
Return CF = 1 if an error has occurred.
A = Error code

PROTATE

IOCS Number 0EH

Function Set the character printing ROTATION used in the graphics mode.

Parameter A = Character printing direction (00H to 03H)



Return CF = 1 if an error has occurred.
A = Error code

PLTYPE

IOCS Number 0FH

Function Set the line type to be used when drawing a line by PAMVDN, PRMVDN or PBOXR routine.

Parameter A = Line type (00H to 09H)

00H	
01H	
02H	
03H	
04H	
05H	
06H	
07H	
08H	
09H	No line is drawn.

PAMVUP

IOCS Number 12H

Function Move the pen in the absolute coordinate mode with the pen up.

Parameter HL = Starting address of the memory locations in which coordinate data are stored

B = Number of coordinate data items stored in the memory locations (Number of bytes/4)

Coordinate data prepared in memory

Address	Contents	Value to be set in B register
HL HL+1 HL+2 HL+3	X0 {Low byte {High byte Y0 {Low byte {High byte	1
HL+4 HL+5 HL+6 HL+7	X1 {Low byte {High byte Y1 {Low byte {High byte	2
. . . .	. . . .	. . . .
HL+20 HL+21 HL+22 HL+23	X5 {Low byte {High byte Y5 {Low byte {High byte	6

Return CF = 1 if an error has occurred.

A = Error code (If the BREAK key has been pressed, then A = 00H is returned.)

Remarks This routine is effective only in the graphics mode. The pen is moved from the start point (X0,Y0) to the coordinate points (X1,Y1), (X2,Y2) ... up to (X5,Y5), one after another.

PRMVUP

IOCS Number 13H

Function Move the pen in the relative coordinate mode with the pen up.

Parameter HL = Starting address of the memory locations in which coordinate data are stored

B = Number of coordinate data items stored in the memory locations (Number of bytes/4)

Coordinate data prepared in memory

Address	Contents	Value to be set in B register
HL HL+1 HL+2 HL+3	X0 {Low byte {High byte Y0 {Low byte {High byte	1
HL+4 HL+5 HL+6 HL+7	X1 {Low byte {High byte Y1 {Low byte {High byte	2
. . . .	. . . .	. . . .
HL+20 HL+21 HL+22 HL+23	X5 {Low byte {High byte Y5 {Low byte {High byte	6

Return

CF = 1 if an error has occurred.
A = Error code (If the BREAK key has been pressed, then A = 00H is returned.)

Remarks

This routine is effective only in the graphics mode. The pen is moved from the start point (X0,Y0) by the relative displacements given as (X1,Y1), (X2,Y2) ... up to (X5,Y5), one after another.

PAMVDN

IOCS Number

14H

Function

Move the pen in the absolute coordinate mode with the pen down.

Parameter

HL = Starting address of the memory locations in which coordinate data are stored
B = Number of coordinate data items stored in the memory locations (Number of bytes/4)

Coordinate data prepared in memory

Address	Contents	Value to be set in B register
HL HL+1 HL+2 HL+3	X0 {Low byte {High byte Y0 {Low byte {High byte	1
HL+4 HL+5 HL+6 HL+7	X1 {Low byte {High byte Y1 {Low byte {High byte	2
. . . .	. . . .	. . . .
HL+20 HL+21 HL+22 HL+23	X5 {Low byte {High byte Y5 {Low byte {High byte	6

Return

CF = 1 if an error has occurred.
A = Error code (If the BREAK key has been pressed, then A = 00H is returned.)

Remarks

This routine is effective only in the graphics mode. The pen is moved from the start point (X0,Y0) to the coordinate points (X1,Y1), (X2,Y2) ... up to (X5,Y5), one after another.

PRMVDN

IOCS Number

15H

Function

Move the pen in the relative coordinate mode with the pen down.

Parameter

HL = Starting address of the memory locations in which coordinate data are stored
B = Number of coordinate data items stored in the memory locations (Number of bytes/4)

Return CF = 0: Normal termination
= 1: An error has occurred.
A = Error code (If the BREAK key has been pressed, then A = 00H is returned.)

PGDOWN

IOCS Number 1FH

Function Move the pen downward in graphics units with the pen up.

Parameter HL = Displacement (Unit: dots) (0 to 2047)

Return CF = 0: Normal termination
= 1: An error has occurred.
A = Error code (If the BREAK key has been pressed, then A = 00H is returned.)

PGLEFT

IOCS Number 20H

Function Move the pen to the left in graphics units with the pen up.

Parameter HL = Displacement (Unit: dots) (0 to 2047)

Return CF = 0: Normal termination
= 1: An error has occurred.
A = Error code (If the BREAK key has been pressed, then A = 00H is returned.)
HL = Number of dots the pen could not move because the pen reached the edge of the printing area

PGRIGHT

IOCS Number 21H

Function Move the pen to the right in graphics units with the pen up.

Parameter HL = Displacement (Unit: dots) (0 to 2047)

Return CF = 0: Normal termination
= 1: An error has occurred.
A = Error code (If the BREAK key has been pressed, then A = 00H is returned.)

PCLEFT

IOCS Number 1CH

Function Move the pen to the left in character units with the pen up. If such a pen movement that will go beyond the printing area is specified, the pen stops at the edge of the printing area.

Parameter A = Displacement (Unit: characters) (00H to FFH)

Return CF = 0: Normal termination
= 1: An error has occurred.
A = Error code (If the BREAK key has been pressed, then A = 00H is returned.)

PCRIGHT

IOCS Number 1DH

Function Move the pen to the right in character units with the pen up. If such a pen movement that will go beyond the printing area is specified, the pen stops at the edge of the printing area.

Parameter A = Displacement (Unit: characters) (00H to FFH)

Return CF = 0: Normal termination
= 1: An error has occurred.
A = Error code (If the BREAK key has been pressed, then A = 00H is returned.)

PGUP

IOCS Number 1EH

Function Move the pen upward in graphics units with the pen up.

Parameter HL = Displacement (Unit: dots) (0 to 2047)

Return CF = 1 if an error has occurred.
A = Error code

Remarks This routine is effective only in the text mode.

ALLOFF

IOCS Number 18H

Function Turn off the printer motor power.

Parameter none

Return none

PCUP

IOCS Number 1AH

Function Move the pen upward in character units with the pen up. If such a pen movement that will go beyond the printing area is specified, the pen stops at the edge of the printing area.

Parameter A = Displacement (Unit: characters) (00H to FFH)

Return CF = 0: Normal termination
= 1: An error has occurred.
A = Error code (If the BREAK key has been pressed, then A = 00H is returned.)

PCDOWN

IOCS Number 1BH

Function Move the pen downward in character units with the pen up. If such a pen movement that will go beyond the printing area is specified, the pen stops at the edge of the printing area.

Parameter A = Displacement (Unit: characters) (00H to FFH)

Coordinate data prepared in memory

Address	Contents	Value to be set in B register
HL HL+1 HL+2 HL+3	X0 {Low byte {High byte Y0 {Low byte {High byte	1
HL+4 HL+5 HL+6 HL+7	X1 {Low byte {High byte Y1 {Low byte {High byte	2
. . . .	. . . .	. . . .
HL+20 HL+21 HL+22 HL+23	X5 {Low byte {High byte Y5 {Low byte {High byte	6

Return CF = 1 if an error has occurred.
A = Error code (If the BREAK key has been pressed, then A = 00H is returned.)

Remarks This routine is effective only in the graphics mode. The pen is moved from the start point (X0,Y0) by the relative displacements given as (X1,Y1), (X2,Y2) ... up to (X5,Y5), one after another.

PTEST

IOCS Number 16H

Function Perform the printing test.

Return CF = 1 if an error has occurred.
A = Error code (If the BREAK key has been pressed, then A = 00H is returned.)

Remarks When this routine is completed, the printer is set in the text mode.

PTAB

IOCS Number 17H

Function Move the pen to the specified column position.

Parameter A = Column number

Return

CF = 0: Normal termination
= 1: An error has occurred.
A = Error code (If the BREAK key has been pressed, then A = 00H is returned.)
HL = Number of dots the pen could not move because the pen reached the edge of the printing area

PCHGPEN

IOCS Number 22H

Function Move the pen to the pen change position, or move the pen back to the position where the pen stayed before the pen change.

Parameter

A = 00H: Move the pen to the pen change position. If the pen is already at the pen change position, move the next color pen to the pen change position.
= 01H: Move the pen back to the original position.

Return

CF = 1 if an error has occurred.
A = Error code

PRESET

IOCS Number 28H

Function Initialize the work area for the printer IOCS routines to the all-reset state.

Parameter none

PCR

IOCS Number 29H

Function Move the pen to the left end (carriage-return action).

Parameter none

Return

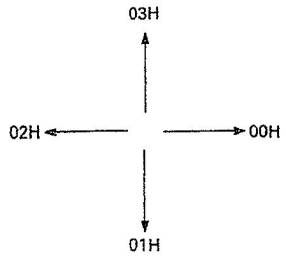
CF = 1 if an error has occurred.
A = Error code

PDIRC

IOCS Number 2AH

Function Set the character printing direction for printing in the graphics mode.

Parameter A = Printing direction (00H to 03H)



Return CF = 1 if an error has occurred.
A = Error code

PCRLF

IOCS Number 2BH

Function Move the pen to the left end of the next line (carriage-return and line-feed action).

Parameter none

Return CF = 1 if an error has occurred.
A = Error code

Remarks This routine causes nothing in the graphics mode.

PMYFD

IOCS Number 2CH

Function Check how many lines the paper can be fed from the current pen position to the −Y (forward) direction.

Parameter none

Return HL = Number of lines the paper can be fed

Remarks This routine is effective only in the text mode.
If roll paper is used, HL = FFFFH is returned.

PPYFD

IOCS Number 2DH

Function Check how many lines the paper can be fed from the current pen position to the +Y (backward) direction.

Parameter none

Return HL = Number of lines the paper can be fed

Remarks This routine is effective only in the text mode.

PBOXA / PBOXR

IOCS Number 2FH (for PBOXA); 30H (for PBOXR)

Function Draw a box in the absolute coordinate mode (PBOXA) or in the relative coordinate mode (PBOXR).

Parameter HL = Starting address of the memory locations in which coordinate data are stored

Coordinate data prepared in memory

Address	Contents
HL HL+1	Start point X coordinate (−2048 to 2047) {Low byte {High byte
HL+2 HL+3	Start point Y coordinate (−2048 to 2047) {Low byte {High byte
HL+4 HL+5	Diagonal point X coordinate (−2048 to 2047) {Low byte {High byte
HL+6 HL+7	Diagonal point Y coordinate (−2048 to 2047) {Low byte {High byte

*Specify a coordinate value in two bytes (a negative value in the complement expression). That is, 0 to 2047 is expressed as 0000H to 07FFH, and −2048 to −1 as F800H to FFFFH.

Return CF = 1 if an error has occurred.
A = Error code (If the BREAK key has been pressed, then A = 00H is returned.)

Remarks This routine is effective only in the graphics mode.
The pen position when the routine has been completed is at the start point.

HARESET

IOCS Number 31H

Function Initialize the printer hardware.

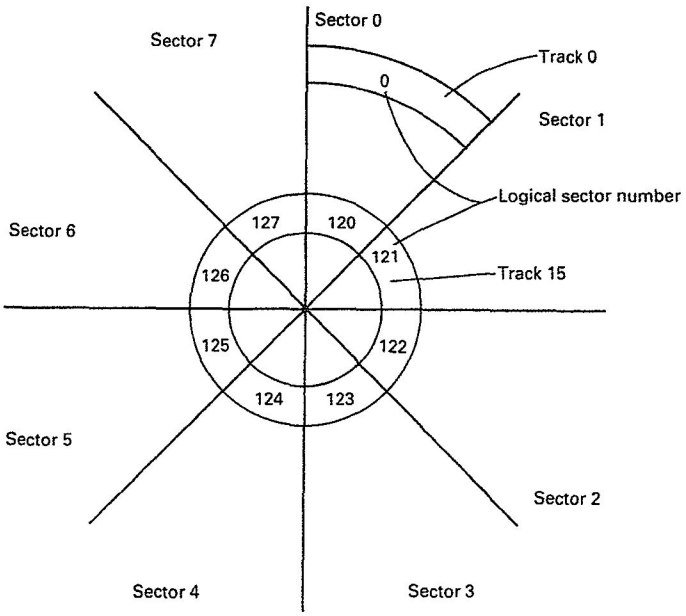
Parameter none

Remarks
1. For the power-on processing, call this routine immediately after an IOCS No. 01H routine.
2. For the all-reset processing, call this routine immediately after an IOCS No. 28H routine.

3.8 DISK

3.8.1 Floppy Disk Format

A floppy disk is formatted to the tracks and sectors as shown below.



The sectors are given a serial number (logical sector number) of from 0 to 127, starting from sector 0 of track 0 through to sector 7 of track 15.

3.8.2 Specifications of Floppy Disk

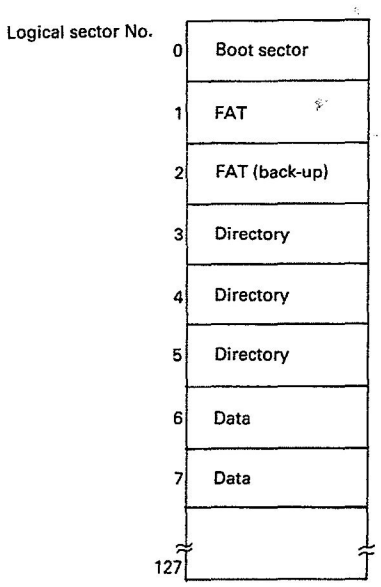
- Number of tracks per side: 16
- Number of sectors per track: 8
- Number of bytes per sector: 512
- Number of sectors per FAT: 1
- Number of FATs: 2
- Number of logical sectors per cluster: 1
- Maximum files per side: 48

3.8.3 File Management

(1) Arrangement of logical sectors

The logical sectors are classified into four areas and are arranged as shown in the figure below.

Logical sectors
- Boot sector area
- File allocation table (FAT) area
- Directory area
- Data area



(2) Boot sector

At power-on time, the contents of the boot sector are read out to the buffer and checked whether they are a boot program.

(3) FAT

The file allocation table (FAT) is a map indicating which files are occupying where on the disk. Files on a disk are physically managed in the units called clusters, and each cluster is managed by one byte of cluster information written in FAT.

The first byte of FAT contains the disk format ID code (F2H) and the second and the following bytes contain the mapping information of the data area. The mapping information consists of 122 bytes of data, each of which represents information of a particular cluster and they are written in the order of from cluster 1 to cluster 122 (the clusters 1 to 122 respectively correspond to logical sectors 6 to 127.) Each cluster information has the following meanings:

00H: This means that the cluster is not used.

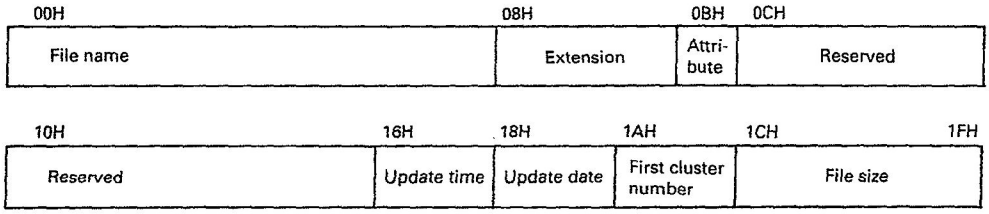
01H to 7AH: This means that the cluster is used and the value designates the cluster number of the cluster that should come after the currently concerned cluster.

F0H: This means that the cluster is the last cluster of the file.

(4) Directory

The directory contains information about each file on the disk and uses 32 bytes of area per file. The directory occupies 3 sectors of area on the disk, and since one sector consists of 512 bytes, 48 files (512 × 3/32) can exist on each side of the disk.

Each file information (32 bytes) consists of the following items:



* The location of each item such as 08H or 0CH is the offset from 00H.

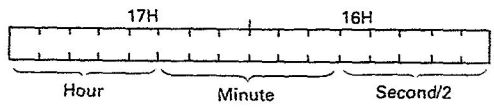
(1) File name (00H to 07H)
Stores the file name. If the file name is less than 8 characters, the space is filled with space characters (code 20H).

(2) Extension (08H to 0AH)
Stores the extension. If no extension is given or if the extension is less than 3 characters, the space is filled with space characters (code 20H).

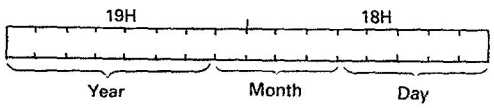
(3) Attribute (0BH)
Stores the attributes of the file. The bits of this byte have the following meanings:
Bit 0: 0 = Read/write 1 = Read only
Bits 1 to 7: Reserved

(4) Reserved area (0CH to 15H)
Always filled with 00H.

(5) Update time (16H and 17H)
Stores the time when the file is created or updated.



(6) Update date (18H and 19H)
Stores the date when the file is created or updated.



(7) First cluster number (1AH and 1BH)
Stores the cluster number of the first cluster used for the file. 1AH stores this cluster number and 1BH always stores 00H.

(8) File size (1CH to 1FH)
Stores the size of the file in bytes. The size data are written from 1CH in the order of low byte and high byte.

3.8.4 IOCS Routines for Floppy Disk

This section describes the IOCS routines related to the floppy disk. These routines can be called as follows:

(1) set the desired parameters,
(2) set the IOCS number in C register, and
(3) call address 4008H of bank 5.

The work area for the IOCS routines must be reserved before they are executed.

The table below lists the IOCS routines for the floppy disk.

The drive number described as a parameter in the following explanation designates one of the following drives (the PC-1600 virtually supports two disk drives):

01H: X drive
02H: Y drive

The error information, if an error has occurred, is returned to A register expressed as an 8-bit code. These bits represent the following errors and they are significant when the corresponding bits are 1.

Bit 0: Drive not ready/time out
Bit 1: Access error
Bit 2: "
Bit 3: The requested sector could not be found.
Bit 4: Head seek error
Bit 5: Disk write protected
Bit 6: Low battery
Bit 7: Other errors (e.g., no disk is set in the drive.)

Name	Function	IOCS No.
DSKINIT	Initialize the disk drive.	80H
CNCTDRV	Read the number of disk drives connected.	81H
RESTORE	Move the disk head to track 0.	82H
FORMAT	Format the floppy disk.	83H
DREAD	Read the contents of specified sectors.	84H
DWRITE	Write data into specified sectors.	85H
DVERIFY	Verify the contents of specified sectors with data in memory.	86H
GETDRVST	Read the state of the disk drive.	87H
HFREAD	Read the contents of the first half (256 bytes) of a specified sector.	88H
HFVERIFY	Verify the contents of the first half (256 bytes) of a specified sector with data in memory.	89H

DSKINIT

IOCS Number 80H

Function Initialize the disk drive specified by A register.

Parameter A = Drive number

Return CF = 0: Normal termination
= 1: An error has occurred. A = Error code

Affected Register All registers except for background registers

CNCTDRV

IOCS Number 81H

Function Read the number of disk drives connected.

Parameter none

Return A = Number of disk drives connected

Affected Register AF, BC, IY

RESTORE

IOCS Number 82H

Function Move the head of the disk drive specified by A register to track 0.

Parameter A = Drive number

Return CF = 0: Normal termination
= 1: An error has occurred. A = Error code

Affected Register All registers except for background registers

FORMAT

IOCS Number 83H

Function Format the floppy disk set in the disk drive specified by A register.

Parameter A = Drive number

Return CF = 0: Normal termination
= 1: An error has occurred. A = Error code

Affected Register All registers except for background registers

DREAD

IOCS Number 84H

Function Read data sequentially from specified sectors into memory.
The first sector to be read is specified by D and E registers, and the number of sectors to be read is specified by B register. The starting address of the memory locations to which the data are written is specified by HL register.

Parameter
A = Drive number
B = Number of sectors
D = Track number of the first sector (00H to 0FH)
E = Sector number of the first sector (00H to 07H)
HL = Starting address of the memory locations

Return CF = 0: Normal termination
= 1: An error has occurred. A = Error code

Affected Register All registers except for background registers

Remarks The sectors to be read can be on more than one track. If an invalid track number or sector number is specified, a proper read action may not be guaranteed.

DWRITE

IOCS Number 85H

Function Write sequentially the contents of consecutive memory locations into specified sectors.
The starting address of the memory locations from which data are read is specified by HL register. The first sector to write is specified by D and E registers, and the number of sectors to write is specified by B register.

Parameter
A = Drive number
B = Number of sectors
D = Track number of the first sector (00H to 0FH)
E = Sector number of the first sector (00H to 07H)
HL = Starting address of the memory locations

Return
CF = 0: Normal termination
= 1: An error has occurred. A = Error code

Affected Register All registers except for background registers

Remarks The sectors to write can be on more than one track. If an invalid track number or sector number is specified, a proper write action may not be guaranteed.

DVERIFY

IOCS Number 86H

Function Verity the contents of specified sectors with the contents of specified memory locations.
The first sector to be verified is specified by D and E registers, and the number of sectors to be verified is specified by B register. The starting address of the memory locations is specified by HL register.

Parameter
A = Drive number
B = Number of sectors
D = Track number of the first sector (00H to 0FH)
E = Sector number of the first sector (00H to 07H)
HL = Starting address of the memory locations

Return
CF = 0: Normal termination
= 1: An error has occurred.
A = Error code (If it is a verity error, then 00H is returned to A register and the number of the sectors that were not verified is returned to B register.)

Affected Register All registers except for background registers

Remarks The sectors to be verified can be on more than one track. If an invalid track number or sector number is specified, a proper verify action may not be guaranteed.

GETDRVST

IOCS Number 87H

Function Read the state of the disk drive and floppy disk.

Parameter A = Drive number

Return
CF = 0: A = State of disk drive
A: Bit 0: This bit is set to "1" if the disk drive is in operation.
Bit 1: Always "0"
Bit 2: This bit is set to "1" if the disk drive door has been opened (for disk change for example) after the previous execution of one of the disk IOCS routines before the execution of this GETDRVST routine. (This bit is significant only when bit 0 = "0".)
Bit 3: This bit is set to "1" if a floppy disk is set in the disk drive.
Bit 4: Always "0"
Bit 5: Always "0"
Bit 6: This bit is set to "0" if the floppy disk is write-protected. (This bit is significant only when bit 3 is "1" and bit 7 is "0".)
Bit 7: This bit is set to "1" if the disk drive is not ready.
CF = 1: The specified disk drive is not connected.

Affected Register All registers except for background registers

HFREAD

IOCS Number 88H

Function Read the contents of the first half 256 bytes of the sector specified by D and E registers into the memory locations whose starting address is specified by HL register.

Parameter
A = Drive number
D = Track number (00H to 0FH)
E = Sector number (00H to 07H)
HL = Starting address of the memory locations

Return CF = 0: Normal termination
= 1: An error has occurred. A = Error code

Affected Register All registers except for background registers

Remarks If an invalid track number or sector number is specified, a proper read action may not be guaranteed.

HFVERIFY

IOCS Number 89H

Function Verify the contents of the first half 256 bytes of the sector specified by D and E registers with the contents of the memory locations whose starting address is specified by HL register.

Parameter A = Drive number
D = Track number (00H to 0FH)
E = Sector number (00H to 07H)
HL = Starting address of the memory locations

Return CF = 0: Normal termination
= 1: An error has occurred.
A = Error code (If it is a verify error, A = 00H is returned.)

Affected Register All registers except for background registers

Remarks If an invalid track number or sector number is specified, a proper verify action may not be guaranteed.

3.8.5 Processing at Power-On Time

If a floppy disk drive is connected to the PC-1600, the following processing is performed to the disk drive when the PC-1600 is powered on.

1. Whether the disk drive is properly connected is checked, and the work area for floppy disk drive is reserved.
2. The contents of the boot sector are read. If there is a boot program, the program is executed.
   If the content of the fourth byte of logical sector 0 (the boot sector) is C3H, this means that there is a boot program in the boot sector. In this case, the program execution starts from the 33rd byte of logical sector 0.
3. If the PC-1600 is in the RUN mode when it is powered on, the PC-1600 searches for a file named "AUTORUN.BAS". If the file exists, it is loaded and executed.

3.9 TIMER / ANALOG PORT

This section describes the IOCS routines for the timer and the analog port. These routines can be called as follows:

(1) set the IOCS number of the desired routine in C register, and

(2) call the address 01D5H:

```
CALL 01D5H
```

For example, to call the SBEEP IOCS routine to generate a key clicking sound, execute the following:

```
LD C,01H
CALL 01D5H
```

The table below lists the names, functions, and IOCS numbers of the IOCS routines related to the timer and the analog port.

IOCS Name	Function	IOCS No.
SINIT	Initialize the timers and the analog input port.	00H
SBEEP	Generate a key clicking sound.	01H
SWRT	Set a date and time for the calendar clock.	02H
SRRT	Read the current date and time of the calendar clock.	03H
SWWT	Set a date and time for the wakeup timer.	04H
SWRT	Read the current settings of date and time of the wakeup timer.	05H
SWA1T	Set a date and time for the alarm1 timer (same as ON TIME$ command).	06H
SRA1T	Read the current settings of dame and time of the alarm1 timer.	07H
SWA2T	Set a date and time for the alarm2 timer (same as ALARM$ command).	08H
SRA2T	Read the current settings of dame and time of the alarm2 timer.	09H
SWMSK	Set the interrupt mask for SC-7852.	10H
SRMSK	Read the current setting of the interrupt mask for SC-7852.	11H
SRIRQ	Read the current interrupt cause for SC-7852.	12H
SRINP	Read the state of the CI signal of RS-232C port.	13H
SWPON	Set a mask for the power-on conditions.	14H
SRA0	Read the digital value of the supply voltage of the PC-1600 main unit.	18H
SRA1	Read the digital value of the voltage input at the analog input port.	19H
SRA2	Read the digital value of the CE-1600P battery voltage.	1AH
SRPON	Read the settings of the mask for the power-on conditions.	21H
SWAB	Set the conditions for alarm beep generation.	22H
SRAB	Read the current settings of the alarm beep generation conditions set by SWAB routine.	23H
SWA1A	Set the software interrupt trigger levels for a digital value of the analog input port.	24H

SINIT

IOCS Number 00H

Function Initialize the timers and the analog input port.

Parameter A = 00H: Initialize to the ALL RESET state.
= 01H: Initialize to the power-on state that is given after the power is turned off by the OFF key.
= 02H: Initialize to the power-on state that is given after the power is turned off by the auto power-off function.

Return none

Affected Register AF

SBEEP

IOCS Number 01H

Function Generate a key clicking sound.

Parameter none

Return none

Affected Register AF

SWRT

IOCS Number 02H

Function Set a date and time for the calendar clock.

Parameter HL = Starting address of the memory locations in which the date and time data are stored

Address	Contents	
	Upper 4 bits	Lower 4 bits
HL	0	Month: 0 to C
HL+1	Day (tenth digit): 0 to 3	Day (unit digit): 0 to 9
HL+2	Hour (tenth digit): 0 to 2	Hour (unit digit): 0 to 9
HL+3	Minute (tenth digit): 0 to 6	Minute (unit digit): 0 to 9
HL+4	Second (tenth digit): 0 to 6	Second (unit digit): 0 to 9

*If the content of the upper or lower 4 bits is set to "F", that item is not updated and remains with the old value. For instance, if the upper 4 bits of address (HL+1) is set to "F", the tenth digit of "day" is not updated.

Return none

Affected Register AF, HL, BC

Remarks The contents of the memory locations specified by HL to HL+4 are destroyed.

SRRT

IOCS Number 03H

Function Read the current date and time of the calendar clock into the memory locations whose starting address is specified by HL register. The data format is the same as used in SWRT routine.

Parameter HL = Starting address of the memory locations

Return none

Affected Register AF, HL, BC

SWWT

IOCS Number 04H

Function Set a date and time for the wakeup timer.

Parameter HL = Starting address of the memory locations in which the date and time data are stored

Address	Contents	
	Upper 4 bits	Lower 4 bits
HL	0	Month: 0 to C
HL+1	Day (tenth digit): 0 to 3	Day (unit digit): 0 to 9
HL+2	Hour (tenth digit): 0 to 2	Hour (unit digit): 0 to 9
HL+3	Minute (tenth digit): 0 to 6	Minute (unit digit): 0 to 9
HL+4	0	0

*If the content of the upper or lower 4 bits is set to "F", that item is not referenced.

*When thisroutine is called, the wakeup interrupt is disabled, that is, the wakeup event is not performed even when it reaches the wakeup time.

Return none

Affected Register AF, HL, BC

SRWT

IOCS Number 05H

Function Read the current settings of date and time of the wakeup timer into the memory locations whose starting address is specified by HL register. The data format is the same as used in SWWT routine except that nothing is written to address HL+4.

Parameter HL = Starting address of the memory locations

Return none

Affected Register AF, HL, BC

SWA1T

IOCS Number 06H

Function Set a date and time for the alarm1 timer. The alarm1 timer is used for ON TIME$ command of BASIC.

Parameter HL = Starting address of the memory locations in which the date and time data are stored

Address	Contents	
	Upper 4 bits	Lower 4 bits
HL	0	Month: 0 to C
HL+1	Day (tenth digit): 0 to 3	Day (unit digit): 0 to 9
HL+2	Hour (tenth digit): 0 to 2	Hour (unit digit): 0 to 9
HL+3	Minute (tenth digit): 0 to 6	Minute (unit digit): 0 to 9
HL+4	0	0

*If the content of the upper or lower 4 bits is set to "F", that item is not referenced.

Return none

Affected Register AF, HL, BC

SRA1T

IOCS Number 07H

Function Read the current settings of date and time of the alarm1 timer into the memory locations whose starting address is specified by HL register. The data format is the same as used in SWA1T routine except that nothing is written to address HL+4.

Parameter HL = Starting address of the memory locations

Return none

Affected Register AF, HL, BC

SWA2T

IOCS Number 08H

Function Set a dame and time for the alarm2 timer. The alarm2 timer is used for ALARM$ command of BASIC.

Other items Same as those of SWA1T routine

SRA2T

IOCS Number 09H

Function Same as SRA1T except that this routine is for the alarm2 timer.

Other items Same as those of SRA1T routine

SWMSK

IOCS Number 10H

Function Set the interrupt mask for SC-7852.

Parameter A = Mask data
(Set those bits corresponding to the interrupt causes to "1" if you want to mask them, or to "0" if you do not want to mask them.)

Interrupt cause	Mask bit set in A register
Interrupt by the wakeup timer	bit 7 (MSB)
Interrupt by the alarm1 timer	bit 6
Interrupt by the alarm2 timer	bit 5
Interrupt by the 1S signal	bit 2
Interrupt by the 0.5S signal	bit 1

Return none

Affected Register AF

SRMSK

IOCS Number 11H

Function Read the current settings of the interrupt mask for SC-7852.

Parameter none

Return A = Maskdata
(The interrupt causes are currently masked if those bits corresponding to the interrupt causes are "1", or they are currently masked if the bits are "0".)

Interrupt cause	Mask bit set in A register
Interrupt by the wakeup timer	bit 7 (MSB)
Interrupt by the alarm1 timer	bit 6
Interrupt by the alarm2 timer	bit 5
Interrupt by the 1S signal	bit 2
Interrupt by the 0.5S signal	bit 1

Affected Register AF

SRIRQ

IOCS Number 12H

Function Read the current interrupt cause for SC-7852.

Parameter none

Return A = Interrupt cause
(If an interrupt has occurred, the bit corresponding to that interrupt cause is set to "1". If an interrupt has not occurred, the bit is set to "0".)

Interrupt cause	Bits set in A register
Interrupt by the wakeup timer	bit 7 (MSB)
Interrupt by the alarm1 timer	bit 6
Interrupt by the alarm2 timer	bit 5
Interrupt by the 1S signal	bit 2
Interrupt by the 0.5S signal	bit 1

Affected Register AF

SRINP

IOCS Number 13H

Function Read the state of the CI signal of RS-232C port.

Parameter none

Return A = State of CI signal
(Bit 5 of the byte in A register indicates the state of the CI signal: If the CI signal is high, then bit 5 = 0. If it is low, then bit 5 = 1.)

Affected Register AF

SWPON

IOCS Number 14H

Function Set a mask for the power-on conditions.

Parameter A = Mask data
(If those bits corresponding to the power-on conditions are set to "1", they are masked.)

b7	0
b6	0
b5	0
b4	0
b3	When the power is on, a beep is generated once at every o'clock.
b2	If b1 is set to "1", a beep is generated every second starting from seven seconds before the PC-1600 is powered on by a wakeup event.
b1	The PC-1600 can be powered on by the wakeup timer.
b0	The PC-1600 can be powered on by the CI signal of RS-232C.

Return none

Affected Register AF

Remarks If the PC-1600 can be powered on by the wakeup timer, the interrupt mask bit 7 must also be set.

SRA0

IOCS Number 18H

Function Read the digital value of the supply voltage of the PC-1600 main unit.

Parameter none

Return A = Supply voltage value

Affected Register AF

Remarks This routine is used for monitoring the supply voltage of the PC-1600 main unit. If the value is less than AFH, it is judged as the low battery. The low battery state is released when the value becomes greater than BEH.

SRA1

IOCS Number 19H

Function Read the digital value of the voltage input at the analog input port.

Parameter none

Return A = Voltage value

Affected Register AF

SRA2

IOCS Number 1AH

Function Read the digital value of the supply voltage of the peripheral device.

Parameter none

Return A = Voltage value

Affected Register AF

Remarks This routine is used for monitoring the Ni-Cd battery supply voltage of the CE-1600P. If the value is less than A8H, it is judged as the low battery.

SRPON

IOCS Number 21H

Function Read the current settings of the mask for the power-on conditions (that is, the current mask data set by SWPON routine.)

Parameter none

Return A = Mask data (The data format is the same as used for the parameter of SWPON routine.)

Affected Register AF

SWAB

IOCS Number 22H

Function Set the conditions for alarm beep generation.

Parameter
A = 00H: No beep is generated even when any alarm event occurs.
= 01H: A beep is generated only when an alarm1 event occurs.
= 02H: A beep is generated only when an alarm2 event occurs.
= 03H: A beep is generatedwhen either alarm1 or alarm2 event occurs.

Return none

Affected Register AF

SRAB

IOCS Number 23H

Function Read the current setting of the alarm beep generation conditions set by SWAB routine.

Parameter none

Return A = Data set by SWAB routine

Affected Register AF

SWA1A

IOCS Number 24H

Function Set the software interrupt trigger levels for a digital value of the analog input port.

Parameter H = Upper limit value
L = Lower limit value

Return none

Affected Register AF

3.10 BEEP

This section describes the IOCS routines for the beep. The table below lists the names, functions, and entry addresses of these routines. The routines can be called by using CALL instruction of Z80 as follows:

CALL <Entry address>

Name	Entry address	Function
BOUT	01B4H	Generate beeps of the specified tone and duration.
SOUT	01B7H	Generate one beep of the specified tone and duration.
SWAIT	01BAH	Wait for a specified period of time.
BONOFF	01BDH	Enable or disable the beep.

BOUT

Entry Address 01B4H

Function Generate beeps of the specified tone and duration.

Parameter
A = Tone data (00H to FFH)
BC = Duration data (0000H to FEFFH)
DE = Number of times (0000H to FFFFH)

Return
CF = 0: Normal termination
= 1: The execution of the routine was halted by the BREAK key.

Affected Register AF

Remarks The frequency of a beep is determined by the tone data (set in A register) as follows:

Frequency = 1300000/(166+22*A) [Hz]

The duration of a beep is determined by the duration data (set in BC register) and the tone data (set in A register) as follows:

Duration = BC*(166+22*A)/1300000
= BC/frequency [second]

When the BREAK key is pressed, the beeping action stops in the middle and the routine is terminated. This routine generates beeps regardless of the BEEP ON/OFF statement.

SOUT

Entry Address 01B7H

Function Generate a beepof the specified tone and duration.

Parameter
A = Tone data (00H to FFH)
BC = Duration data (0000H to FEFFH)

Other items Same as BOUT

SWAIT

Entry Address 01BAH

Function Wait for a specified period of time.

Parameter BC = Time data (0000H to FFFFH)
The actual wait time is BC/64 [second].

Return CF = 0: Normal termination
= 1: The execution of the routine was halted by the BREAK key.

Affected Register BC, flags

BONOFF

Entry Address 01BDH

Function Enable or disable the beep.

Parameter Bit 0 of the content of MODEF (address F86BH)
= 1: Beep OFF
= 0: Beep ON

Return none

Affected Register none

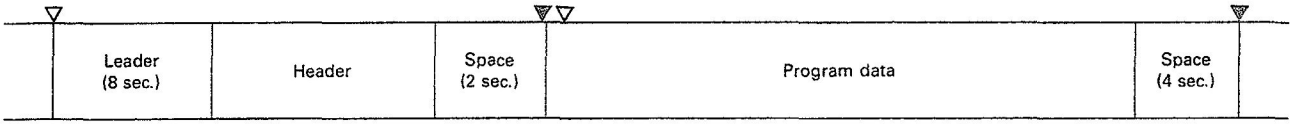
3.11 TAPE RECORDER

3.11.1 PC-1600 Mode (Mode 0)

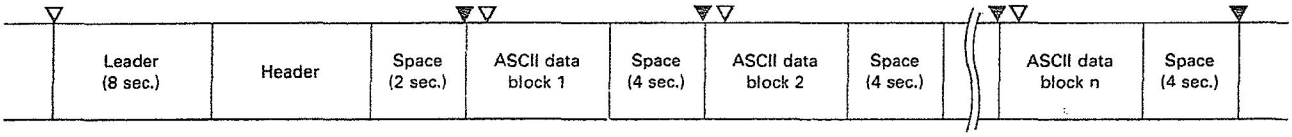
[1] Cassette tape logical format

(1) Recording format of a file

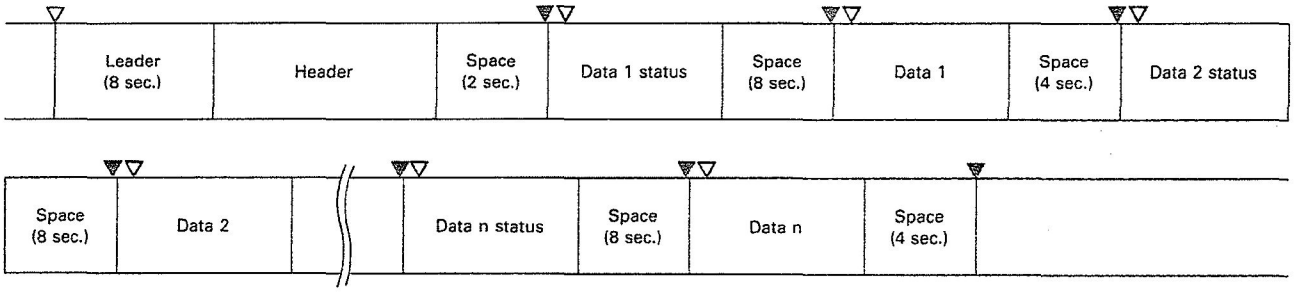
(a) BASIC program (intermediate code format), Machine language program, RESERVE contents



(b) BASIC program (ASCII saved format), Text data (ASCII)



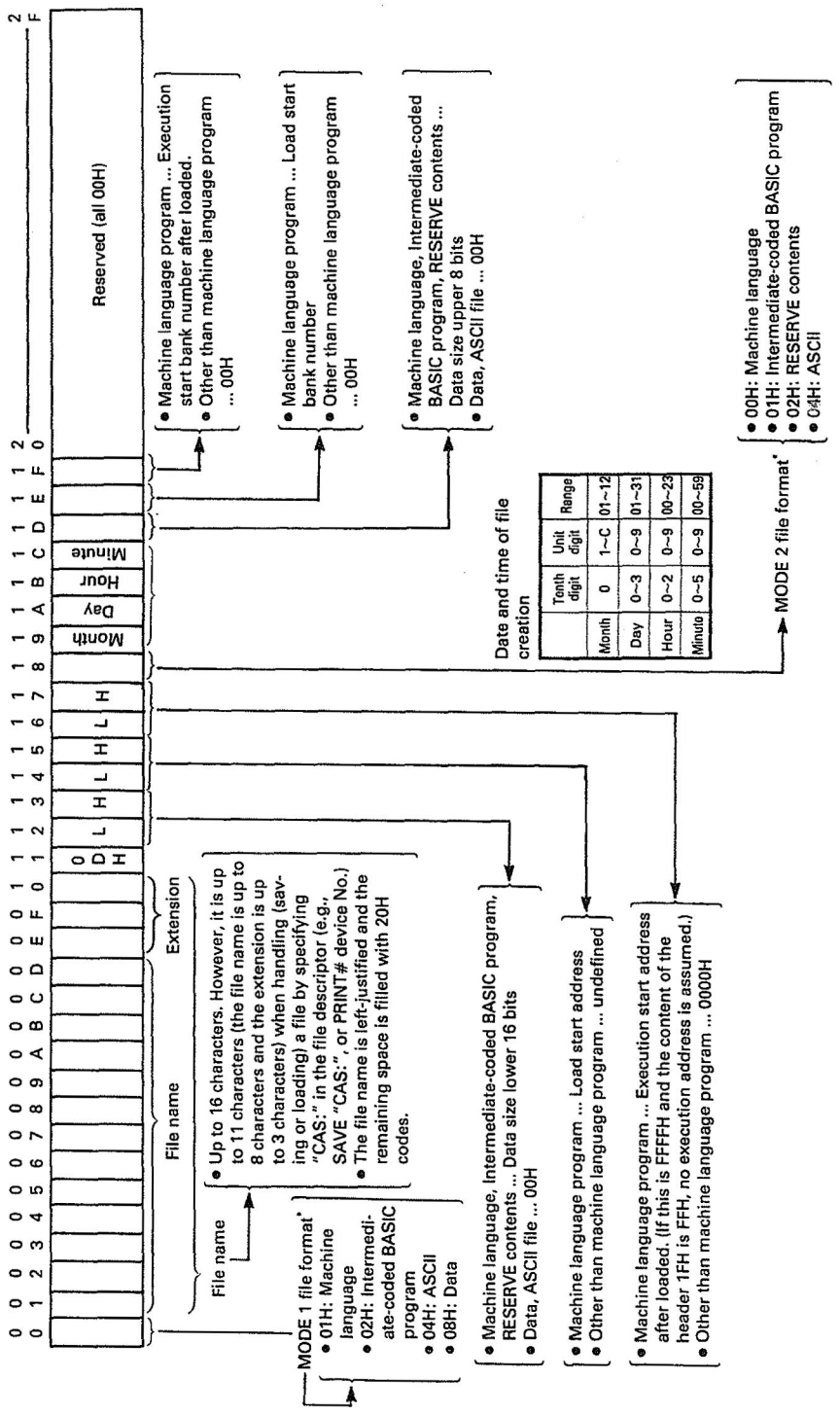
(c) Data (special format)



*▼: The cassette tape stops.

▽: The cassette tape starts.

(2) Header structure

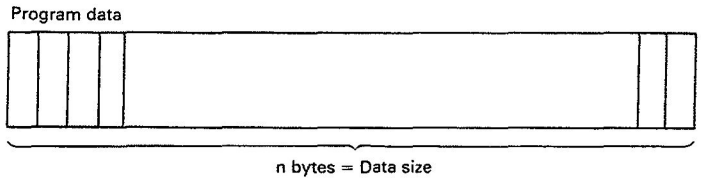


※

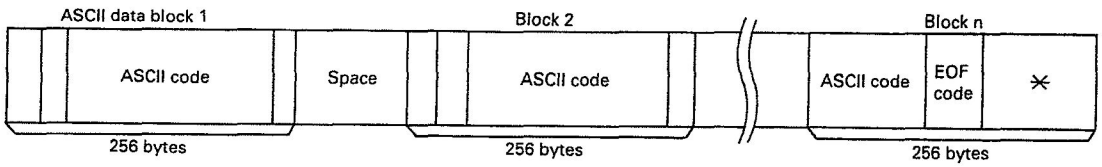
MODE 1	MODE 2	File type	Recording format (See the above item (1) "Recording format of a file".)
01H	00H	Machine language program	(a)
02H	01H	BASIC program (intermediate code format)	(a)
02H	02H	RESERVE contents (internal code format)	(a)
04H	04H	BASIC program (ASCII format)	(b)
04H	04H	Data (ASCII format)	(b)
08H	04H	Data (special format)	(c)

(3) Data block structure (See item (1) above.)

(a) BASIC program (intermediate-coded), Machine language program, RESERVE contents



(b) BASIC program (ASCII saved), Data (ASCII format)

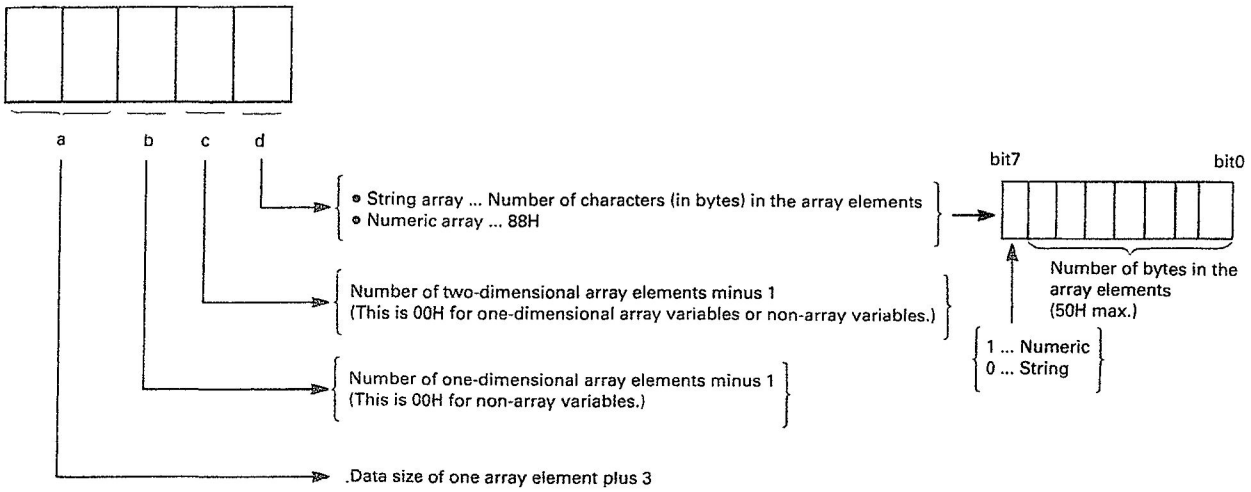


- ASCII data are handled in blocks of 256 bytes.
- The end of a file is detected when an EOF code (1AH) is encountered.
- If the last block (the block that includes an EOF code) is less than 256 bytes, meaningless data are automatically added at the end of the data (the part indicated by * in the above figure) to make it as a complete 256-byte block.

(c) Data (Special format)

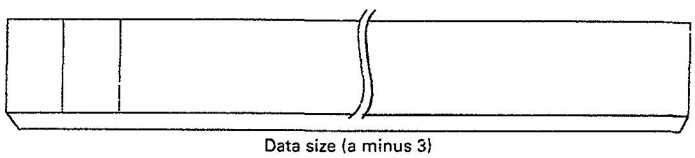
- Status part

  The status part consists of 5 bytes and has the following structure.



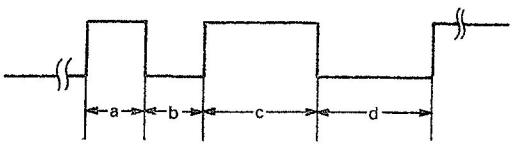
Value of a, b, c, and d (hex)				Variable type	Data size (bytes)
a	b	c	d		
000B	00	00	88	Numeric data (e.g., A, B)	8
0013	00	00	10	String data (e.g., A$, B$)	16
00D3	19	00	88	@ (*)	208
01A3	19	00	10	@$ (*)	416
a	b	c	88	Numeric array	(b+1)*(C+1)*8
a	b	c	d	String array	(b+1)*(C+1)*d

- Data part
  The data part contains the contents of one element of an array variable in the internal code format (see section 4.1.3).



[2] Cassette tape physical format

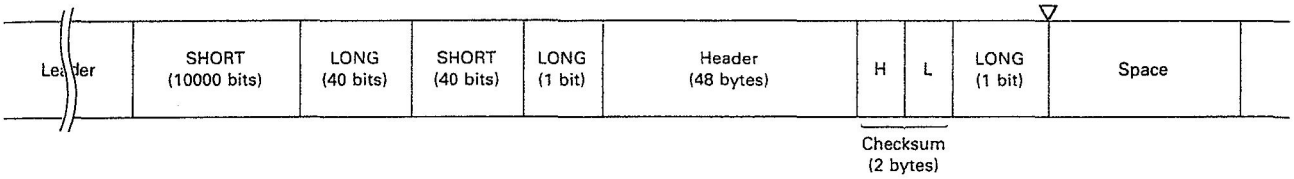
(1) Bit structure
One bit (SHORT...expresses the value "0"; LONG ... expresses the value "1") consists of the following pulses.



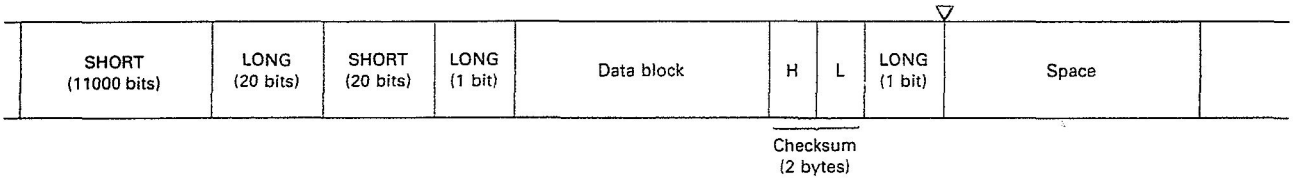
	Pulse width (μs)
a	163
b	164
c	409
d	409

※These pulse widths are the default values. They can be changed by changing the contents of the cassette tape work area. (See section 3.11.3 for details.)

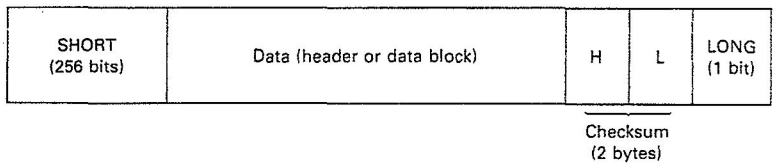
(2) Header structure



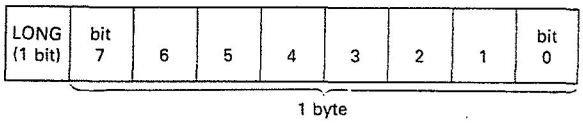
(3) Data block structure



※In the double write mode, the following contents are added at the position indicated by ▽.
In principle, however, the PC-1600 does not use the double write mode.



(4) Byte structure



(5) Checksum structure

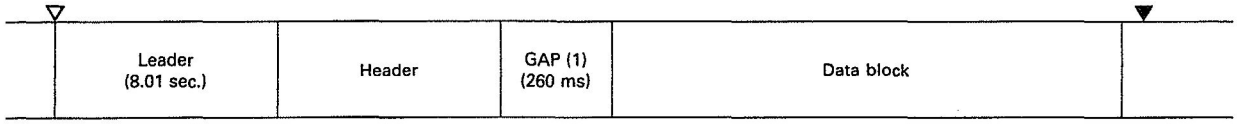
The checksum is the value of the lower 2 bytes of the sum of all LONG bits appearing in the data excluding the first LONG bit at the beginning of each byte within the header or data block.

3.11.2 PC-1500/PC-1500A Mode (Mode 1)

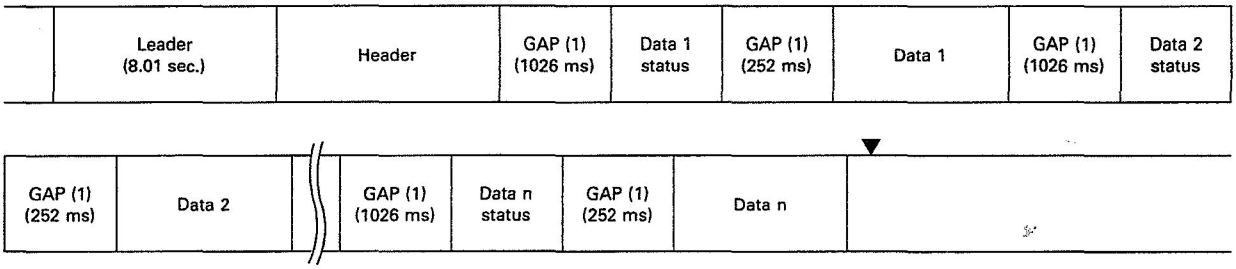
[1] Cassette tape logical format

(1) Recording format of a file

(a) BASIC program, Machine language program, RESERVE contents



(b) Data

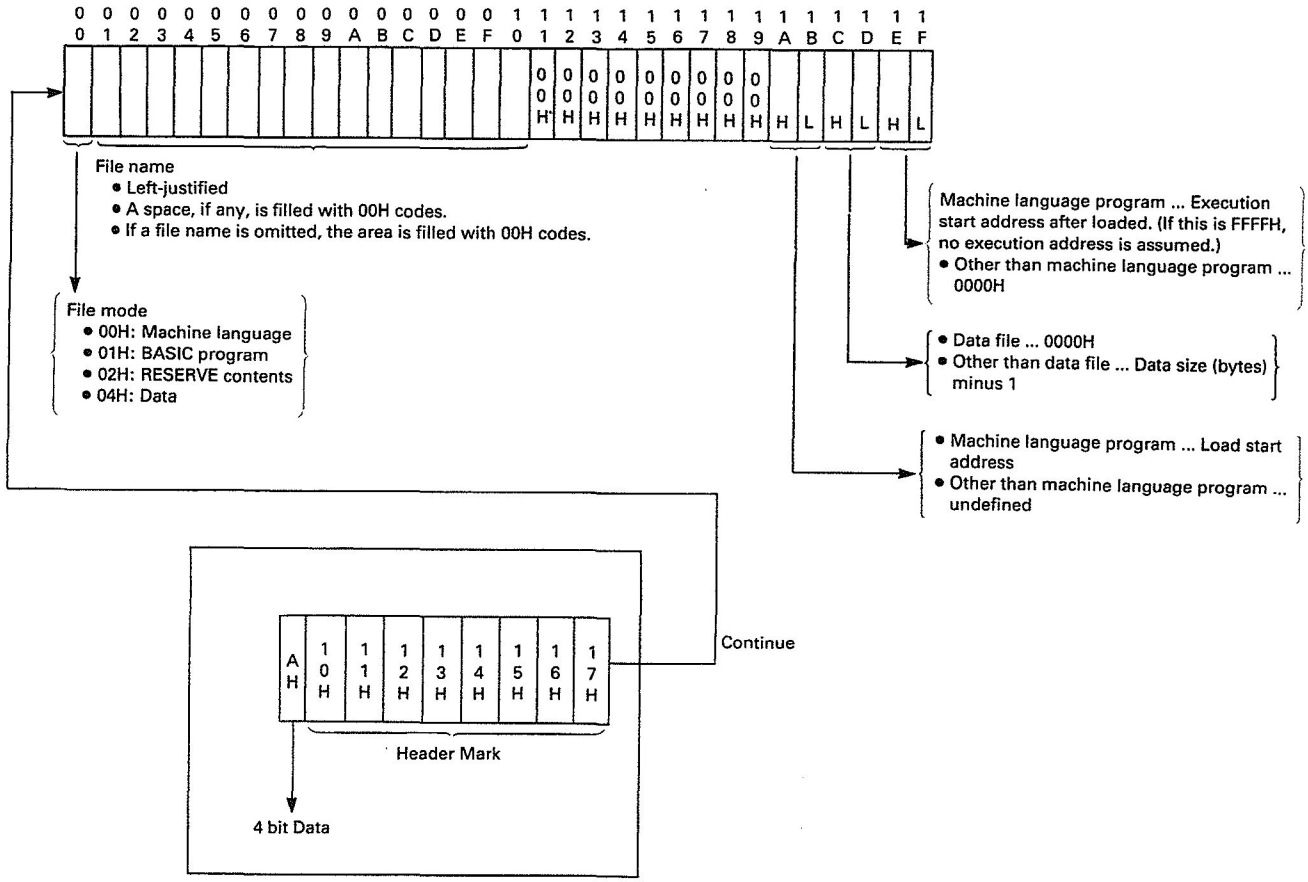


*▼: The cassette tape stops.

▽: The cassette tape starts.

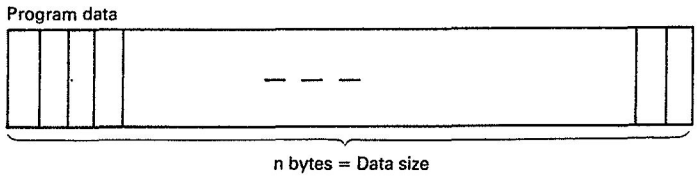
●: Leader/GAP is Stop bit "1". (see Bit structure)

(2) Header structure



(3) Data block structure (See item (1) above.)

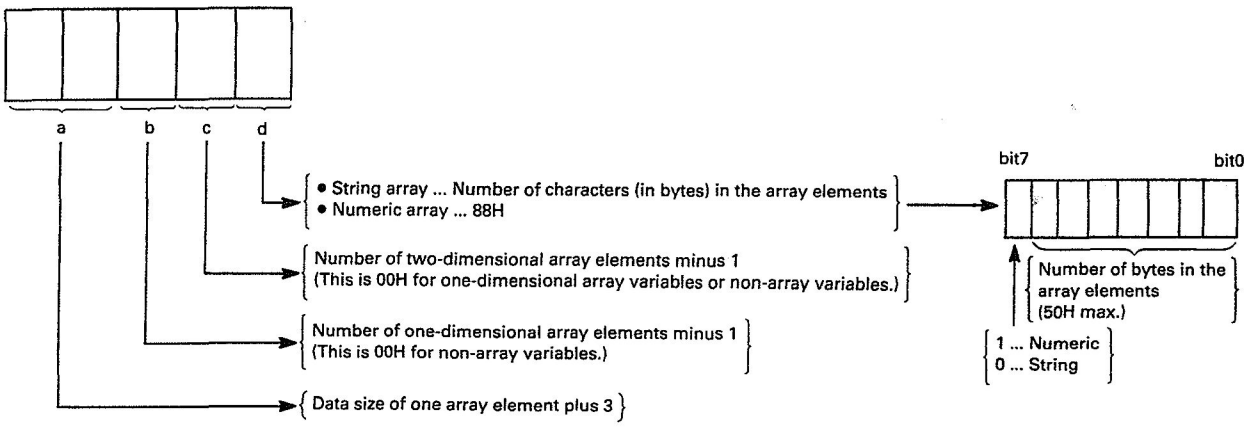
(a) BASIC program, Machine language program, RESERVE contents



(b) Data

- Status part

The status part consists of 5 bytes and has the following structure.

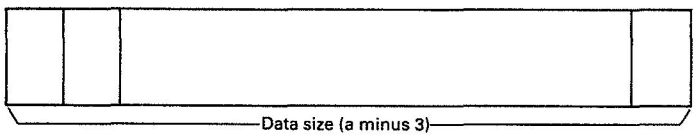


Value of a, b, c, and d (hex)				Variable type	Data size (bytes)
a	b	c	d		
000B	00	00	88	Numeric data (e.g., A, B)	8
0013	00	00	10	String data (e.g., A$, B$)	16
00D3	19	00	88	@ (*)	208
01A3	19	00	10	@$ (*)	416
a	b	c	88	Numeric array	(b+1)*(C+1)*8
a	b	c	d	String array	(b+1)*(C+1)*d

- Data part

The data part contains the contents of one element of an array variable in the internal code format.

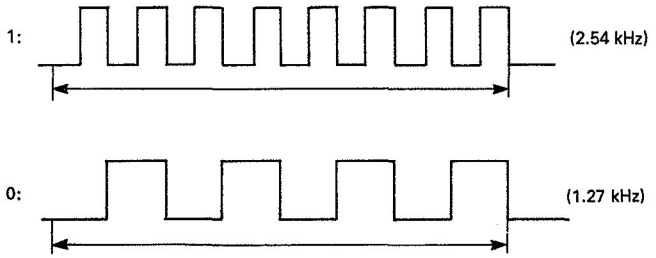
(For details of the internal code format, see section 4.1.3.In section 4.1.3, it is explained that the MSB of the string starting address is inverted. However, this does not apply here.)



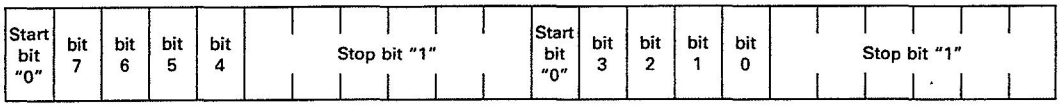
[2] Cassette tape physical format

(1) Bit structure

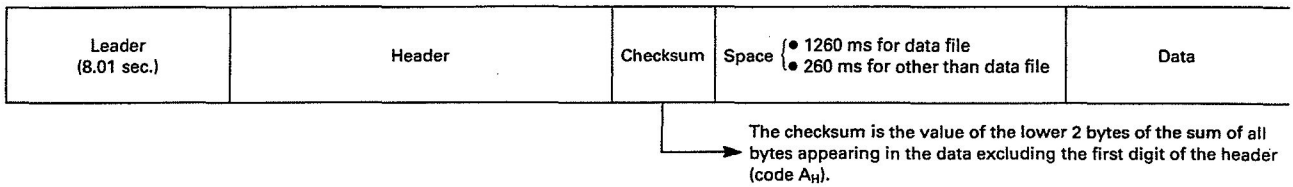
One bit (value "0" or "1") consists of the following pulses.



(2) Byte structure

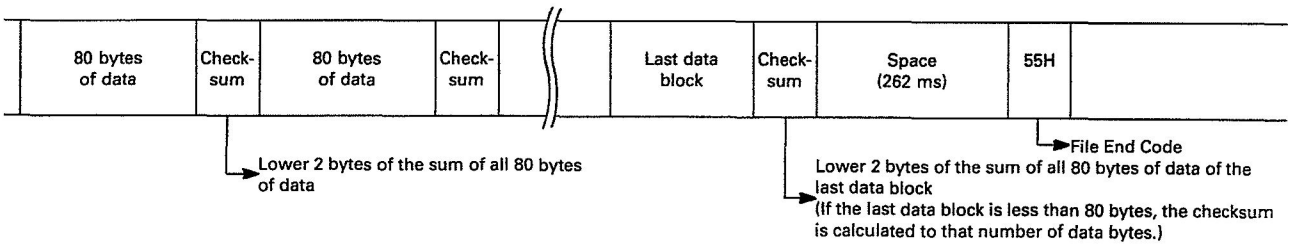


(3) Header structure

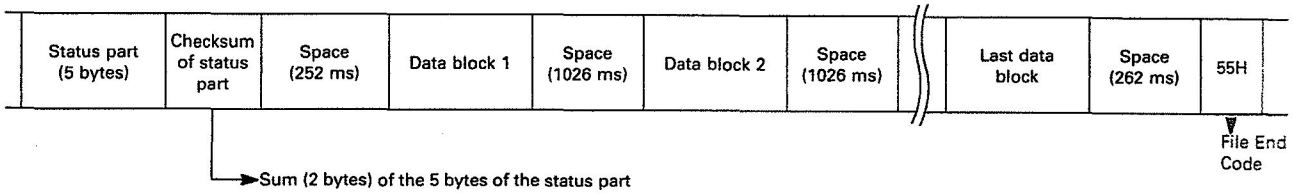


(4) Data block structure

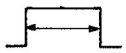
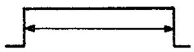
(a) BASIC program, Machine language program, RESERVE contents



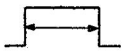
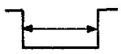
(b) Data



3.11.3 Work Area used for Cassette Tape Recorder

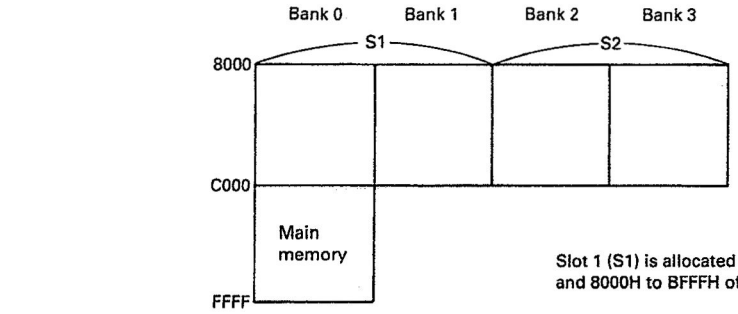
Name	Address	Contents
SHORT HIGH	F197H	• Specify the "high" period of the SHORT (0) pulse in the PC-1600 mode. • The default is 30.  (See note 1)
SHORT LOW	F198H	• Specify the "low" period of the SHORT (0) pulse in the PC-1600 mode. • The default is 23. (See note 1)
LONG HIGH	F199H	• Specify the "high" period of the LONG (1) pulse in the PC-1600 mode. • The default is 79.  (See note 1)
LONG LOW	F19AH	• Specify the "low" period of the LONG(1) pulse in the PC-1600 mode. • The default is 71. (See note 1)
CHECK SUM	F19B~CH	• Checksum calculation register
INFORMATION LEADER	F19DH	• Specify the length of the leader (no signal) part of the header. • The default is 80. (See note 2)
INFORMATION TRAILER	F19EH	• Specify the length of the trailer (no signal) part of the header. • The default is 20. (See note 2)
INFORMATION SHORT 1	F19F~A0H	• A header gap parameter which specifies the number of continuous SHORT pulses for the first time. • The default is 10000.
INFORMATION LONG 1	F1A1H	• A header gap parameter which specifies the number of continuous LONG pulses for the first time. • The default is 40.
INFORMATION SHORT 2	F1A2H	• A header gap parameter which specifies the number of continuous SHORT pulses for the second time. • The default is 40.
DATA SHORT 1	F1A3~4H	• A data block gap parameter which specifies the number of continuous SHORT pulses for the first time. • The default is 11000.
DATA LONG 1	F1A5H	• A data block gap parameter which specifies the number of continuous LONG pulses for the first time. • The default is 20.
DATA SHORT 2	F1A6H	• A data block gap parameter which specifies the number of continuous SHORT pulses for the second time. • The default is 20.
DATA TRAILER	F1A7H	• Specify the length of the trailer (no signal) part of a data block. • The default is 40. (See note 2)
RPOINT	F1A8H	• Specifythe threshold level to be used for judgment whether a bit data read is a LONG or a SHORT pulse. If the bit data is less than this threshold level, it is judged as a SHORT pulse. • The default is 22.

3.11.3 Work Area used for Cassette Tape Recorder

Name	Address	Contents
SHORT HIGH	F197H	• Specify the "high" period of the SHORT (0) pulse in the PC-1600 mode. • The default is 30.  (See note 1)
SHORT LOW	F198H	• Specify the "low" period of the SHORT (0) pulse in the PC-1600 mode. • The default is 23.  (See note 1)
LONG HIGH	F199H	• Specify the "high" period of the LONG (1) pulse in the PC-1600 mode. • The default is 79. (See note 1)
LONG LOW	F19AH	• Specify the "low" period of the LONG(1) pulse in the PC-1600 mode. • The default is 71. (See note 1)
CHECK SUM	F19B~CH	• Checksum calculation register
INFORMATION LEADER	F19DH	• Specify the length of the leader (no signal) part of the header. • The default is 80. (See note 2)
INFORMATION TRAILER	F19EH	• Specify the length of the trailer (no signal) part of the header. • The default is 20. (See note 2)
INFORMATION SHORT 1	F19F~A0H	• A header gap parameter which specifies the number of continuous SHORT pulses for the first time. • The default is 10000.
INFORMATION LONG 1	F1A1H	• A header gap parameter which specifies the number of continuous LONG pulses for the first time. • The default is 40.
INFORMATION SHORT 2	F1A2H	• A header gap parameter which specifies the number of continuous SHORT pulses for the second time. • The default is 40.
DATA SHORT 1	F1A3~4H	• A data block gap parameter which specifies the number of continuous SHORT pulses for the first time. • The default is 11000.
DATA LONG 1	F1A5H	• A data block gap parameter which specifies the number of continuous LONG pulses for the first time. • The default is 20.
DATA SHORT 2	F1A6H	• A data block gap parameter which specifies the number of continuous SHORT pulses for the second time. • The default is 20.
DATA TRAILER	F1A7H	• Specify the length of the trailer (no signal) part of a data block. • The default is 40. (See note 2)
RPOINT	F1A8H	• Specifythe threshold level to be used for judgment whether a bit data read is a LONG or a SHORT pulse. If the bit data is less than this threshold level, it is judged as a SHORT pulse. • The default is 22.

3.12 MEMORY

3.12.1 Slots and Memory Modules



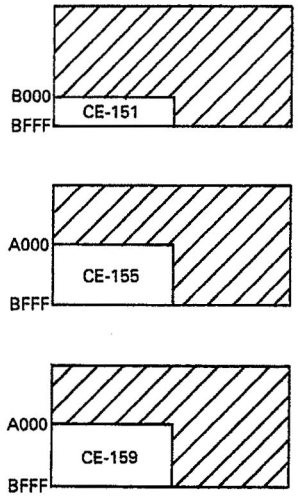
Slot 1 (S1) is allocated to the memory locations of 8000H to BFFFH of bank 0 and 8000H to BFFFH of bank 1.

Slot 2 (S2) is allocated to the memory locations of 8000H to BFFFH of bank 2 and 8000H to BFFFH of bank 3.

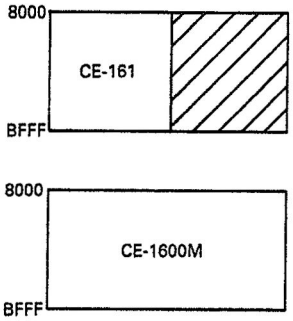
Each module uses a part of or the entire memory locations allocated to that slot as shown in the figures. Those memory locations of the shaded part are not used.

Note: Even when a memory module is installed in S1 or S2, if it is used as an extension memory, it is referred to as S0.

These modules can be installed only in slot 1 (S1).



These modules can be installed in either slot 1 (S1) or slot 2 (S2).



3.12.2 Work Area used for Memory

(1) Work area used to specify the order in which a BASIC program is loaded into banks.

Name	Address	Contents
S0MTb	F02AH	First bank of S0
S1MTb	F016H	First bank of the program module in S1
S1MBb	F018H	Last bank of the program module in S1
S2MTb	F020H	First bank of the program module in S2
S2MBb	F022H	Last bank of the program module in S2

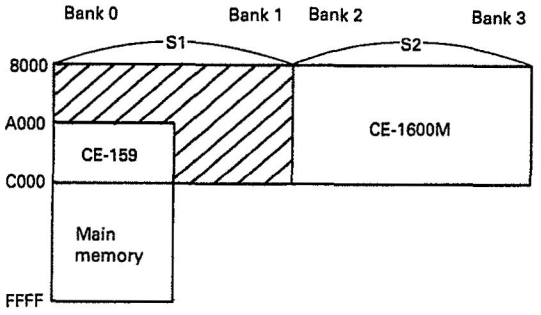
Name	Address
ADTBL+1	F1D6H
ADTBL+2	F1D7H
ADTBL+3	F1D8H
ADTBL+4	F1D9H
ADTBL+5	F1DAH

The bank information is stored in bits 4 and 5 of each of ADTBL+1 to ADTBL+5. If the value of bits 4 and 5 is 00H, this means "unused".

(Example) 01H ... Indicate bank 0.
32H ... Indicate bank 3.
00H ... unused

If S0MTb, S1MTb or S2MTb contains a value between 1 to 5, the bank information is stored in the memory locations starting from ADTBL plus that value. For instance, if the value is 5, the bank information is in locations starting from ADTBL+5. If the value is other than 1 to 5, this means "unused".

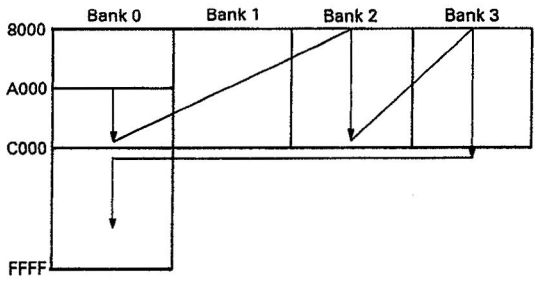
Example 1: When installing CE-159 in S1 and CE-1600M in S2 and using them as an extension memory



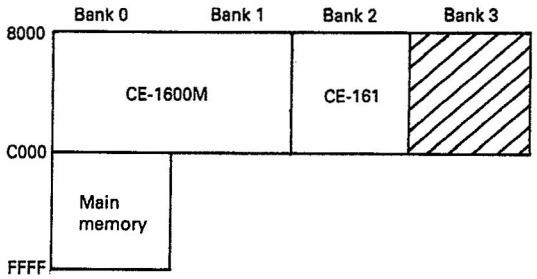
S0MTb	03H	The bank information is in the locations from ADTBL+3.
S1MTb	FEH	S1 is not used as a program module.
S1MBb	?	(It is used as an extension memory.)
S2MTb	FEH	S2 is not used as a program module.
S2MBb	?	(It is used as an extension memory.)

ADTBL+1	00H	Unused
ADTBL+2	00H	Unused
ADTBL+3	01H	Bank 0
ADTBL+4	22H	Bank 2
ADTBL+5	32H	Bank 3

In this example, the program is loaded into bank 0, bank 2, bank 3, and the main memory in that order.

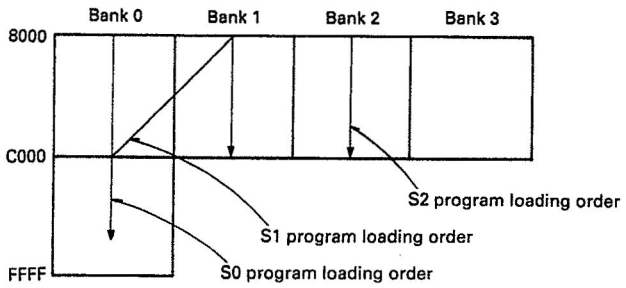


Example 2: When installing CE-1600M in S1 and CE-161 in S2 and using them as an program module



S0MTb	05H	S0 is from ADTBL+5.
S1MTb	02H	S1 is from ADTBL+2 to ADTBL+3.
S1MBb	03H	
S2MTb	04H	S2 is ADTBL+4.
S2MBb	04H	

ADTBL+1	00H	Unused
ADTBL+2	81H	Bank 0
ADTBL+3	11H	Bank 1
ADTBL+4	A2H	Bank 2
ADTBL+5	00H	Unused



3.12.3 IOCS Routines for Memory Control

This section describes the IOCS routines to be used for memory control. The table below lists the names, entry addresses, and functions of these routines.

Name	Entry address	Function
MEMORYCHK	018DH	Check whether or not memory exists at a specified location (bank and address).
BANKSET	0190H	Change the bank of a specified page.
BANKREAD	0193H	Read the bank number of a specified page.
SLOT1MAP	0196H	Set the mapping of slot 1.
SLOT2MAP	0199H	Set the mapping of slot 2.
BANKJUMP	019CH	Inter-bank jump
BANKCALL	019FH	Inter-bank call
BANKCALL2	0020H	Inter-bank call using RST 20H

F650 to 9
OUT (&31), A
LD A,(DE)
Push AF
LD A,&00
OUT (&31), A
Pop AF
RET. (10)

To Peek Bank 0-7 0000 to 3FFF.
CALL
LD A, Bank no
CALL F65C
LD (F8C0), A
RET.

MEMORYCHK

Entry Address 018DH

Function Check whether or not memory exists at a specified location (bank and address).

Parameter
D = Bank number (00H to 07H)
E = Address: Upper 5 bits (40H to B8H)
The other bits are all "0".

Return
CF = 1 : No memory exists at the specified location.
CF = 0 : Bit 0 of A register = 1 ..... Memory exists.
Bit 1 of A register = 1 ..... The memory is ROM.
= 0 ..... The memory is RAM.

Affected Register AF

BANKSET

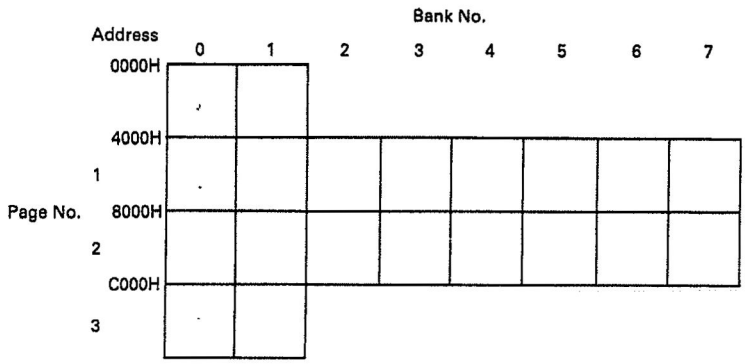
Entry Address 0190H

Function Change the bank number of the page specified by B register to the new bank number specified by A register.

Parameter
B = Page number (01H to 03H)
A = Bank number (00H to 07H)

Return CF = 1 : An invalid page number was specified.

Affected Register AF



BANKREAD

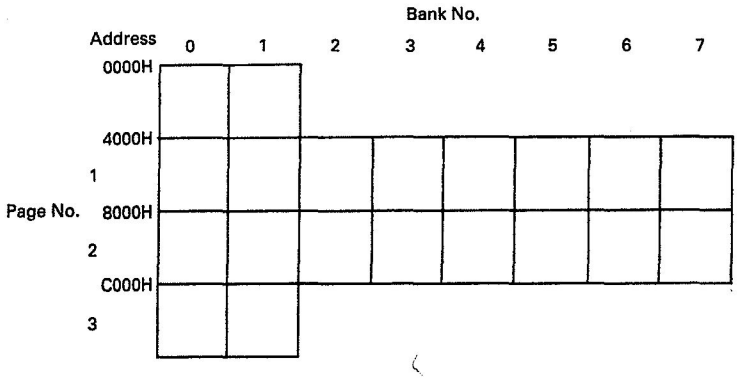
Entry Address 0193H

Function Read the banknumber of the page specified by B register.

Parameter B = Page number (01H to 03H)

Return A = Bank number (00H to 07H)

Affected Register AF, B



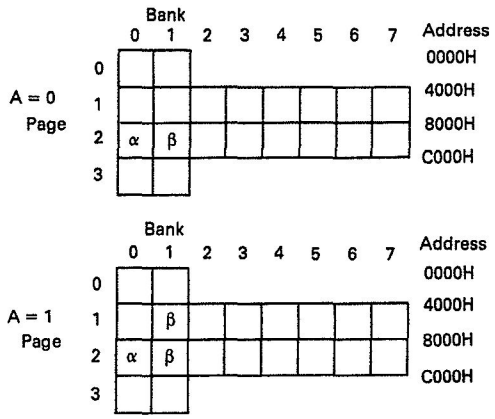
SLOT1MAP

Entry Address 0196H

Function Specify to which bank and page the last half 16 KB of slot 1 is mapped.

Parameter A = 00H : The last half 16 KB of slot 1 is mapped to bank 1, page 2.
A = 01H : The last half 16 KB of slot 1 is mapped to bank 1, page 2 and bank 1, page 1.

The following shows the mapping diagrams of the above two cases. In the diagram, "α" and "β" represent the first half 16 KB and the last half 16 KB of slot 1, respectively.



Return none

Affected Register AF

SLOT2MAP

Entry Address 0199H

Function Specify to which banks and pages the first half 16 KB and the last half 16 KB of slot 1 are mapped.

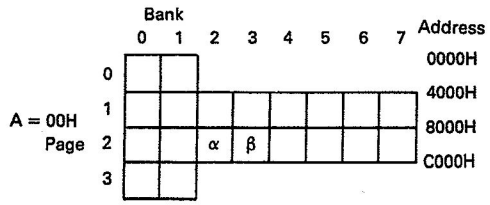
Parameter

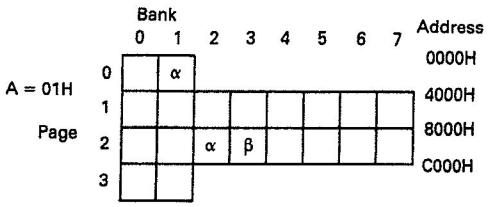
A = 00H : The first half 16 KB of slot 2 is mapped to bank 2, page 2, and the last half 16 KB is mapped to bank 3, page 2.

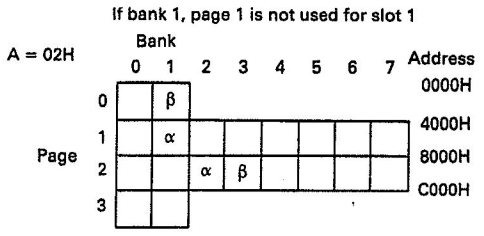
A = 01H : The first half 16 KB of slot 2 is also mapped to bank 1, page 0.

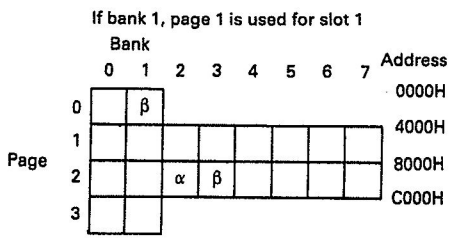
A = 02H : The first half 16 KB of slot 2 is mapped to bank 1, page 1, and the last half 16 KB is mapped to bank 1, page 0. However, if bank 1, page 1 is used for slot 1, the first half 16 KB of slot 2 is not mapped to that location.

The following shows the mapping diagrams of the above three cases. In the diagram, "α" and "β" represent the first half 16 KB and the last half 16 KB of slot 2, respectively.









Return none

Affected Register AF

BANKJUMP

Entry Address 019CH

Function Jump to an address (specified by HL' register) of a bank (specified by A' register).

Parameter A' = Bank number
HL' = Address

Return Control does not return because this is a jump operation.

Affected Register AF', BC', DE', HL'

BANKCALL

Entry Address 019FH

Function Call an address (specified by HL' register) of a bank (specified by A' register).

Parameter A' = Bank number
HL' = Address

Return none

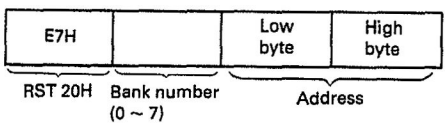
Affected Register AF', BC', DE', HL'

BANKCALL2

Entry Address 20H (To call this routine, use the format: RST 20H.)

Function Same as BANKCALL routine except for the call format

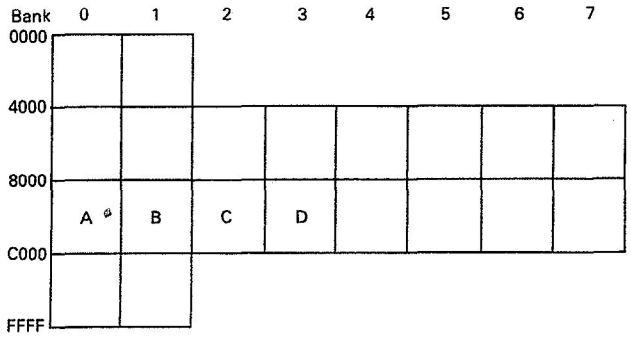
Parameter Write 3 bytes of data (bank number and address) immediately after the RST 20H instruction.



3.13 MEMORY MODULE

3.13.1 Location of Memory Module

A memory module is a ROM or RAM module of a minimum of 4 KB and is used by setting it in slot 1 or 2. The following describes the relationship between the memory capacities, the slot numbers, and the memory locations.



(1) Memory module of 16 KB or less
- Slot 1: The memory module is mapped to location A.
- Slot 2: The memory module is mapped to location C.

(2) Memory module of 32 KB
- Slot 1: The memory module is mapped to locations A and B. .
- Slot 2: The memory module is mapped to locations C and D.

(3) Memory module of 48 KB or more
- Slot 1: Cannot be used.
- Slot 2: The 32 KB part of the memory module is mapped to locations C and D, and the rest of the memory module can be mapped to locations C and D through the bank switching.

3.13.2 Type of Memory Module

The memory modules are classified into four types depending on how they are used.

(1) Extension module (RAM module only)
This module is used to extend the main memory.

(2) Program module (ROM or RAM module)
This module is used to store a program.

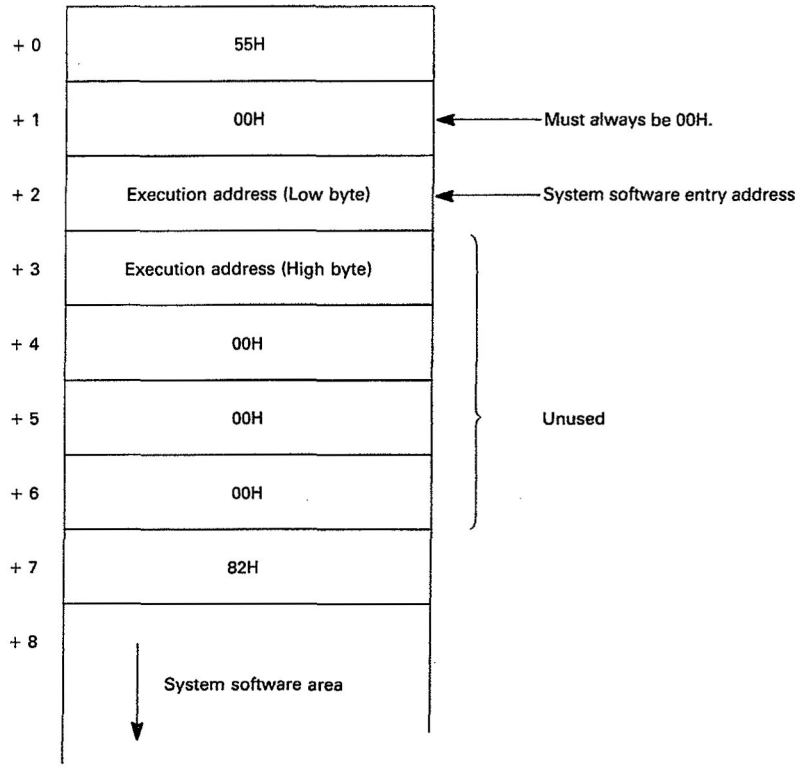
(3) File module (ROM or RAM module)
This module is used to save and load data, like a disk unit.

(4) System software module (ROM or RAM module)
This module is used to store a machine language program that can be automatically runat power-on time.

The type of memory module is identified by the header data (the first 8 bytes of the module). If it is a ROM module, it cannot be used as a different type of module (e.g., a ROM program module can be used only as the program module,) however, a RAM module can be used as a different type of module by specifying it by a BASIC command. For any type of module, only one header is placed in location A or C (see the figure in section 3.13.1).

3.13.3 Header Structure of Memory Module

(1) Header of system software module



CHAPTER 4

BASIC INTERPRETER

4.1 FUNCTIONS HANDLING AND INTERNAL EXPRESSION

4.1.1 Intermediate Codes of Functions

The reserved words of BASIC are developed to 2-byte intermediate codes (internal codes) in memory. The 2-byte intermediate code of a BASIC function has a value between 50H to 7FH in its low byte, depending on the kind of function. The functions are classified as follows:

50H to 51H: Operators
52H to 5FH: System variables
60H to 7FH: Functions

The intermediate code table of the functions is given below.

High byte: E8H

Low byte — Upper 4 bits / Lower 4 bits

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2																
3																
4																
5																
6																
7						DEV$										
8						COM$										
9						INSTAT										
A						INKEY$										
B																
C																
D																
E																
F																

High byte: F0H

Low byte — Upper 4 bits

Lower 4 bits	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1							SPACE$									
2						ERN										
3						ERL										
4																
5																
6																
7																
8																
9																
A																
B																
C																
D																
E																
F																

High byte: F1H

Low byte — Upper 4 bits

Lower 4 bits	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0						AND	ASC	ABS								
1						OR	STR$	INT								
2							VAL	RIGHT$								
3							CHR$	ASN								
4							LEN	ACS								
5							DEG	ATN								
6							DMS	LN								
7							STATUS	LOG								
8						MEM	POINT	EXP								
9								SGN								
A								LEFT$								
B						TIME	SQR	MID$								
C						INKEY$		RND								
D						PI	NOT	SIN								
E							XPEEK#	COS								
F							XPEEK	TAN								

High byte: F2H

Low byte — Upper 4 bits / Lower 4 bits

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0						MOD										
1																
2																
3							HEX$									
4						AIN										
5																
6																
7																
8																
9						DATE$										
A						TIME$										
B																
C							INSTR									
D																
E						ALARM$										
F						WAKE$										

4.1.2 Arithmetic Registers

When you want to use an IOCS routine to perform an arithmetic operation, you must prepare in advance necessary arguments of numeric values or strings in the arithmetic registers provided in the BASIC work area. There are seven arithmetic registers: X, Z, Y, U, V, W, and S. Each register uses 8 bytes of space in the work area. An argument can be set into an arithmetic register (i.e., into the memory location allocated to a particular arithmetic register) in the format described in section 4.1.3.

The following shows the arithmetic registers and their memory locations.

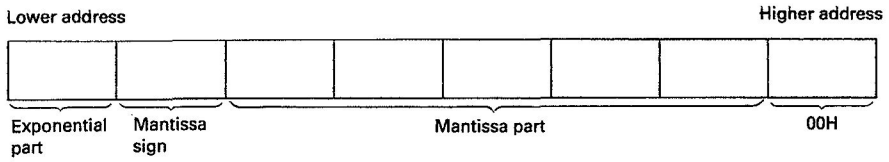
Register	Address
X	FA00H ~ FA07H
Z	FA08H ~ FA0FH
Y	FA10H ~ FA17H
U	FA18H ~ FA1FH
V	FA20H ~ FA27H
W	FA28H ~ FA2FH
S	FA30H ~ FA37H

(These are the addresses viewed from SC-7852 (Z80).)

4.1.3 Internal Expression of Numeric Values and Strings

(1) Decimal (BCD) expression of a numeric value

A numeric value is expressed in an 8-byte format which consists of the exponential part, the mantissa sign, and the mantissa part. This expression format can express a value between $-9.999999999 \times 10^{99}$ and $9.999999999 \times 10^{99}$.



Exponential part: Expressed in one binary byte. (A negative value is expressed as a complement.)

Mantissa sign: 00H ... represents the plus sign.
80H ... represents the minus sign.

Mantissa part: Expressed in BCD.

[Example]

- When setting 123 (or 1.23×10^{2}) in X register

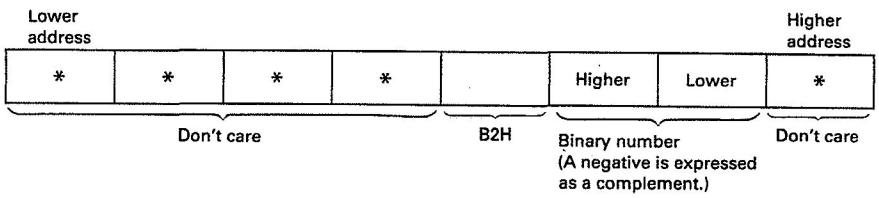
FA00							FA07
02H	00H	12H	30H	00H	00H	00H	00H

- When setting -0.0123 (or -1.23×10^{-2}) in Y register

FA10							FA17
FEH	80H	12H	30H	00H	00H	00H	00H

(2) Binary expression of a numeric value

A numeric value is expressed in the following 8-byte format (although 5 bytes of it are not used.) This expression format can express a value between -32768 and 32767.



[Example]

- When setting 123 (or 007BH) in X register

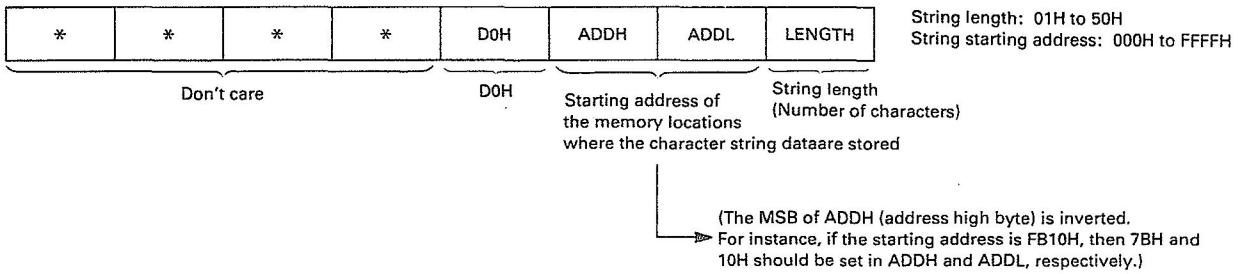
FA00H							FA07H
00H	00H	00H	00H	B2H	00H	7BH	00H

- When setting -123 (or FF85H) in Y register

FA10H							FA17H
00H	00H	00H	00H	B2H	FFH	85H	00H

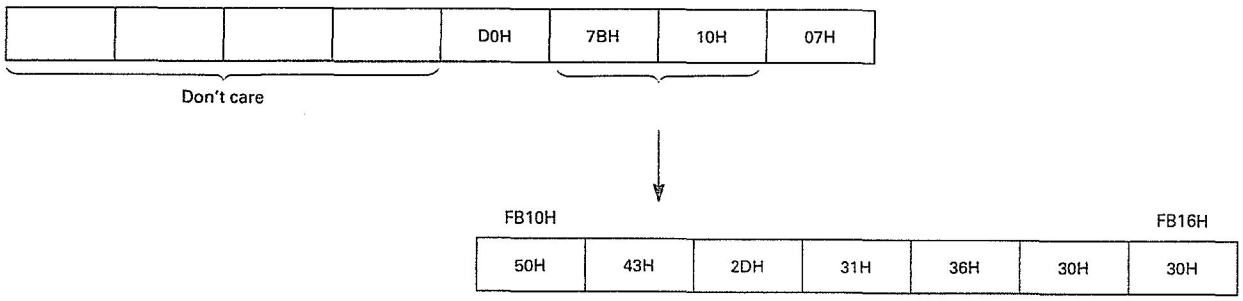
(3) Internal expression of a string

A string of characters are represented by the string information that specifies the memory locations where the actual character string data are stored. This string information is expressed in the following 8-byte format (although its lower 4 bytes are not used.)



[Example]

When the character string "PC-1600" is in the string buffer (FB10H to FB5FH) and you want to set the string information in X register



4.1.4 Function Operation Subroutines

(1) Numericfunctions

(a) One-variable functions (in the case of a function which accepts only one numeric value as the argument)

1. Set the argument (in BCD expression) in X register.
2. Set the intermediate code of the desired function in DE register.
3. Set "01H" in address F88CH (the work area that specifies the number of arguments).
4. Call 0202H (with the CALL instruction of Z80).

When the function subroutine is completed properly, CF is set to "0" and the result is stored in X register. If the subroutine has resulted in an error, CF is set to "1" and the error code (same as used for BASIC) is stored in A register.

Affected registers: HL, DE, BC, AF, AF'

The PC-1600 has the following one-variable functions.

Square root	SQR X → X
Logarithm	LN X → X
	LOG X → X
Exponent	EXP X → X
Trigonometric functions	SIN X → X
	COS X → X
	TAN X → X
Inverse trigonometric functions	ASN X → X
	ACS X → X
	ATN X → X
DMS (degree, minute, second) conversion	DEG X → X
	DMS X → X
Absolute	ABS X → X
Sign	SGN X → X
Integer	INT X → X
Negate	NOT X → X
Random number	RND X → X
Machine language	XPEEK (n) X → X
	PEEK X → X
Machine language	STATUS X → X
Dot pattern	POINT X → X

(b) Two-variable functions

1. Set the arguments (in BCD expression) in X and Y registers. The operand should be set in X register. (For instance, to compute 10 - 9, set "10" in X and "9" in Y.)
2. Set "02H" in address F88CH (the work area that specifies the number of arguments).
3. Call the entry address of the desired function.

Function	Entry address
+	021AH
−	021DH
*	0220H
/	0223H
^	01F3H
AND	01F6H
OR	01F9H

When the function subroutine is completed properly, CF is set to "0" and the result is stored in X register. If the subroutine has resulted in an error, CF is set to "1" and the error code (same as used for BASIC) is stored in A register.

Affected registers: HL, DE, BC, AF, AF'

(c) Relational operations

1. Set the arguments (in BCD expression) in X and Y registers. Theoperand (the value to be compared) should be set in X register. (For instance, to compute 10 < 9, set "10" in X and "9" in Y.)
2. Set "80H" in D register and set the internal code of the desired function (relational operator) in E register.

Operator	Internal code (to be set in E)
< >	00H
<	01H
>	02H
=	04H
< =	05H
> =	06H

3. Set "02H" in address F88CH.
4. Call 01FCH.

If the comparison is true, "1" is stored in Xregister. If it is false, "0" is stored.

Affected registers: HL, DE, BC, AF, AF'

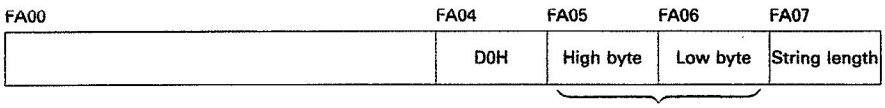
(2) String functions

(a) One-variable function (in the case of a function which accepts only one numeric value as the argument)

1. Set the argument (in either BCD or binary expression) in X register.
2. Set the intermediate code of the desired function in DE register.
3. Set "01H" in address F88CH and "10H" in address F894H (the string buffer pointer).
4. Call 0202H.

When the function subroutine is completed properly, CF is set to "0" and the result (the string information expressed in the following internal format) is stored in X register.

X register



Starting address of the memory locations in which the resultant character data are stored
The MSB of the high byte of the starting address is inverted.
(Example) If the following result is stored in X register:
FA04H = D0H
FA05H = 7BH (interpreted as FBH)
FA06H = 10H
FA07H = 10H
this string information indicates a string of 16 (=10H) characters stored in the memory locations starting from FB10H.

The PC-1600 has the only one string function of this kind: STR$.
Affected registers: HL, DE, BC, AF, AF'

(b) One-variable functions (in the case of a function which accepts only one string data item as the argument and gives a numeric value as the result.)

1. Set the argument (string information) in X register. The actual string data must be prepared in the string buffer (the area starting from FB10H).
2. Set the intermediate code of the desired function in DE register.
3. Set "01H" in address F88CH.
4. Call 0202H (with the CALL instruction of Z80).

When the function subroutine is completed properly, CF is set to "0" and the result (expressed in either BCD or binary expression) is stored in X register. If the subroutine has resulted in an error, CF is set to "1" and the error code (same as used for BASIC) is stored in A register.
Affected registers: HL, DE, BC, AF, AF'
The PC-1600 has three functions of this kind: VAL, ASC and LEN.

(c) Two-variable functions ... RIGHT$(string, numeric value) and LEFT$(string, numeric value)

1. Check whether there is a free space of 8 bytes in the BASIC stack area (FA38H to FAFFH). This function subroutine can be executed if the following relation is satisfied:
   (Content of F890H) < (Content of F891H) − 8
2. Set the string information into addresses from (content of F890H)+4 to (content of F890H)+7:

Address	Data
(Content of F890H)+4	D0H
(Content of F890H)+5	Starting address of the memory locations where the actual string data are stored (High byte)
(Content of F890H)+6	Starting address of the memory locations where the actual string data are stored (Low byte)
(Content of F890H)+7	String length

and set the actual string data in the string buffer (FB10H to FB5FH). When setting the high byte of the string starting address into address (content of F890H)+5, invert the MSB of the high byte.

For example, when a string of eight characters are already stored in the string buffer starting from its top, set the following string information:

Address	Data
(Content of F890H)+4	D0H
(Content of F890H)+5	7BH
(Content of F890H)+6	10H
(Content of F890H)+7	08H

3. Set "(content of F890H)+8" into address F892H (the data pointer).
4. Set the numeric argument (in either BCD or binary expression) in X register.
5. Set the intermediate code of the desired function in DE register.
6. Call 0202H (with CALL instruction of Z80).

When the function subroutine is completed properly, CF is set to "0" and the result (the string information expressed in the internal format) is stored in X register. The actual resultant character data are stored in the string buffer.

If the subroutine has resulted in an error, CF is set to "1" and the error code is stored in A register.

Affected registers: HL, DE, BC, AF, AF'

(d) Three-variable function ... MID$(string, numeric1, numeric2)

1. Check whether there is a free space of 16 bytes in the BASIC stack area (FA38H to FAFFH). This function subroutine can be executed if the following relation is satisfied:

   (Content of F890H) < (Content of F891H) − 16

2. Set the string information into addresses from (content of F890H)+4 to (content of F890H)+7:

Address	Data
(Content of F890H)+4	D0H
(Content of F890H)+5	Starting address of the memory locations where the actual string data are stored (High byte)
(Content of F890H)+6	Starting address of the memory locations where the actual string data are stored (Low byte)
(Content of F890H)+7	String length

and set the actual string data in the string buffer (FB10H to FB5FH). When setting the high byte of the string starting address intoaddress (content of F890H)+5, invert the MSB of the high byte.

For example, when a string of eight characters are already stored in the string buffer starting from its top, set the following string information:

Address	Data
(Content of F890H)+4	D0H
(Content of F890H)+5	7BH
(Content of F890H)+6	10H
(Content of F890H)+7	08H

3. Set the third argument numeric2 (expressed in either BCD or binary expression) into addresses from (content of F890H)+8 to (content of F890H)+15.
4. Set "(content of F890H)+16" into address F892H (the data pointer).
5. Set the second argument numeric1 (in either BCD or binary expression)in X register.
6. Set the intermediate code of the desired function (i.e., F17BH in the case of MID$ function) in DE register.
7. Call 0202H (with CALL instruction of Z80).

When the function subroutine is completed properly, CF is set to "0" and the result (the string information expressed in the internal format) is stored in X register. The actual resultant character data are stored in the string buffer.

If the subroutine has resulted in an error, CF is set to "1" and the error code is stored in A register.

Affected registers: HL, DE, BC, AF, AF'

(e) Relational operations (string1, operator, string2)

1. Set string1 and string2 in X and Y registers, respectively. These string arguments must be expressed as the string information in the internal format.
2. Set the internal code of the desired function (relational operator) in DE register.
   D = 80H

Operator	Value to be set in E
< >	D0H
<	01H
>	02H
=	04H

3. Call 01FFH (with CALL instruction of Z80).

If the comparison is true, "1" is stored in X register. If it is false, "0" is stored.

Affected registers: HL, DE, BC, AF, AF'

4.2 BASIC PROGRAM TEXT HANDLING

4.2.1 Subroutines for Numeric Value Handling

This section describes the subroutines to convert between the internal decimal (BCD) codes and the internal binary codes. In the following explanations, the content of X register is an address for Z80, specifying a memory location between FA00H and FA07H. For the format of data written in X register, refer to section 4.1.3 "Internal Expression of Numeric Values and Strings".

BCDBIN

Entry Address 0247H

Function Convert a numeric value (between 0 and 65535) expressed in an 8-byte BCD code into a 2-byte binary code, with the fractional part rounded off.

Parameter X = Numeric value expressed in the BCD code

Return
CF = 0: Normal termination
DE = Conversion result (If the result is "0", then ZF is set to "1".)
CF = 1: An error has occurred. (The error code is returned in A register.)
A = 13H: The value of the operand is out of the range.
A = 07H: The value of the operand is not expressed in the BCD code.

Affected Register AF, BC, DE, HL

BCBINS

Entry Address 0241H

Function Convert a numeric value (between −32768 and 32767) expressed in an 8-byte BCD code into a 2-byte binary code (with a negative value expressed as a complement). The fractional part is rounded off.

Parameter X = Numeric value expressed in the BCD code

Return
CF = 0: Normal termination
DE = Conversion result (If the result is "0", then ZF is set to "1".)
CF = 1: An error has occurred.

Affected Register All registers

BI2BCD

Entry Address 024AH

Function Convert a numeric value expressed in a 2-byte binary code into a BCD code.

Parameter DE = Numeric value expressed in the binary code

Return X = Conversion result (BCD code)

Affected Register All registers

4.2.2 Subroutines for ASCII Code Conversion

HTOA

Entry Address 0283H

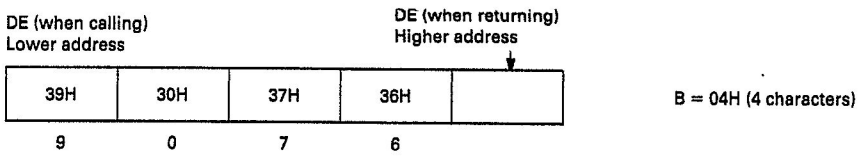
Function Convert a numeric value expressed in a 2-byte binary code into an ASCII string.

Parameter HL = Binary number to be converted (0000H to FFFFH)
DE = Starting address of the memory locations to which the converted ASCII string is stored

Return DE = Address in which the last character is stored
B = Number of characters

Affected Register HL, DE, BC, IX, AF

Remarks (Example) When HL = 2374H (or 9076)



BCDASC

Entry Address 0244H

Function Convert a numeric value expressed in a BCD code into an ASCII string. At the end of the converted ASCII string is always added 00H. If the absolute value of the exponent part is 10 or greater, the converted result is expressed in the exponential notation (like 1.23×10^{10}). If it is less than 10, the converted result is expressed in the fixed decimal notation (like 123). A plus sign, if any, is converted to a space character.

Parameter HL = Starting address of the memory locations in which the operand (a numeric value expressed in a BCD code) is stored
DE = Starting address of the memory locations to which the converted ASCII string is stored

Return The converted string is stored in the memory location specified by DE.

Affected Register AF, BC, DE, HL, IX, IY, AF'

4.2.3 Subroutines for Evaluation of Expressions

EXPRESS

Entry Address 0274H

Function Evaluate the expression (in the intermediate code format) stored in the memory location specified by HL register, and return the result (in the BCD format) into X register.

Parameter HL = Text read address

Return CF = 0: Normal termination
X = Result
CF = 1: An error has occurred.
A = Error code

Affected Register AF, BC, DE, HL, IX, IY

EXPREX

Entry Address 0277H

Function Same function as EXPRES, except that this routine does not result in an error even if the expression to be evaluated does not have a left parenthesis which should correspond to the right-most parenthesis.

Parameter HL = Text read address

Return
CF = 0: Normal termination
X = Result
CF = 1: An error has occurred.
A = Error code

Affected Register AF, BC, DE, HL, IX, IY

SUSING

Entry Address 0298H

Function Interpret the contents of the memory locations whose starting address is specified by HL register as the USING format data, and store the format data in the work area.

Parameter HL = Starting address of the memory locations in which the USING format data are stored

Return
CF = 0: Normal termination
HL = (Last address of the memory locations where the USING format data are stored) + 1
CF = 1: An error has occurred.
(Example)

```
---- USING "###   ##"; A ----
```

The starting address of the USING format data which should be specified by HL register

When returning from the routine, HL register specifies the address of this point.

Affected Register AF, HL

USGCNT

Entry Address 029EH

Function Convert the numeric value (in the internal format) in X register into the ASCII string formatted according to the USING format data given by SUSING routine, and store the ASCII codes into Y, U, V, W and S registers (i.e., locations from FA10H to FA37H).
If no USING format data has been given, the numeric value is converted in the same manner as BCDASC routine (i.e., converted simply to an ASCII string) and the ASCII codes are stored in Y, U, V, W and S register.

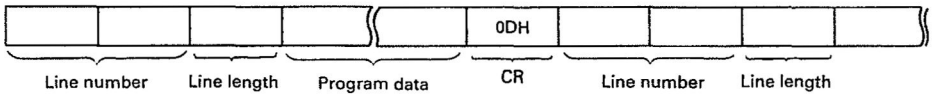
Parameter none

Return CF = 0: Normal termination
A = Number of the resultant ASCII haracters (excluding the 00H)
CF = 1: An error has occurred.

Affected Register AF, AF', BC

4.2.4 Subroutines for BASIC Text

Structure of BASIC program text
Each command line of BASIC program consists of a line number, line length, and program data. The BASIC reserved words are expressed in intermediate codes and the other program data are expressed in ASCII codes. For the intermediate codes of BASIC reserved words, see section 4.2.5 "Intermediate Code Table".




```
10 PRINT A
20 END
```

The program shown right is expanded in the memory as follows:

Address	Data	
C0C5	00	} 10
C0C6	0A	
C0C7	04	--Line length
C0C8	F0	} PRINT
C0C9	97	
C0CA	41	--A
C0CB	0D	--CR
C0CC	00	} 20
C0CD	14	
C0CE	03	--Line length
C0CF	F1	} END
C0D0	8E	
C0D1	0D	--CR
C0D2	FF	--Code indicating the end of a BASIC program

GETCD1

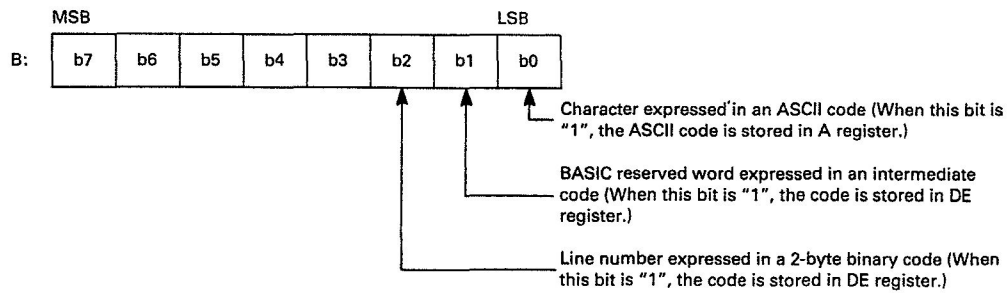
Entry Address 027DH

Function Read the attribute (such as line number or reserved word) of the content of a memory location (specified by HL register) within the BASIC program text area.

Parameter HL = Text read address

Return B = Attribute

The respective bits of the attribute value in B register have the following meanings (these bits are significant only if they are "1".)



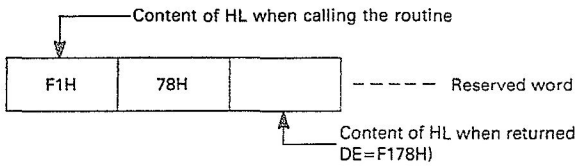
The PC-1600 uses two internal formats for line numbers:

	No. of bytes	Example: Internal expression of "GOTO 100"
ASCII format	Variable-length	F1 92 31 30 30
Binary format	4 bytes	F1 92 1F 00 64 00

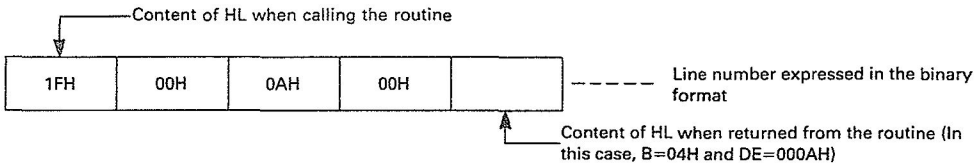
GOTO

In the binary format, "1F" is the header, the next "0064" is the binary expression of decimal number "100", and the last "00" is a dummy.
In the PC-1600 mode, usually the binary format is used.
When returned from the routine, the content of HL register specifies the address of the block next to the currently referenced block (line number, reserved word, etc.) This address is also stored in the work area (TEXTRP).
(Example)

1)



2)



CF = 1: The end-of-text code (FFH) was detected.

Affected Register AF, B, DE, HL

EOSCHK

Entry Address 026EH

Function Check the end-of-statement code (colon or CR code) based on the bit pattern set in B register by GETCD1 routine.

Parameter B = Bit pattern set by GETCD1
A = ASCII code

Return ZF = 1: The end-of-statement code was detected.
ZF = 0: The end-of-statement code was not detected.

Affected Register All registers

DATSKP

Entry Address 0262H

Function Skip from the specified text read pointer to the next end-of-line code (CR, or "0DH"). The text read pointer to be specified must not at the 00H code of a binary-expressed line number or in the middle of a line number or string constant.

Parameter HL = Text read pointer

Return HL = Address of the location where the end-of-line code is.

Affected Register AF, HL

LINSRH

Entry Address 028CH

Function Search for the line of the line number specified by DE register.

Parameter DE = Line number

Return

CF = 0: The line of the specified or greater line number has been found. In this case,
SRLINE (F8A8H: high byte, F8A9H: low byte) stores the line number found.
SRADR (F8A6H: high byte with MSB inverted, F8A7H: low byte) stores the starting address of the memory locations where the line is stored. (For example, if the starting address is 80C5H, then "00H" is stored in F8A6H and "C5H" is stored in F8A7H.)
SRADRb (F1C3H) stores the logical bank number.

CF = 1: The line of the specified or greater line number has not been found. In this case SRLINE, SRADR and SRADRb store the line information of the greatest among the existing line numbers.

Affected Register AF

NXTADR

Entry Address 02EEH

Function Return the starting address and bank number of the line next to the one that specified by HL register.

Parameter
HL = Starting address of a line
A = Logical bank number of that line
(FE2AH, FE2BH) = Address of the last line in the area to be searched for (Specify the address in the order of the low byte and the high byte)
(FE2CH) = Logical bank number of the last line

Return
CF = 0: The next line was found within the specified area.
HL = Starting address of the next line
A = Logical bank number of the next line
CF = 1: The next line was not found within the specified area.

Affected Register AF, AF', HL, DE

TADCNV

Entry Address 02D4H

Function Convert a logical bank number to a physical bank number.

Parameter
C = Logical bank number
DE = Address

Return
CF = 0: Normal termination
A = Physical bank number
DE = Address
CF = 1: An error has occurred (e.g., a non-existing address was specified.)

Affected Register AF

LBSCH2

Entry Address 02D1H

Function Return the line number of the line that contains the label specified by HL and BC.

Parameter

HL = Address of the first character in the label
(Example) "START"
↑
HL

BC = Number of characters in the label excluding the double quotes (")
(Example) "START" - - - BC=0005H
" " - - - BC=0000H

Return

CF = 0: Normal termination
STLINE (F8A8H: high byte, F8A9H: low byte) stores the line number found.
SRADR (F8A6H: high byte with MSB inverted, F8A7H: low byte) stores the starting address of the line found. (For example, if the starting address is 80C5H, then "00H" is stored in F8A6H and "C5H" is stored in F8A7H.)
SRTOP (F8AAH: high byte, F8ABH: low byte) stores the same contents as SRADR.

CF = 1: The specified label has not been found.

Affected Register

AF, BC, DE, HL, AF'

Remarks

Because this routine uses the bank switching, the calling side may need to switch back to the original bank.

4.2.5 Intermediate Code Table

BASIC Reserved Word Intermediate Code Table

High byte: E3

Upper 4 bits

Low byte

Lower 4 bits

Low \ High	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1									PAPER							
2																
3																
4																
5																
6																
7																
8																
9																
A																
B																
C																
D																
E																
F																

High byte: E6

Upper 4 bits

Low byte

Lower 4 bits

Low \ High	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0									CSIZE							
1									GRAPH							
2									GLCURSOR							
3									LCURSOR							
4									SORGN							
5									ROTATE							
6									TEXT							
7																
8																
9																
A																
B																
C																
D																
E																
F																

High byte: E7

Upper 4 bits

Low byte

Lower 4 bits

Low \ High	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2																
3																
4																
5																
6																
7																
8																
9											PMT					
A																
B																
C																
D																
E																
F																

High byte: E8

Upper 4 bits

Low byte

Lower 4 bits

Low \ High	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0									OUTSTAT							
1																
2									SETCOM							
3									TERMINAL							
4									DTE							
5									TRANSMIT							
6									SETDEV							
7						DEV$										
8						COM$										
9						INSTAT										
A						RINKEY$										
B																
C																
D																
E																
F																

High byte: F0

Upper 4 bits

Low byte

Lower 4 bits

Low \ High	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0										LIST	LFILES	FEED				
1							SPACE$			INPUT		CONSOLE				
2						ERN				GCURSOR		CHAIN				
3						ERL						BREAK				
4									CURSOR		PITCH	ZONE				
5									USING	CSAVE	LCURSOR	COLOR				
6												LF				
7									WIDTH	PRINT		LLINE				
8									CLS	FILES		LLIST				
9									CLOAD	LINE		LPRINT				
A										PRESET		RLINE				
B										PSET		TAB				
C												TEST				
D																
E																
F									MERGE	GPRINT						

BASIC INTERPRETER

High byte: F1

Upper 4 bits

Low byte / Lower 4 bits

Low \ High	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0						AND	ASC	ABS	AREAD		XPOKE#	TROFF				
1						OR	STR$	INT	ARUN		XPOKE	TO				
2							VAL	RIGHT$	BEEP	GOTO	PAUSE					
3							CHR$	ASN	CONT			WAIT				
4							LEN	ACS		GOSUB	RUN	ERROR				
5							DEG	ATN			FOR	LOCK				
6							DMS	LN	GRAD	IF	READ	UNLOCK				
7							STATUS	LOG	CLEAR		RESTORE					
8						MEM	POINT	EXP		LET	RANDOM					
9								SGN		RETURN						
A								LEFT$	XCALL	NEXT	RADIAN					
B						TIME	SQR	MID$	DIM	NEW	REM					
C						INKEY$		RND	DEGREE	ON	STOP					
D						PI	NOT	SIN	DATA	OPN	STEP					
E							XPEEK#	COS	END	OFF	THEN					
F							XPEEK	TAN			TRON					

High byte: F2

Upper 4 bits

Low byte / Lower 4 bits

Low \ High	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0						MOD			ADIN	BLOAD	PHONE					
1						XOR	WAKE$	EOF		BSAVE	SNDBRK	PCONSOLE				
2								LOC	CALL	CLOSE	SNDSTAT					
3								LOF	ELSE	COPY	COM	MODE				
4								DSKF	KBUFF$	INIT	RCV STAT	PZONE				
5							HEX$		KEY	LOAD		RENUM				
6						RXD$	INP		KEY STAT	OPEN		AUTO				
7						DATE$	INSTR		KILL	NAME		ERASE				
8						TIME$			MAX FILES	SET		PASS				
9										SAVE		DELETE				
A						AIN			OUT			TITLE				
B									POWER							
C						ALARM$			POKE			AOFF				
D							PEEK		RESUME			AS				
E							PEEK#		RETI			OUTPUT				
F												APPEND				

Functions and symbols	Intermediate code
,	60 2CH
AND	70 50H
OR	70 51H
< >	80 00H
<	80 01H
>	80 02H
=	80 04H
< =	80 05H
> =	80 06H
+	81 2BH
−	81 2DH
MOD	82 55H
\	83 5CH
*	84 2AH
/	84 2FH
+ (sign)	85 2BH
− (sign)	85 2BH
^	86 5EH

Functions and symbols	Intermediate code
√	F1 6BH
Function name	FX XXH
(	20 28H
Variable name	40 XXH ~ 5A XXH

Functions and symbols	Intermediate code
)	10 29H
End-of-statement (CR, colon, etc.)	00 XXH

CHAPTER 5

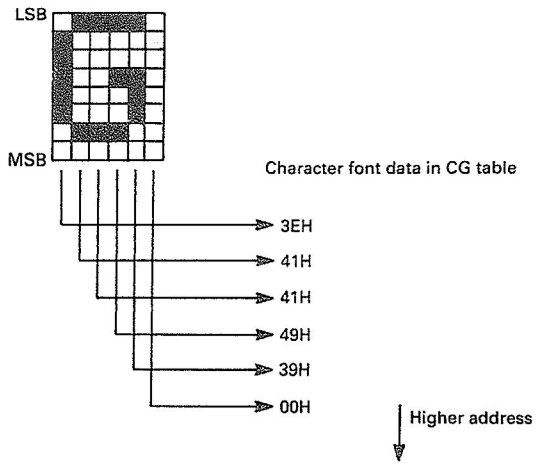
OTHER FUNCTIONS AND PRECAUTIONS

5.1 AUTOMATIC LOADING AND RUNNING OF BASIC PROGRAM FILE (AUTORUN.BAS)

The PC-1600 can automatically load and run a specific BASIC program file (named "AUTORUN.BAS") from the file device (CE-1600F or memory file).
When the PC-1600 is powered on, if it is in the RUN mode and the S symbol is not on, the PC1600 searches for a file named "AUTORUN.BAS" in the CE-1600F, the slot S1 memory file and the slot S2 memory file, in that order. If the file exists, it is automatically loaded into the currently selected memory module (program module or extension memory module) and executed.

5.2 CHANGING DISPLAY CHARACTER FONT

In the PC-1600, a display character is composed of a dot matrix of 6 columns by 8 lines. The font data of each character is stored in the CG (character generator) table, in which one 8-dot column data of a character isexpressed in one byte (or 8 bits) and the six one-byte column data forming one complete character are stored in sequential memory locations with the data of the first column of that character stored first, as shown below.



The PC-1600 has three CG tables:

1. GG table that stores the font data for the character codes of from 20H to 7FH
   This CG table is in the ROM of the main unit, and the fonts of these characters cannot be changed by the user.
2. CG table that stores the font data for the character codes of from 80H to FFH
   This CG table is in the ROM of the main unit, and the fonts of these characters can be changed by the user by preparing a user defined font set in RAM and changing the contents of UPACGA and UPACGB.
3. CG table that stores the font data for the character codes of from 00H to 1FH
   This CG table is not in the ROM of the main unit. If you want to define your own character fonts to these codes of 00H to 1FH, change the contents of CTRCGA and CTRCGB, prepare a user defined font set in RAM, and set LCDWK (F05DH) bit 3 to "1".

As described, the CG tables of items 2 and 3 above can be replaced with user-defined CG table prepared in memory. In this case, since the character fonts of all character codes allocated to that table are to be changed, you must prepare fontdata for all these character codes even if you want to change the fonts for only some of the character codes.

The following shows the details of the work area used for the CG tables.

Name	Address	Bytes	Contents
CTRCGA	F061H (Low byte) F062H (High byte)	2	Starting address of the memory locations where the character font data for the character codes of 00H to 1FH are stored (The starting address must not be lower than 8000H.)
CTRCGB	F063H	1	Bank number (0 to 7) of the memory locations where the character font data for the character codes of 00H to 1FH are stored
UPACGA	F064H (Low byte) F065H (High byte)	2	Starting address of the memory locations where the character font data for the character codes of 80H to FFH are stored (The starting address must not be lower than 8000H.)
UPACGB	F066H	1	Bank number (0 to 7) of the memory locations where the character font data for the character codes of 80H to FFH are stored

5.3 EXTENDED FUNCTION OF KEYSTAT COMMAND

The PC-1600 can accept key input from the serial port instead of the main unit keyboard.

(1) Setting procedure

Execution of KEYSTAT command with the first parameter set to "2" lets the PC-1600 to accept key input from the serial port currently set by SETDEV command. To do this, set appropriate communication parameters (such as the baud rate) by SETCOM command. The use of the second and the third parameters of KEYSTAT command is the same as for the keyboard.

(2) Key codes

The key codes to be input from the serial port must be the same as those sent from the keyboard (see section 10.2.) Code 10H has a special meaning:

10H: Indicates that the [↓], [↑] or [OFF] key has been released. Since the key input routine needs to check whether or not the [↓], [↑] or [OFF] key has been currently pressed, a code 10H must be sent each time one of these keys is released.

With the keyboard, key codes 11H and 13H are allocated to the function keys 1 and 3 on the keyboard, respectively. With the serial port, however, codes 11H and 13H are usually used as XON and XOFF codes, respectively. Because of this, usually you cannot send the key codes corresponding to the function keys 1 and 3 through the serial port. However, execution of the following POKE statements lets the PC-1600 to interpret codes F1H and F3H sent from the serial port as the key codes of the function keys 1 and 3, respectively.

```
POKE &FF40,&33,&33,&F5,&79,&FE,&07,&28,&02
POKE &FF48, &F1,&C9,&F1,&E3,&CD,&61,&FF,&67,&F5
POKE &FF51,&FE,&F1,&20,&02,&26,&11,&FE,&F3
POKE &FF59,&20,&02,&26,&13,&F1,&7C,&E1,&C9,&E9
POKE &F3C1,&C3,&40,&FF
```

This example uses codes F1H and F3H for the function keys 1 and 3, however, you can use different codes instead of F1H and F3H by changing the contents of addresses FF52H and FF58H.

FF52H: F1H ← Change to a different value.
FF58H: F3H ← Change to a different value.

(3) Precaution

Even when "KEYSTAT 2" statement is executed to accept key input from the serial port, the key input from the keyboard can still be accepted: the PC-1600 accepts key input from both keyboard and serial port.

5.4 SEGMENTING ONE RAM MODULE FOR DIFFERENT USES

A RAM module (CE-1600M or CE-161) is usually used for only one of the following three uses:

(1) extension memory
(2) program module
(3) RAM disk

However, a RAM module (if two RAM modules are installed in slots 1 and 2, only one of them) can also be used as:

(4) program module + extension memory

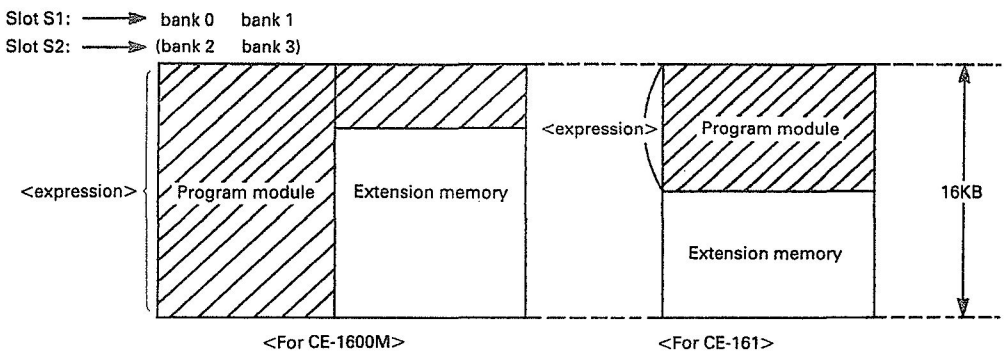
In the fourth case, CE-1600M or CE-161 is segmented into two parts. The first part of the RAM module is used as a program module and the last part as the extension memory. To do this, execute the following statement:

INIT "S1:(or S2:)","P",<expression>

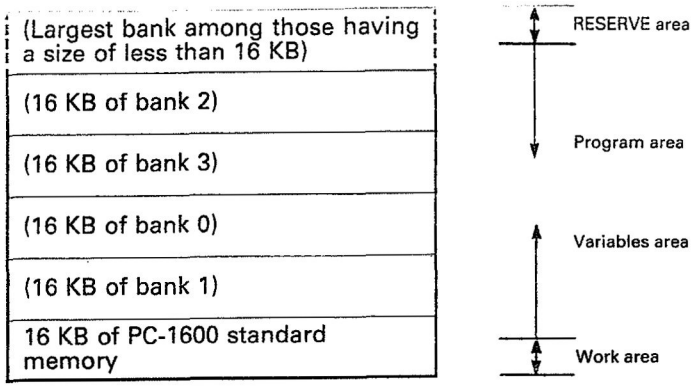
where <expression> specifies the size of the area to be used as the program module. The program module area can have a size of a multiple of 2 KB, and <expression> should be specified by a value of a multiple of 2 in kilobytes.

This module segmentation is possible only when the following four conditions are satisfied:
(1) No program exists in the extension memory.
(2) No program exists in the program module.
(3) No files exist in the RAM disk.
(4) The other module, if there are two modules in the slots, is not segmented.

<Segmentation example>



* When a RAM module in a slot is segmented, if there is another RAM module in the other slot and it is used as the extension memory, the total size of the extension memory varies as follows.



(1) First, the extension memory areas are stacked up in the order of banks 1, 0, 3 and 2. However, if the extension memory area of a bank is less than 16 KB, that bank is not stacked.
(2) After those banks of 16 KB are stacked, finally the largest bank among those having a size of less than 16 KB, if any, is stacked. The other banks of less than 16 KB are not used as extension memory. (If there are two or more banks that have the same size of less than 16 KB, only one of those banks is stacked being selected in the order of banks 1, 0, 3 and 2.)

5.5 FILE FORMAT

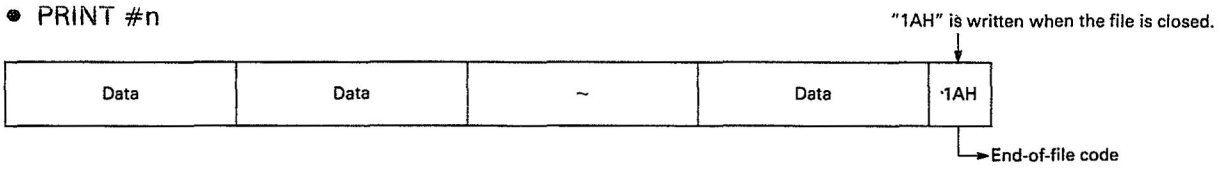
The PC-1600 can handle three kinds of files:

(1) ASCII file : Program file or data file saved by PRINT #n, SAVE with A option, or SAVE* command
(2) Program file : Program file saved by SAVE command
(3) Machine language file : Machine language file saved by BSAVE command

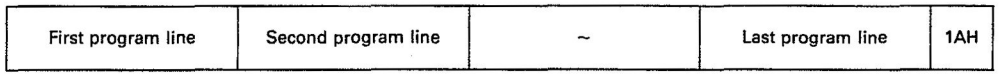
These files have the following formats:

(1) ASCII file

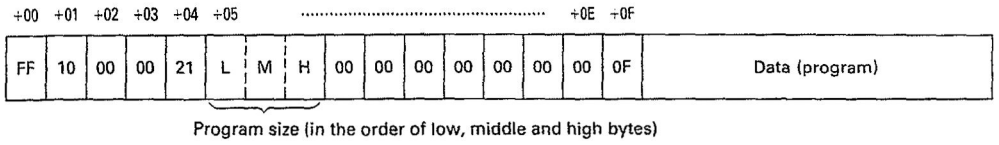
- PRINT #n



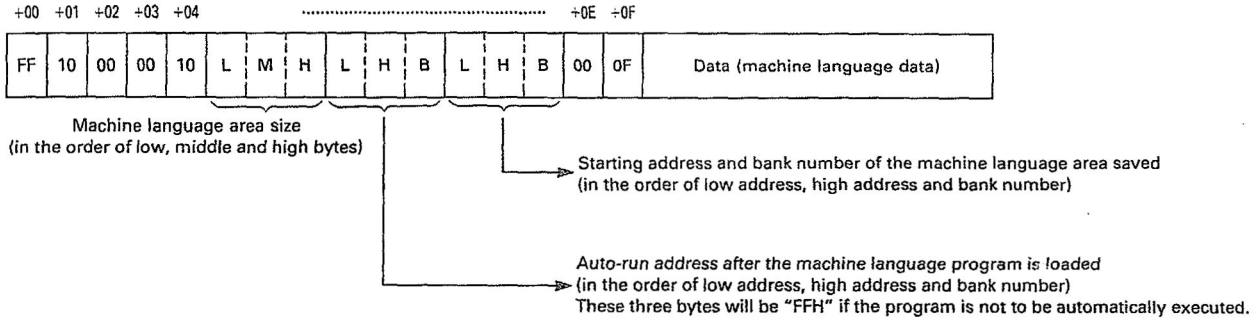
- SAVE with A option, and SAVE*



(2) Program file



(3) Machine language file



5.6 DATA INPUT/OUTPUT TO FILE DEVICE

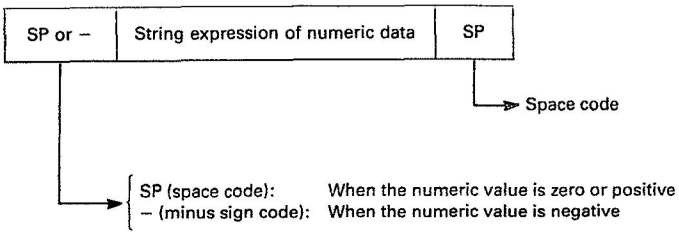
(1) There are two commands to for data input/output to a file device:
PRINT #n ..... Write data to a file device.
INPUT #n ..... Read data from a file device into a variable.

(2) The following outlines the procedure for data input/output to a file device.
1. MAXFILES=m ..... m is the number of files that can be opened at the same time. (m is 1 to 15.)
Usually this command is placed at the beginning of a program.
2. OPEN "<device name><file name>" FOR { INPUT / OUTPUT / APPEND } AS #n
..... n must be a unique number among the currently opened files.
3. PRINT #n ..... Write data.
or
INPUT #n ..... Read data into a variable.
4. CLOSE #n ..... When the file operation is completed, the file must be closed.

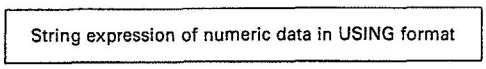
(3) End of file
While you are reading data from a file by INPUT #n command, if all data in the file have already been read and you attempt to read data from the file, this execution of INPUT #n command results in an error.
You can check whether all data in the file have been read, by using EOF function. Thus, when you attempt to sequentially read data from a file that you do not know how many data items exist in that file, you should check the end of the file by EOF function each time before you execute INPUT #n command.

1. When writing data by PRINT #n command

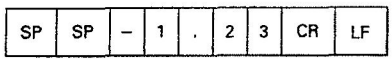
(1) ①Numeric data (without USING)



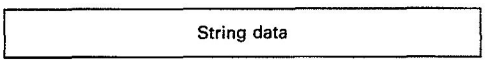
②Numeric data (with USING)



(Example) PRINT#1, USING "+####.##"; −1.234



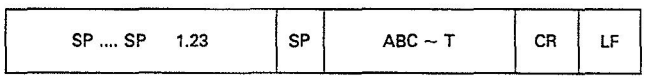
③String data



If a USING format is given before the string, the string data are arranged according to the USING format.

(2) PRINT #n,data1,data2 (when the data delimiter is a comma)
When each data items are separated by a comma, each data item is written in units of 20 bytes. If a data item exceeds 20 bytes, it is written over to the subsequent 20 bytes. A numeric data is right-justified in a block.

(Example) PRINT#1, 1.23, "ABC ~ T"



(3) PRINT #n,data1;data2 (when the data delimiter is a semicolon)
When each data items are separated by a semicolon, each data item is written in the format of (1) above, and the data items are written immediately following one another.

(4) PRINT #n,data1
When the last data item is not followed by a comma or semicolon, the data item is written with CR and LF codes added at the end.

2. When reading data by INPUT #n command

(1) INPUT #n,<numeric variable>

This statement read data as a numeric value from a file device and stores it to the specified numeric variable. If a data item cannot be read as a numeric value, "0" is stored to the variable. A space code and a data delimiter code preceding to each data item are ignored. The end of each data item is recognized by a comma, space, or CR+LF.

(2) INPUT #n,<string variable>

This statement read data as a string from a file device and stores it to the specified string variable. Consecutive space codes or delimiter codes preceding to each data item are ignored. The end of each data item is recognized by a comma or CR+LF. A comma enclosed with double quotes is recognized as part of data (that is, not as a data delimiter).

3. Examples of PRINT #n and INPUT #n commands

Example 1

When you execute

PRINT #1,A$,B$ (where A$="ABC" and B$="CD")

the data are written in the following format:

A B C sp ..... sp	CD	CR	LF

Then, when you execute the following statement to read data from the previous file:

INPUT #2,A$(0) (assume DIM A$(0)*80 has been executed)

the data are stored to A$(0) as follows:

A$(0) = "ABC sp ..... sp CD"

Example 2

When you execute the following statement to read the same data as Example 1:

INPUT #2,A$,B$

the data are read into A$ as follows:

A$= "ABC sp ..... sp"

(16 bytes of data)

then the INPUT statement results in an error because there is no data to be read to B$. In this case, the data "ABC sp ..... sp CD" are treated as one data item, and only the first 16 bytes of data are stored into A$, therefore, no data is left for B$.

Example 3
Suppose you have executed the following statement:

```
PRINT #1,1.23,456
```

Now, you read these recorded data by the following statement:

```
INPUT #2,A,B
```

Then you get the results:

A=1.23 and
B=456

This is because, different from examples 1 and 2, a space code is treated as a data delimiter if the input variable is a numeric variable.

4. PRINT #n and INPUT #n commands to serial port (RS-232C or SIO)

(1) The data format is the same as item 1 above.
(2) With PRINT #n command, the data are actually output to the serial port when one of the following events occurs:
- When the amount of output data reaches 256 bytes
- When the serial port is closed

(3) With INPUT #n command, the received data are actually stored to the variable when one of the following events occurs:
- When the amount of received data reaches 256 bytes
- When an EOF code (1AH) is received

5.7 PRECAUTIONS FOR USE OF SERIAL PORT (RS-232C AND SIO)

(1) Transmission speed

For communication at higher transmission speed, set the size of the receive buffer as follows (by using INIT "COMn:" statement.

Transmission speed (bps)	Receive buffer size (bytes)
4800 to 9600	80 or more
19200	130 or more
38400	1100 or more

(2) XON/XOFF flow control

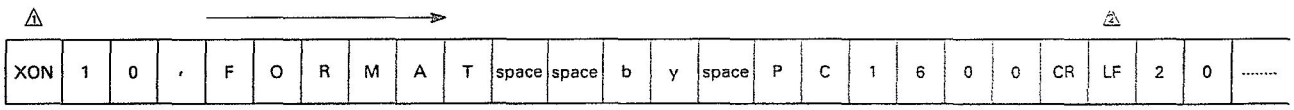
When XON/XOFF control is used, execution of any of the following commands to a serial port may cause an XON (11H) code to be generated at the beginning of the command execution.

OPEN "COMn:" SAVE BSAVE LOAD
LLIST INPUT BLOAD LPRINT

The following shows an example in which an XON (11H) code is not generated when the OPEN "COMn:" statement is executed.

```
SETCOM "COMn:",1200,8,N,1,N,N ... (XON/XOFF is disabled)
OPEN "COMn:" FOR INPUT AS #1
SETCOM "COMn:",1200,8,N,1,X,N ... (XON/XOFF is enabled)
```

(3) Data format



<1> When the XON/XOFF control is enabled, opening a serial port or executing a SAVE, BSAVE, LLIST or LPRINT command may cause an XON code to be generated.
When the XON/XOFF control is disabled, an XON code is not generated.

<2> With a file, the end-of-line code (CR or CR+LF) is always converted to CR+LF. However, for data transmission with LPRINT, LLIST or INPUT command, the end-of-line code can be specified to CR or CR+LF by PCONSOLE command.

(4) Serial port

(a) To select the RS-232C port, execute

```
SETDEV "COM1:"
    or
OPEN "COM1:"
```

The default port at power-up is the SIO port (COM2:).

(b) To select the SIO port, execute

```
SETDEV "COM2:"
    or
OPEN "COM2:"
```

(c) RS-232C control signals

1) The load of the outgoing control signal RST or DTR should be 3 to 7 kilo-ohms. Thus, do not connect the RST signal, for instance, to two signal lines on the remote system.

2) Control of RTS signal

The RTS signal is usually held at low, but can be set to high by OUTSTAT "COM1:" statement.

- OUTSTAT "COM1:"

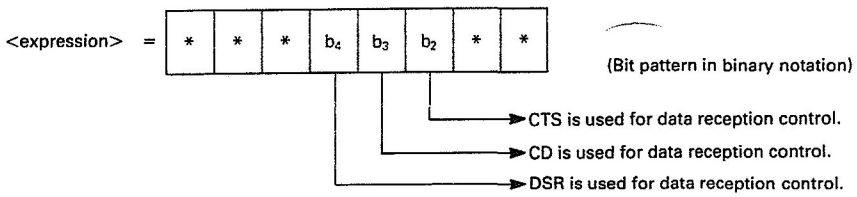
  While an RS-232C input/output command (SAVE, BSAVE, LOAD, BLOAD, LLIST, or LPRINT) is being executed or while the RS-232C port is open as a file, the RTS signal is held at high. It goes low when the command execution is terminated or when the RS-232C port is closed.

The RTS signal also works as the busy signal for data reception. The RTS signal goes low when the receive buffer is getting full and the free space in the buffer becomes 8 bytes, and it goes high when the received data in the buffer are read into the system and the number of remaining data in the buffer becomes 8 bytes.

Control of CTS, CD, and DSR incoming signals

- RCVSTAT "COM1:",<expression>

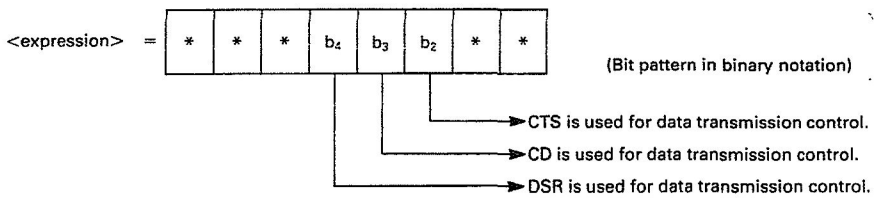
  This statement specifies which incoming signals (CTS, CD and/or DSR) are used to control data reception. Bits 2 to 4 of <expression> correspond to STC, CD and DSR.



Bit = { 0 : This signal is used for the data reception control. (When this signal is high, the data reception is enabled. When this signal is low, the data reception is disabled.)
1 : This signal is not used for the data reception control. (The data reception is enabled regardless of this signal.)

- SNDSTAT "COM1:",<expression>

  This statement specifies which incoming signals (CTS, CD and/or DSR) are used to control data transmission. Bits 2 to 4 of <expression> correspond to STC, CD and DSR.



Bit = { 0 : This signal is used for the data transmission control. (When this signal is high, the data transmission is enabled. When it is low, the data transmission is suspended.)
1 : This signal is not used for the data transmission control. (The data transmission is enabled regardless of this signal.)

- OUTSTAT "COM1:",<expression>
  This statement sets the RTS signal to high or low according to the value of <expression>. (For the relationship between the value of <expression> and the state of RTS signal, see the table in item 3) below.

3) Control of DTR signal
   The DTR signal is usually held at low, but it can be set to high by OUTSTAT "COM1:" statement.
   - OUTSTAT "COM1:"
     While an RS-232C input/output command is being executed or while the RS-232C port is open as a file, the DTR signal is held at high.
   - OUTSTAT "COM1:",<expression>
     This statement sets the DTR signal to high or low according to the value of <expression>.

<expression>	RTS	DTR
0	High	High
1	High	Low
2	Low	High
3	Low	Low

4) CTS signal (incoming signal)
   The CTS signal can be used for the data reception control and the data transmission control of the RS-232C port.
5) DSR signal (incoming signal)
   The DSR signal can be used for the data reception control and the data transmission control of the RS-232C port.
6) CD signal (incoming signal)
   The CD signal can be used for the data reception control and the data transmission control of the RS-232C port.
7) CI signal (incoming signal)
   The CI signal is used as a calling indicator from the modem. For example, you can use the CI signal to power on the PC-1600 and execute a specified program, or to interrupt the current program execution and move the control to a modem handling program.
   - WAKE$(1)="<command string>"
     When this statement has been executed, if the CI signal goes high, the PC-1600 is turned on and the specified command string is executed.
   - ON PHONE GOSUB ..... PHONE ON/OFF/STOP
     When these statement have been executed, if the CI signal goes high, control is moved to the specified subroutine.

   Note: To let the PC-1600 to be powered on by the CI signal, the CI signal must be held at high for more than one second.
8) INSTAT "COM1:"
   This statement reads the states of the RS-232C control signals: the incoming signals (CTS, DSR, CD and CI) and the outgoing signals (RTS and DTR). With this statement, you can know the state of the remote machine. The following shows the meaning of a value to be returned by this statement.

b_7	0
b_6	0
b_5	State of CI ("0" means "high"; "1" means "low")
b_4	State of DSR ("0" means "high"; "1" means "low")
b_3	State of CD ("0" means "high"; "1" means "low")
b_2	State of CTS ("0" means "high"; "1" means "low")
b_1	State of RTS ("0" means "high"; "1" means "low")
b_0	State of DTR ("0" means "high"; "1" means "low")

9) Timeout value for data reception and transmission

- RCVSTAT "COMn:",<expression1>,<expression2>

  The <expression2> of this statement specifies the timeout value for data reception.

  <expression2> = 0 : If no data has been received in the receive buffer, a command to input data from the serial port waits until data comes in.

  <expression2> = 1 to 255 : If no data has been received in the receive buffer, a command to input data from the serial port waits for a maximum of <expression2>/2 seconds. If no data comes in within that period, the command execution results in a timeout error.

- SNDSTAT "COMn:",<expression1>,<expression2>

  The <expression2> of this statement specifies the timeout value for data transmission.

  <expression2> = 0 : If the data transmission to the serial port is disabled when a command to output data to the serial port is executed, the command waits until the data transmission is enabled.

  <expression2> = 1 to 255 : If the data transmission to the serial port is disabled when a command to output data to the serial port is executed, the command waits for a maximum of <expression2>/2 seconds. If the data transmission is not enabled within that period, the command execution results in a timeout error.

  A timeout error occurs in either of the following cases:

  (1) When the XON/XOFF control is used, if the PC-1600 does not receive an XON code (11H) within the specified period of time since it has received an XOFF code (13H).

  (2) For data transmission to the RS-232C port (i.e., COM1:), if the incoming control signal that has been specified for the data transmission control does not go high within the specified period of time.

- RXD$

  Even when the PC-1600 has received 11H code or 13H code, an RXD$ command may not return these codes.

- SETCOM "COM2:"

  For data transmission/reception to the SIO port (i.e., COM2:), specify the same XON/XOFF control setting to both COM1: and COM2:.

- RCVSTAT and SNDSTAT

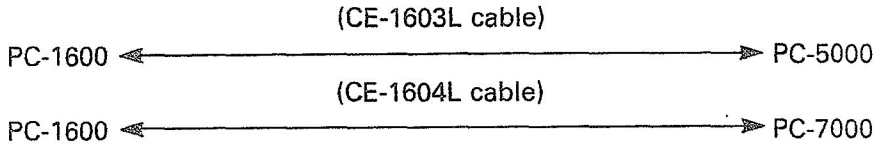
  Do not omit the <expression1> when you use these commands.

5.8 TRANSFERRING A BASIC PROGRAM BETWEEN PC-1600 AND OTHER MACHINE

The data format for transmission/reception of a BASIC program through RS-232C is different between the PC-1600 and other personal computers. Therefore, you cannot transfer a BASIC program through RS-232C simply by using load and save operations. This section describes a method to transfer a BASIC program between the PC-1600 and a different personal computer (PC-7000 or PC-5000), in which the PC-1600 uses the LOAD or SAVE command and the other personal computer uses a special file transfer BASIC program (described later).

(1) Connecting the computers

The method described here uses only three signal lines of RS-232C (transmit data line, receive data line and ground line).
Connect the PC-1600 and the other personal computer through their RS-232C connectors with the appropriate RS-232C cable.



(2) Setting up the PC-1600

To transfer a BASIC program, the PC-1600 uses the LOAD or SAVE command. Before starting the program transfer, you must initialize the PC-1600's RS-232C port as described in item (4) "Communication procedures" below.
You can use any baud rate, however, you must use the same baud rate between the two computers. If you use a baud rate of 4800 bps or higher, set the PC-1600's receive buffer size to the following value (by using INIT command):

Baud rate	Receive buffer size (minimum value)
4800, 9600	80
19200	130

(3) Setting up the remote computer

The remote computer uses a special BASIC program to transfer a BASIC program file. The following lists the file transfer programs and the setting conditions for different computers.

Personal computer	Transfer direction	File transfer program	Setting conditions
PC-7000	→ PC-1600	List 1	GW BASIC
	← PC-1600	List 2	GW BASIC
PC-5000	→ PC-1600	List 3	GW BASIC
	← PC-1600	List 4	GW BASIC

Note: Some personal computers do not suspend the data transmission even when the PC-1600 sends an XOFF code. For data transmission from such a computer to the PC-1600, take the following measures:

PC-1600 side:

Set the receive buffer size to 600 bytes or more.

Remote side:

Let the computer to wait for about 0.1 to 1 second each time the computer has sent one line of data, so that the PC-1600 does not have to send an XOFF code to the remote computer.

List 1

```
10 INPUT "FILE NAME (PC-1600 to PC-7000)";LECCEE$
20 OPEN LECCEE$ FOR OUTPUT AS #2
30 OPEN "com1:1200,n,8,1,lf" AS #1
40 PRINT "READY!!":BEEP
50 IF LOF(1)=0 THEN 50
60 PRINT #1,CHR$(&H11);
70 X2$=INPUT$(1,#1)
80 IF X2$=CHR$(&H11) THEN 70
90 IF X2$=CHR$(&HA) THEN 70
100 IF X2$=CHR$(&H1A) THEN 160
110 LINE INPUT #1,X1$:X1$=X2$+X1$
120 PRINT #1,CHR$(&H13);
130 PRINT X1$
140 PRINT #2,X1$
150 GOTO 60
160 CLOSE:END
```

List 2

```
20 INPUT "FILE NAME (PC-7000 to PC-1600)";LECCEE$
50 OPEN LECCEE$ FOR INPUT AS #1
60 OPEN "COM1:1200,N,8,1,lf" FOR OUTPUT AS #2
70 IF EOF(1) THEN 120
80 LINE INPUT #1,X1$
90 PRINT X1$
100 PRINT #2,X1$
110 GOTO 70
120 PRINT #2,CHR$(&H1A);:CLOSE :END
```

List 3

```
10 INPUT "FILE NAME (PC-1600 to PC-5000)";LECCEE$
20 OPEN LECCEE$ FOR OUTPUT AS #2
30 OPEN "com1:1200,n,8,1,LF,CS0" AS #1
40 PRINT "READY!!":BEEP
60 X2$=INPUT$(1,#1)
70 IF X2$=CHR$(&H11) THEN 60
80 IF X2$=CHR$(&HA) THEN 60
90 IF X2$=CHR$(&H1A) THEN 160
100 LINE INPUT #1,X1$:X1$=X2$+X1$
110 PRINT #1,CHR$(&H13);
120 PRINT X1$
130 PRINT #2,X1$
140 PRINT #1,CHR$(&H11);
150 GOTO 60
160 CLOSE:END
```

List 4

```
10 INPUT "FILE NAME (PC-5000 to PC-1600)";LECCEE$
20 OPEN LECCEE$ FOR INPUT AS #1
25 BEEP:PRINT "READY!!"
30 OPEN "COM1:1200,N,8,1,LF" FOR OUTPUT AS #2
40 LINE INPUT #1,X1$
50 PRINT X1$
60 PRINT #2,X1$
70 IF EOF(1)=0 THEN 40
80 PRINT #2,CHR$ (&H1A);
90 CLOSE
100 END
```

(4) Communication procedures

(a) Procedure to transfer a BASIC program from PC-1600 to the remote computer

	Operations on PC-1600	Operations on remote computer
1	(Create a BASIC program.)	
2	(Save the BASIC program.) Example: SAVE "X:TEST"	
3	Set up the communication protocol. 1) OUTSTAT "COM1:",0 2) RCVSTAT "COM1:", 63 3) SNDSTAT "COM1:", 59 4) INIT "COM1:",,"",""	
4	Load a BASIC program you want to transfer. Example: LOAD "X:TEST"	
5		Execute the appropriate data reception BASIC program. (Loading program from RS232C)
6		Enter a file name to save the received program data into a file with that name.
7	Start sending the program by executing the following statement. SAVE "COM1:",A	

Note: To send another program after you have sent one, repeat the above steps 4 to 7.

(b) Procedure to transfer a BASIC program from the remote computer to PC-1600

	Operations on PC-1600	Operations on remote computer
1	Set up the communication protocol. 1) OUTSTAT "COM1:",0 2) RCVSTAT "COM1:",63 3) SNDSTAT "COM1:",59 4) INIT "COM1:", 600,"",""	
2		Execute the appropriate data transmission BASIC program. (Saving program to RS232C)
3	Execute LOAD "COM1:"	
4		Enter a file name of the file you want to transfer.
5	After the program has been loaded into memory, save it. Example: SAVE "X:TEST"	

Note: To send another program after you have sent one, repeat the above steps 2 to 5.

5.9 MERGING PROGRAM FILES

The merge operation described in this section basically has the same function as MERGE command of BASIC, therefore, the following rules apply to the merge operation.

- The merge operation loads a program (specified in LOAD command) into memory without deleting the programs previously existing in memory. That is, different programs may coexist in memory.
- These coexisting programs may have the same line numbers.
- Each loaded program must be labeled (i.e., defined to a key on the keyboard (such as "A" or "S"). For details of the labeling, see OPERATION MANUAL, section IV 8 "Program Labels and the [DEF] Key".
- If an unlabeled program is loaded, label it immediately.
- The programs must be labeled to different keys.
- When you load a program by this merge operation, that program will be the object of editing. To edit the other programs, specify a program explicitly by LIST "label" command.
- This merge operation will result in an error if a password is set to the PC-1600 main unit.
- If READ and DATA statements are used in a program and the DATA items are to be re-read by a RESTORE command, label the first line of the DATA block and specify that label when you re-read the DATA block by a RESTORE command.
  If DATA statements are used in more than one program, unexpected data items may be read depending on the program execution state.
- After the merge operation is performed, the fixed numeric variable Z contains the value "0" (because it is used for the merge operation.)

File merge operation procedure

1. Execute the machine language program I (by CALL command of BASIC).
2. Execute LOAD "<d:filename>" (where <d:filename> is the drive and the file names of the file you want to merge.)
3. Execute the machine language program II (by CALL command of BASIC.)

Note: Perform steps 1, 2 and 3 in that order and do not perform any other operations between them. Even if the LOAD operation of step 2 has resulted in an error, be sure to perform step 3.

Machine language program I

```
D000:3A D5 F1 FE 01 30 28 3A 2B F0 32 CA F9 2A 65 F8
D010:22 C8 F9 ED 5B 67 F8 B7 ED 52 2A 2C F0 20 03 BD
D020:28 04 14 20 01 1C ED 53 65 F8 7D 32 2B F0 C9 21
D030:19 F0 28 03 21 23 F0 E5 11 C8 F9 01 03 00 ED B0
D040:EB E1 06 03 1A BE 77 28 01 0C 13 23 10 F6 79 B7
D050:C8 5E 23 56 13 2B 2B 2B 72 2B 73 C9
```

Machine language program II

```
D05C:3A D5 F1 FE 01 30 22 3A 2B F0 32 C4 F1 3A CA F9
D06C:32 2B F0 2A 65 F8 22 69 F8 2A C8 F9 22 65 F8 7D
D07C:6C F6 80 67 22 C8 F9 18 20 21 1B F0 28 03 21 25
D08C:F0 7E 32 C4 F1 2B 7E E6 7F 5F 2B 56 ED 53 69 F8
D09C:EB 21 C8 F9 01 03 00 ED B0 11 3C FE 21 C8 F9 01
D0AC:03 00 ED B0 21 C8 F9 AF 06 08 77 23 10 FC 2A 69
D0BC:F8 22 9E F8 3A C4 F1 32 C1 F1 C9
```

Note: The starting addresses of the programs I and II are "D000H" and "D05CH", respectively. However, they can be placed in any other locations because they are relocatable.
It would be useful if you save the programs in files by BSAVE command after loading them by POKE command.

5.10 SAVING AND LOADING THE RESERVE AREA

You can save and load the contents of the RESERVE area to and from a file by using BSAVE and BLOAD commands.

(1) Saving the RESERVE area from the extension memory

Extension memory module in slot S1	Extension memory module in slot S2	Saving procedure
None	None	BSAVE "<devicename:filename>",#0,&C008,&C0C4
None or CE-1600M or CE-161	CE-1600M or CE-161	BSAVE "<devicename:filename>",#2,&8008,&80C4
CE-159 or CE-155	None or CE-1600M or CE-161	BSAVE "<devicename:filename>",#0,&A008,&A0C4
CE-151	None	BSAVE "<devicename:filename>",#0,&B008,&B0C4
CE-1600M or CE-161	None	BSAVE "<devicename:filename>",#0,&8008,&80C4

(2) Saving the RESERVE area from the program module

RESERVE area in the program module in slot S1	BSAVE "<devicename:filename>",#0,&8008,&80C4
RESERVE area in the program module in slot S2	BSAVE "<devicename:filename>",#2,&8008,&80C4

(3) Loading a file data into the RESERVE area

(a) BLOAD "<devicename:filename>"
This statement loads the data from the specified device and file into the same RESERVE area as saved in items (1) or (2) above.

(b) BLOAD "<devicename:filename>",<#bank number>,<starting address>,<end address>
This statement loads the data from the specified device and file into the specified memory locations. The locations of the RESERVE area are the same as described in items (1) and (2) above.

5.11 DISABLING THE KEY INTERRUPT DUE TO ON KEY STATEMENT

If a function key is pressed for an INPUT command, the key operation is memorized as a key interrupt even if the key is pressed after a KEY(n) STOP statement has been executed, and the key interrupt becomes effective when a KEY(n) ON statement is executed later.
To avoid a function key operation for an INPUT command from being accepted as an interrupt (that is, to disable a function key operation for an INPUT command), execute the following lines:

```
KEY(n) STOP
INPUT command
POKE &F1D4,PEEK(&F1D4) AND &C0
KEY(n) ON
```

5.12 CE-153 CONTROL UTILITY (FOR PC-1600)

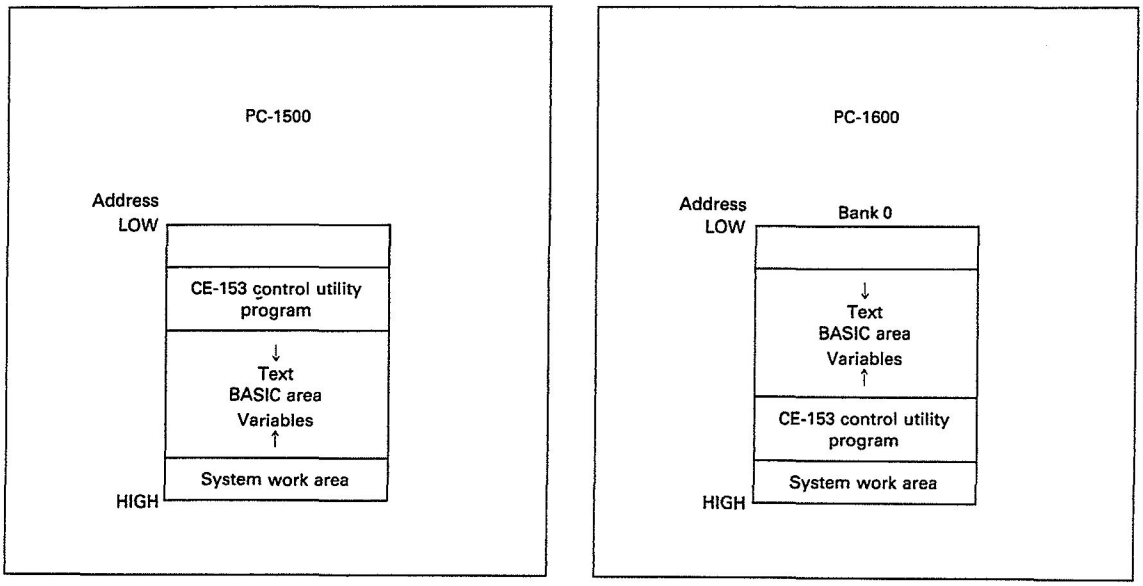
(1) Basic specifications

The CE-153 control utility program allows the CE-153 (which is a peripheral device for PC-1500) to be used with the PC-1600.

1. The basic specifications of the utility program are the same as those of the utility program written for PC-1500 which comes with CE-153, except for the following points. Thus, if you want to use a CE-153 application program written for PC-1500 for the PC-1600, you may need to change part of the program according to the following changes.

	For PC-1500	For PC-1600
Differences	When operating the keys on the CE-153, letters and numbers entered from the PC-1500 are stored into the variable Z$(0). This variable can contain up to 80 characters.	When operating the keys on the CE-153, letters and numbers entered from the PC-1600 are stored into the variable Y$. This variable can contain up to 16 characters.

2. As shown in the following memory maps, in the PC-1500, the CE-153 control utility program (written for PC-1500) is loaded from the top of the RAM area (i.e., in the machine language area before the BASIC area). In the PC-1600, however, the utility program (written for PC-1600) is loaded in the middle of the RAM area (before the system work area and after the BASIC variables area).



(2) Entering and saving the CE-153 control utility program

List A shows the dump list of the CE-153 control utility program. This utility program is written in the LH-5803 machine codes in the relocatable addressing structure. Enter and save the utility program in the following procedure:

1. Execute: MODE 0 [ENTER]
2. Reserve 1.2 KB of machine language area.
   Example: When no memory module is installed
   NEW "S0:",1392 [ENTER]
   This statement reserves the machine language area of 1200 (=1392−192) bytes starting from C0C6H.

3. Enter the codes of list A by using POKE command.
   The following BASIC program would be helpful to enter these codes.

```
10: A=&C0C6
20: CURSOR 0,0: PRINT HEX$(A);: INPUT X
30: POKE A,X: A=A+1
40: GOTO 20
```

   (To terminate the program, press the [BREAK] key.)
4. When the codes have been entered completely, save the program into a cassette or disk.
   Example 1: Saving to a disk

```
BSAVE "X:filename", #0,&C0C6,&C54F
```

   Example 2: Saving to a cassette (through CE-1600P)

```
CSAVEM "filename"; #0,&C0C6,&C54F
```

   Example 3: Saving to a cassette (through CE-150 in MODE 1)

```
MODE 1
CSAVEM "filename"; &40C6,&454F
```

List A

Memory address	data							
C0C6-CD	4A	00	8E	06	4A	01	8E	02
C0CE-D5	8E	35	04	AE	78	5A	FD	58
C0D6-DD	CA	8E	B5	11	FD	CA	58	7B
C0DE-E5	5A	A8	6A	06	F5	88	03	8E
C0E6-ED	DC	CC	8E	24	FD	CA	FD	5E
C0EE-F5	ED	78	6B	01	89	0A	6A	08
C0F6-FD	48	01	4A	4B	BE	E6	6F	9A
C0FE-05	6A	FF	CD	A6	88	04	9A	FD
C106-0D	98	FD	A8	58	80	B5	FC	5A
C10E-15	0D	FD	1E	B5	00	52	FD	1E
C116-1D	5A	08	FD	1E	FD	AE	80	0F
C11E-25	AE	77	F0	0A	94	FD	1E	6A
C126-2D	03	88	02	FD	ED	80	0E	FF
C12E-35	89	15	FD	ED	80	0F	03	89
C136-3D	0E	40	D5	91	18	FD	1E	4E
C13E-45	09	83	22	5A	0F	9E	23	04
C146-4D	F1	0A	5A	0F	FD	15	BF	02
C14E-55	89	21	40	BF	01	89	1C	40
C156-5D	52	FD	15	BF	80	89	14	40
C15E-65	D5	83	10	99	06	FB	B5	FF
C166-6D	8E	50	A5	7B	0F	FD	98	FD
C16E-75	A8	8E	05	04	F1	AE	7B	08
C176-7D	58	7A	5A	22	FD	C8	B9	0F
C17E-85	0A	48	00	BE	DD	2F	60	5A
C186-8D	20	FD	8A	FD	C8	F1	B9	0F
C18E-95	0A	48	00	BE	DD	2F	60	48
C196-9D	7A	4A	21	58	77	5A	F0	6A
C19E-A5	03	F5	88	03	59	00	FD	8A
C1A6-AD	F9	B3	C0	ED	78	5A	01	8B
C1AE-B5	08	FD	58	F9	B3	57	FD	CA
C1B6-BD	05	F9	0A	FD	2A	FD	1A	AE
C1BE-C5	7B	08	9A	8E	A7	8E	A1	8E
C1C6-CD	A1	38	C0	C1	C2	C3	C4	C5
C1CE-D5	C6	C7	C8	B4	B9	A1	A4	A2
C1D6-DD	09	09	D0	D1	D2	D3	D4	D5
C1DE-E5	D6	D7	D8	AC	BC	84	9A	94
C1E6-ED	09	09	E0	E1	E2	E3	E4	E5
C1EE-F5	E6	E7	E8	9C	9E	91	81	9D
C1F6-FD	09	09	F0	F1	F2	F3	F4	F5
C1FE-05	F6	F7	F8	AD	B2	AA	8C	BA
C206-0D	09	09	00	01	02	03	04	05
C20E-15	06	07	08	92	A9	8A	89	82
C216-1D	09	09	10	11	12	13	14	15
C21E-25	16	17	18	B1	A8	99	9F	83
C226-2D	09	09	20	21	22	23	24	25
C22E-35	26	27	28	95	85	BD	B5	B5
C236-3D	09	09	30	31	32	33	34	35
C23E-45	36	37	38	96	86	BE	8E	AE
C246-4D	09	09	40	41	42	43	44	45
C24E-55	46	47	48	97	87	BF	A7	B7
C256-5D	09	09	50	51	52	53	54	55
C25E-65	56	57	58	90	8F	A8	98	98
C266-6D	8E	64	8E	74	48	7B	ED	7B
C26E-75	0E	40	89	05	B5	B0	AE	7B
C276-7D	09	B5	80	AE	7B	0D	68	F8
C27E-85	CD	A6	89	59	BE	E4	18	89
C286-8D	47	B5	FF	FD	AE	80	08	FD

Memory address	data							
C28E-95	AE	80	0F	B5	FC	FD	AE	80
C296-9D	0D	FD	E9	80	0C	00	6A	03
C29E-A5	88	02	FD	ED	80	0F	03	89
C2A6-AD	06	FD	A5	80	0E	8B	33	6A
C2AE-B5	2F	BE	7B	A8	83	2C	ED	7B
C2B6-BD	0E	01	89	74	EB	7B	0E	01
C2BE-C5	04	AE	7B	0F	BF	40	89	AC
C2C6-CD	BF	89	8B	A8	8E	80	8E	DD
C2CE-D5	BE	E4	2C	83	0D	6A	50	CD
C2D6-DD	A6	88	04	9E	27	B5	0E	9A
C2DE-E5	8E	FC	ED	7B	0E	01	88	0E
C2E6-ED	ED	7B	0E	40	89	35	FD	60
C2EE-F5	81	04	E9	7B	0E	FE	ED	7B
C2F6-FD	7C	01	8B	1D	EF	7B	0D	01
C2FE-05	81	17	EB	7B	0D	80	FD	A8
C306-0D	CC	7E	EF	78	7C	80	B5	7F
C30E-15	81	03	A5	78	7D	CD	8A	FD
C316-1D	2A	B5	57	FD	CE	FD	81	FD
C31E-25	81	9E	A3	B5	B0	AE	7B	09
C326-2D	E9	7B	0E	9F	68	F8	9E	3A
C32E-35	ED	7B	0E	40	9B	0A	04	A7
C336-3D	7B	0F	99	19	48	7B	4A	09
C33E-45	05	DD	0E	91	2C	B5	FB	0E
C346-4D	A5	7B	0F	0A	9E	8A	68	75
C34E-55	6A	4E	4E	83	8B	4E	4E	B0
C356-5D	8B	2A	4E	B3	8B	19	48	FE
C35E-65	BE	E3	66	B7	0D	8B	03	9A
C366-6D	9E	FE	B5	9E	AE	7B	0F	6A
C36E-75	92	BE	7B	A8	B5	0D	9A	25
C376-7D	BD	80	B9	FD	2E	6A	18	BE
C37E-85	7B	A8	9E	1C	FD	ED	F0	0F
C386-8D	08	89	0D	A5	78	5D	D9	89
C38E-95	0E	25	BD	04	B9	77	9E	1C
C396-9D	25	BD	08	B9	7B	9E	23	25
C39E-A5	B9	73	9E	3C	25	BD	02	B9
C3A6-AD	7F	9E	2F	8E	87	A5	E0	02
C3AE-B5	B7	68	89	23	ED	71	BC	40
C3B6-BD	89	1D	EB	78	74	01	A5	78
C3BE-C5	75	FD	C8	D9	F9	A3	78	75
C3C6-CD	D9	AE	78	75	6A	EF	BE	78
C3CE-D5	A8	FD	8A	AE	78	75	9A	A5
C3D6-DD	78	75	FD	C8	9E	12	CC	83
C3DE-E5	FD	88	CC	84	FD	88	FD	58
C3E6-ED	B5	88	FD	CA	FD	5A	CE	58
C3EE-F5	4F	6C	80	83	D5	DA	E9	78
C3F6-FD	76	00	6A	19	A5	78	75	8B
C3FE-05	0F	DF	68	06	F8	A0	81	05
C406-0D	88	05	FB	8E	8B	A2	89	01
C40E-15	60	24	9B	0A	F9	BD	FF	AE
C416-1D	78	77	EB	78	8A	10	B5	40
C41E-25	AE	78	80	BE	D0	34	58	7B
C426-2D	5A	B0	E9	7B	B0	00	ED	78
C42E-35	74	01	89	35	6A	EB	BE	7B
C436-3D	A8	58	7B	E9	78	00	0D	EB
C43E-45	78	00	0D	B7	20	81	2C	5E
C446-4D	B0	81	1E	1E	5E	FF	83	14
C44E-55	50	ED	78	74	01	89	09	ED

Memory address	data							
C456-5D	78	5A	02	89	07	BE	EE	71
C45E-65	EB	78	5A	02	6A	18	BE	7B
C466-6D	A8	BE	E8	CA	9E	3A	8E	D0
C46E-75	59	24	0D	E9	78	0E	BF	B7
C476-7D	0E	8B	A0	B7	0D	8B	86	B7
C47E-85	18	8B	75	B7	1A	8B	71	5E
C486-8D	B0	91	E0	B7	08	8B	62	B7
C48E-95	0C	8B	4E	B7	10	8B	35	B7
C496-9D	1C	99	50	15	B7	0D	8B	24
C49E-A5	F9	B5	FF	10	81	1B	FD	A8
C4A6-AD	FD	88	2A	5A	FF	FD	18	46
C4AE-B5	05	B7	E0	81	02	B5	0D	0E
C4B6-BD	47	53	88	04	FB	FD	0A	FD
C4BE-C5	2A	B5	27	1E	8E	37	8E	78
C4C6-CD	8E	74	8E	6E	FD	98	FD	18
C4CE-D5	14	BD	FF	2A	44	8E	01	F5
C4D6-DD	88	03	B5	0D	1E	FD	1A	8E
C4DE-E5	1C	15	B7	0D	8B	17	5E	FF
C4E6-ED	88	13	54	EB	7B	0E	40	8E
C4EE-F5	0C	5E	81	81	08	56	9E	0D
C4F6-FD	8E	D0	34	5A	B0	ED	78	5A
C4FE-05	02	9B	CF	9E	A1	48	7B	4A
C506-0D	B0	05	B7	0D	88	03	40	91
C50E-15	08	04	B3	4F	4A	B0	CD	24
C516-1D	CD	08	25	E9	78	8A	EF	BE
C51E-25	D0	34	6A	18	BE	7B	A8	FD
C526-2D	0A	CA	84	FD	0A	CA	83	F9
C52E-35	A5	7B	0F	B3	40	0A	48	00
C536-3D	FB	9A	68	07	8E	02	68	20
C53E-45	FD	1A	FD	8A	FD	1A	FD	1A
C546-4D	FD	1A	FD	1A	E9	78	8A	EF
C54E-4F	E0	00						

(3) Loading the CE-153 control utility program

1. Turn off the power of the PC-1600 main unit, then remove all memory modules.
2. Connect CE-150 to the PC-1600 main unit.
3. Turn on the power of the PC-1600 main unit.
4. Set the PC-1600 in the RUN mode, and execute the following commands:

```
MODE 1
A=1200
CALL &2DD,A
```

(These commands reserve a memory area in which the utility program is loaded.)

5. Connect the cassette tape recorder to the PC-1600, set the cassette tape containing the utility program, and execute the following statement.

```
CLOADM XPEEK&7035*256+XPEEK&7034-32768
```

Now the utility program has been loaded in memory.
The loaded program will be destroyed if you do ALL RESET.
Now, turn off the power of the PC-1600 and install the necessary memory modules.

(4) Executing the CE-153 control utility program

1. Execute the following statement:

```
XCALL (XPEEK&7035*256+XPEEK&7034-32768)
```

This statement is equivalent to the statement CALL &40C6 of the CE-153 control utility program written for PC-1500.

2. Special usage 1 of software keyboard (See CE-153 INSTRUCTION MANUAL.)

XCALL (XPEEK&7035*256+XPEEK&7034−32768)+4

3. Special usage 2 of software keyboard (See CE-153 INSTRUCTION MANUAL.)

XCALL (XPEEK&7035*256+XPEEK&7034−32768)+8

Notes 1: You must specify a character display position by CURSOR command before you run the utility program.
Example: 100: CURSOR 0,0: XCALL &6B50
2: Characters entered from the software keyboard are always displayed on the third line (bottom line) of the screen. Therefore, CURSOR command is effective only to the X coordinate value.
3: When the software keyboard is in use, the cursor display is always on and the cursor type is the two-character composite cursor (an underline and a space).

5.13 RST COMMANDS OF SC-7852 (Z-80)

Some of the RST commands of Z-80 are open to the user. These commands are used to jump to a system work area. (When the system is in the reset state, RET command is written in the system work areas.) A system work area of 3 bytes is prepared for each user RST command. By writing a JP instruction into these three bytes, the user can move control to an arbitrary routine.

RST command	Description	
RST 00H		System reset
RST 08H		Reserved by SHARP
RST 10H		Reserved by SHARP
RST 18H		Reserved by SHARP
RST 20H		IOCS for inter-bank call
RST 28H		Reserved by SHARP
RST 30H	JP F0D4H	Open to user
RST 38H	JP F0D7H	Open to user

5.14 SC-7852 (Z-80) AND LH-5803 MICROPROCESSORS

(1) Switching the microprocessors

The PC-1600 has two main CPUs: SC-7852 and LH-5803. The system uses one of the two CPUs at a time: when one is in the operation state, the other is in the non-operation state. Usually the system uses SC-7852 (Z-80) as the main CPU. For control of a peripheral device for PC-1500 or for execution of a program written in the LH-5803 machine language, the CPU is switched to LH-5803.

(2) How to call an LH-5803 machine language program subroutine from SC-7852

To call an LH-5803 machine language program from SC-7852, use the following IOCS routine.

CALLH

Entry Address 01C6H

Function Move control from SC-7852 to an LH-5803 machine language program subroutine. When the program execution is completed, control returns to SC-7852.

Parameter Set the parameters in the appropriate addresses as described in the following table.

Name	Address*1	Contents
CMDZ	F002H (7002H)	Operation mode*2
PARA	F005H (7005H)	Value to be passed to A register of LH-5803
PARXL	F006H (7006H)	Value to be passed to XL register of LH-5803
PARXH	F007H (7007H)	Value to be passed to XH register of LH-5803
PARYL	F008H (7008H)	Value to be passed to YL register of LH-5803
PARYH	F009H (7009H)	Value to be passed to YH register of LH-5803
PARUL	F00AH (700AH)	Value to be passed to UL register of LH-5803
PARUH	F00BH (700BH)	Value to be passed to UH register of LH-5803
PARPCL	F00CH (700CH)	Entry address of a called subroutine (low byte)*3
PARPCH	F00DH (700DH)	Entry address of a called subroutine (high byte)*3
PARBAN	F00EH (700EH)	Bank of a called subroutine*4

*1: The addresses are those viewed from SC-7852. Those enclosed in brackets are the addresses viewed from LH-5803.
*2: CMDZ (operation mode) must be either 20H or 30H.
CMDZ=20H: When control is moved to LH-5803, no parameters are passed to the registers of LH-5803.
=30H: When control is moved to LH-5803, the values set in PARA to PARUH (i.e., F005H to F00BH) are passed to the registers of LH-5803.
*3: The entry address must be an address viewed from LH-5803.
*4: PARBAN=00H: $\overline{PV}$
=01H: PV

Return

When the LH-5803 machine language subroutine is completed and control returns to SC-7852, the contents of the LH-5803's registers are stored in memory.
When the operation mode is CMDZ=20H

Address	Contents
F005H	Content of A register of LH-5803
F004H	Content of STATUS(T) register of LH-5803

When the operation mode is CMDZ=30H

Address	Contents
F005H	Content of A register of LH-5803
F004H	Content of STATUS(T) register of LH-5803
F006H	Content of XL register of LH-5803
F007H	Content of XH register of LH-5803
F008H	Content of YL register of LH-5803
F009H	Content of YH register of LH-5803
F00AH	Content of UL register of LH-5803
F00BH	Content of UH register of LH-5803

Affected Register All registers

5.15 COMPATIBILITY WITH PC-1500

The BASIC interpreter of PC-1600 is basically compatible with programs written for PC-1500/PC-1500A and compatible with peripheral devices for PC-1500/PC-1500A, although there are some restrictions.

(1) Restrictions in running a BASIC program written for PC-1500/PC-1500A

1. To run a BASIC program written for PC-1500/PC-1500A in the same conditions (such as the single line display) as when run on PC-1500/PC-1500A, set the PC-1600 in MODE 1 before executing the program.
2. To use the PC-1600 in MODE 1, the following conditions must be satisfied:
   - A RAM module of more than 16 KB must not be installed in slot 1 and any RAM module must not be installed in slot 2. However, if a RAM module is used as a RAM disk, CE-1600M can be installed in slot 1 and CE-1600M or CE-161 can be installed in slot 2. In this case, the formatting of a RAM disk (INIT "Sn:","F") must be done in MODE 0.
3. Some BASIC command names are different between PC-1600 and PC-1500/PC-1500A.

	PC-1500/PC-1500A command names	PC-1600 command names
1)	LCURSOR	TAB
2)	LINE	LLINE
3)	CALL	XCALL
4)	POKE	XPOKE
5)	PEEK	XPEEK
6)	POKE#	XPOKE#
7)	PEEK#	XPEEK#

   - When you enter a BASIC program text written for PC-1500/PC-1500A from the keyboard, change the command names as described above.
   - When you load a BASIC program written for PC-1500/PC-1500A from a cassette tape recorder, the PC-1500/PC-1500A command names are automatically rewritten to the corresponding PC-1600 command names, except for LCURSOR. In this case, after loading the program, rewrite "LCURSOR" to "TAB".
4. Loading a BASIC program written for PC-1500/PC-1500A from a cassette tape recorder
   - When the CE-150 or CE-162E cassette interface is used, you can load a program.
   - When the CE-1600P cassette interface is used, you can load a program in the MODE 1 but you cannot load data.
5. TIME command
   In the PC-1500/PC-1500A BASIC, you can specify TIME=0. In the PC-1600 BASIC, however, TIME=0 will result in an error.
6. Several new reserved words (NAME, AS, XOR, etc.) have been added to the PC-1600 BASIC. If these words are used as a variable name in a PC-1500/PC-1500A BASIC program, change them to other non-reserved words.
7. The use of the work areas for the BASIC interpreter is different between the PC-1500/PC-1500A BASIC and the PC-1600 BASIC. Because of this, a PC-1500/PC-1500A BASIC program including a machine language program may not run properly on the PC-1600.
8. The PC-1500A memory area from 7C01H to 7FFFH is used as the system work area in the PC-1600. Thus, a PC-1500A BASIC program using any memory locations between 7C01H and 7FFFH cannot run on the PC-1600.
9. An array variable cannot be used for the assignment variable of an INPUT statement which is to be performed to CE-158.

(2) Restrictions in using a peripheral device for PC-1500/PC-1500A

1. RAM module

The following RAM modules can only be installed in the memory slots.
Memory slot 1: CE-1600M, CE-161, CE-159, CE-155, CE-151
Memory slot 2: CE-1600M, CE-161

2. CE-150, CE-158, CE-162E

- Use in MODE 0
  a) LLIST, CSAVE, CLOAD, CSAVEM and CLOADM cannot be executed to these peripheral devices.
  b) If a command with a variable other than a fixed variable specified in the operand is executed to these peripheral devices, the command will result in an error. For instance,

```
LPRINT A1
```

  this command will result in an error. Change this to the following:

```
A=A1
LPRINT A
```

  c) TERMINAL and DTE commands cannot be used to CE-158.
- Use in MODE 1
  Execution of a command to CE-150, CE-158, or CE-162E is effective only in the scope supported in the PC-1500/PC-1500A BASIC. For instance,

```
LPRINT TIME$
```

  this command will result in an error since TIME$ is not supported in the PC-1500/PC-1500A BASIC.

3. CE-153

For the use of CE-153, see section 5.12 "CE-153 CONTROL UTILITY". The utility program which comes with CE-153 cannot be used on the PC-1600.

4. Differences between CE-150 and CE-1600P

- Since the printable area is different between these printers, the format of the printout may differ. (You can change the printing area by PCONSOLE "LPT1:" command and PAPER command.)
- CE-150 has two remote control terminals, while CE-1600P has only one.

5.16 PRECAUTIONS FOR APPLICATION PROGRAM DEVELOPMENT

The IOCS routines covered in this manual will not be changed even when the system software is revised in the future. Therefore, when you develop a machine language program for the PC-1600, it is recommended to write it using the IOCS calls.
There are a couple of versions of PC-1600 BASIC interpreters and they are different in the minor specifications. When you develop a program using the BASIC, keep the following rules so that the program you make can be compatible with all these versions.

(1) USING format

To print a numeric value in the exponent format using a USING specification, the number of characters specifying the exponent part must be 2.
Example: USING "##.####^"

(2) INPUT "<message>"; <variable> statement

Avoid the end of the message being displayed at the right most column on the screen.

(3) INPUT statement

To specify an input position for an INPUT statement by CURSOR statement, execute a CURSOR statement before each INPUT statement.
Example:
```
100: CURSOR 10,1
110: INPUT "A=";A
120: CURSOR 10,1
130: INPUT "B=";B
```

If an array variable or @ is used for the assignment variable of INPUT statement, the subscript must not be specified by an expression that includes a string.

(4) Addition of string data

When adding string items, each item must not consist of a function that contains a string constant. For instance,

```
B$="ABC"+STR$ LEN"XYZ"
```

should be written in the following two lines:

```
A$=STR$ LEN"XYZ"
B$="ABC"+A$
```

(5) LINE statement

When a LINE statement is given with the dot toggle parameter of X and the option B, the corners of the box may not be displayed on or off properly.

(6) WAKE$(0) statement

To disable the wakeup function by using an WAKE$(0)=" " statement, do as follows:

```
Example: 100: A=LEN ALARM$
         110: WAKE$(0)=" "
         120: IF A=0 LET A=A: ALARM$=" "
```

In this example, the program checks whether the alarm function is on before executing WAKE$(0)= " ". After execution of WAKE$(0)=" ", the program resets the alarm function.

(7) CALL statement

When a CALL statement is executed with a string variable, if the machine language program is to pass a null string back to the string variable when returning to the calling BASIC program, the null string must be expressed as a 00H code of length 1.

(8) XCALL statement

If you want to give a parameter of XCALL statement by a variable, use a fixed or a simple variable only.

(9) DIM statement

When declaring a two-dimensional array variable, the subscript must not exceed 254.

(10) LLINE/RLINE statement (to CE-1600P)

When executing a LLINE or RLINE statement, be sure to specify a line type.

(11) CSIZE statement (to CE-1600P)

If you change the character size to a larger one by CSIZE statement when the printer is in the text mode and the pen is at a position other than the left end, ensure the correct pen position by executing the appropriate LCURSOR statement.

(12) PZONE "LPT1:" statement (to CE-1600P)

In the text mode, when you specify a print zone length by a PZONE "LPT1:" statement, the zone length (characters per zone) must be given such a value that an integer multiple of the zone length equals the current line length (characters per line).

(13) PCONSOLE "LPT1:" statement (to CE-1600P)

When a left margin has been set by a PCONSOLE "LPT1:" statement, if the number of characters to be printed on a line is less than or equal to the value shown in the table below, you must append to the characters as many spaces as the total number of characters including these spaces becomes that value plus 1 before performing CR+LF operation.

Character size	No. of characters
CSIZE 1	3
CSIZE 2 to 5	1

(14) LET statement

Do not use a LET statement to assign a value to a system variable (TIME, TIME$, DATE$, ALARM$, WAKE$).

Example: LET TIME=102513.45 (Not allowed)

CHAPTER 6

WORK AREA USED FOR BASIC

6.1 OVERVIEW OF WORK AREA

The PC-1600 uses a work area of 4 KB, from F000H to FFFFH of bank 0 viewed from SC-7852 (or from 7000H to 7FFFH viewed from LH-5803). The work area consists mainly of five blocks from block A to block E as shown in the table below.

Block name	Address (viewed from SC-7852)	Address (viewed from LH-5803)	Contents (Z-80 address)	
A	F000H	7000H	F05CH	
			F1B2H	IOCS work
			F21DH	Interpreter work I
			F31DH	Edit buffer (256 bytes)
			F3C7H	Interpreter work II
			F500H	Default FCB (313 bytes)
	F5FFH	75FFH		Z-80 stack area (256 bytes)
B	F600H	7600H	Same usage as for PC-1500/PC-1500A F650H	PC-1500/PC-1500A display area I
			F700H	Fixed variables area (E$ to O$)
			F750H	PC-1500/PC-1500A display area II
	F7FFH	77FFH		Fixed variables area (P$ to Z$)
C	F800H	7800H	Same work area and usage as for PC-1500/ PC-1500A	
	FBFFH	7BFFH		
D	FC00H	7C00H	FCB0H	RAM file work
	FF20H			Interpreter work III
E	FF21H FFFFH	7FFFH	Reserved by system	

Blocks A to E are used for the following purposes.

Block A: This block is used for the IOCS, interpreter, default FCB, and Z-80 stack. (Extended for PC-1600)

Block B: This block is used in the same way as for PC-1500/PC-1500A, and consists of two parts:

- PC-1500/PC-1500A display memory space area
- Fixed variables (E$ to Z$) area

Block C: This block is used in the same way as for PC-1500/PC-1500A, and consists of five parts:

- LH-5803 stack area ..... F800H to F84FH
- Fixed variables area: A to Z ..... F900H to F9CFH
  A$ to D$ ..... F8C0H to F8FFH
- Arithmetic operation work ..... FA00H to FA37H
- Buffers: String buffer ..... FB10H to FB5FH
  Output buffer ..... FB60H to FBAFH
  Input buffer ..... FBB0H to FBFFH
- Interpreter work

Block D: This block is used for the RAM file work, etc. (Extended for PC-1600)

Block E: PC-1600 system reserved area, which will be used by CE-1F01A (barcode pen reader software)

Note: Since the block C is used in the same way as for PC-1500/PC-1500A, bear the following points in mind.

- A 2-byte data is written in the order of the high byte and the low byte.
- The addressing uses an address viewed from LH-5803 (the MSB of almost all LH-5803 addresses is "0".) Thus, when viewed from SC-7852, the MSB of each address is inverted.

6.2 EXPANSION OF WORK AREA AND BUFFER

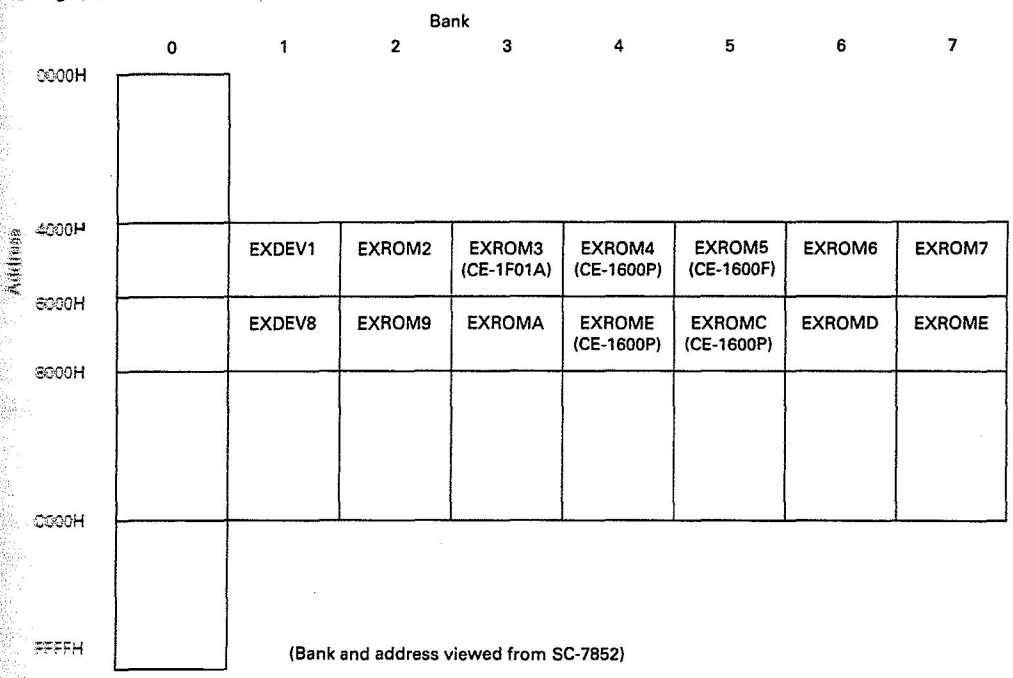
The standard work area of PC-1600 is from F000H to FFFFH, however, it can be expanded to the lower address direction (down to C000H) in the following cases:

- When a peripheral device is connected
- When a buffer is explicitly expended by the appropriate command

(1) Connection of a peripheral device

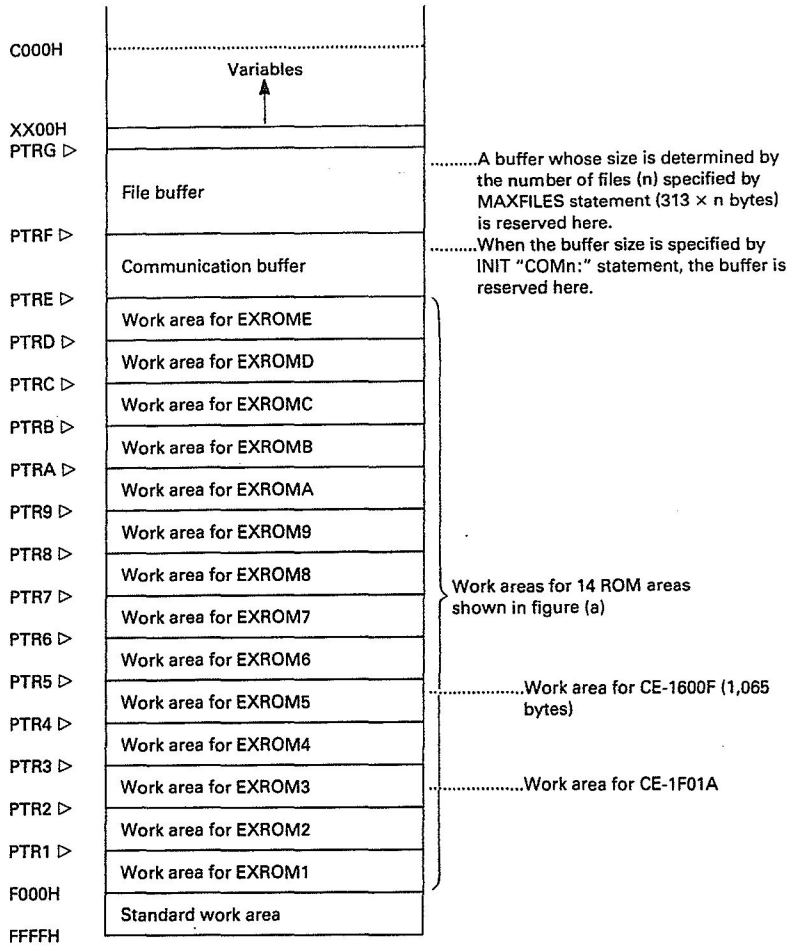
When a peripheral device is connected to the PC-1600, the program (ROM) to control the device is mapped as shown in figure (a) and is loaded into the expanded work area as shown in figure (b).

1) Fig. (a)



(Bank and address viewed from SC-7852)

1) Fig. (b)



As shown in figure (a), there are 14 memory blocks for the peripheral devices (8 KB per block). The following five blocks among these 14 blocks are used for particular peripheral devices:

EXROM3: Used for CE-1F01A
EXROM5: Used for CE-1600F
EXROM4:
EXROMB:
EXROMC: } Used for CE-1600P (The work area is not expanded even when CE-1600P is connected. It uses the standard work area.)

The other nine memory blocks are reserved for future peripheral devices.

(2) Reserving and releasing the work areas and buffers

1. Work area

When the PC-1600 is all-reset or reset or powered on, if there is a peripheral device connected, the relevant work area is reserved.

The reserved work area is released when you power off the PC-1600 and remove the peripheral device and power on again. However, the work areas for EXROM3 and EXROMA are released when the PC-1600 is all-reset or when the work areas are released explicitly by using the appropriate commands.

2. **Buffer**

	Expanding command	Releasing command	ALL RESET event	RESET event	POWER ON event
Communication buffer	INIT "COMn:", m (m≥80)	INIT "COMn:", 0	Released	Released	Released
File buffer	MAXFILES=m (m≥1)	MAXFILES=0	Released	Maintained	Maintained

(3) Pointers to point the work areas and buffers (PTR1 to PTRG in figure (b) above)

Each pointer points to the starting address of the relevant work area, which is stored in the standard work area. The following table shows the pointers and the standard work area locations where the contents of each pointer are stored (the address data are stored in the order of the low address byte and the high address byte.)

Pointer name	Locations	Pointer name	Locations	Pointer name	Locations
		PTR7	F03C, 3DH	PTRE	F04A, 4BH
PTR1	F030, 31H	PTR8	F03E, 3FH	PTRF	F04C, 4DH
PTR2	F032, 33H	PTR9	F040, 41H	PTRG	F04E, 4FH
PTR3	F034, 35H	PTRA	F042, 43H		
PTR4	F036, 37H	PTRB	F044, 45H		
PTR5	F038, 39H	PTRC	F046, 47H		
PTR6	F03A, 3BH	PTRD	F048, 49H		

You can check whether or not the work area for a particular peripheral device or a buffer is reserved in the following method: First, read the content of the pointer for the work area or buffer concerned, and read the content of the pointer for the work area or buffer which is reserved (at the higher adjacent address) next to the former work area or buffer (if you want to check EXDEV1, use F000H.) Then, compare the contents of these two pointers. If they are different, the work area or buffer concerned is reserved. If they are the same, it is not reserved.

(4) Work area for EXROM3

This work area is used for CE-1F01A (the barcode pen reader software). If CE-1F01A is not connected, you can use this memory area as a machine language program area.

How to reserve	Set A=(size to be reserved), then execute: CALL &02DD,A
How to release	Set A=0, then execute: CALL &02DD,A

(5) Variables area

The variables area is reserved in the memory locations lower than xx00H, next to the file buffer. The memory locations from xx00H to the beginning of the file buffer is not used.

Since the variables area is reserved in units of 256 bytes, reservation or expansion of work areas and buffers does not cause the user area (program and variables areas) to be reduced simply by that much.

6.3 WORK AREA MAP

The following shows the work area map used for the BASIC.

Address	Name	Contents
F02D	FBNO	MAXFILES value
F04C F04D	FBBP FBBP	Communication buffer start address (L) Communication buffer start address (H)
F04E F04F	FCBPTR FCBPTR	FCB buffer start address (L) FCB buffer start address (H)
F05C	DSPLPTR	LCD display start line
F05D	LCDWK1	LCD work area 1 Bit 0: LCD mode (fixed to "0") Bit 2: Character generator mode ("0"=PC-1600, "1"=PC-1500) Bit 3: Control characters ("0"=Not displayed, "1"=Displayed) Bit 4: Cursor blinking speed ("0"=slow, "1"=fast)
F05E	LCDWK2	LCD work area 2 Bit 0: Cursor blinking work area Bit 1: Interrupt request mask for LCD
F05F	CRSRY	Cursor X coordinate
F060	CRSRX	Cursor Y coordinate
F061 F062	CTRCGA CTRCGA	Start address (L) of CG table for control characters Start address (H) of CG table for control characters
F063	CRTCGB	Bank number of CG table for control characters
F064 F065	UPACGA UPACGA	Start address (L) of CG table for character codes from 80H to FFH Start address (H) of CG table for character codes from 80H to FFH
F066	UPACGB	Bank number of CG table for character does from 80H to FFH
F067	CRSRST	Cursor type "00H"=Cursor display off "01H"=Underline cursor "02H"=Square cursor "03H"=Space cursor
F068	CBLCTR	Cursor blinking counter
F079	KEYWK1	Key work area 1 Bit 1: Clicking sound ("0"=OFF, "1"=ON) Bit 2: Repeat ("0"=OFF, "1"=ON) Bit 3: Repeated key ("0"=Other than special keys, "1"=All keys) Bit 4: Repeat delay time ("0"=1 sec., "1"=0.8 sec.) Bit 5: Bit 6: Bit 7: Key code conversion ("0"=ON, "1"=OFF)
F07A	KEYWK2	Key work area 2
F07B	KEYWK3	Key work area 3
F182H	PITCHX	Character pitch value of PITCH command
F183H	PITCHY	Line spacing value of PITCH command

Address	Name	Contents
F184H	COLORP	Bits 0 to 3: Pen color of plotter/printer Bits 4 to 7: Must not be changed
F185H	WIDTH	Characters per line
F187H	FLAGA	Bit 0: Fixed to 1 Bits 1 to 4: Must not be changed Bits 5 and 6: Line-feed code specification bit 6 = "1" and bit 5 = "1": LF bit 6 = "1" and bit 5 = "0": CR+LF bit 6 = "0" and bit 5 = "1": CR
F188H F189H	CUSZL CUSZH	Scissoring counter (L) Scissoring counter (H)
F18FH F190H	SZXRL SZXRH	X direction right end scissoring area (L) X direction right end scissoring area (H)
F191H F192H	SZXL SZXH	X direction left end scissoring area (L) X direction left end scissoring area (H)
F194H	INZONE	Pen position (Number of characters counted from the left end)
F88FH	OUTPUT BUFFER POINTER	Pointer for the output buffer
F890H	FOR POINTER	Stack pointer for FOR...NEXT
F891H	GOSUB POINTER	Pointer for GOSUB
F894H	STRING BUFFER POINTER	Pointer for the string buffer
F895H	USING F/F	USING format (control of decimal point and comma)
F896H	USING M	Integer part of USING
F897H	USING &	USING for string
F898H	USING m	Decimal point of USING
F899H	VARIABLE POINTER H	Pointer for variables
F89AH	VARIABLE POINTER L	
F89BH	ERL	Error number of the error occurred
F89CH	CURRENT LINE H	Current program line number
F89DH	CURRENT LINE L	
F89EH	CURRENT TOP H	Start address of the program of the current line
F89FH	CURRENT TOP L	
F8A6H	SEARCH ADDRESS H	Address of the line found in the search operation
F8A7H	SEARCH ADDRESS L	
F8A8H	SEARCH LINE H	Line number of the line found after the search operation
F8A9H	SEARCH LINE L	
F8AAH	SEARCH TOP H	Start address of the program block searched
F8ABH	SEARCH TOP L	

Work area for CE-1600P

WORK AREA USED FOR BASIC

Address	Name	Contents
F8ACH	BREAK ADDRESS H	Address where the break occurred
F8ADH	BREAK ADDRESS L	
F8AEH	BREAK LINE H	Line number where the break occurred
F8AFH	BREAK LINE L	
F8B0H	BREAK TOP H	Start address of the program block where the break occurred
F8B1H	BREAK TOP L	
F8B2H	ERROR ADDRESS H	Address where the error occurred
F8B3H	ERROR ADDRESS L	
F8B4H	ERROR LINE H	Line number where the error occurred
F8B5H	ERROR LINE L	
F8B6H	ERROR TOP H	Start address of the program block where the error occurred
F8B7H	ERROR TOP L	
F8B8H	ON ERROR ADDRESS H	Address to which control jumps when an error occurs
F8B9H	ON ERROR ADDRESS L	
F8BAH	ON ERROR LINE H	Line number to which control jumps when an error occurs
F8BBH	ON ERROR LINE L	
F8BCH	ON ERROR TOP H	Start address of the program block where the error occurred
F8BDH	ON ERROR TOP L	
F8C0-F8CF	ADOLAR	Content of variable A$
F8D0-F8DF	BDOLAR	Content of variable B$
F8E0-F8EF	CDOLAR	Content of variable C$
F8F0-F8FF	DDOLAR	Content of variable D$
F900-F907	AVAR	Content of variable A
F908-F90F	BVAR	Content of variable B
F910-F917	CVAR	Content of variable C
F918-F91F	DVAR	Content of variable D
F920-F927	EVAR	Content of variable E
F928-F92F	FVAR	Content of variable F
F930-F937	GVAR	Content of variable G
F938-F93F	HVAR	Content of variable H
F940-F947	IVAR	Content of variable I
F948-F94F	JVAR	Content of variable J
F950-F957	KVAR	Content of variable K

Address	Name	Contents
F958-F95F	LVAR	Content of variable L
F960-F967	MVAR	Content of variable M
F968-F96F	NVAR	Content of variable N
F970-F977	OVAR	Content of variable O
F978-F97F	PVAR	Content of variable P
F980-F987	QVAR	Content of variable Q
F988-F98F	RVAR	Content of variable R
F990-F997	SVAR	Content of variable S
F998-F99F	TVAR	Content of variable T
F9A0-F9A7	UVAR	Content of variable U
F9A8-F9AF	VVAR	Content of variable V
F9B0-F9B7	WVAR	Content of variable W
F9B8-F9BF	XVAR	Content of variable X
F9C0-F9C7	YVAR	Content of variable Y
F9C8-F9CF	ZVAR	Content of variable Z
F9D1H	OPN DV	Specification of peripheral device
F9E0H	USER COUNTER XH	Counter to specify the pen position X coordinate
F9E1H	USER COUNTER XL	
F9E2H	USER COUNTER YH	Counter to specify the pen position Y coordinate
F9E3H	USER COUNTER YL	
F9E4H	SCISSORING COUNTER YH	Scissoring counter for Y direction
F9E5H	SCISSORING COUNTER YL	
F9E6H	ABSOLUTE POSSITION X	X direction absolute position counter
F9E7H	SCISSORING COUTNER XL	Scissoring counter for X direction
F9E8H	SCISSORING COUTNER XH	
F9EAH	LINE TYPE	Line type
F9EBH	DOT LINE COUNTER	Dotted-line counter
F9ECH	UP/DOWN	Pen up/down state
F9EDH	X MOTOR HOLD COUNTER	X motor hold counter
F9EEH	PORT C	Current motor phase

Work area for CE-150

WORK AREA USED FOR BASIC

Address	Name	Contents
F9EFH	Y MOTOR HOLD COUNTER	Y motor hold counter
F9F0H	GRAPH/TEXT	Printer mode specification ("255"=Graphics mode, "0"=Text mode)
F9F2H	ROTATE	Printing direction specification
F9F3H	COLOR	Color specification
F9F4H	CSIZE	Print character size specification
F9E0H F9E1H	ABSXL ABSXH	Pen physical position in X direction (L) Pen physical position in X direction (H)
F9E2H F9E3H	OVRXL OVRXH	X direction scissoring counter (L) X direction scissoring counter (H)
F9E4H F9E5H	OVRYL OVRYH	Y direction scissoring counter (L) Y direction scissoring counter (H)
F9E6H F9E7H	SZMYL SZMYH	−Y direction scissoring area (L) −Y direction scissoring area (H)
F9E8H F9E9H	SZPYL SZPYH	+Y direction scissoring area (L) +Y direction scissoring area (H)
F9EAH F9EBH	SRXL SRXH	Pen position X coordinate in graphics mode where the coordinate origin is specified by SORGN (Range: −4069 to +4069. A negative value is expressed as a complement of 2.) (L) Pen position X coordinate in graphics mode where the coordinate origin is specified by SORGN (Range: −4069 to +4069. A negative value is expressed as a complement of 2.) (H)
F9ECH F9EDH	SRYL SRYH	Same as SRXL except this is for Y coordinate Same as SRXH except this is for Y coordinate
F9EFH	MODE	Plotter/printer mode Bit 0: "0"=Text mode, "1"=Graphics mode Bit 1: "0"=Cur sheet, "1"=Roll paper Bits 2 to 4: Must not be changed Bit 5: "0"=Printer ready, "1"=Printer not ready Bit 6: "0"=Pen not in exchange state "1"=Pen in exchange state Bit 7: "0"=The printer hardware is not initialized. "1"=The printer hardware is initialized.
F9F4H	CHR	Value set by ROTATE statement Bits 4 to 7: ROTATE 0 to 3 Bits 0 to 3: DIRECTION (pen movement) 0 to 3
F9F5H	CSIZEP	Value set by CSIZE statement Bits 0 to 3: CSIZE 1 to 9 Bits 4 and 5: Must not be changed Bit 6: Fixed to "0" Bit 7: Fixed to "0"
F9F6H	LINE	Line type Bits 0 to 3: Line type 0 to 9 Bits 4 to 7: Must not be changed
F9F7H	ZONE	Value set by PZONE statement

Work area for CE-1600P

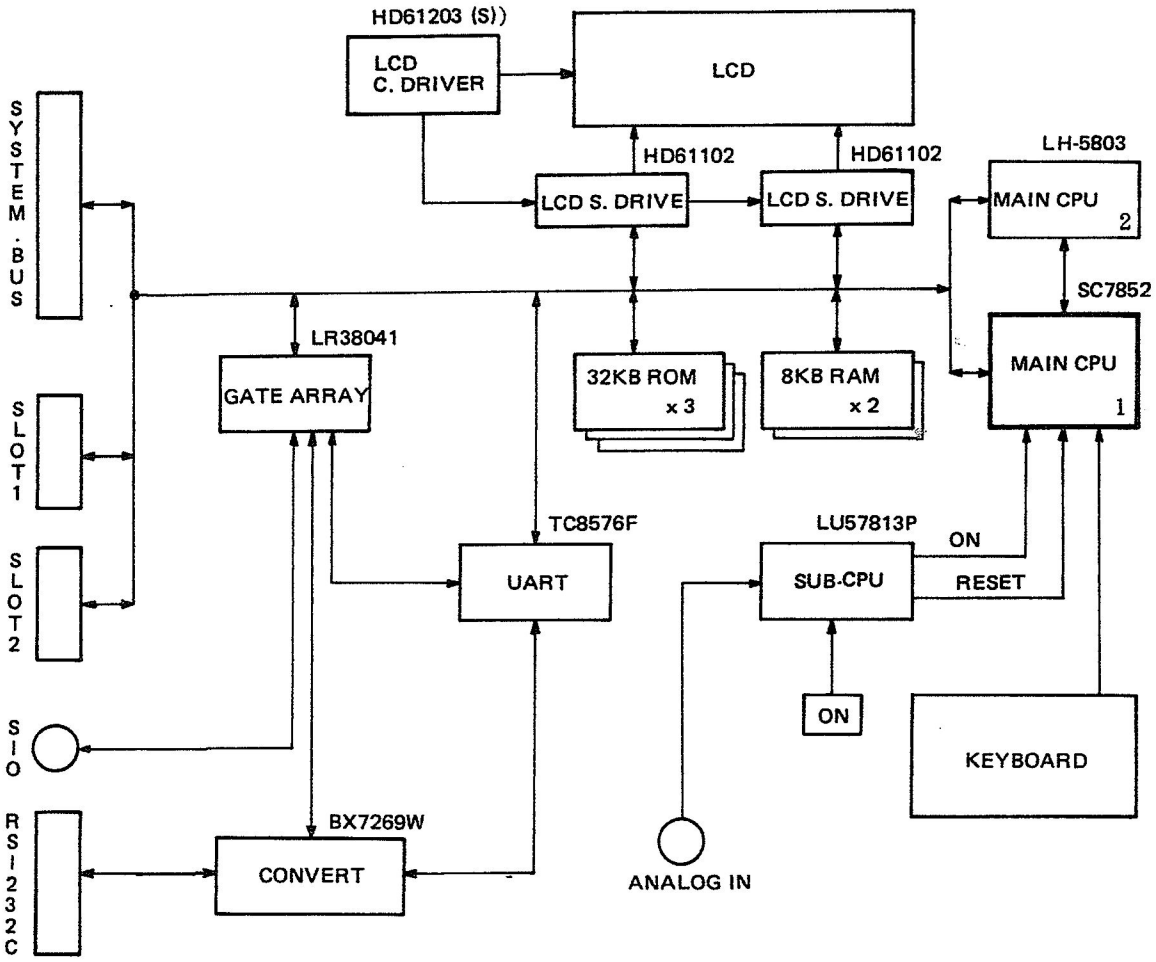
Address	Name	Contents
F9F8H	PWORK	Special work area Bit 0: "1"=For LLIST, the list is printed in characters of the special size. "0"=Normal mode Bit 1: "1"=The pen is not lifted up after an LLINE or RLINE statement is executed. (This state is given when line type "20" is specified in an LLINE or RLINE statement.) "0"=Normal mode Bit 2: Fixed to "0" Bit 4: Must not be changed Bit 5: "1"=The pen is not moved when paper is fed by the [↑] key. "0"=Normal mode Bit 6: "1"=The scissoring in the -Y direction is not performed when the printer is in the graphics mode and roll paper is used. "0"=Normal mode Bit 7: Fixed to "0"
F9FFH	LOCK	LOCK/UNLOCK specification
FB00H	RND NUMBER	For generation of a random number
FB01H	RND NUMBER	
FB02H	RND NUMBER	
FB03H	RND NUMBER	
FB04H	RND NUMBER	
FB05H	RND NUMBER	
FB06H	RND NUMBER	
FB07H	RND NUMBER	

CHAPTER 7

PC-1600 HARDWARE

The PC-1600 has three CPUs and a large memory, however, it has been made very compact with use of a gate array and a one-chip CPU. The following block diagram shows the PC-1600 hardware architecture. This chapter describes the details of each hardware component of the PC-1600.

PC-1600 block diagram



7.1 CPU

The PC-1600 has three CPUs: two main CPUs and one sub-CPU. The two main CPUs are SC-7852 (equivalent to Z-80A: 3.58 MHz clock) and LH-5803 (equivalent to LH-5801: 1.3 MHz clock). The sub-CPU is LU-57813P, which is a 4-bit CMOS processor of 307.2 KHz clock.

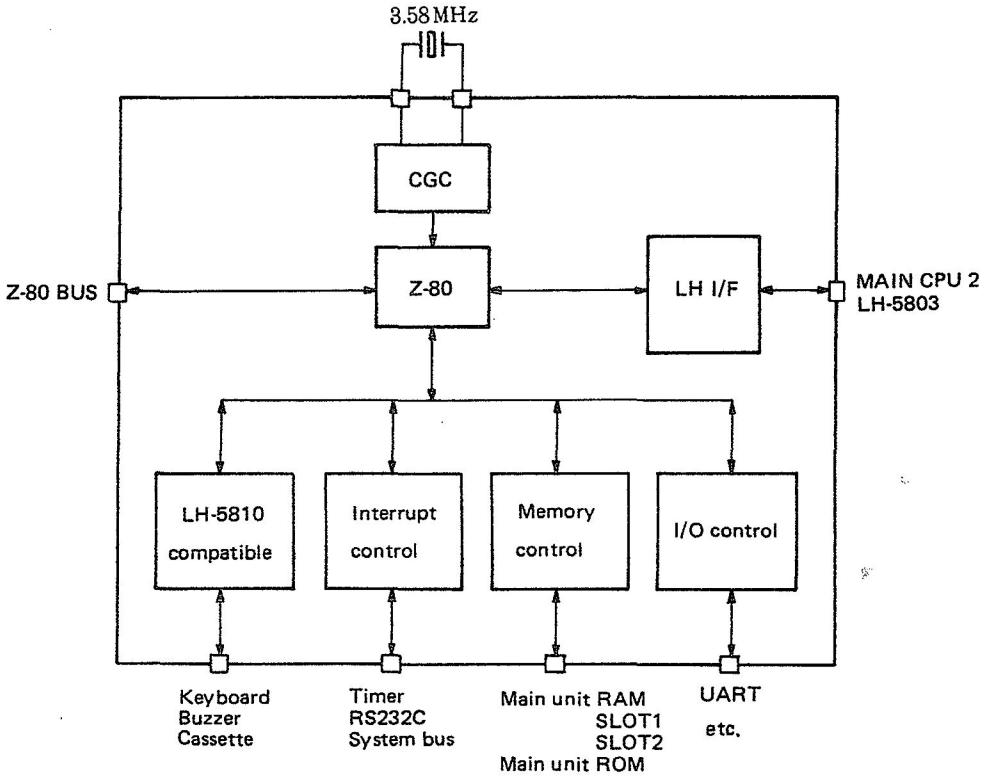
7.1.1 Specifications of SC-7852

SC-7852 has been developed specially for PC-1600 and contains a Z-80A equivalent circuitry and interfaces.

(1) Internal block diagram

SC-7852 is a one-chip CPU consisting of a Z-80A equivalent circuitry and interfaces.

SC-7852 Internal Block Diagram



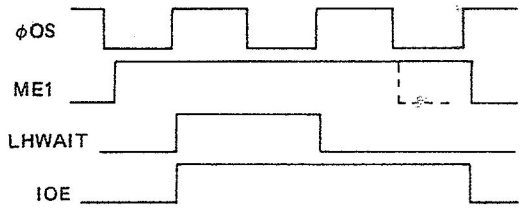
As shown in the block diagram, this chip contains Z-80A, clock generator, memory select circuit, interrupt control circuit, I/O port, and interface circuit for LH-5803.

(2) Terminal signals

SC-7852 is a 100-pin LSI, having several terminals for those signals that Z-80A does not have. The table below describes the terminal signals of SC-7852.

SC-7852 Terminal Signals

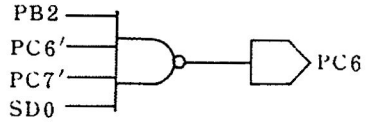
Pin No.	Symbol	In/Out	Active level	Function
95~ 100~2	$\overline{KIN0}$~$\overline{KIN7}$	▽ In	Low	(1) Internally pulled up to VCC by the resistor (200K ~ 5000K). (2) T input = Low (normal mode) keyboard input. A key in the low input line is pressed. (3) T input = High (emulation mode). Used for connection of the Z-80 ICE.
3	LHWAIT	Out	High	Wait output to the LH-5803. The signal goes high in one of the following: (1) When the WAIT input is at a high level. (2) When the LH-5803 accesses **0*H or 8000H~FFFFH of the ME1 space, it goes high for one cycle time to insert one wait. (3) When the Z-80 is running with the LH-5803 at halt.
4	φOS	In	⎍⎍	LH-5803 basic clock (1.3MHz). This clock is used for the sync signal of the internal LH-5810 corresponding port and generation of the LCD CLOCK (217KHz).
5	PT	Out		Memory bank signal.
6	PU	Out		Memory bank signal.
7	PVOUT	Out		Memory bank signal.
8	PVIN	▲ In		LH-5803's PV signal input. As PV is kept in the floating state when the Z-80 is operating, it is internally pulled down by the resistor.
9	$\overline{WR}$	In/Out	Low	(1) When the Z-80 is in operation, the Z-80's $\overline{WR}$ is a direct output on this line. (2) When the LH-5803 is in operation, it becomes an input to enable R/W for the LH-5803.
10~25	A15~A0	▲ In/Out		(1) When the Z-80 is in operation, the Z-80 address bus is an output on this line. (2) When the LH-5803 is in operation, the LH-5803 address bus is an input on this line.
26~33	DB7~DB0	In/Out		Data bus.
34	IORQ	▲ In/Out		(1) When the Z-80 is in operation, the Z-80 $\overline{IORQ}$ is an output on this line. (2) When the LH-5803 is in operation, the LH-5803 ME1 is an input on this line.
35	MREQ	▲ In/Out		(1) When the Z-80 is in operation, the Z-80 $\overline{MREQ}$ is an output on this line. (2) When the LH-5803 is in operation, the LH-5803 ME0 is an input on this line.
36	RD	In/Out		(1) When the Z-80 is in operation, the Z-80 $\overline{RD}$ is an output on this line. (2) When the LH-5803 is in operation, the LH-5803 OD is an input on this line.
37	WAIT	▲ In	High	WAIT input to the Z-80 and LH-5803. Pulled down internally by a resistor.
38	LHA90	Out		Among the RAMs (the bank of the spaces C000H~FFFFH) connected to the RAM3, it is an input to the address A9 of the RAM of E000H~FFFFH (the side A13A is input to $\overline{CE1}$). (1) When the Z-80 is in operation, "LHA90 = A9" is established. (2) Except that "LHA90 = high" is established when the LH-5803 accesses 7400H~744FH and 7500H~754FH. In other words, when the LH-5803 tries to access 7400H~744FH and 7500H~754FH, it actually accesses 7600H~764FH and 7700H~774FH.

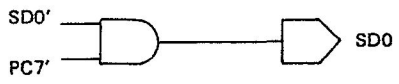
Pin No.	Symbol	In/Out	Active level	Function
39	$\overline{M1}$	Out	Low	(1) When the Z-80 is in operation, the Z-80 $\overline{M1}$ is an output on this line. (2) When the LH-5803 is in operation, the signal created from the OPF signal of the LH-5803 is sent on this line.
40	$\overline{RFSH}$	Out	Low	Refresh signal. (1) The Z-80 $\overline{RFSH}$ signal is on this line. (2) When the LH-5803 is in operation, the signal created from the OPF signal of the LH-5803 is sent on this line.
41	VDD			VCC
42	$\overline{IOE}$	Out	High	This signal is issued when the LH-5803 tries to access **00H~**0FH and 8000H~0FFFH of the ME1 space. When this signal is sent out, one wait is sent to the LH-5803. In terms of timing, the signal is sent with a half clock delay on the ME1. 
43	$\overline{CS001}$	Out	Low	Z-80 control ROM select signal. 0000H~7FFFH memory space (bank 0).
44	$\overline{CS123}$	▽Out	Low	(1) Z-80 control ROM select signal. 8000H~BFFFH memory space (bank 6). (2) LH-5803 control ROM select signal. C000H~FFFFH memory space.
45	$\overline{CS24}$	Out	Low	Z-80 control ROM select signal. 4000H ~ 7FFFH memory space (bank 3).
46 47 48	$\overline{LHS3}$ $\overline{LHS2}$ $\overline{LHS1}$	Out ▽ Out ▽ Out	Low Low Low	Memory select signal. Depending on the state of bit "6" of I/O 3CH, the memory space selected differs.

	b6 = 0	b 6 = 1
$\overline{LHS1}$	A800H~AFFFH (bank 0)	B000H~B7FFH (bank 0)
$\overline{LHS2}$	B000H~B7FFH (bank 0)	A800H~AFFFH (bank 0)
$\overline{LHS3}$	B800H~BFFFH (bank 0)	A000H~A7FFH (bank 0)

$\overline{LHS1}$ and $\overline{LHS2}$ are pulled up internally.
$\overline{LHS3}$ needs to be pulled up externally. (pulled up externally.)

49	RAM3	Out	High	Memory select signal (internal 16KB RAM). C000H~FFFFH (bank 0).
50	$\overline{RAM2}$	Out	Low	Memory select signal (S1:). 8000H~BFFFH (bank 0, bank 1). 8000H~BFFFH (bank 2, bank 3).
51	$\overline{RAM1}$	Out	Low	Memory select signal (S2:). 8000H~BFFFH (bank 2, bank 3).
52	SLCT	In	High	When this signal is at low, output of the memory and I/O select signal is disabled. Disabled signals are: $\overline{CS001}$, $\overline{CS123}$, $\overline{CS24}$, RAM3, $\overline{RAM2}$, $\overline{RAM1}$, IOE, $\overline{IOSU}$, $\overline{KA2}$, $\overline{KA1}$, $\overline{KA0}$, C/D, and $\overline{IORP}$. This input is an output to the subcontroller and is at a high level when the system is on.

Pin No.	Symbol	In/Out	Active level	Function
53	$\overline{KA2}$	Out	Low	Goes low when the Z-80 I/O 28H~2FH is written.
54	$\overline{KA1}$	Out	Low	Goes low when the Z-80 I/O 28H~2FH is read.
55	$\overline{KA0}$	Out	Low	Goes low when the Z-80 I/O 60H~6FH is accessed.
56	CK0	Out		A 217KHz ϕOS output. This signal is supplied to the HD61203 (S) LCD driver. This signal is issued only when bit "b4" of the Z-80 I/O 37H is at "1". Bit "b4" is at "0" at power-on, but turns to "1" in the power-on routine to activate the LCD.
57	$\overline{IORP}$	Out	Low	Goes low when the Z-80 reads 33H of I/O. This signal is used by the Z-80 to read the return data from the LU57813P.
58	C/$\overline{D}$	Out	High	Goes high when the Z-80 writes 3DH of I/O. Data are latched at a low to high transition of C/$\overline{D}$. When the signal rises with a half clock delay from IORQ, the data bus is stable.
59	$\overline{IOSU}$	Out	Low	Goes low when the Z-80 I/O 20H~27H is accessed. This signal is used for selection of the TC8576F UART.
60	E	Out	High	Goes high when the Z-80 I/O 40H~5FH is accessed. This signal is used to interface with the 6800 series LSI and is connected to the HD61202 LCD driver input. This signal is issued with a half clock delay slower than IORQ.
61	DMF0	Out	High	LH-5803 memory select signal. This signal goes high when the LH-5803 accesses the memory.
62~70	PA0~PA7	▽ In/Out		Corresponds to the port PA of the LH-5810 I/O port. This signal is used for the key strobe signal. To restore the original state of the low-forced strobe signal, this signal must be turned high and then set in the input mode. The input signal is pulled up internally.
65	VSS			±0V
71	PB2	▽ In		Used for the cassette tape to reproduce a signal. Pulled up internally.
72	PB5	▽ In/Out		Used for an input port by the PC-1600. Input to this line is a 1/64 second pulse which is issued from the LU57813P sub-controller. Pulled up internally.
73	PB6	▽ In/Out		Used for the key strobe signal. Application is the same as for the PA7~PA0. Pulled up internally.
74	PB7	▲ In		Receives the state of the BREAK/ON key sent from the subcontroller. Pulled down internally.
75	PC6	Out		Used by the Z-80 for a beep generation. The following circuit is internally composed in the LSI.  When either the PB2, PC6', PC7', or SD0 goes low, PC6 becomes high. To drive the buzzer, one of signals issues a pulse.

Pin No.	Symbol	In/Out	Active level	Function
				PB2: Cassette reproducing signal. PC6': Beep disable signal. PC7': Cassette recording signal (PC-1600). SD0': Cassette recording signal (PC-1500).
76	SD0	Out		Cassette recording signal output.  SD0' is the cassette recording output by the CE-150. PC7' is the cassette recording output by the CE-1600P.
77	$\overline{\text{ELH}}$	Out		(1) A low state of this signal indicates that the LH-5803 is in operation. (2) A high state of this signal indicates that the Z-80 is in operation.
78	PCSTB	In/Out		(1) Goes into the input mode when reset. This current state is latched in the PB3 flip-flop. In PC-1600, this terminal is pulled up through an external resistor. (2) Goes to the output line in the normal mode. The signal goes high when the Z-80 writes 18H or I/O or the LH-5803 is F008H of the ME1. This signal is not used in the output mode with the PC-1600.
79	$\overline{\text{RSTIN}}$	In	Low	A reset input to the SC7852. This signal is forced low for 30 milliseconds by the sub CPU when ACL or RESET is issued or at power-on.
80	IRQ	▲ In	High	An interrupt to the CPU (Z-80, LH-5803). This line is input as an interrupt request from the PC-1500 peripheral.
81	INT0	In	High	An interrupt to the CPU. This line is input as an interrupt request from the T8576F.
82	$\overline{\text{INT1}}$	▽ In	Low	An interrupt to the CPU. This line is input as an interrupt request from the PC-1600 peripheral. Pulled up internally.
83	$\overline{\text{INT4}}$	In	⌐↓	An interrupt to the CPU. An interrupt is sent to the CPU at a high to low transition. This line is input at a 1/64 second pulse from the sub CPU. It is externally shorted with PB5. But, the sub CPU output, which is a P-ch open drain, is pulled down by the external resistor to assure a low output.
84	INT6	▲ In	High	An interrupt to the CPU. This line inputs the output from the sub CPU.
85	PCTRL	Out	Low	At the time the power-off command is sent to the sub CPU, the sub CPU turns the power off (active low). This signal goes low after the Z-80 completes the following: (I) 11H written to I/O 37H (II) OUT (38H), A (III) HALT
86	CLK	Out		Z-80 clock output. 3.58MHz for the PC-1600.
87	T	▲ In		(1) It is in the normal mode when a low signal is received and the Z-80 is operating normally. Pulled down internally. (2) It is in the simulation mode when a high signal is received. The Z-80 bus is in the floating state, and the Z-80 (or Z-80 ICE) can be connected externally.
88 89	XOUT XIN	Out In		The 3.58MHz Z-80 clock is supplied when the oscillator is attached across these lines.
90	VDD			Power input to the high side (4~5.5V).

(3) I/O map

Similar to Z-80A, SC-7852 has a 256-byte I/O space from 00H to FFH. The table below shows the I/O map of SC-7852.

I/O Map of SC-7852

00H 0FH	Use prohibited.
10H 1FH	Port corresponding to LH-5810 (LH-5811) contained in the SC7852 (not synchronized with φOS).
20H 27H	TC8576F UART selection
28H 2FH	S2 (slot 2)
30H 3FH	SC7852 internal LSI control register port
40H 4FH	System reserve
50H 58H	HD61102 (IC2), (IC3) HD61102 (IC3)
5BH	HD61102 (IC2)
	System reserve
60H 6FH	S2 (slot 2)
78H 7FH	CE-1600F
80H 83H	CE-1600P
84H BFH	Reserved for future extension

Z-80 I/O address	LH-5803 address	Read	Write
30H	#A030H	IOR MOD	IOW MOD
31H	#A031H	IOR MAP	IOW MAP
32H	#A032H	IOR INT	IOW PRI
33H	#A033H	IOR P	IOW CDF
34H	#A034H	IOR LHMSK	IOW LHMSK
35H	#A035H	IOR ZMSK	IOW ZMSK
36H	#A036H	IOR ADRS	IOW CL1
37H	#A037H	IOR KB	IOW CGC CGC register write
38H	#A038H		IOW STP
39H	#A039H		IOW VCT
3AH	#A03AH		IOW KA Not used
3BH	#A03BH		IOW KS Not used
3CH	#A03CH		IOW SLT
3DH	#A03DH		IOW C/D
3EH	#A03EH		
3FH	#A03FH		

Note: When writing to an I/O space between 30H and 3DH, if the setting is incorrect, the PC-1600 does not operate properly.

NOTES:

(Rreg): Indicates the contents of the memory (ME1 accessed) which are implied by the LH-5803 CPU internal register (R register).

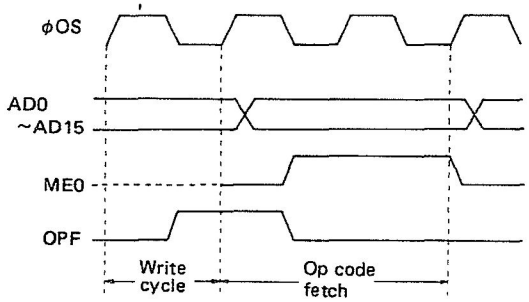
☐ : Vacancy in the Z-80 I/O map which is not used at present.

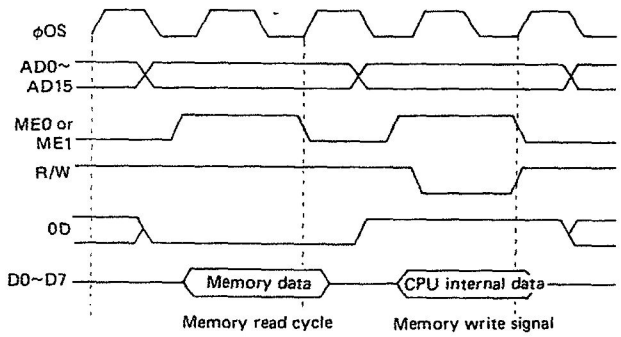
Note: In the machine cycle, 1 wait is automatically inserted.

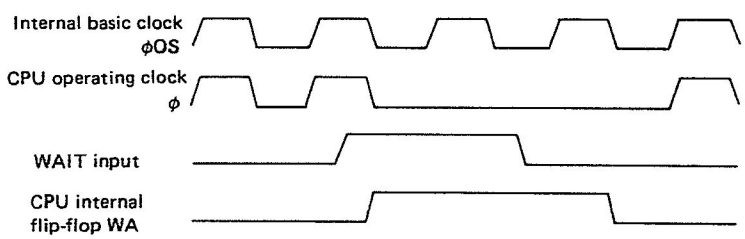
7.1.2 Specifications of LH-5803

LH-5803 is an 8-bit C-MOS CPU, which is an upper version of LH-5801. Therefore, LH-5803 supports almost all LH-5801 machine language instructions, except that the SDP, RDP and OFF instructions of LH-5801 operate as a NOP instruction in LH-5803.
The table below describes the terminal signals of LH-5803.

Main CPU 1 (LH5803) pin description

Pin No.	Symbol	In/Out	Active level	Function
1	RESET	In		CPU reset input. A high on this line causes the reset. The contents of the address FFFEH are transferred to the PH register and the contents of FFFFH to the PL register. When the reset input changes from high to low, the program starts to execute from the address set in the program counter.
2	(NC)	–		
3	BRQ	In		Bus request. Connected to $\overline{ELH}$ of the SC7852 output.
4	BFI	In		BF flip-flop output (BFO) and input (BFI). The BF flip-flop is reset by the OFF command of the CPU. It can be reset when the BFI is set high. The BFO is at a low level when the BF flip-flop is active and at a high level when not active. The contents of the BF flip-flop are protected as long as VGG is in supply. Because VGG is VCC in the PC-1600, this function is not used and VCC is used for an input.
5	VGG			Power supply (system's VCC input).
6	BFO	Out		See Pin No.4.
7	OPF	Out		Op code fetch signal which appears when the CPU fetches the OP code. OPF is the signal that is issued only when the operation code is fetched and is not therefore issued in fetching the address data, immediate data, and the second byte of a 2-step command. 
8	BAK	Out		Bus acknowledge signal. When BRQ is set at a high level, the CPU issues a high BAK state in response to it. When BAK is at a high level, the CPU sets the address bus (AD0 ~ AD15), data bus (D0~D7), ME0, ME1, R/W, and OD in high impedance.
9	VCC			Power supply (system's VCC input).

Pin No.	Symbol	In/Out	Active level	Function
10	VGG			Power supply (system's VCC input).
11	VM	In		LCD backplate power supply input.
12	VDis	In		LCD backplate power supply input. Not used by the PC-1600.
13	VA	In		LCD backplate power supply input.
14	VB	In		LCD backplate power supply input.
15	NMI	In		Non-maskable interrupt input. A high input state causes an interrupt to the CPU. The CPU unconditionally accepts the request and starts to execute the interrupt routine from the address whose high order address is represented by the contents of the address FFFCH and the low order address by the contents of FFFDH.
16	MI	In		Maskable interrupt input. When the IE flag (Interrupt Enable) is set on, an interrupt request is caused by a high M1 input state, and the CPU starts to execute the interrupt routine from the address whose high order address is represented by the contents of the address FFF8H and the low order address by the contents of FFF9H.
17	$\overline{HIN}$	In		Input to the counter by which the LCD and backplate signals, H0~H7, are generated. Normally connected to the HA pin of the CPU. With the PC-1600, this function is not used.
18	HA	Out		CPU internal divider output through which is delivered the basic clock for the LCD driver and connected to $\overline{HIN}$ and the segment signal generator LSI.
19	DISP	Out		LCD display on/off control signal output. Can be set and reset by means of a command. With the PC-1600, this function is not used.
20~27	H7~H0	Out		LCD backplate signal output. When the LCD is driven by the backplate signal and the segment signal, the backplate signal is issued by the CPU.
28	OD	Out		Output disable signal. When OD is at a high level, the CPU disables the data output onto the data bus for the external device. This signal is issued when writing data in the memory. 
29 30	ME0 ME1	Out Out		Memory enable signal. This signal is enabled to directly access the 128KB memory area; ME0 accesses a 64KB area and ME1 accesses a 64KB area. The memory area accessible by the program counter P and stack pointer S is 64KB, for ME0 is used by the fetch and stack commands. For accessing data, both ME0 and ME1 memory areas can be accessed by the CPU command.
31~38	D0~D7	In/Out		Bidirectional data bus which is used to write data in the external memory or to read data from the external memory.
39~46	A0~A7	Out		Address bus which may be in three states. Goes to high impedance with the BRQ (bus request) signal. It is possible to access the memory area of 64KB. It is also possible to access the memory of 128KB using the ME0 or ME1 signal.

47	GND			Power supply.
48	A8	Out		Address bus (see Pin No.39).
49	VGG			Power supply.
50~56	A9~A15	Out		Address bus (see Pin No.39).
57	(NC)	–		
58	R/W	Out		Memory write signal. With a low R/W state, the data in the CPU are sent on the data bus.
59	Pφ	Out		External latch clock. With a high state of this clock, the contents of the accumulator are transferred onto the data bus. Use of the latch IC permits its use as the output port (see the ATP command).
60 61	PV PU	Out Out		These are the CPU internal flip-flop output pins (PU, PV). There are commands to set and reset PU and PV.
62	φOS	Out		The clock, in the same phase as the CPU internal basic clock, is on this line to supply clock pulse to the external system. When a 2.6MHz crystal is connected across XL0 and XL1, a 1.3MHz clock is supplied.
63 64	XL0 XL1	In Out		Crystal connection pins. XL0 is an input and XL1 is an output. Inside the CPU, the clock is divided in half. When a 2.6MHz crystal is connected, the machine cycle within the CPU is at 1.3MHz.
65	WAIT	In		CPU wait signal. When this input is high, the CPU's internal operation clock "φ" stops and the CPU therefore stops executing a command. When it resumes a low state, the CPU starts to execute a command.  NOTE: WA is the CPU internal flip-flop for WAIT. At a high to low transition of the clock φOS, input of WAIT is accepted. The CPU operating clock φ stops when WA is at high; the CPU halts a command execution temporarily as a result.
66~73	IN7~IN0	In		Input port. The CPU can send the signal input on the IN0~IN7 to the CPU accumulator as an 8-bit data. It has an internal pull-up resistor. When not connected, the CPU assumes the line to be in high impedance.
74~76	(NC)	–		

NOTE: NC: No Connection

7.1.3 Specifications of LU57813P

LU57813P is a 4-bit CMOS CPU. This section first describes the functions of LU57813P, then shows the table of LU57813P terminal signals.

(1) PC-1600 main power ON/OFF control

LU57813P turns off the power of the PC-1600 by receiving a command from the main CPU, and turns off the power when the [ON] key is pressed.

(2) Timer management

LU57813P supports one wakeup timer and two alarm timers (counted up every 0.5 second) and manages the calendar clock.

(3) Battery voltage monitoring

LU57813P has an A/D converter, with which LU57813P monitors the supply voltages of the PC-1600 main unit and the peripheral devices. LU57813P turns on the BATT symbol on the LCD when the supply voltage goes below a certain level.

(4) Analog input

LU57813P converts an analog voltage input at the analog port of the PC-1600 into a digital value and passes the value to the main CPU.

(5) Key click sound generation

If the key clicking is enabled, when a key input occurs, the main CPU sends a command to make LU57813P generate a click sound.

(6) Reset signal handling

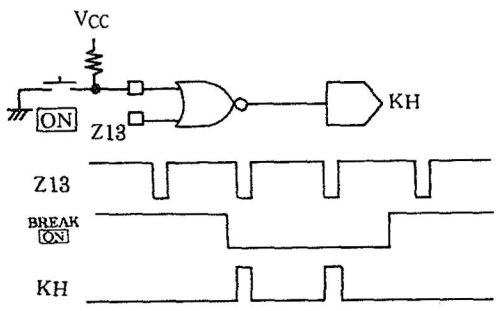
LU57813P manages a reset signal from two reset switches (one on the rear side of PC-1600 and the other on the rear side of CE-1600P). Receiving a reset signal from one of the reset switches, LU57813P sends a reset signal for 30 ms to the system.

(7) Receiving a CI signal from RS-232C, LU57813P turns on the system.

(8) Timer (1/64 second) signal output

Sub CPU (LU57813P) pin description

Pin No.	Symbol	In/Out	Active level	State at ACL	Function
1	Q0	In	Low	In	When the system-off command is received from the Z-80, the system is turned off after this signal goes low. It has PCTRL output from the SC7852 as its input.
2	VDD				High side VGG is supplied.
3	ACL	In	High		The pulse width of ACL must be greater than 1 microsecond in duration to be recognized by the hardware. It takes about 80 microseconds before the LSI starts to operate after input of ACL. This pin is used as reset input from the ALL RESET switch of the PC-1600.
4 5	CL1 CL2	In Out			The system clock generating ceramic oscillator is attached across these two lines. With the PC-1600, a 1.229MHz oscillator is used for the basic clock of the RS-232C baud rate.
6	FOUT	Out			System clock output. Not used.
7	P0	Out	Low	In	Reset input to the SC7852. This line is maintained low for 30 milliseconds during system-on and reset.
8	P1	Out	Low	In	In a low state when the main CPU is permitted to access the memory and I/O.
9	P2	Out	High	In	In an opposite level of P1. Input to SLCT of the SC7852.
10	P3	Out	Low	In	In a low state during system-on. Used to turn on the system.
11	KH	In	High		This signal goes high with an input of the ON key. When the system is off, this LSI is in the standby mode, and it turns on the system with a high KH state.
12	KI	In	High		A command request from the main CPU. Interrupt is caused by a high K1 state.
13	T	In			Test pin which is NC.
14 15	OSCOUT OSCIN	Out In			The 32.768 KHz timer crystal oscillator is attached across these lines.
16	KL	In	High		Reset input from the peripheral unit. As monitored by the software, if this input is high for more than the given.time, the reset is executed.
17	Z15	Out	High	In	Z15 and KL are shorted outside and externally pulled down by the resistor. Z15 is turned high for 1 millisecond in the reset routine to be converted into the RSTE signal, and sent to peripherals as the reset signal via the system bus. So, both Z15 and KL can be handled as an input/output line, which may be used to apply reset to the peripheral or to receive reset from the peripheral. This signal is used as the reset input of the CE-1600P.
18	Z14	Out	High	In	The sub CPU monitors the state of the BREAK/ON key via the KH input line and its state is sent through Z14 and supplied to PB7 of the SC7852. Therefore, key chattering and bouncing of the BREAK/ON key are completely controlled the sub CPU.
19	Z13	Out	Low	In	The sub CPU goes into the power-down mode except when one of the conditions mentioned below holds true. (1) When a command is received from the main CPU. (2) When a timer interrupt is received. (3) If the BREAK/ON key sensing KH input is at a high level. To prevent these conditions from occurring, Z13 is set low at every time interrupt (1/128 second). If KH is at a high level, depression of the BREAK/ON key is sensed.

Pin No.	Symbol	In/Out	Active level	State at ACL	Function
					
20	Z12	Out	Low	In	*For the PC-1600, a high signal state is normally issued (at ON).*
21	Z11	In		In	NC.
22	Z10	Out	High	In	This signal is used to interface with the main CPU. it goes high when the sub CPU waits for a command (ready), and goes low when busy.
23	Z9	Out	High	In	This signal is also used to interface with the main CPU. A high pulse is issued when the sub CPU terminates a command execution.
24	Z8	Out		In	Used to setup the analog input mode. (1) Voltage is A/D converted when low (initial value). The input impedance is 100K ohms. (2) Current is A/D converted when high. This signal goes low when the interrupt cause status is read.
25	Z7	Out	High	In	The sub CPU sets this signal high when the command specified interrupt has been acknowledged. This signal is connected to INT6 input of the SC7852. There are four causes which force this signal level to high. (I) The weke-up time matched the real-time timer. (II) The time of alarm-1 or -2 matched the real-time timer. (III) Receipt of an input from the external keyboard. (IV) At 0.5 second cycle of the real-time timer. This signal goes low when the interrupt cause status is read or when all inputs from the external keyboard have been read.
26	GND				±0V
27	Z6	Out	Low	In	A 1/64 second pulse of 50% duty is sent. Connected to $\overline{\text{INT4}}$ and PB5 inputs of the SC7852.
28	Z5	Out	High	In	NC.
29	Z4	Out	High	In	Shorted with the analog input KC1. Normally an open output.
30	Z3	In		In	Not used.
31	Z2	In		In	
32	Z1	In		In	
33	Z0	In		In	
34	SOUT	Out			
35	SCLOCK	In/Out		In	
36	F	Out			Used for generation of click and alarm sounds.
37	VRH	In			A/D conversion high side reference voltage (2.495V in supply).

Pin No.	Symbol	In/Out	Active level	State at ACL	Function
38	KC3	In			Not used.
39	KC2	In			Used for checking the CE-1600P power supply level. VPP supplied from the CE-1600P via the system bus is A/D converted. If it is below the given level, the peripheral is assumed to have a weak battery condition.
40	KC1	In			Used for checking the PC-1600 main power supply level. The level of the main power supply is A/D converted and checked. If it is below the given level, a weak battery condition is assumed.
41	KC0	In			Receives the signal input from the analog input connector. (1) For the analog input, A/D conversion is done. (2) For the external keyboard input, its logic level is interrogated.
42	R33	Out		In	MSB } Return data to the Z-80.
43	R32	Out		In	
44	R31	Out		In	
45	R30	Out		In	
46	R23	Out		In	
47	R22	Out		In	
48	R21	Out		In	
49	R20	Out		In	LSB
50	VRL				NC
51	SIN	In			NC
52	VDD				High power supply voltage level (VGG).
53	R13	In		In	MSB } Command from the Z-80.
54	R12	In		In	
55	R11	In		In	
56	R10	In		In	
57	R03	In		In	
58	R02	In		In	
59	R01	In		In	
60	R00	In		In	LSB
61	Q3	In	Low	In	Hardware sensed weak battery detection signal. A high on this line causes the CPU to force the system to go down. The only means to turn the system on after recovery of power supply is the depression of the BREAK/ON key or ALL RESET switch. The time in the real-time timer would not be revised.
62	Q2	In		In	Not used. Pulled up.
63	Q1	In	Low	In	Opposite polarity as CI of the RS-232C interface. It is possible with CI to turn on the system when the system is off (when this line is at a low level).
64	Q0	In	Low	In	If this signal is at a low level when the system-off command is received from the Z-80, the system is turned off.

7.1.4 Interface Between SC-7852 (Z-80) and LH-5803

The two CPUs are connected directly with each other through a bus. The bus signals of one CPU being in operation are output to the system bus while the bus signals of the other CPU in non-operation are not output to the system bus. (The two main CPUs cannot be operated at the same time.) The table below shows the signals of the bus to which the two CPUs are connected.

SC7852 signal name	Z-80 signal name	LH-5803 signal name
A15 ~ A0	A15 ~ A0	A15 ~ A0
DB7 ~ DB0	D7 ~ D0	D7 ~ D0
MREQ	Opposite polarity of $\overline{\text{MREQ}}$	ME0
IORQ	Opposite polarity of $\overline{\text{IORQ}}$	ME1
$\overline{\text{RD}}$	$\overline{\text{RD}}$	OD*
$\overline{\text{WR}}$	$\overline{\text{WR}}$	R/$\overline{\text{W}}$

Which CPU is currently in operation can be known by checking the state of the ELH signal.

$\overline{\text{ELH}}$ = low : LH-5803 is in operation.
$\overline{\text{ELH}}$ = high: SC-7852 is in operation.

When the system is reset, ELH goes high and SC-7852 starts operating.

To move control from SC-7852 to HL-5803, execute the following:

```
OUT (38H),A      where the content of A can be anything.
HALT
```

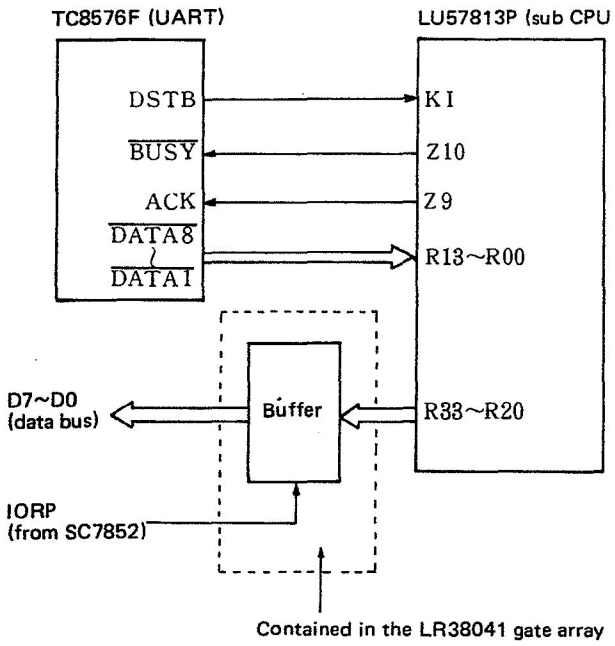
To move control from HL-5803 to SC-7852, execute the following:

```
STA #A038H
```

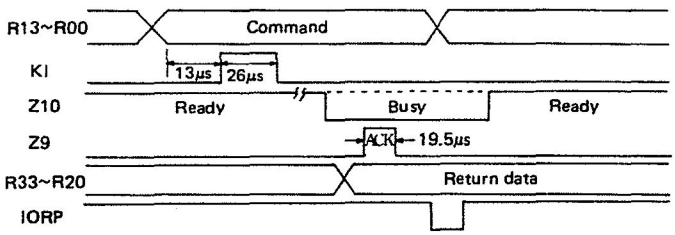
7.1.5 Interface Between Sub-CPU and Main CPU

Figure (a) shows the simplified diagram of the interface circuit between the sub-CPU and the main CPU. Figure (b) shows the timing of the signals shown in figure (a).

Fig. (a)



(Fig. b)



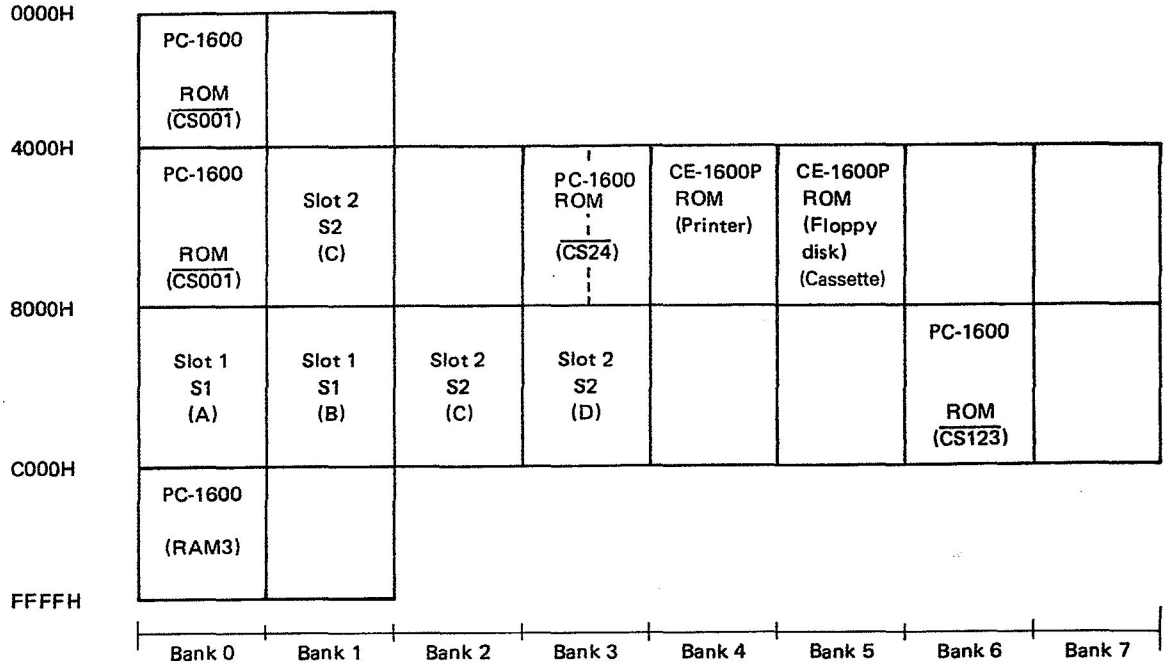
When SC-7852 sends a command to the sub-CPU, it waits until the sub-CPU becomes ready. SC-7852 presumes that the sub-CPU is ready if the BUSY signal of TC-8576F (UART) is high.
Then, within 500 us after a change of the 1/64S signal, SC-7852 sends a command data (8 bits) to the UART ports DATA1 to DATA8, and then sends the DSTB signal of UART. The sub-CPU receives this command data through the terminals R13 to R00, and processes the command by the interrupt service routine. There are two types of commands that SC-7852 sends to the sub-CPU:
(1) Command that requires a return data from the sub-CPU
(2) Command that does not require any return data from the sub-CPU
The sub-CPU judges the type of the command and performs one of the following processes.
For a command of type (1): When the command processing completes, the sub-CPU sends the return data to R33 to R20, then sends a pulse to Z9 to notify that the command execution has been completed.
For a command of type (2): Receiving a command, the sub-CPU sends a pulse to Z9.
SC-7852 waits until Z9 becomes high.
For reading the return data, SC-7852 performs a read operation to the I/O port 33H. The OPRP signal goes low and the return data is read into the bus signals D_0 to D_7 of SC-7852.

7.2 MEMORY

7.2.1 Memory Map Viewed from SC-7852 (Z-80)

The memory space that SC-7852 can directly access is 64 KB. In PC-1600, however, the memory space is expanded to 320 KB through the bank switching technology. The bank switching is accomplished by changing the content of the I/O port 31H.
The following diagram and table show the memory map viewed from SC-7852, and the SC-7852 access memory areas and the contents of I/O port 31H.

Memory Map viewed from SC-7852



SC-7852 Access Memory Areas and Contents of I/O Port 31H

Bank No.	Z-80 accessing space	Status in the I/O address 31H								PT	PU	PV OUT	
		b7	b6	b5	b4	b3	b2	b1	b0				
0	0000H~3FFFH	*	*	*	*	*	*	*	*	*	*	0	
1	†	*	*	*	*	*	*	*	1	*	*	1	
0	4000H~7FFFH	*	*	*	*	0	0	0	*	0	0	0	
1	†	*	*	*	*	0	0	1	*	0	0	1	
2	†	*	*	*	*	0	1	0	*	0	1	0	
3	†	*	*	*	*	0	1	1	*	0	1	1	
C	†	*	*	*	*	1	0	0	*	1	0	0	
5	†	*	*	*	*	1	0	1	*	1	0	1	
6	†	*	*	*	*	1	1	0	*	1	1	0	
7	†	*	*	*	*	1	1	1	*	1	1	1	
0	8000H~BFFFH	*	0	0	0	*	*	*	*	0	0	0	
1	†	*	0	0	1	*	*	*	*	0	0	1	
2	†	*	0	1	0	*	*	*	*	0	1	0	
3	†	*	0	1	1	*	*	*	*	0	1	1	
C	†	*	1	0	0	*	*	*	*	1	0	0	
5	†	*	1	0	1	*	*	*	*	1	0	1	
6	†	*	1	1	0	*	*	*	*	1	1	0	
7	†	*	1	1	1	*	*	*	*	1	1	0	
0	C000H~FFFFH	0	*	*	*	*	*	*	*	*	*	0	
1	†	1	*	*	*	*	*	*	*	*	*	1	

*: DON'T CARE

7.2.2 Memory Chip Select Signals

There are six memory-chip select signals.

(1) $\overline{\text{CS001}}$ signal

When bank 0 in the memory space from 0000H to 7FFFH is accessed, the CS001 signal goes low. This signal is entered to the CS signal of ROM.

(2) $\overline{\text{CS123}}$ signal

When bank 6 in the memory space from 8000H to BFFFH is accessed, the CS123 signal goes low. This signal is entered to the CS signal of ROM. The ROM to be selected by CS001 and CS123 signals can be made not to be selected by setting the INH terminal (which is entered to the system bus terminal) to high. (In the PC-1600 main unit, the INH terminal is pull down to the ground and entered to the OE signal of ROM.)

(3) $\overline{\text{CS24}}$ signal

When bank 3 in the memory space from 4000H to 7FFFH is accessed, the CS24 signal goes low. This signal is entered to the CS signal of 256K-bit ROM, and to the OE signal of this ROM is entered the A15 signal. This 16 KB of memory space is expanded to 32 KB by the A16A signal.

(4) RAM3 signal

When bank 0 in the memory space from C000H to FFFH is accessed, the RAM3 signal goes high. This signal is entered to the CE2 signal of each of the two 8KB RAMs.

(5) RAM2 signal

The RAM2 signal is a memory select signal for the memory slot 1 (S1). This signal goes low when bank 0 or 1 in the memory space from 8000H to BFFFH is accessed.

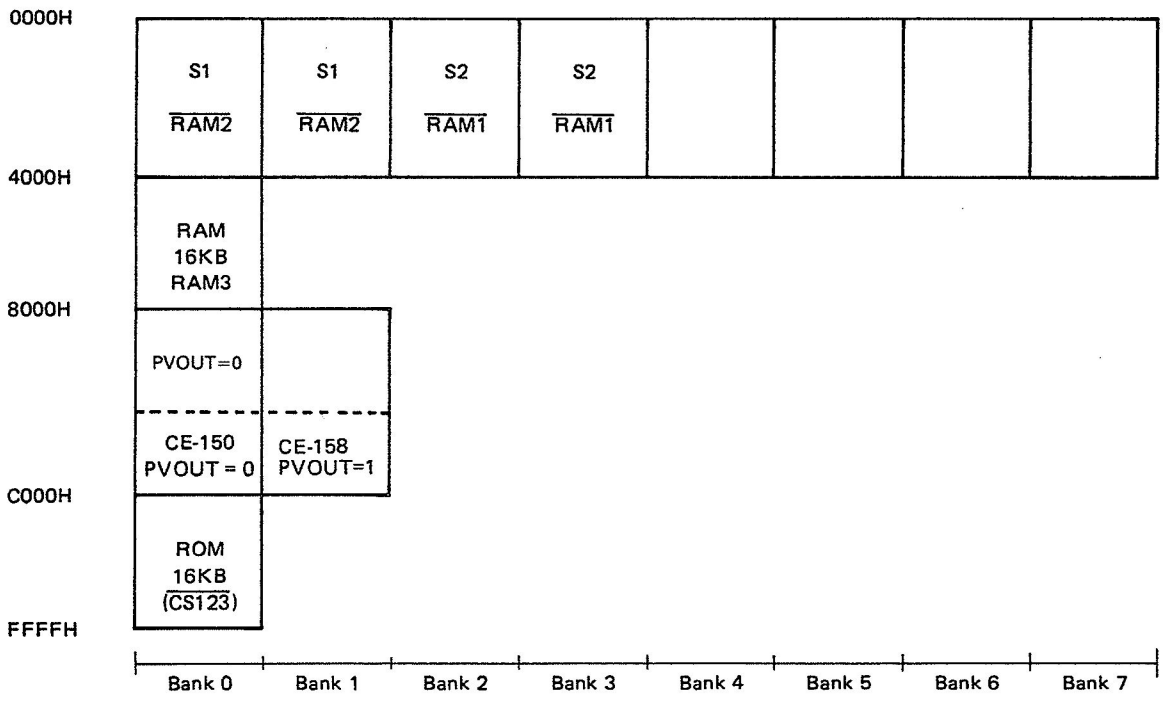
(6) $\overline{\text{RAM1}}$ signal

The RAM1 signal is a memory select signal for the memory slot 2 (S2). This signal goes low when bank 2 or 3 in the memory space from 8000H to BFFFH is accessed.

7.2.3 Memory Map Viewed from LH-5803

The following diagram shows the memory map viewed from LH-5803.

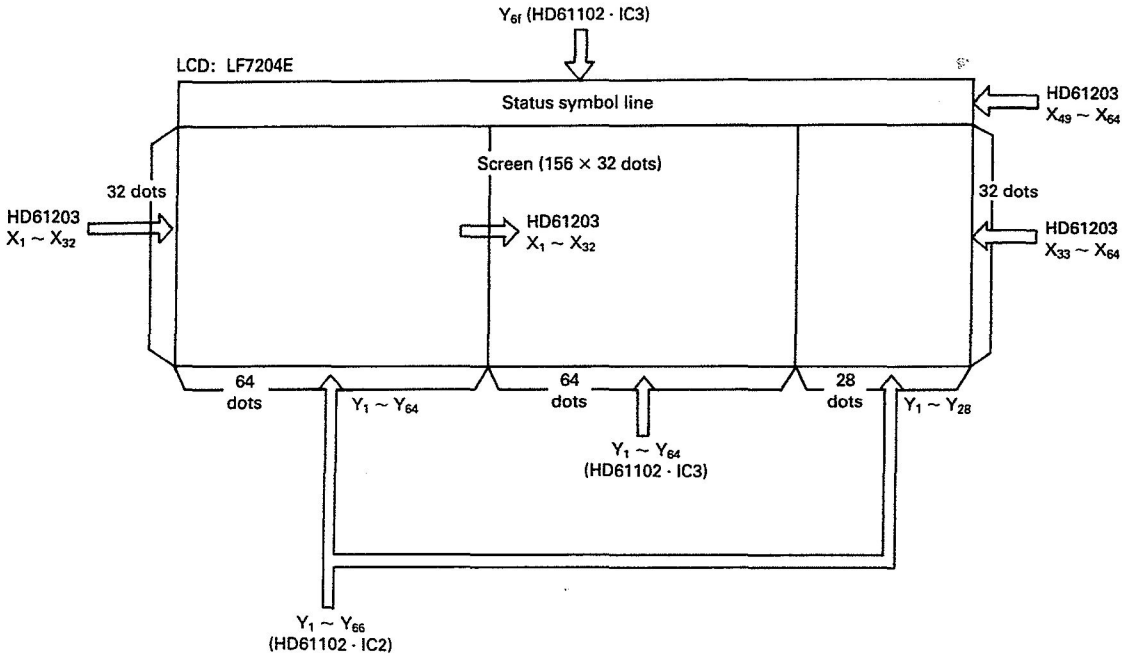
Memory Map Viewed from LH-5803



The memory space from 0000H to 3FFFH is the same as the SC-7852 memory space from 8000H to BFFFH, and the accessing method is also the same.
The memory space from 4000H to 7FFFH is the same as the bank 0 of SC-7852 memory space from C000H to FFFFH.
The memory space from 8000H to FFFFH is the same as the PC-1500 memory space from 8000H to FFFFH. That is, the bank selection of 8000H to BFFFH is accomplished by the PV signal of LH-5803.

7.3 LCD

The LCD (liquid crystal display) used in PC-1600 consists of a screen (156 × 32 dots) and a status symbol line (16 symbols) and is operated in 1/64 duty.
The LCD is driven by HD61203 and HD61102. The diagram below shows the connection between the LCD and the two drivers.



The LCD uses the basic clock of 217 KHz, which is supplied from the CK0 terminal of SC-7852.
The LCD driver HD61102 has three chip-select signals (CS1, CS2 and CS3). This driver is selected only when all of three select signals are enabled. The table below shows the I/O address space allocated for HD61102.

	HD61102 (IC2)	HD61102 (IC3)
$\overline{CS1}$	A2	A3
$\overline{CS2}$	A5	A5
CS3	A4	A4
Selected I/O space	50H ~ 53H 58H ~ 5BH	50H ~ 53H 54H ~ 57H

7.4 KEYBOARD

The keys on the keyboard are scanned by interrupting SC-7852 every 1/64 second by the 64 Hz pulse supplied from the Z6 terminal of the sub-CPU LU57813P.

7.5 BUZZER

The built-in buzzer can be activated by either of two signals: PC6 signal of SC-7852 and F signal of LU57813P. These signals are connected to the buzzer as follows.

The buzzer beeps when either of these signals vibrates. To the PC6 is connected the cassette playback signal (PB2), the cassette record signal, the BEEP ON signal of BEEP command (PC7), and CE-150 or CE-162E record signal (SD0'). When a BEEP OFF statement is executed, the PC6 signal is held at high so that the buzzer cannot vibrate. When the buzzer is in the silent state, the PC6 signal is in the high state. To the F signal is connected the clicking ON signal, the wakeup ON signal and the alarm ON signal. When the buzzer is in the silent state, the PC6 signal is in the low state.

7.6 RS-232C/SIO INTERFACE

The PC-1600 has two serial ports, RS-232C and SIO, and has an LSI of TC8576F (UART) for the interface of these serial ports. The table below shows the terminal signals of TC8576F (UART). Since TC8576F has only one serial interface, RS-232C and SIO cannot be used at the same time. Which serial port is used is specified by an OPEN or SETDEV statement in BASIC, or by the PRIM signal.
When the PRIM signal is set to high, RS-232C is enabled, the RS-232C interface supply voltage VDD is supplied, both SDF and RDF go low, and the TXD and RXD signals of UART are respectively sent, with their polarity inverted, to the TXD and RXD signal lines of RS-232C.
When the PRIM signal is set to low, SIO is enabled, the VDD stops being supplied, and the TXD and RXD signals of UART are respectively sent, with their polarity inverted, to the SDF and RDF signal lines of SIO. The RS-232C output signals are held at the high impedance state or at the low level. When the system is powered on or reset, the PRIM signal is set to low.

Note: Use of the RS-232C port consumes more power than the SIO port. When your application program has finished using the RS-232C port, change the selection of the serial port to the SIO port to prevent the battery power from being wasted.

The TC8576P is a single chip C-MOS LSI which supports the RS-232C serial interface and parallel interface.
In the LSI is contained the RS-232C ART (Asynchronous Receiver Transmitter), its baud rate generator, and the Centronics transmitter/receiver interface.
When the ART receives data from the CPU, the data are converted into serial form and sent out on the TXD line. On the other hand, the serial data received on the RCD line are converted into parallel form before being handed to the CPU. The ART is able to inform the CPU at any time of the completion of sending the data received from the CPU or the reception of the data to be handed to the CPU.
The clock input of the IC is divided by a 4-bit programmable prescaler and becomes the internal clock (SYS-CLK), which is further divided by the baud rate generator composed of a 12-bit programmable divider, for the creation of any baud rate of 50 to 38,400 bauds.
The transmission/reception handshake pins are provided for the Centronics parallel interface. When the 8-bit data are received from the CPU in the transmit mode, a strobe of the programmed pulse width is automatically issued. In the receive mode, when data are received with a strobe singal from the external source, a busy singal is returned to automatically inform the CPU.

Pin No.	Symbol	In/Out	Active level	Function
1	(NC)	–	–	Not used.
2	$\overline{RD}$	In	Low	A low on this line causes the CPU to read data or status information from the TC8576F.
3	$\overline{WR}$	In	Low	A low on this line causes the TC8576F to receive data or control words sent from the CPU via the data bus.
4	$\overline{CS}$	In	Low	A low on this line causes theTC8576F to be activated. When $\overline{CS}$ is at a high level, both $\overline{RD}$ and $\overline{WR}$ are disabled.
5, 6	A1, A0	In	–	In combining this signal with $\overline{RD}$ or $\overline{WR}$, the CPU selects the contents of the data transfer with the TC8576F.
7	GND	Power supply	–	Power supply.
8	INT	Out	High	Logical OR of four internal signals (RXRDY, TXRDY, PRRDY, and PTRDY) which is used to cause an interrupt to the CPU.
9~16	D7~D0	In/Out	–	Data bus.
17	VCC	Power supply	–	Power supply.
18	GND	Power supply	–	Power supply.

A1	A0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Function
0	0	0	1	0	RXD → data bus, serial
0	0	1	0	0	Data bus → TXD, serial
0	1	0	1	0	PIN → data bus, parallel
0	1	1	0	0	Data bus → PVOUT, parallel
1	0	0	1	0	Serial status → data bus
1	0	1	0	0	Data bus → parameter register
1	1	0	1	0	Parallel status → data bus
1	1	1	0	0	Data bus → command + parameter address
*	*	*	*	1	Data bus, high impedance
*	*	1	1	0	Data bus, high impedance

* don't care

Pin No.	Symbol	In/Out	Active level	Function
19~26	$\overline{DATA1}$~$\overline{DATA8}$	In/Out	–	Bidirectional parallel data bus fixed to the output mode. The input mode is established with a high CDS state and the output mode is established with a low CDS state. The contents of data are in the reverse phase.
27	DSTB	In/Out	–	Parallel mode data strobe signal. Data strobe is sent when CDS is "1." Data strobe is received when CDS is "0."
28	ACK	In/Out	–	Parallel mode acknowledge signal. ACK is sent when CDS is "1." ACK is received when CDS is "0."
29	FAULT	In/Out	–	Parallel mode fault signal. When CDS is "1," the contents of the bit "0" of the command byte are sent out. When CDS is "0," the contents of this signal line can be known by the status bit "0." Mainly used for detection of a fault in the device.
30	$\overline{BUSY}$	In/Out	–	Parallel mode busy signal. When CDS is "1," a busy signal is sent. When CDS is "0," a busy signal is received.
31	PRIME	In/Out	–	Parallel mode input PRIME signal. When CDS is "1," it serves as a single bit input port. When CDS is "0," it serves as a single bit output line, but it still would be possible to choose a high level, low level, or one-shot pulse signal.
32	$\overline{SLCT}$	In/Out	–	Parallel mode select signal. When CDS is "1," the contents of the bit "1" of the command byte are sent out. When CDS is "0," the contents of this signal line can be known by the status bit "1." Mainly used for a device select.
33	$\overline{RTS}$	Out		Serial mode request to send signal. A general purpose 1-bit output port in the reverse phase. By programming the bit "5" of the command byte, it is set to "0." The signal is normally used by the modem control as a request to send.
34	$\overline{DSR}$	In		Serial mode data set ready signal. A general purpose 1-bit input port in the reverse phase. It is possible to know the state of the signal by interrogating the status information (bit 7) of the serial interface. This signal is normally used for tests by the modem for such as a data set ready. With the PC-1600, this signal is connected with the RXD line.
35	$\overline{CTS}$	In		Serial mode clear to send signal. Connected to GND. If the TXEN bit of the command byte has been set to "1," a high on this line enables the SMI transmit (serial).
36	$\overline{DTR}$	Out	–	Serial mode data terminal ready signal. A general purpose 1-bit input port in the reverse phase. By programming the bit "5" of the command byte, it is set to "0." The signal is normally used for the modem control as a data terminal ready.
37	TXD	Out	-	Serial mode transmit data signal. Serial data output for the serial interface.
38	RXD	In	–	Serial mode receive data signal. Serial data input for the serial interface.
39	VCC	Power supply	–	Power supply.
40	CDS	In	–	Parallel mode direction selection line. Fixed to GND, it's an input signal to determine the direction of the parallel interface. When the signal is at a low level, the parallel interface is operated in the output mode. When the signal is at a high level, the parallel interface is operated in the input mode.
41	XCLK	In	–	Fixed to GND. This line is an input to the internal 4-bit programmable prescaler. Its output becomes the system clock (SYS-CLK) which is used for internal timing generation and baud rate generation. Normally, 400KHz~10MHz is used as the system clock frequency.

Pin No.	Symbol	In/Out	Active level	Function
42	$\overline{\text{RESET}}$	In	Low	IC reset pin. A low on this line disables all the functions of the IC.
43	P5V	In/Out	–	Parallel mode signal which is fixed to GND. When CDS is "1," it serves as a 1-bit output port. When CDS is "0," it serves as a supply voltage input of the external device.
44	$\overline{\text{PE}}$	In/Out	–	Parallel mode paper end signal. When CDS is "1," it serves as a 1-bit output port. When CDS is "0," it receives the paper end signal from the external device.

RS-232C Interface Signals

The RS-232C interface signals of PC-1600 conform to the EIA and JIS standards, however, the control is somewhat different to the general RS-232C interface.

The RS-232C interface signals of PC-1600 conform to the EIA and JIS standards, however, the control is somewhat different to the general RS-232C interface.

(1) Incoming signal

The incoming signals are shaped as shown below by the hybrid IC (BX7269W), so that they can be processed as a logic signal.

(2) Outgoing signal

The outgoing signals are converted to the VDD to VEE level by the hybrid IC.

7.7 POWER SUPPLY

7.7.1 Kinds of Supply Voltages

The table below shows the kinds of the supply voltages used in the PC-1600.

Power supply	Voltage range	Description
VGG	4.0 ~ 4.7V	• Logic driving power which is on while the system is not operating. Power is supplied to the chips that need protection. (1) RAM16KB Memory protection (2) LU57813P Real-time timer and wake-up timer protection (3) HD61102 Display data protection which is required to activate the display at power-on after auto power-off. (4) LR38041 To maintain the signal level of such as the memory select signal at a non-active level.
VCC	4.0 ~ 4.7V	• Logic driving power which is shut off when the system is turned off. Power is supplied to the chips that do not need protection when the system is off. (1) ROM 256Kbit (2) CPU SC7852, LH5803 (3) HD61203(S) LCD common driver chip (4) TC8576F UART LSI
VEE	Approx. −8.5V	• For creation of a low voltage to the LCD drive voltage and the RS-232C interface signals.
VDD	Approx 6.0V	• For creation of a high voltage to the RS-232C interface signals. This voltage, however, is supplied when PRIME is at a high level (RS-232C is chosen) and shut off when PRIME is a low level.

7.7.2 Kinds of Power Supplies

The voltages described above are supplied from the following power supplies:

(1) Battery in the main unit
(2) AC power adapter
(3) VBAT of the system bus

If more than one power supply is connected, the one that has the highest voltage level among the power supplies is used.

7.8 GATE ARRAY

The gate array is an LSI (LR38041) consisting of a number of gate circuits necessary for the connection between the LSIs in PC-1600. The table below shows the terminal signals of the gate array.

Gate array (LR38041) pin description

Pin No.	Symbol	In/Out	Active level	Function
1	SLCB	In		It is an input of the sub CPU-issued signal PI which indicates commencement of the system operation. PI (SLCB) goes high when the system is off and does the following. (1) Data buses, D2~D0, are fixed at a low level. (2) Except for A13A, all output levels are fixed to low or high.
2	Q3	In		Hardware weak battery detect signal. (1) When a weak battery condition is detected, it forces Q3 high; S1, S2, S3, K0, K1, and K2 outputs are set high; KH output is set low; and RD is set to high impedance (inactive). (2) Q3 is at a low level when a weak battery is not established.
3	Z12	In		When on, the input is high.
4	Z13	In	Low	The sub CPU is normally in the sandby mode to save power when a command is not received. But, it would not go into the power save mode if the BREAK/ON key is continuously depressed, as it goes out of the standby mode if KH is at a high level. To prevent this, the state of the BREAK/ON key must be interrogated with Z13 when required. During the power save mode, Z13 is set to high to keep the KH output at a low level.
5	ON	In	Low	BREAK/ON key input. The signal goes low when the BREAK/ON key is depressed; otherwise, it is in a high state.
6	KC1	In		Signal input from the analog input connector.
7	CL2	In		Sub CPU 1.229MHz clock input.
8	RD	Out	Low	Read signal created by five signals (ME0, ME1, OD, CK0S, and BR0) which are externally wired OR with RD of the Z-80. When the Z-80 is in operation, RD is at a high impedance.
9~16	R20~R33	In		Return data from the sub CPU. The data becomes the Z-80 data when the Z-80 reads 33H of I/O.
17	PRIM	In		The PC-1600 has two serial input/output interface: the RS-232C interface and the SIO interface. But, either one must be assigned as only one hardware is for the serial input/output. (1) The SIO interface is selected with a low PRIM state. (2) The RS-232C interface is selected with a high PRIM state.
18	TXD	In	Low	Transmit data which is an output from the T8576F UART.
19	RXD	Out	Low	Receive data which is an input to the T8576F UART.
20	RDA	In	Low	Receive data which is an input from the RS-232C interface.
21	SDA	Out	High	Transmit data output is sent through the RS-232C interface connector. A low signal state is sent when PRIM is at a low level.
22	RDF	In	High	Receive data which is an input from the SIO interface.
23	SDF	Out	High	Transmit data which is an output to the SIO interface connector. A low signal state is sent when PRIM is at a high level.

	PRIME = "Low"	PRIME = "High"
Output SDA	LOW	$\overline{TXD}$
Output SDF	$\overline{TXD}$	LOW
Input RXD	RDF	RDA

Pin No.	Symbol	In/Out	Active level	Function
24	CKOS	In		The OD signal indicates that the CPU (LH5803) read timing is at a low level when not writing, So, it may possibly be at a low level when not reading, and it also may not match the Z-80's timing during data input/output of the SC7852 internal data. To prevent these problems, the signal goes low only when the LH-5803 is reading the memory or I/O is created with five signal (ME0, ME1, OD, CKOS, and BR05).
25	VCC	–		Power supply (input of VGG of the system).
26	GND	–		Power supply.
27	BRQ	In		See Pin No. 24.
28~30 31~35	D0~D2 D3~D7	In/Out Out		Z-80 data bus. When the sub CPU output P1 (SLCB), which is sent out when the system is off, is at a high level, D2~D0 become low, thus fixing the level of the input sinal during system-off as D2~D0 are inputs also.
36	C/$\overline{D}$	In	High	The C/$\overline{D}$ line of the SC7852 goes high when the Z-80 wirites 3DH of the I/O.
37	$\overline{IORP}$	In	Low	The signal used to send R33~R20 on the Z-80 data bus. It goes low when the Z-80 reads 33H of the I/O.
38	$\overline{CL}$	In	Low	System reset input. When this signal is at a low level, it forces A16A to high, A15A to low, and A14A to high.
39	A13	In		CPU address A13.
40	A14	In		Input of the CPU address A14 (insignificant).
41	DSR	Out		Not used.
42	CLK1	Out		When the system is on, CL2 is issued on this line and becomes the basic clock for the UART. A low is on this line when the system is off.
43	KH	Out	High	Opposite polarity of the BREAK/ON key input is sent to the sub CPU.
44	A13A	Out		Opposite polarity of the A13 input is sent.
45~47	A14A~A16A	Out		C/D latched D2~D0 output. Also, A16A is used for separating the $\overline{CS24}$ selected 16KB memory space 4000H~7FFFH (bank 3) into two banks.
48~50 51~53	LHS1~LHS3 KA0~K2	In In	Low Low	Memory select and I/O select signals from the SC7852 are sent to slots of S1: and S2: via the gate array. The reason why is that it has to be set at a high level at system-off as the SC7852 power supply is shut off when the system is off. (1) When the system is on but not in a weak battery condition the status of each signal appears on S1, S2, S3, KD, K1, and K2. (2) All are high when the system is off or a weak battery is detected.
54~56 59~61	S1~S3 K0~K2	Out Out	Low Low	LHS1~KA2 outputs. All are high outputs when the system is off.
62 63 64	ME1 ME0 OD	In In In		See pin No.24.

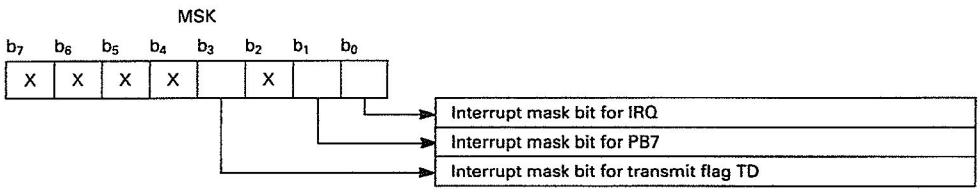
7.9 CONTROL OF I/O PORT CONTROLLER

The I/O port controller has seven read/write registers, each allocated to an I/O address of Z-80. Reading from or writing to a register is accomplished by performing a read or write operation to the appropriate I/O address.
The table below shows the registers and their I/O addresses.

Register	I/O address
OPC register	18H
MSK	1A
IF	1B
DDA	IC
DDB	1D
OPA	1E
OPB	1F

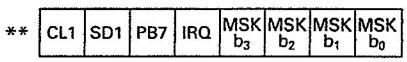
The I/O controller can be controlled by performing a read/write operation to these registers. The details of each register are described below.

(1) MSK register (Z-80 I/O address: 1AH)



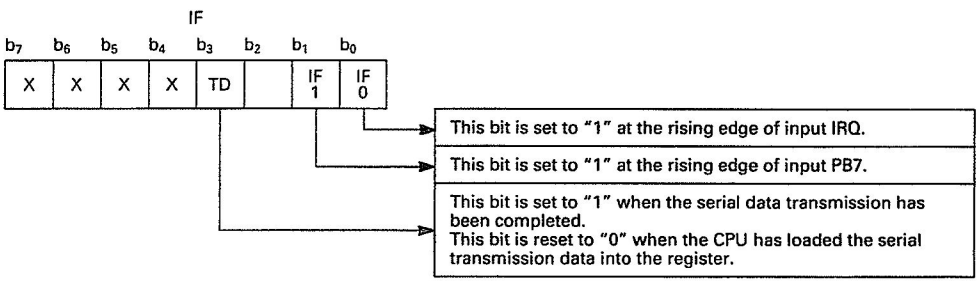
When the bit is "1", the interrupt is enabled.

Note: When the contents of MSK register are read, the upper 4 bits contain the contents of CL1, SD1, PB7 and IRQ.



MSK read contents

(2) IF register (Z-80 I/O address: 1BH)



TD is read-only.

(3) DDA register (Z-80 I/O address: 1CH)

DDA

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

This register determines the direction (input or output mode) of the PA port.
Bit i of DDA register

When bit i is "0"	PAi is in the input mode.
When bit i is "1"	PAi is in the output mode and the content of OPAi is output.

(4) DDB register (Z-80 I/O address: 1DH)

DDA

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
X			X	X	X	X	X

This register determines the direction (input or output mode) of the PB port.
Bit i of DDB register

When bit i is "0"	PBi is in the input mode.
When bit i is "1"	PBi is in the output mode and the content of OPBi is output.

(5) OPA register (Z-80 I/O address: 1EH)

OPA

7	6	5	4	3	2	1	0

The OPA register is a buffer register used when transferring data with the PA port. In the output mode, when you set DDAi to "1" (for the output mode) then send data to 1EH port, the data on the bus line are loaded to OPAi and output to PAi. In the input mode, when you set DDAi to "0" (for the input mode), the data transmission from OPAi is disabled, the content of PAi is loaded to OPAi and the data are sent out to the bus line.

(6) OPB register (Z-80 I/O address: 1FH)

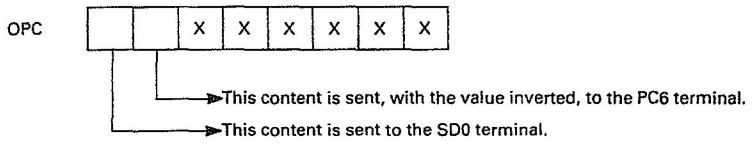
OPB

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

The OPB register is a buffer register used when transferring data with the PB port. In the output mode, when you set DDBi to "1" (for the output mode) then send data to 1FH port, the data on the bus line are loaded to OPBi and output to PBi. In the input mode, when you set DDBi to "0" (for the input mode), the data transmission from OPBi is disabled, the content of PBi is loaded to OPBi and the data are sent out to the bus line.

(7) OPC register (Z-80 I/O address: 18H)

The OPC register ia a buffer register used when sending data to the PC port. Also, the data on the data bus can be latched into the OPC register by the falling edge of the clock signal applied at the ϕos terminal of SC-7852.



CHAPTER 8

HARDWARE OF PERIPHERAL DEVICES

8.1 CE-1600P

8.1.1 Specifications

Model name:
CE-1600P
Type:
Printer/cassette interface
Print method:
X-Y plotting
Print capacity:
160 printing positions/line (with minimum size print characters)
Printing colors:
Four colors of black, blue, green, and red
Printing character size:
Nine sizes (0.8 mm x 1.2 mm ~ 7.2 mm x 10.8 mm).
Printing directions:
Four directions.
Minimum print pen moving distance:
0.2 mm
Printing speed:
5 characters/second, average (printing the size 2 characters in black with all kinds of ASCII characters (96)).
The printing speed is subject to variation depending on the print contents and program.
Print form:
210 mm wide roll paper whose roll size is up to 40 mm (EA-4AR1).
216 mm wide roll paper (EA-1LR1).
Cut sheet (A4 or letter size)
Power supply:
From the internal rechargeable batteries which can be recharged through the AC adaptor (EA-160).
Power consumption:
6VDC , 5.7W
Maximum printable lines per charge:
About 250 lines after 8 hours of recharge (continuous printing 40 digits of "5" of the print size 2 in black on a single line under the operating temperature of 20°C).
Operating temperature:
5°C ~ 40°C
Physical dimensions:
320 mm (W) x 221.5 mm (D) x 46 mm (H)

Weight:
About 1.6 kg including the pocket.
Accessories:
EA-160 AC adaptor, hard case, roll paper (1 pc), pen (2 pcs each of black, blue, green, and red), tape recorder interfacing cable (1 pc), paper holder (1 set), shaft (1 pc), instruction book

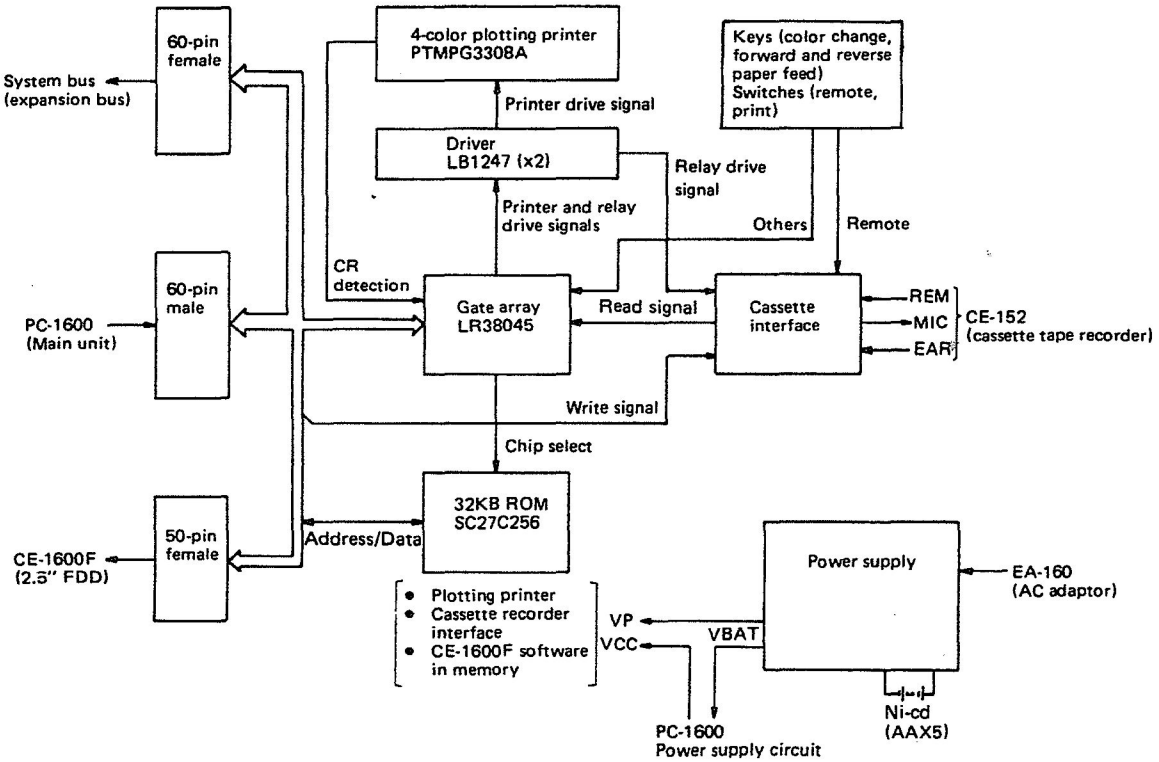
= About output error =
On account of a mechanical accuracy, a slight error may appear on the output. The error is larger in the direction Y (vertical) than in the direction X (horizontal). It is preferable to have accurate output to avoid repeated operation in the direction Y (paper feeding direction) when programming.

Options
The following options are available for the CE-1600P.

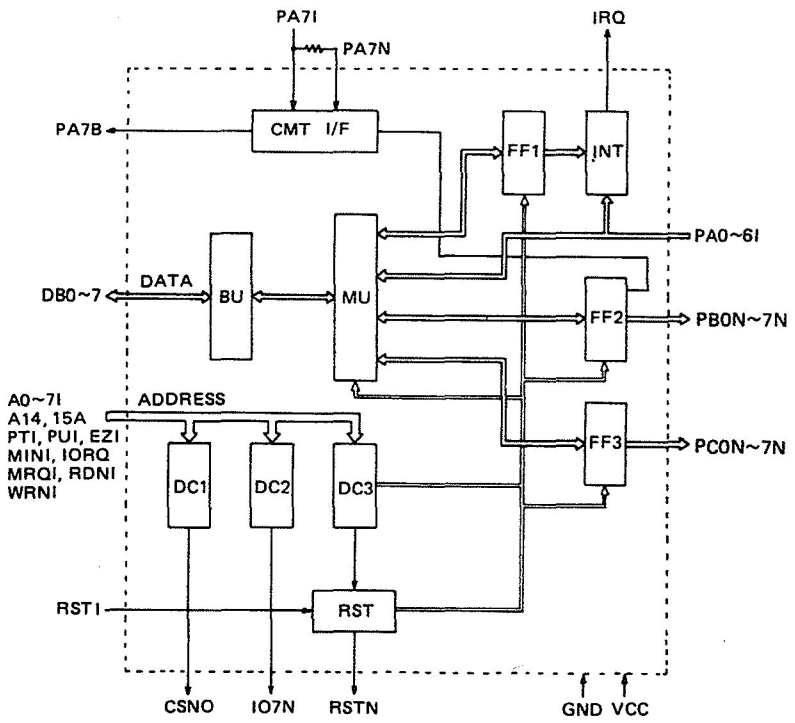
	Item	Product name	Note
1	Roll paper	EA-4AR1	210mm wide, 14m long, 40mm roll
2	Roll paper	EA-1LR1 (only in U.S.A. and Canada)	216mm wide, 40mm roll
3	Print pen	EA-850B	Contents of 4 pens of black.
4	Print pen	EA-850C	Contents of one each pen of black, blue, green, and red.
5	Floppy disk drive	CE-1600F	2.5" floppy disk drive unit
6	Cassette tape recorder	CE-152	

8.1.2 Block diagram

Since the printer, cassette, and floppy disk drive are all controlled by the PC-1600, the CE-1600P and PC-1600F can not operate by itself.
Battery, however, can be recharged without intervention of the PC-1600.
A 32KB ROM within the CE-1600P contains the program to operate the printer, cassette, and floppy disk drive.



(Fig.1) CE-1600P block diagram



Bu: 8 bits I/O Buffer
Mu: Multiplexer
FF1~3: 8 bits Latch
DC1~3: Decorder
INT: Interrupt circuit
RST: Reset circuit
CMT I/F: Serial input
PA0~7I: 8 bits input
PA0N~7N: 8 bits output port
PC0N~7N: 8 bits output port

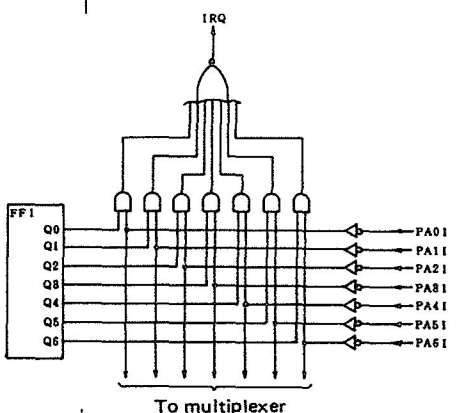
(Fig.2) LR33045 gate array block diagram

8.1.3 Description of each block

(1) LR38045 gate array

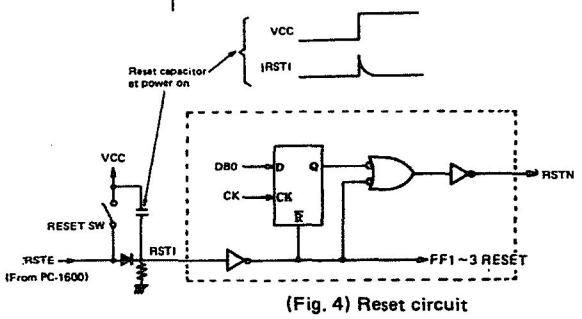
Table below shows the functions and port address of the gate array block.

Block	Function
Bu (8-bit I/O buffer)	A bidirectional 8-bit input/output buffer.
Mu (multiplexer)	Used to select FF1, FF2, FF3, or PA port when data are read from the gate array.
FF1~FF3 (8-bit latch)	FF1: The interrupt circuit is controlled with an FF1 output. For instance, when a certain bit is set to "1", the input signal to the PA port (PA0~61) which corresponds to the bit is sent on the IRQ line as an interrupt signal. FF2: PB port (PB0~7N) latch FF3: PC port (PC0~7N) latch
DC1~3 (decoder)	DC1: For generation of 32KB ROM chip select signal. (CSNO) DC2: For generation of 2.5" FDD select signal. (IO7N) DC3: For selection of FF1~FF3 and FFD reset latch at the time of data write. Or selection of FF1~FF3 or PA port at the time of data read.
INT (interrupt circuit)	Inputs to the PA port (PA0~61) are ORed and sent on the IRQ line as an interrupt signal. As PA0~61 correspond to Q0~Q6 of FF1, the interrupt is enabled when FF1 is set with "1". (Fig. 3 shows the quivalent circuit of the interrupt circuit.)
RST (reset circuit)	FF1~3 are reset by this circuit, when a reset signal is received on RSTI. At the same time, the 2/5" FDD reset signal (RSTN) is issued which will be kept active until cleared by software.

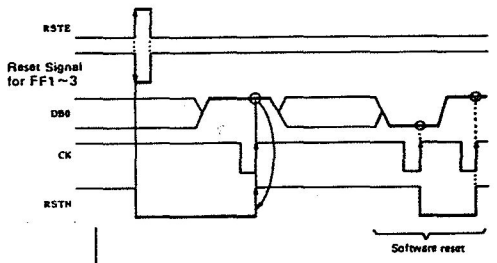


(Fig. 3) Interrupt circuit

Block	Function
	It is possible without an input on RSTI to output RSTN by means of software. (Fig. 4 shows the equivalent circuit of the reset circuit and Fig. 5 shows its timings.)

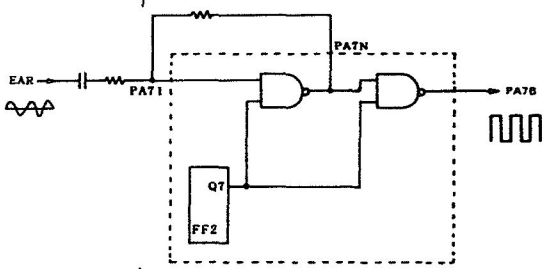


(Fig. 4) Reset circuit



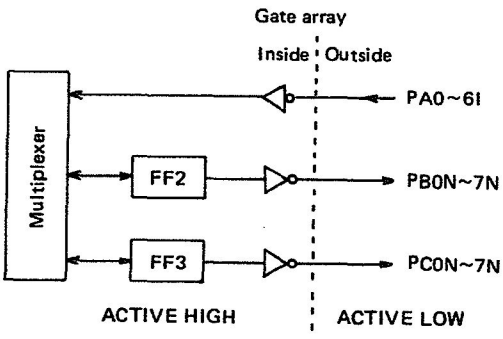
(Fig. 5) Reset circuit timings

CMT I/F (cassette interface circuit)	The cassette signal received from the EAR jack is amplified and waveform shaped, to be sent on PA7B. (See Fig.6 for its equivalent circuit.)



(Fig. 6) Cassette interface circuit equivalent circuit

NOTE: Ports, PA, PB, and PC, are all active high within the gate array, but they are converted to active low signals outside of the gate array.



For instance, if "1" is set to Q0 of FF2, the PB0N output becomes low.

TABLE-3

Port address

Table-1

IORQ	Address								WR	RD	Operation	Data							
M1	A7	A6	A5	A4	A3	A2	A1	A0				D7	D6	D5	D4	D3	D2	D1	D0
1	1	0	0	0	0	0	0	0	0	1	Write data to FF1	0	*	Printer CR INT Enable	Printer SW INT Enable	FD INT Enable	Reverse PF key INT Enable	PF key INT Enable	CC key INT Enable
									1	0	*Read data from FF1*	†	†	†	†	†	†	†	†
					0	0	0	1	0	1	Reset FD (reset with "0")								FD Reset
									1	0	Read $\overline{PA0 \sim 7}$	CMT input	(0)	Printer CR	Print SW	FD INT	Reverse PF key	PC key	CC key
					0	0	1	0	0	1	Write data to FFD ($\overline{PB0 \sim 7}$)	CMT in Enable	*	RMT OFF	RMT ON	Motor ZD	Motor ZB	Motor ZC	Motor ZA
									1	0	Read data from FFE ($\overline{PB0 \sim 7}$)	†	†	†	†	†	†	†	†
					0	0	1	1	0	1	Write data to FFE ($\overline{PC0 \sim 7}$)	Motor YD	Motor YB	Motor YC	Motor YA	Motor XD	Motor XB	Motor XC	Motor XA
									1	0	Read data from FF3 ($\overline{PC0 \sim 7}$)	†	†	†	†	†	†	†	†

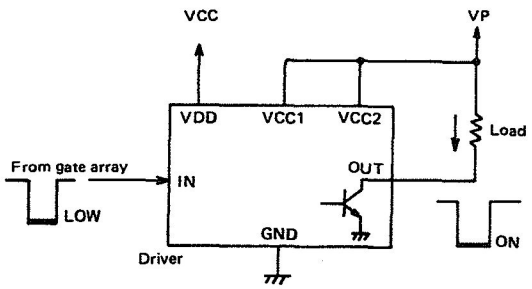
NOTE: Above are all high active as seen from the CPU side, except that FD reset is low active.

• Gate array (LR38045) pin description

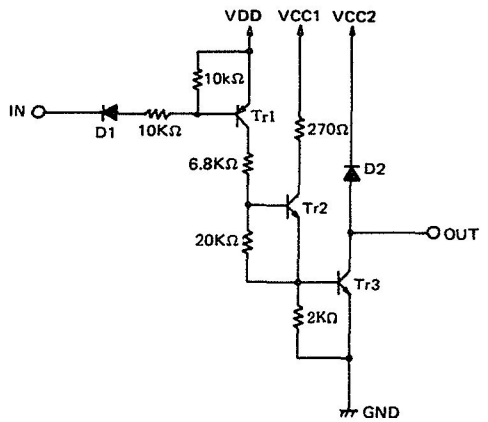
Pin No.	Symbol	I/O	Active level	Level at reset	Description
1 ~ 8	PC7N ~ PC0N	Out	Low	High	8-bit output port (port address: 83H). D0 ~ D7 correspond to PC0 ~ 7N via FF3'
9 ~ 16	PB7N ~ PB0N	Out	Low	High	8-bit output port (port address: 82H). D0 ~ D7 correspond to PB0 ~ 7N via FF2.
17	(NC)				
18 19 20	PUI PTI EZI	In In In	(Low) (High) (High)		PU signal input. PT signal input. $\overline{ELH}$ signal input. [Used for creation of a 32KB ROM signal (CSNO).]
21 22	MINI IORQ	In In	(High) High		$\overline{M1}$ signal input. IORQ signal input. [Used for creation of the IO7N and gate array internal enable signal.]
23	MRQI	In	High		MREQ signal input (used for generation of CSNO).
24	RSTI	In	High		Reset signal input. When the reset signal is received on this line, it issues the internal flipflop reset signal and RSTN (2.5" FDD reset signal).
25 26	VCC GND				Power supply.
27	IRQ	Out	Low	High impedance	Interrupt signal output. The output is N-channel open drain type and is pulled up to VCC on the PC-1600 side.
28 29	RDNI WRNI	In In	Low Low		$\overline{RD}$ signal input. $\overline{WR}$ signal input.
30 ~ 39	A0I ~ A7I A14I, A15I	In			Address input.
40	(NC)				
41	RSTN	Out	Low	Low	2.5" FDD reset signal output. The active state of the signal is unconditionally issued with a reset signal and it must be cleared by means of software. It is also possible to create the signal by software. (Address: 81H, D0, $\overline{WR}$)

Pin No.	Symbol	I/O	Active level	Level at reset	Description
42	IO7N	Out	Low	(High)	2.5" FDD select signal. (Out through the address 70H — 7FH)
43	CSNO	Out	Low	(High)	32KB ROM select signal. [$\overline{ELH}$, MREQ, PT . . . . . High PU . . . . . Low Address . . . . . 4000H ~ 7FFFH (PV ... Low; printer, High; FDD, CMT)]
44 ~ 51	DB0 ~ DB7	In/Out			(8-bit) data input/output.
52 ~ 57 60	PA0I ~ PA5I PA6I	In	Low		Input port (port address: 81H). (PA0I ~ 6I correspond to D0 ~ 6. (Interrupt controlled by FF1 (address: 80H) outputs Q0 ~ 6.)
58	GND				Power supply.
59	(NC)				
61	PA7B	Out	(High)	High	CMT I/F circuit output. (The cassette signal that has been amplified and waveform shaped is sent from this line.) (See Fig. 6.)
62 64	PA6N PA7I	(Out) In	(Low)	High	Comprise an amplifier when a feed back resistor is connected across PA7N and PA7I. (See Fig. 6.) (Input signal is given from PA7I.)
63	(NC)				

(2) Printer drive IC (LB-1247)

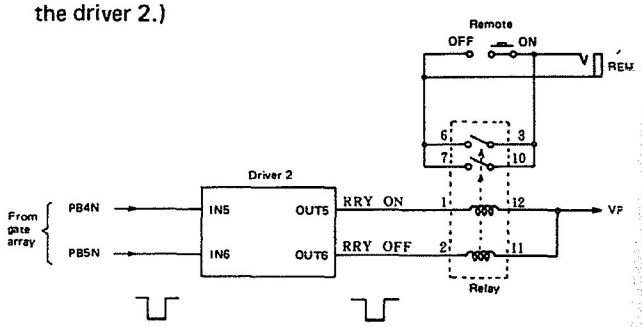


As shown above, the output transistor within the driver is turned active in the period that the driver input signal (signal from the gate array) is low level, so that current flows across the load connected to the output terminal of the driver. Fig.7 shows the equivalent circuit of the driver.



(Fig.7) Driver (LB1247) equivalent circuit

Eight circuits shown in Fig.7 are contained in a single driver circuit. The driver 1 is sued for driving of printer X and Z motors and the driver 2 is used for driving of printer Y motor and and remote relay. (Two circuits are not used for the driver 2.)



(Fig.8) Remote circuit

To increase the torque of the printer Y motor, a 5V zener diode (HZ5C1) is inserted across two VCC2 (which contains reverse surge absorb diode) of the driver 2.

(3) Printer (PTMPG3308A)

The PTMPG3308A ball point pen type, 4-color, plotting printer consists of three stepping motors which are used to control the direction X (horizontal pen movement), the direction Y (vertical pen movement), and the direction Z (pen up/down and color change). Each motor is driven by coils of A, B, C, and D. The CR detect switch is attached to the left side of the printer for detection of a CR via the pin PA5I of the gate array. The X and y motors are 1-2 phase excited and the Z motor 2-2 phase excited.

Printer, for detail of PTMPG3308A printer specifications, characteristics, drive method, etc.

8.1.4 CMT interface

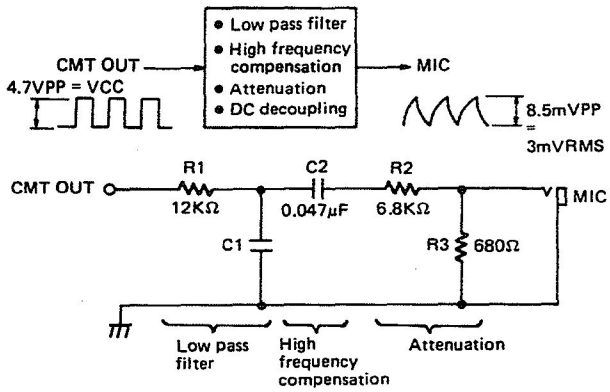
The CMT interface consists of the following circuits:

- Write circuit
- Read circuit
- Remote circuit

(1) Write circuit

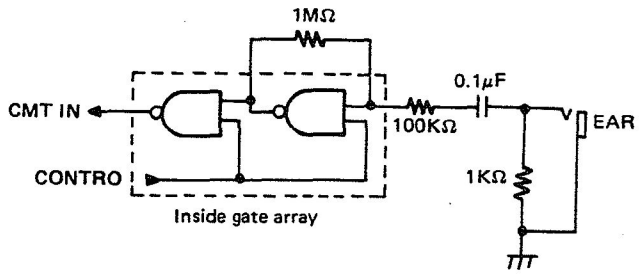
As shown below, the logic level signals are converted into signals of micro level.

- High frequency component of signal is eliminated. → Low pass filter
- As a 3KHz component drops 6dB than a 1.5KHz component because of the low pass filter, compensation is therefore done. → High frequency compensation
- The output level is set to the micro level. → Attenuation
- DC component is cut. → DC decoupling



C1, C2: DC decuppling
Output level: 3mV rms
Output impeadance: APROX 600Ω

(2) Read circuit



The read signal amplifier circuit consists of the same type as that of the CE-150. The circuitry is contained inside the gate array in the case of the CE-1600P.

4-3. Remote circuit

For the relay (AG8229 or G5AK-287P) is a two-coil latching type, A ON (or OFF) pulse must be given to the activate (or deactive) the relay through the driver of the gate array, in order to turn the relay active. (See Fig.8.)
The width of pulse must be more than 5 milliseconds than that mentioned in the relay specification. With the CE-1600P, it is set to about 10 milliseconds.

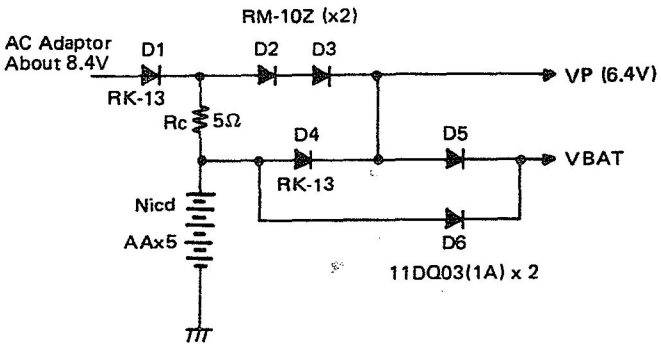
The following signal formats are used for the cassette interfacing signals.

Write . . . . . PWM method (1600 method)
Read . . . . . PWM method (1600 method) and 1500 method

8.1.5 Power supply circuit

(1) Power supply

VP, VBAT, and battery recharge circuits



D1: For prevention of reverse current to the rechargeable battery to the adaptor.
To achieve efficient recharging of the battery, a Schottky barrier type diode RK-13 (1.7A) is used.

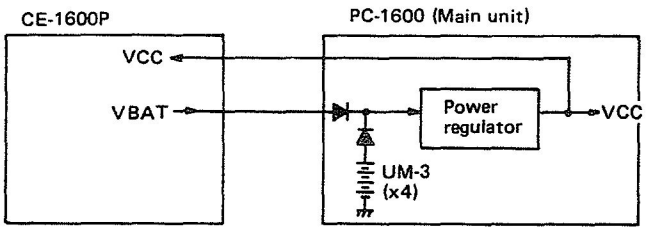
D2, D3: These diodes are used to drop the voltage from the printer to less than printer driving voltage (7.15V max.).

D4: For prevention of reverse current from VP to the rechargeable battery, when the adaptor is being used.

D5: For prevention of reverse current from VBAT (PC-1600) to VP (printer).
The diode is a Schottky battery type for avoiding battery exhaustion when the adaptor is used.

D6: To avoid exhaustion of the battery in the main unit when the rechargeable battery is used, D6 is used to bypass D4 and D5.
To meet the printer drive voltage (5.0V, min.), the rechargeable battery low voltage is set to 5.65V limit (1.13V per battery).

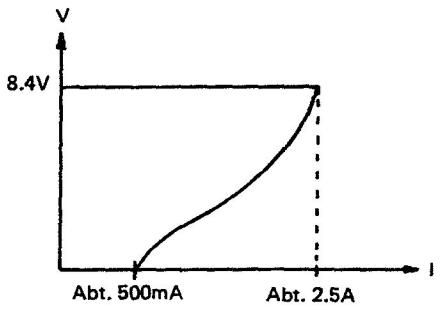
After the main unit battery is ORed with the VBAT supply from the CE-1600P, VCC is regulated to 4.7V before supplied to the CE-1600P. (See the figure below.)
When the main unit power is turned off, VCC is not supplied.



(2) AC adaptor (EA-160)

The following is a brief specification.

- Primary side input rating
  100VAC, 50/60Hz, 20VA (Japan use)
- Secondary side output
  Rated voltage: 8.4VDC
  Rated current: 1A
  Peak current: 2A
  Overcurrent protection: About 2.5A
  (Output short protection)



  Regulator type: Chopper
- Size of case and weight
  67.2 mm (W) x 115.2 mm (D) x 53.5 mm (H) excluding the stand of 1 mm high.
  695 g

8.2 CE-1600F/CE-1650F

8.2.1 Specifications

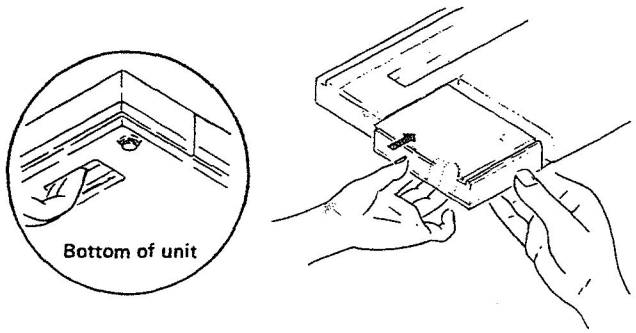
Model name: CE-1600F
Product name: Floppy disk drive
Drives: One drive (one side)/unit
Recording media: 2.5" two-sided floppy disk
Recording method: GCR (4/5)
Tracks: 16 tracks/side
Capacity: 64KB (one side)
(8 sectors/track)
Power supply: 6VDC: Supplied from the unit connected.
Power consumption: 2.5W
Operating temperature: 10°C ~ 35°C
(drive operating requirement)
Humidity: 20% ~ 80% (without moisture condensation)
Physical dimensions: 96mm(W) x 122mm(D) x 39mm(H)
Weight: 470 grams
Accessories: 2.5" two-sided floppy disk (x 1), instruction book (x 1)
Option: CE-1650F
(contents of 10 2.5" two-sided floppy disks)

NOTE: '2.5" (63.5 mm)' indicates the diameter of the floppy disk media.

8.2.2 Cautions in installing and removing the CE-1600F

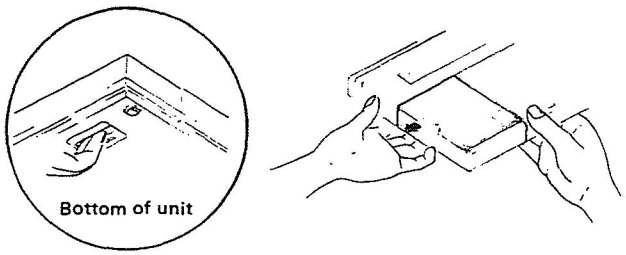
(1) Cautions in installing the CE-1600F

Power must be shut off to the CE-1600P before connecting the CE-1600F to the CE-1600P.
Pay special attention to hold the unit in a way as shown in the figure below with care not to touch the disk holder, in order to avoid a read/write failure because of center deviation.

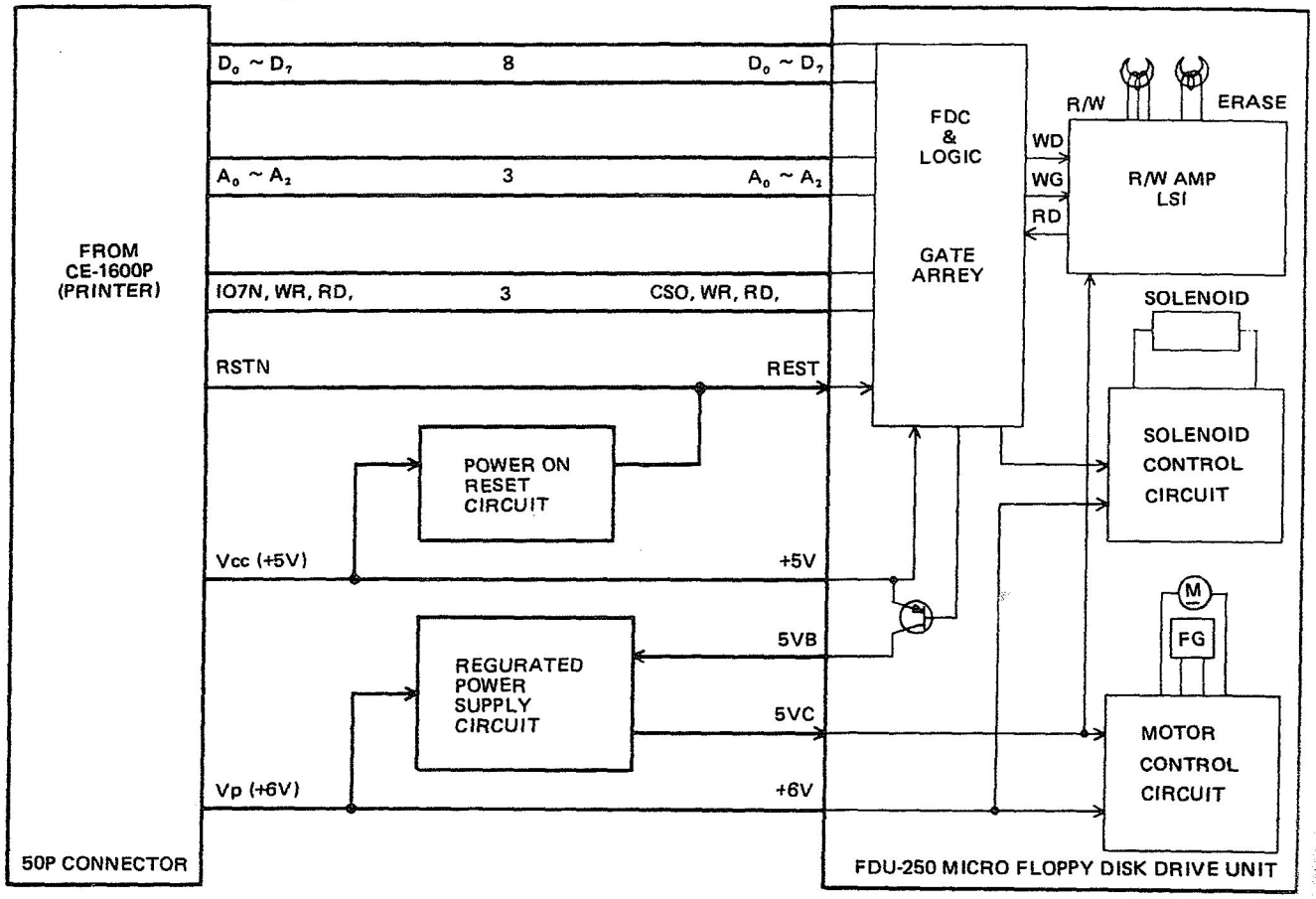


(2) Cautions in removing the CE-1600F

Before the removal of the CE-1600F, make sure that the power is off and remove it without adding force to the disk holder (see the figure below).



8.2.3 Block diagram



8.2.4 Circuit description

(1) Internal operation

Since the floppy disk controller is contained within the 2.5" floppy disk drive unit and directly interfaced with the bus line, data line and control signals are directly connected.

So, only the power-on-reset signal generation circuit and the amp's 5VC (+5V) supply regulator circuit are provided for circuit.

(2) Power-on-reset signal generation circuit

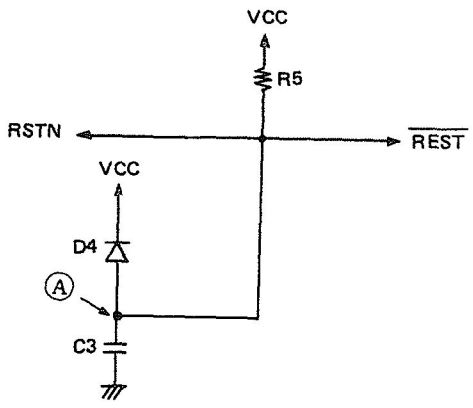


Fig. 1 Reset circuit

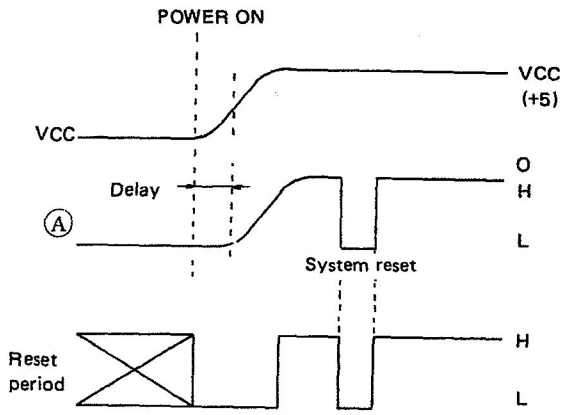


Fig. 2 Reset timings

Fig. 1 shows the reset circuit and Fig. 2 shows its timings. R5 is a charge current regulating resistor C3 which is used for pullup and delay. D4 is a diode which is used to bypass the charge in C3 to VCC line when VCC is off.

The reason why the reset signal is required at power on is to hold it in the standby mode so as to avoid malfunction in the floppy disk controller inside the floppy disk unit.

(3) Regulated power supply circuit

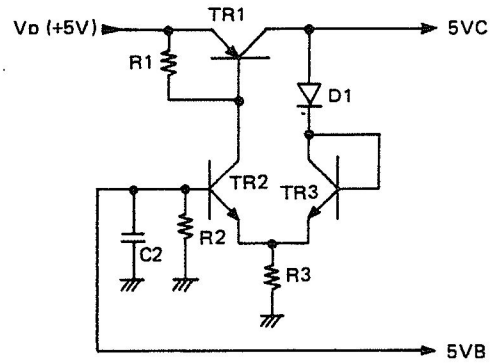


Fig. 3 Regulated power supply circuit

Fig.3 shows the regulated voltage supply circuit. In this circuit, floppy disk unit's 5VC (5V of amp) is supplied from VP (battery voltage), because 5VC can not be supplied from VCC on account of current restriction.
For the voltage of 5VC is used with a voltage difference of 0.5V minimum against VCC, the power is produced in reference to 5VB through the differentiation circuit composed of TR2 and TR3, not merely the regulator circuit. 5VB is a transistor output which is employed to turn on/off VCC with the MOTOR ON signal, and it has less voltage drop caused in the transistor, as compared with VCC. So, D1 is inserted to the output voltage feedback transistor TR3 to correct 5VC to be 0.2 to 0.3 volts higher then 5VB in appearance. (A schottky barrier diode is used for D1.)

8.2.5 Brief description of floppy disk drive

The floppy disk controller is implemented within the 2.5" floppy disk drive, and the floppy disk driving and head seeking are done by one motor. The floppy disk is driven by the belt and the head is seeked using the solenoid and cam.
The floppy disk controller and its peripheral logic are contained in a single chip gate array (2700 gates) and the read/write amplifier is also in a single chip LSI, which are directly bus connected to permit a low voltage driving.
Floppy disk format and write method are unique to the floppy disk. Though the floppy disk drive is for one-sided operation, both sides of the media can be used.

Specification of FDU 250

1) Memory capacity:
   64KB (512 Bytes/sector, 8 sectors/track)
2) Recording method:
   GCR (4/5)
3) Transfer speed:
   250K bits (25K Bytes/sec)
4) Track density:
   48 TPI
5) Total tracks:
   16
6) Revolutions:
   270 rpm
7) Access time:
   One step 80 milliseconds from track 00 to track 15.
   170 milliseconds to restore from track 15 to track 00.
   Settling time:
   50 milliseconds
8) Motor startup time:
   0.5 second

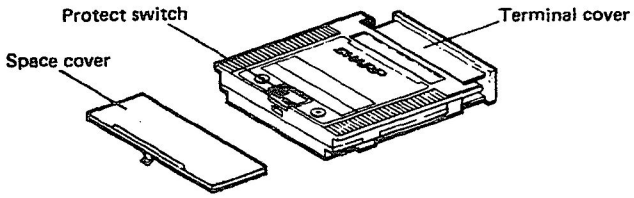
NOTE: GCR is an abbreviation of of Group Coded Recording. A single byte, 8 bits, data are divided into two 4-bit data which is also converted onto a 5-bit data. Thus, a single byte (8 bits) is recorded on the media as a 10-bit data.

8.3 CE-1600M

8.3.1 Specifications

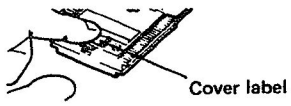
Product name: Program module
Model name: CE-1600M
Type: Module (RAM)
Capacity: 32KB
Backup battery: 3V(DC) lithium battery (CR2032 x 1)
Battery life: About 5 years in the pocket computer, or, about 24 month when removed from the pocket computer under temperature of 20°C. (Subject to variation depending on the usage and environment.)
Operating temperature: 0 to 40°C
Physical dimensions: 40.9mm (W) x 42.8 mm (D) x 8.5 mm (H)
Weight: 15 grams, including the battery cell
Accessories: Case, cover label (x 3), space cover, lithium battery (in the main unit), instruction book.

8.3.2 Parts identification



Protect switch
When the switch is set to the side marked with "●", memory write is prohibited so that it disables to write, erase, and revise the memory contents.
When the switch is at the side not marked, the write protect is cleared.

* When it has been write protected, cover the switch with the cover label to avoid incidental manipulation of the switch.



8.3.3 Use

This RAM module may be used in the following way:
① For expansion of user's area.
② For program module separate from the computer's internal memory.
③ RAM file
The INIT statement of BASIC must be used to assign it to the above mode.

User area
The maximum size of the user memory run under the PC-1600 memory only is 11,834 bytes. (Fig.1) If the machine language area is reserved or a buffer is reserved using the command 'MAXFILES' or 'INIT"COMn:"', it will become less than 11,834 bytes.

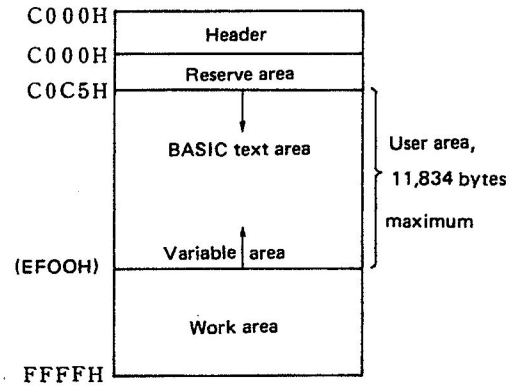


Fig. 1 Bank 0 user area map

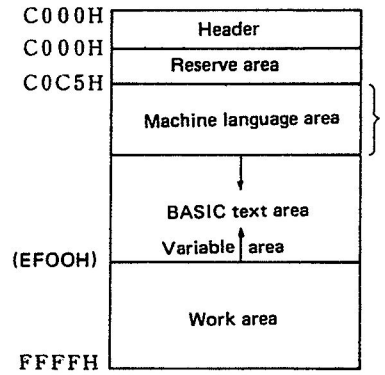


Fig. 2 Bank 0 user area map

Expansion of user area

When "M" is specified with the INIT statement after connecting the RAM module into the memory slot, the computer will acknowledge the RAM module as the user area.

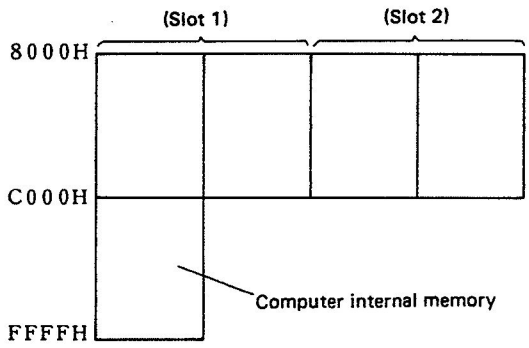
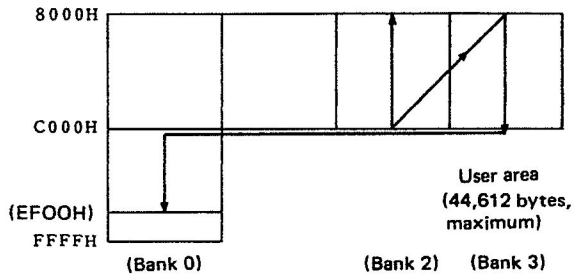
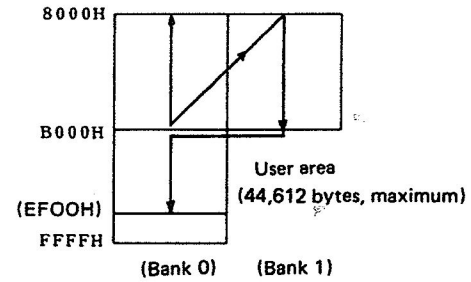


Fig. 3 Computer internal memory and memory slot memory map

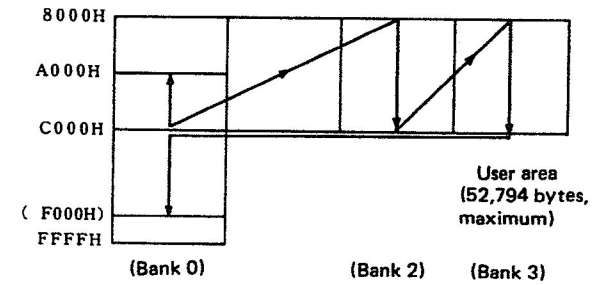
When the CE-1600M is connected to S2:



When the CE-1600M is connected to S1:



When the CE-159 is connected to S1: and the CE-1600M is connected to S2:

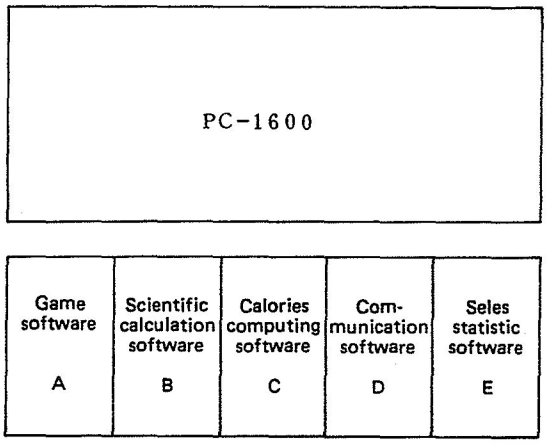


When the RAM module is connected to both slots of S1: and S2:, connection is made from the smallest memory module to larger module and to the main memory. If they have the same capacity, connection is made in order of S1:, S2: and main memory. A larger capacity memory must be the CE-161 or CE-1600M. Otherwise, the control assumes as if only the larger module is connected.

Program module

The program modules discussed here is the one that used as a software cartridge. The already compiled programs are stored in the module and connected with the computer for operation.

Assume now that there are five program modules as an example.



Program modules

According to the need, the desired program module is connected for an immediate program execution.

① Two program modules can be used at the same time.
② When used as the program module, no user area can be contained. But, if the module has been divided into a program module and a user area using the INIT statement, only the declared user area conforms to the user area of (1).
③ Creating the program module.
After declaring the program area with the INIT statement, the program is written or loaded to that area.
④ The memory protected CE-159, CE-161, or CE-1600M must be used for the program module.

RAM file module

With this usage, the completed program or data are saved into the memory module, to be loaded onto the user area when so required.

If used as a RAM module, the module is not included in the area.

① The module that can be used the RAM file module is the CE-161 and CE-1600M.
② The RAM file module can be accessed free from the main unit. While it is removed from the main unit, the contents are retained by the internal battery.

- What program and data are contained within the RAM file module can be known by means of the FILES statement or LFILES statement. It is possible to change the name or delete the program or data.

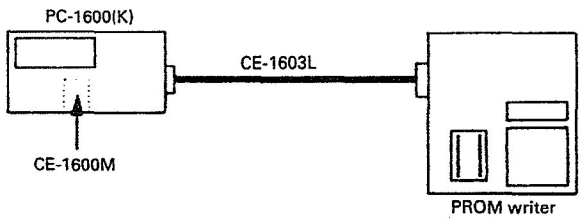
8.4 CE-1620M/CE-1601E/PROM PROGRAMMER

8.4.1 Specifications of CE-1620M

Product name: PROM module
Model name: CE-1620M
Type: Module (EPROM)
Capacity: 32 KB
Operating temperature: 0 to 40°C
External dimensions: 40.9 mm (W) × 42.8 mm (D) × 8.5 mm (H)
Weight: Approx. 12 g

8.4.2 Program Transfer to PROM Writer

Turn off the power of the PC-1600 and the PROM writer, then connect them through the CE-1603L cable. Use the BASIC program listed in the next section 8.4.3 to transfer your program data to the PROM writer.

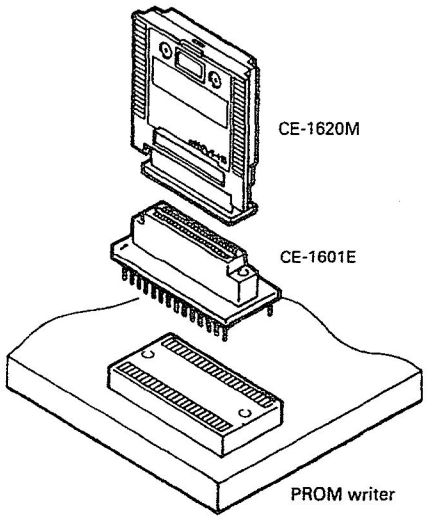


8.4.3 Transfer Program

```
 10:REM   *** INTEL HEX FORMAT UPLOAD PROGRAM ***
 20:DIM A$(0)*80
 30:SETCOM "COM1:",4800,7,E,2,X
 40:SETDEV "COM1:",PO
 50:OUTSTAT "COM1:",0
 60:REM  * MAIN *
 70:"R"GOSUB "I"
 80:B=E-A+1:IF B>15LET B=16:A$(0)=":10":GOTO 110
 90:IF B<0THEN "R"
100:A$(0)=":0"+HEX$ B
110:GOSUB 320:A$(0)=A$(0)+"00"
120:REM  * DATA SEND *
130:S=0
140:IF B<>16THEN 160
150:GOSUB 500:GOSUB 500:GOTO 210
160:FOR I=0TO B-1
170:D=PEEK# (K,A):S=S+D
180:IF D<16LET A$(0)=A$(0)+"0"
190:A$(0)=A$(0)+HEX$ D:A=A+1
200:NEXT I
210:S=S+B+AH+AL
220:SL=256-(S-INT (S/256)*256)
230:IF LEN HEX$ SL=1LET A$(0)=A$(0)+"0"
240:A$(0)=A$(0)+RIGHT$ (HEX$ SL,2)
250:REM  * SEND DATA *
260:LPRINT A$(0)
265:PRINT A$(0)
270:IF A-1<>ETHEN 80
280:REM  * END *
290:LPRINT ":00000001FF"
300:BEEP 2
310:END
320:REM  ** ADDR OUT **
330:ADDR$=HEX$ A
340:L=LEN ADDR$
350:IF L=4THEN 370
360:FOR I=0TO 3-L:A$(0)=A$(0)+"0":NEXT I
370:A$(0)=A$(0)+ADDR$
380:AH=INT (A/256):AL=A-AH*256
390:RETURN
400:"I"REM   * INPUT BANK,ADDR *
410:CLS :PRINT "       *** UPLOAD ***"
420:ON ERROR GOTO "H"
430:INPUT "    BANK  No.  #";K
440:INPUT "    START ADDR ";A
450:INPUT "    END   ADDR ";E
460:ON ERROR GOTO 0
470:RETURN
480:"H"REM  * INPUT ERROR *
490:RESUME "I"
500:REM
505:S$=""
510:FOR I=0TO 7
520:D=PEEK# (K,A):S=S+D
530:IF D<16LET S$=S$+"0"
540:S$=S$+HEX$ D:A=A+1:NEXT I
550:A$(0)=A$(0)+S$
560:RETURN
```

8.4.4 Writing and Erasing of CE-1620M

To write to CE-1620M, connect it through CE-1601E to the PROM writer as shown below. To erase the contents of CE-1620M by ultraviolet rays, remove the front cover of the CE-1620M.



8.4.5 Recommended EPROM Programmer

In principle, you can use any EPROM programmer if it supports FUJITSU 27C256. However, we recommend the following products of Data I/O Corp. This recommendation is based on permission from FUJITSU Ltd. and Data I/O Corp.

Recommended model: Model 29B or Model 201

The following address list may help you find your nearest Data I/O office.

DATA I/O Corporation
10525 Willows Road N.E.
P.O.Box 97046
Redmond, WA 98073-9746
(206) 881-6444
Telex Dom. 15-2167,
Int'l 4740166 dio ui
Fax (206) 882-1043

WESTERN REGION

Data I/O
3505 Cadillac Avenue
Suite L-1
Casta Mesa, CA 92626
(714) 662-1182 (Sales)
(714) 662-2498 (Service)

EASTERN REGION

Data I/O
Birch Pond Business Center
22 Cotton Road
Nashua, NH 03063
(603) 889-8511 (Sales)
(603) 889-8513 (Service)
Telex 943431

CENTRAL REGION

Data I/O
701 N. Glenville Drive
Suite 101
Richardson, TX 75081
(214) 235-0044
Telex 792474
TWX 910-997-1767

ALABAMA

Pen Tech Associates
Huntsville, AL
(205) 881-9298
Telex 62826048

ALASKA

Northwest T & M
Beaverton, OR
(503) 646-9966
Telex 910-467-8775

ARIZONA

Zeus Electronics, Inc.
Phoenix, AZ
(602) 263-6022
1-800-528-4512
Telex 910-951-1362

ARKANSAS

Testech, Inc.
Richardson, TX
(214) 644-5010

CALIFORNIA, NORTHERN

Data I/O
Santa Clara, CA
(408) 727-0641

CALIFORNIA, SOUTHERN

Data I/O
Costa Mesa, CA
(714) 662-1182

COLORADO

Zeus Electronics, Inc.
Denver, CO
(303) 321-4246
1-800-521-4512

CONNECTICUT

Data I/O
Nashua, NH
(603) 889-8511
Telex 943431

DELAWARE

Scl-Rep
Ballimore Office
(301) 321-1411

FLORIDA

Pen Tech Associates
Deerfield Beach, FL
(305) 421-4989

Casselberry, FL
(305) 678-6809
Telex 62821019

FLORIDA, NORTHERN

Pen Tech Associates
Huntsville, AL
(205) 881-9298
Telex 628260458

GEORGIA

Pen Tech Associates
Marletta, GA
(404) 424-1931
Telex 62826058

HAWAII

Northwest T & M
Beaverton, OR
(503) 646-9966
Telex 910-467-8775

IDAHO

Zeus Electronics, Inc.
Salt Lake City, UT
(801) 534-0500
(801) 534-0503

ILLINOIS, NOTHERN

Torkelson Associates
Deerfield, IL
(312) 945-8700
Telex 910-992-1438

ILLINOIS, SOUTHERN

Palatine Eng. & Sales
Hazelwood, MO
(314) 839-0800
Telex 910-762-0627

Blue Spring, MO
(816) 229-4007

INDIANA

Torkelson Associates
Indianapolis, IN
(317) 244-7867
Telex 810-341-3141

IOWA
Tarkelson Associates
Cedar Rapids, IL
(319) 373-0200

KANSAS
Palatine Eng. & Sales
Hazelwood, MO
(314) 839-0800
Telex 910-762-0627

Bule Spring, MO
(816) 229-4007

KENTUCKY, EASTERN
Electro Sales Associates
Allison Park, PA
(412) 487-3801

LOUISIANA
Testech, Inc.
Houston, TX
(713) 956-0837

MAINE
Data I/O
Nashua, NH
(603) 889-8511
Telex 943431

MARYLAND
Scl-Rep
Baltimore Office
(301) 321-1411
(301) 666-5223

MASSACHUSETTS
Data I/O
Nashua, NH
(603) 889-8511
Telex 943431

MICHIGAN
Electro Sales Associates
Livonia, MI
(313) 474-7320
Telex 510-1006711

Portage, MI
(616) 323-2416

MINNESOTA
Torkelson Associates
Minneapolls, MN
(612) 835-2414
Telex 910-576-2740

MISSISSIPPI
Pen Tech Associates
Huntsville, AL
(205) 881-9298
Telex 62826048

MISSOURI
Palatine Eng. & Sales
Hazelwood, MO
(314) 839-0800
Telex 910-762-0627

Blue Spring, MO
(816) 229-4007

MONTANA
Zeus Electronics, Inc.
Salt Lake City, UT
(801) 534-0500
(801) 534-0503

NEBRASKA
Palatine Eng. & Sales
Hazelwood, MO
(314) 839-0800
Telex 910-762-0627

Blue Spring, MO
(816) 229-4007

NEVADA
Zeus Electronics, Inc.
Phoenix, AZ
(602) 263-6022
1-800-528-4512
Telex 910-951-1362

NEW HAMPSHIRE
Data I/O
Nashua, NH
(602) 889-8511
Telex 943431

NEW JERSEY, NORTHERN
Data I/O
1-800-858-5803

NEW JERSEY, SOUTHERN
Scl-Rep, Inc.
Pennsauken, NJ
(609) 662-5222
Telex 710-892-1297

NEW MEXICO
Zeus Electronics, Inc.
Albuquerque, NM
(505) 842-6633
1-800-528-4512

NEW YORK, METRO
Data I/O
1-800-858-5803

NEW YORK, UPSTATE
DB Associates, Inc.
Fayettevile, NY
(315) 446-0220

NORTH CAROLINA
Pen Tech Associates
Greensboro, NC
(919) 852-6000
(305) 645-3444
Telex 62826053

NORTH DAKOTA
Torkelson Associates
Minneapolls, MN
(612) 835-2414
Telex 910-576-2740

OHIO
Electro Sales Associates
Dayton, OH
(513) 426-5551
Telex 510-1001610

Chesterland, OH
(216) 729-0190

OKLAHOMA
Testech, Inc.
Tulsa, OK
(918) 665-7788

OREGON
Northwest T & M
Beaverton, OR
(503) 646-9966
Telex 910-467-8775

PENNSYLVANIA, WESTERN
Electro Sales Associates
Allison Park, PA
(412) 487-3801

PENNSYLVANIA, EASTERN
Scl-Rep, Inc.
Pennsauken, NJ
(609) 662-5222
Telex 710-892-1297

PUERTO RICO
Data I/O
Nashua, NH
(603) 889-8511
Telex 943431

RHODE ISLAND
Data I/O
Nashua, NH
(603) 889-8511
Telex 943431

SOUTH CAROLINA
Pen Tech Associates
Greensboro, NC
(919) 852-6000
(305) 645-3444
Telex 62826053

SOUTH DAKOTA
Torkeison Associates
Minneapolls, MN
(612) 835-2414
Telex 910-576-2740

TENNESSEE
Pen Tech Associates
Marletta, GA
(404) 424-1931
Telex 6286058

Huntsville, AL
(205) 881-9298
Telex 62826048

TEXAS
Testech, Inc.
Richardson, TX
(214) 644-5010

Houston, TX
(713) 956-0837

Austin, TX
(512) 327-7033

TEXAS, EL PASO CO.
Zeus Electronics, Inc.
Albuquerque, NM
(505) 842-6633
1-800-528-4512

UTAH
Zeus Electronics, Inc.
Salt Lake City, UT
(801) 534-0500
(801) 534-0503

VERMONT
Data I/O
Nashua, NH
(603) 889-8511
Telex 943431

VIRGINIA
Scl-Rep, Inc.
Falrfax, VA
(703) 385-0600
Telex 710-833-0361

WASHINGTON, WESTERN SPOKANE
Northwest, T & M
Redmond, WA
(206) 881-8857

WASHINGTON, TRI-CITIES, VANCOUVER
Northwest T & M
Beaverton, OR
(503) 646-9966
Telex 910-467-8775

WASHINGTON D.C.
Scl-Rep, Inc.
Falrfax, VA
(703) 385-0600
Telex 710-833-0361

WEST VIRGINIA
Electro Sales Associates
Allison Park, PA
(412) 487-3801

WISCONSIN
Torkelson Associates
Waukesah, WI
(414) 784-7736

WYOMING
Zeus Electronics, Inc.
Denver, CO
(303) 321-4246
1-800-521-4512

U.S. SERVICE CENTERS

Redmond, WA	(206) 881-6444
Santa Clara, CA	(408) 727-0659
Costa Mesa, CA	(714) 662-2498
Nashua, NH	(603) 889-8513
Richardson, TX	(214) 235-0044

U.S. CORPORATE OFFICE

Data I/O Corporation
10525 Willows Road N.E.
P.O.Box 97046
Redmond, WA 98073-9746
(206) 881-6444
Telex Dom. 15-2167,
Int'l 4740166 DIO UI
FAX (206) 882-1043

EUROPE

Data I/O Europe
World Trade Center
Strawinskylaan 633
1077 XX Amsterdam
The Netherlands
(20) 622866
Telex 16616 DATIO NL
FAX 020-627255

FEDERAL REPUBLIC OF GERMANY

Data I/O Germany GmbH
Bahnhofstrasse 3
D-6453 Sellgenstadt
Federal Republic of Germany
(6182) 3088/89
Telex 4184962 DATA D

JAPAN

Data I/O Japan Company, Ltd.
Ginza Orient Bldg. 6F
8-9-13, Ginza, Chuo-ku
Tokyo 104 Japan
(03) 574-0211
Telex 2522685 DATAIO J
FAX 0118135740280

AUSTRALIA

Anltech
Adelalde
(08) 356-7333
Telex AA82579

Anltech
Brisbane
(07) 275-1766
Telex AA40141/AA51052

Anltech
Melbourne
(03) 795-9011
Telex AA31370

Anltech
Sydney
(02) 648-1711
Telex AA120238

Anltech
Perth
(09) 277-7000
Telex AA92908

Anltech
Corporate Office
Lidcombe, N.S.W. 2141
(02) 647-2266
Telex AA121299

AUSTRIA

Ing. Ernst Steiner
A-1130 Wlen
(222) 827474
Telex 135026 ES A

BELGIUM

Slmac Electronics
B-1210 Brussels
(2) 2192451
Telex 23662 SIMEIP B

BRAZIL

Sistronics Instrumentacao E
Sistemas Ltdg.
04726 Sao Paulo SP
(11) 247-5588
Telex (011) 38044

Newtec Produtos Electronicos
Ltda.
20.511-Rio de Janeiro-RJ
(21) 284-1248
Telex (021) 33619

CANADA

Allan Crawford Associates
Mississauga, Ontario L4Z1Y2
(416) 890-2010
Telex 06 961235

Allan Crawford Associates
Ottawa, Ontario K2B8K2
(613) 596-9300
Telex 053 3600

Allan Crawford Associates
St. Laurent, P.Q. H4T 1E7
(Montreal)
(514) 731-8564
Telex 05-824944

Allan Crawford Associates
Calgary, Alberta T2E 6Z5
(403) 291-3417
Telex 03 821186

Allan Crawford Associates
Burnaby, B.C. V5C6A7
(604) 294-1326
Telex 04 54247

CHINA

Dorado Company
Seattle, WA 98104
(206) 583-0000
Telex 329473 (Burgess Sea)
880212 (DORADO CO UD)

Kowloon, Hong Kong
(3) 770-2021
Telex 47833 (RULIN HX)

Beljing, The People's Rep. of China
507766 (Ext. 4017)
Telex 22163 TOMEN CN

DENMARK

ITT Multikomponent A/S
DK-2600 Glostrup
(2) 451822
Telex 33355 ITT DL
FAX 45 07 86

FINLAND

Instrumentarium Elektroniikka
SF-02631 Espoo 63
(0) 5284312
Telex 124426 HAVUL SF
FAX 09-358-0-524986

FRANCE

M.B. Electronique
F-78530, Buc
(1) 39568141
Telex 695414 MB F

GERMANY, FEDERAL REPUBLIC OF

Instrumatic Electronic GmbH
D-8032 Graefelfing
(89) 85802-0
Telex 524298 INSTR D

GREECE

Eletronics Ltd.
GR-106 75 Athens 139
(1) 7249511/15 or 7210669
Telex 216589 DARX GR

HONG KONG

Eurotherm (Far East) Ltd.
Aberdeen
5-546391
Telex 72449 EFELD HX

INDIA

Transmarketing Private Ltd.
Bombay 40018
022 4921874, 4920320 or 4926044
Telex 011 73724 TMPL IN

Bangalore
560-046

ISRAEL

Telsys Ltd.
IL-69010 Tel-Avlv
(3) 494891/5 or
(3) 494881 and 2
Telex 371279 TLSYS IL or
32392 TSEE IL
FAX 972-3-497407

ITALY

Sistrel SPA
1-00143 Roma
(6) 5915551
Telex 680356 SISTRL I

Sistrel SPA
1-20092 Cinisello Balsamo (MI)
(2) 6120129 or 618193
Telex 334643 SISTRL I

KOREA

Elcom Systems Inc.
Seoul ZIP135
(2) 555-5222
Telex K25227

MALAYSIA

GEA Technology PTE., Ltd.
Singapore 0511
65 2729412
Telex RS 37162

MEXICO

Christensen, S.A.
06470-Mexico, D.F.
546-25-95, 546-29-55
Telex 017-75612 Mycome

NETHERLANDS

Simac Electronics
5503HR Veldhoven
(40) 582911
Telex 51037 SIMAC NL

NEW ZEALAND

Warburton Franki, Ltd.
Auckland
(649) 444-2645
Telex NZ 60893

NORWAY

Teleinstrument A/S
N-1371 Asker
(2) 789460
Telex 72919 TELIN N

PORTUGAL

Decada Espectral
P-1495 Lisboa
(1) 2103420
Telex 15515 ESPEC P

SINGAPORE

GEA Technology PTE., Ltd.
(65) 2729412
Telex RS 37162

SOUTH AFRICA

Electronic Building Elements (PTY), Ltd. (EBE)
Pretorla 0001
(12) 46-9221/7
Telex 3 227868

SPAIN

Unitornics
Madrid-13
(1) 2425204
Telex 22596 UTRON E or
46786 UTRON E

SWEDEN

Macrotek AB
172 02 Sundbyberg
(8) 7330220
Telex 12543 MATEK S

SWITZERLAND

Instrumatic SA
CH-1207 Geneve
(22) 360830
Telex 28667 INSR CH

Instrumatic AG
CH-8800 Thalwil/ZH
(1) 7231410
Telex 826801 INBC CH

TAIWAN

Sertek International, Inc.
Taipei, 10479, R.O.C.
(2) 5010055
Telex 23756 SERTEK

THAILAND

Dynamic Supply Engineering
R.O.P.
Bangkok 10110
(2) 3928532, 3925313 or 3919571
Telex 82455 DYNASUP TH

TURKEY

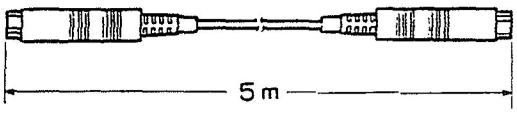
Data I/O Europe
(20) 622866

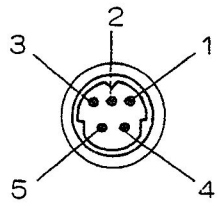
UNITED KINGDOM

Microsystem Services
High Wycombe
(494) 41661
Telex 837187 MICSYS G

Addresses on this list subject to change without notice.

8.5 CE-1600L/CE-1601T





Pin No.	Signal name
1	RD
2	GND
3	SD
4	Vcc
5	Vcc

8.6 CE-1601L ... CE-1605L

(1) CE-1601L

Pin description

PC-1600		MODEM SIDE	
Pin No.	Signal name	Pin No.	Signal name
1	FG	FG	1
2	SD	TXD(SD)	2
3	RD	RXD(RD)	3
4	RS	RTS(RS)	4
5	CS	CTS(CS)	5
6	DR	DSR(DR)	6
7	GND	SG	7
8	CD	CD	8
14	ER	DTR(ER)	20
9	CI	CI	22

(2) CE-1602L

Pin description

PC-1600		MZ-5600, MZ-5500	
Pin No.	Signal name	Pin No.	Signal name
3	RXD(RD)	SD	2
2	TXD(SD)	RD	3
8	CD	RS	4
4	RTS(RS)	CS	5
5	CTS(CS)	READY	6
7	SG	GND	7
14	DTR(ER)	DR	8
6	DSR(DR)	ER	12

(3) CE-1603L

Pin description

PC-1600		PC-5000, CE-158	
Pin No.	Signal name	Pin No.	Signal name
1	FG	FG	1
3	RXD(RD)	TXD	2
2	TXD(SD)	RXD	3
8	CD	RTS	4
8	CD	CTS	5
14	DTR(ER)	DSR	6
7	SG	GND	7
4	RTS(RS)	CD	8
		[RR]	11
5	CTS(CS)	DTR	20

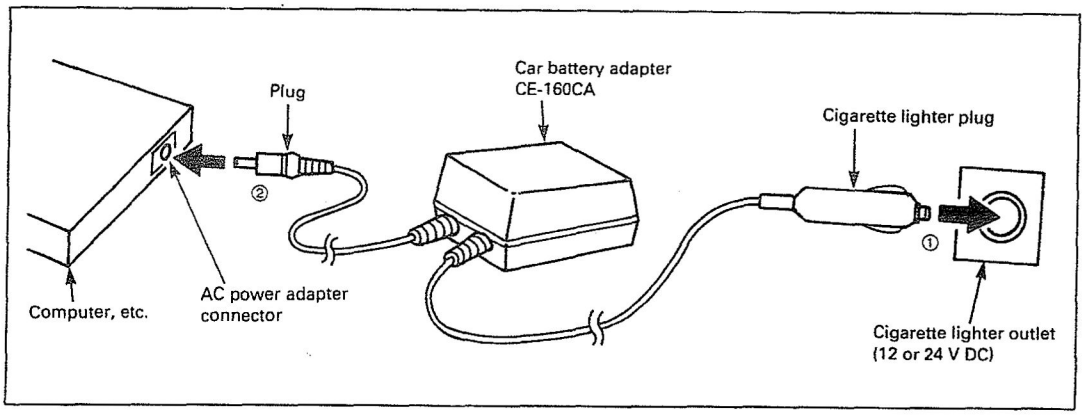
(4) CE-1605L

Pin discription

PC-1600		Open side	
Pin No.	Signal name	Pin No.	Color
1	FG	FG	Sield
3	RXD (RD)	TXD	Orange
2	TXD (SD)	RXD	Red
8	CD	RTS	Gray
8	CD	CTS	Gray
14	DTR (ER)	DSR	Cyau
7	SG	GND	Purple
4	RTS (RS)	CD	Yellow
5	CTS (CS)	DTR	Pink

8.7 CE-160CA

8.7.1 Connection



8.7.2 Specifications

Model name:	CE-160CA (Car battery adapter)
Input voltage:	12 to 24 V DC
Output voltage:	8.4 V DC
Output current:	1 A
Operating temperature:	0 to 40°C (Storage: −20 to 60°C)
External dimensions:	52 mm (W) × 76 mm (D) × 46 mm (H)
Weight:	Approx. 200 g
Connectable devices:	PC-1600, CE-1600P, CE-140P, PC-1500/PC-1500A, CE-150, etc.

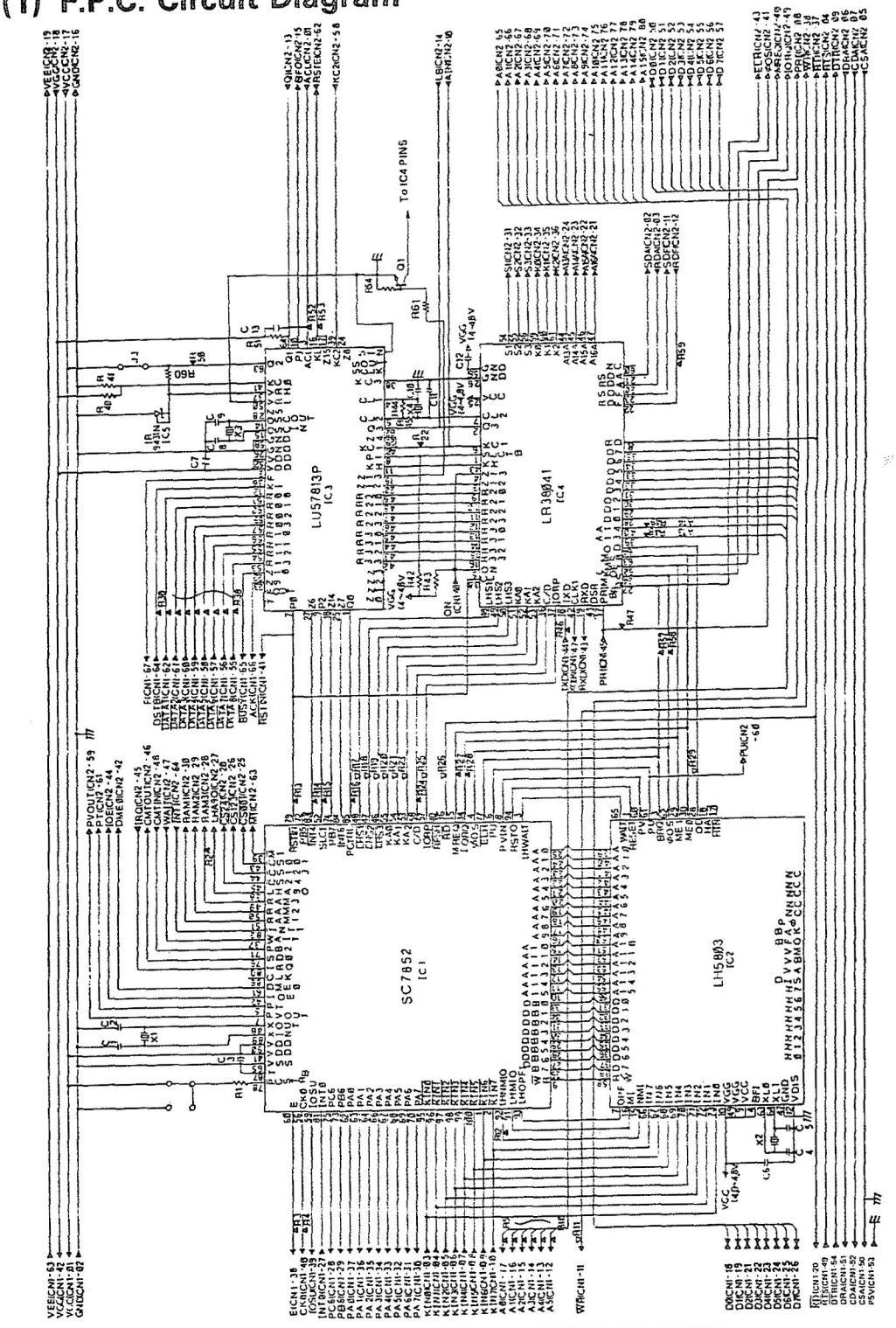
CHAPTER 9

CIRCUIT DIAGRAM

Notes 1. The names of the parts given in the following circuit diagrams are merely the general names of the parts.
2. The circuit diagrams are subject to change to functionally equivalent ones without notice.

9.1 CIRCUIT DIAGRAM OF PC-1600

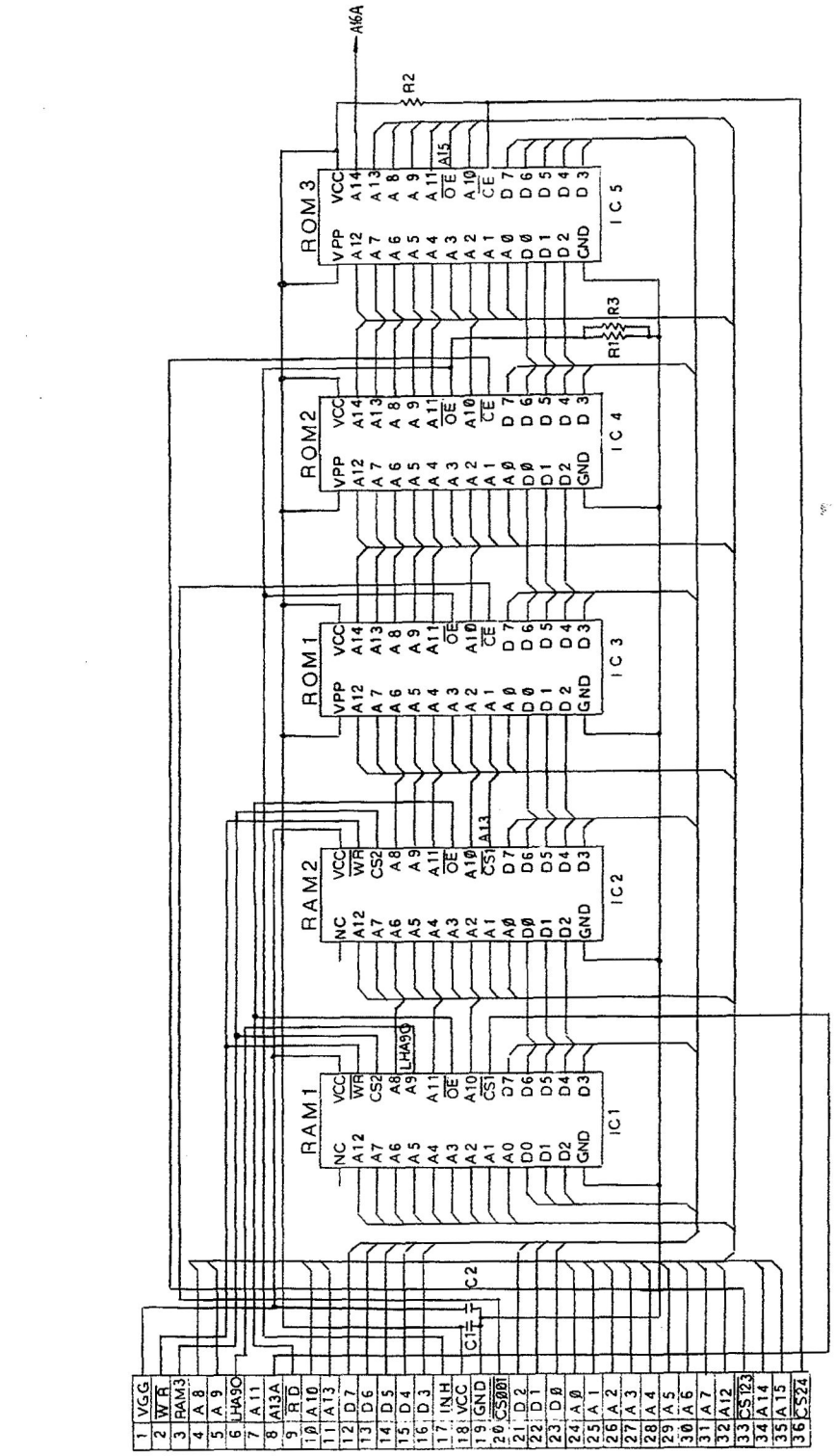
(1) F.P.C. Circuit Diagram



Unit code: DUNTK1035EC2Z

Part No.	Part name
IC1	SC7852
IC2	LH5803
IC3	LU57813P
IC4	LR38041
IC5	JR9401N
R1~13	100K Ω ± 5 % 1/10W
R14	4.7K Ω ± 5 % 1/10W
R15~38	100K Ω ± 5 % 1/10W
R40	47K Ω ± 2 % 1/10W
R41	33K Ω ± 2 % 1/10W
R42,43	100K Ω ± 5 % 1/10W
R44	1M Ω ± 5 % 1/10W
R45	1K Ω ± 5 % 1/10W
R46~50	100K Ω ± 5 % 1/10W
R51	10K Ω ± 5 % 1/10W
R52,53	100K Ω ± 5 % 1/10W
R54	22K Ω ± 5 % 1/10W
R57~59	100K Ω ± 5 % 1/10W
C1,2	100P ± 5 % CH50V
C3	0.1μ %
C4,5	100P ± 5 % CH50V
C6,7	0.1μ %
C8,9	220P ± 5 % CH50V
C10,11	47P ± 5 % CH50V
C12	0.1μ %
C13	10F
X1	CSA series 3.58M
X2	CSA series 2.45M
X3	Covered by shrinking tube 3×8 type 32.768K
X4	CSB series 1.229M
Q1	2SD1048
R60	1K Ω ± 5 % 1/4W
R61	56K Ω ± 5 % 1/4W

(2) Memory Circuit Diagram

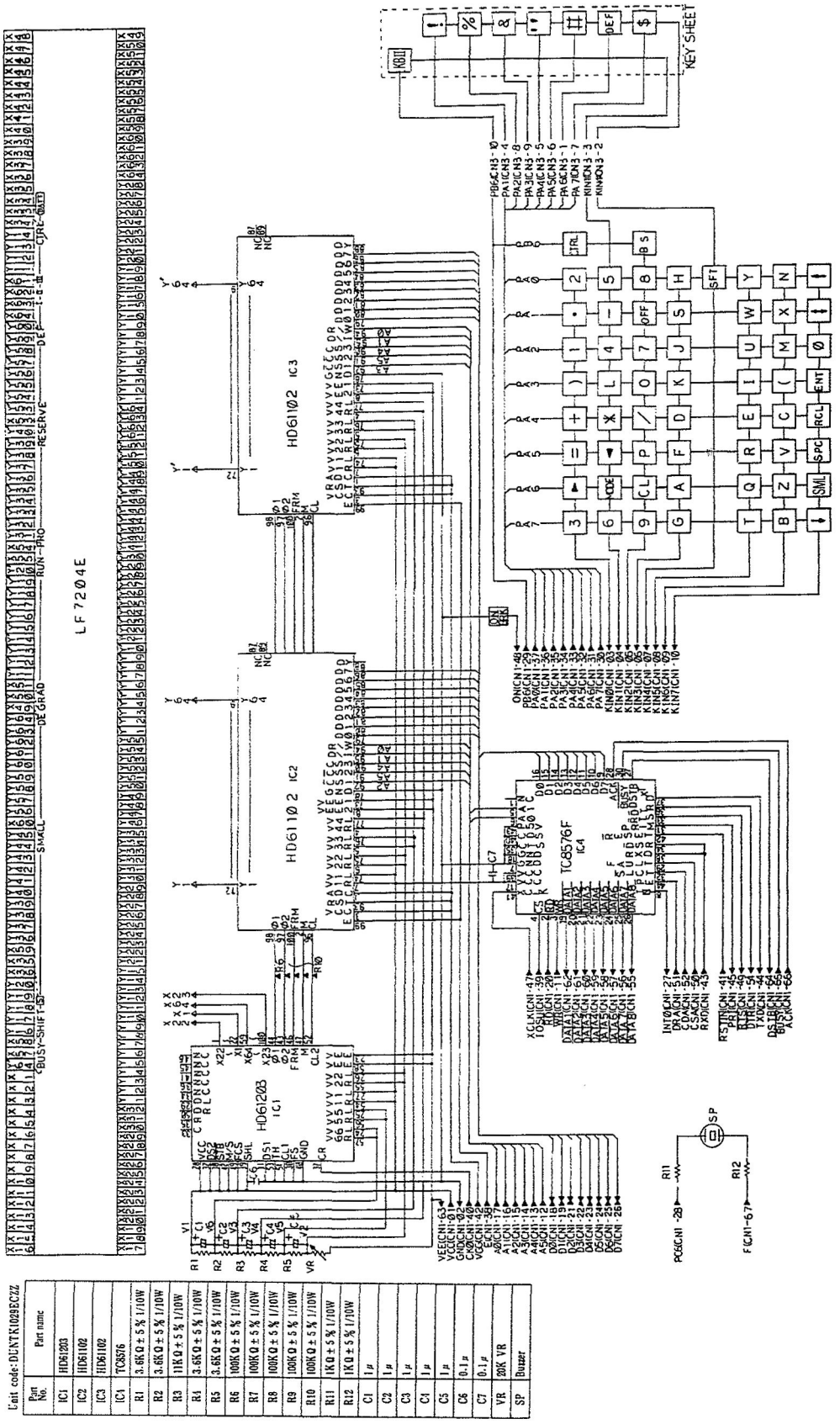


Unit code DUNTK1031EC2Z

Board name QPWBF1013EC2Z

Part No.		Part name
IC1	64K	SRAM
IC2	64K	SRAM
IC3	256K	ROM11
IC4	256K	ROM12
IC5	256K	ROM13
R1	100K Ω ± 5 % 1/10W	
R2	100K Ω ± 5 % 1/10W	
C1	0.1 μ	
C2	0.1 μ	

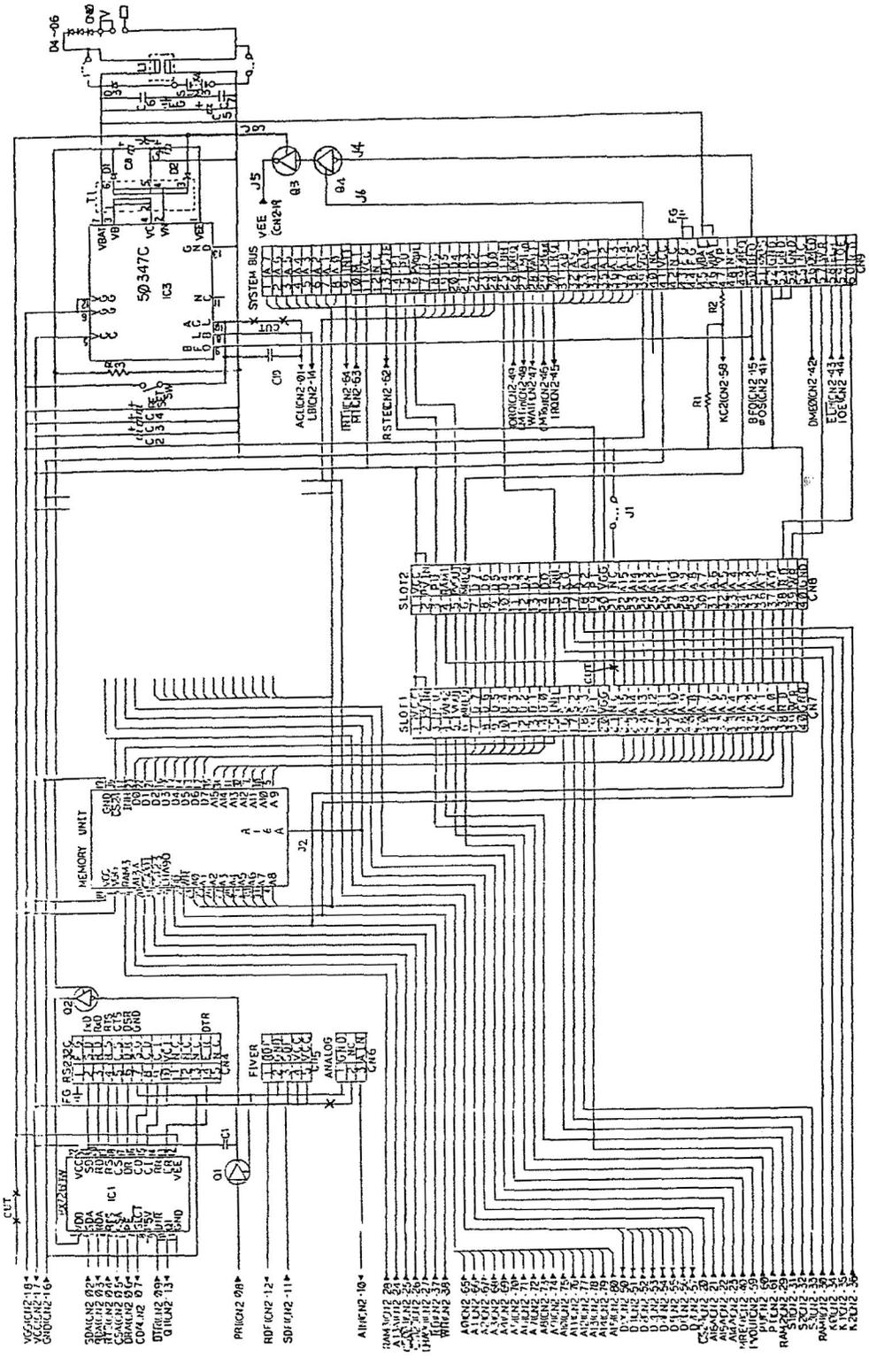
(3) Key Circuit Diagram



Unit code: DUNTK1029ECZZ

Part No.	Part name
IC1	HD61203
IC2	HD61102
IC3	HD61102
IC4	TC8576
R1	3.6KΩ ± 5% 1/10W
R2	3.6KΩ ± 5% 1/10W
R3	11KΩ ± 5% 1/10W
R4	3.6KΩ ± 5% 1/10W
R5	3.6KΩ ± 5% 1/10W
R6	100KΩ ± 5% 1/10W
R7	100KΩ ± 5% 1/10W
R8	100KΩ ± 5% 1/10W
R9	100KΩ ± 5% 1/10W
R10	100KΩ ± 5% 1/10W
R11	1KΩ ± 5% 1/10W
R12	1KΩ ± 5% 1/10W
C1	1 μ
C2	1 μ
C3	1 μ
C4	1 μ
C5	1 μ
C6	0.1 μ
C7	0.1 μ
VR	20K VR
SP	Buzzer

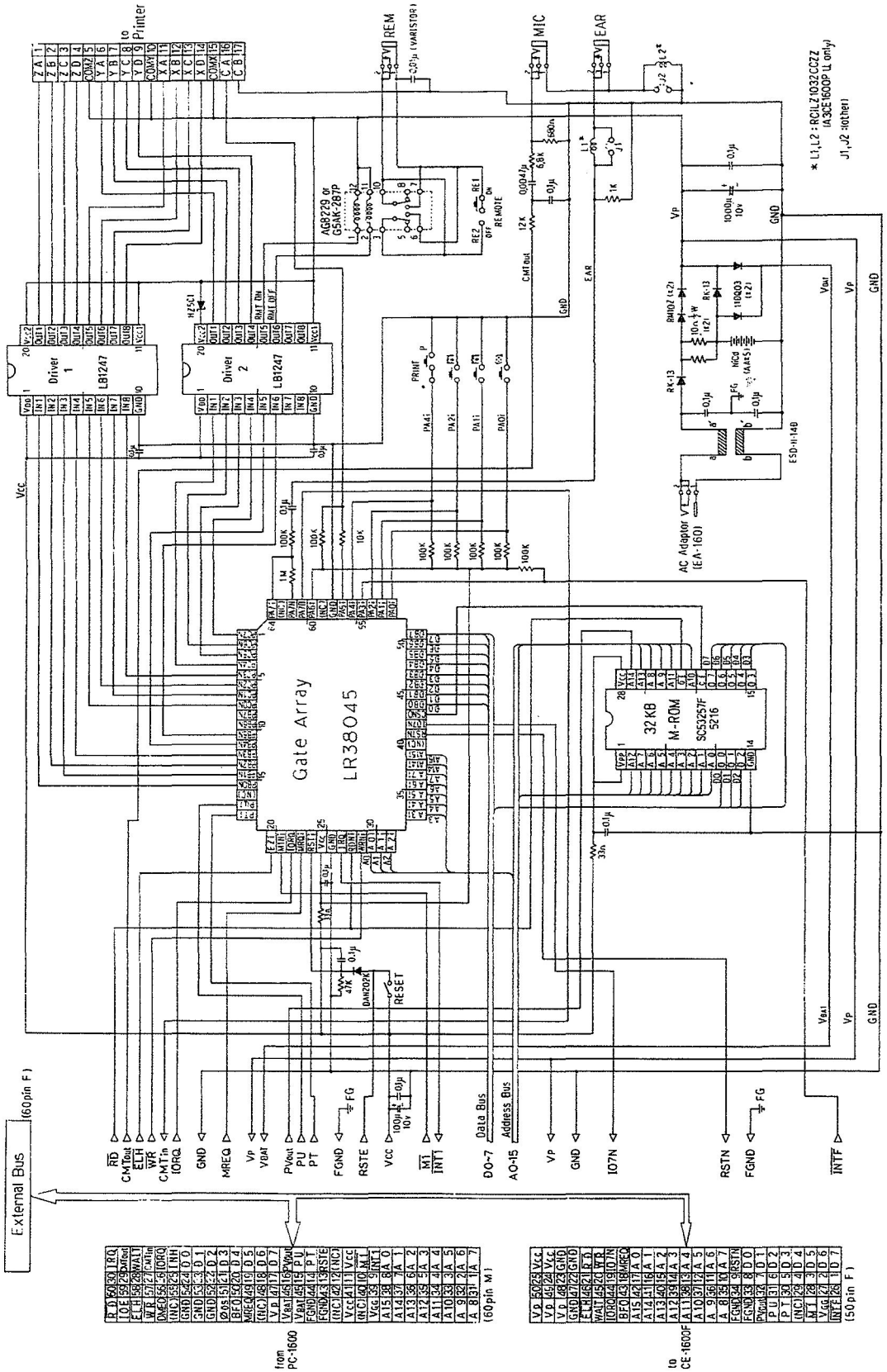
(4) Connector Circuit Diagram



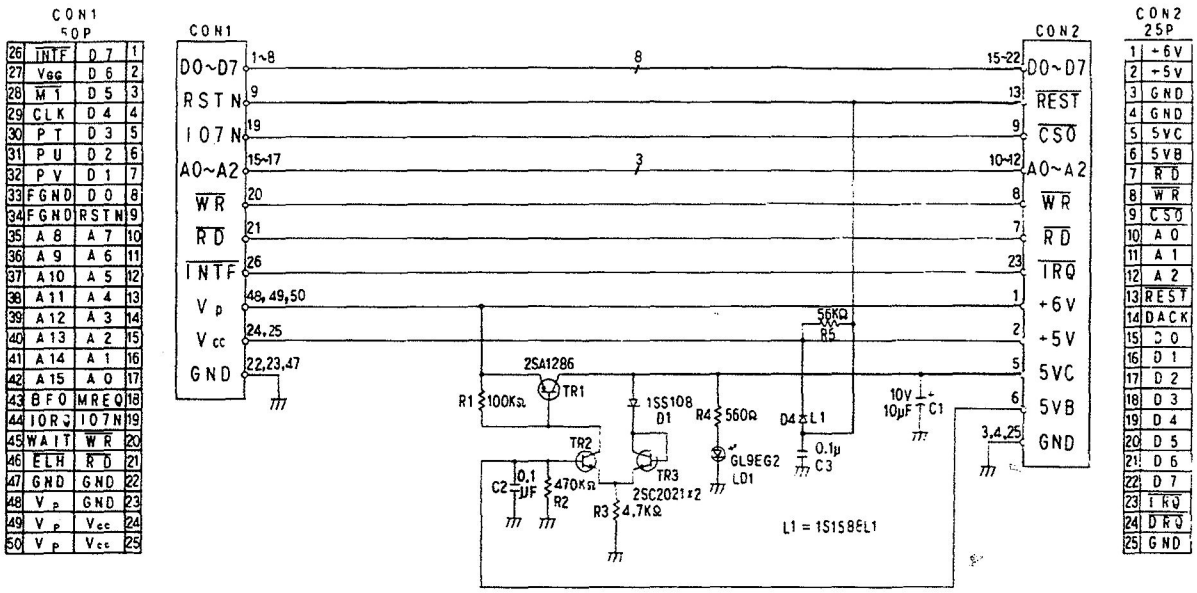
Part No.	Part name
IC1	Hybrid IC2
IC3	Hybrid IC1
Q1	Digital transistor
Q2	Digital transistor
Q3	Digital transistor
Q4	Digital transistor
R1	15K Ω
R2	33K Ω
R3	22K Ω
C1	Ceramics 470P
C2	6.3V 47 μ
C3	6.3V 22 μ
C4	6.3V 22 μ
C5	10V 47 μ
C6	0.1 μ
C7	0.1 μ
C8	10V 22 μ
C9	10V 22 μ
C10	0.01 μF
T1	Converter trasnformer
L1	Noise filter
D1	L1
D2	L1
D3	1104Q03
D4	10BIN
D5	10BIN
D6	10BIN
SW	Reset SW
CN1	RS - 232C
CN5	FIVER
CN6	ANALOG
CN7	SLOT1
CN8	SLOT2
CN9	SYSTEM BUS
CN10	Adaptor

9.2 CIRCUIT DIAGRAM OF PERIPHERAL DEVICES

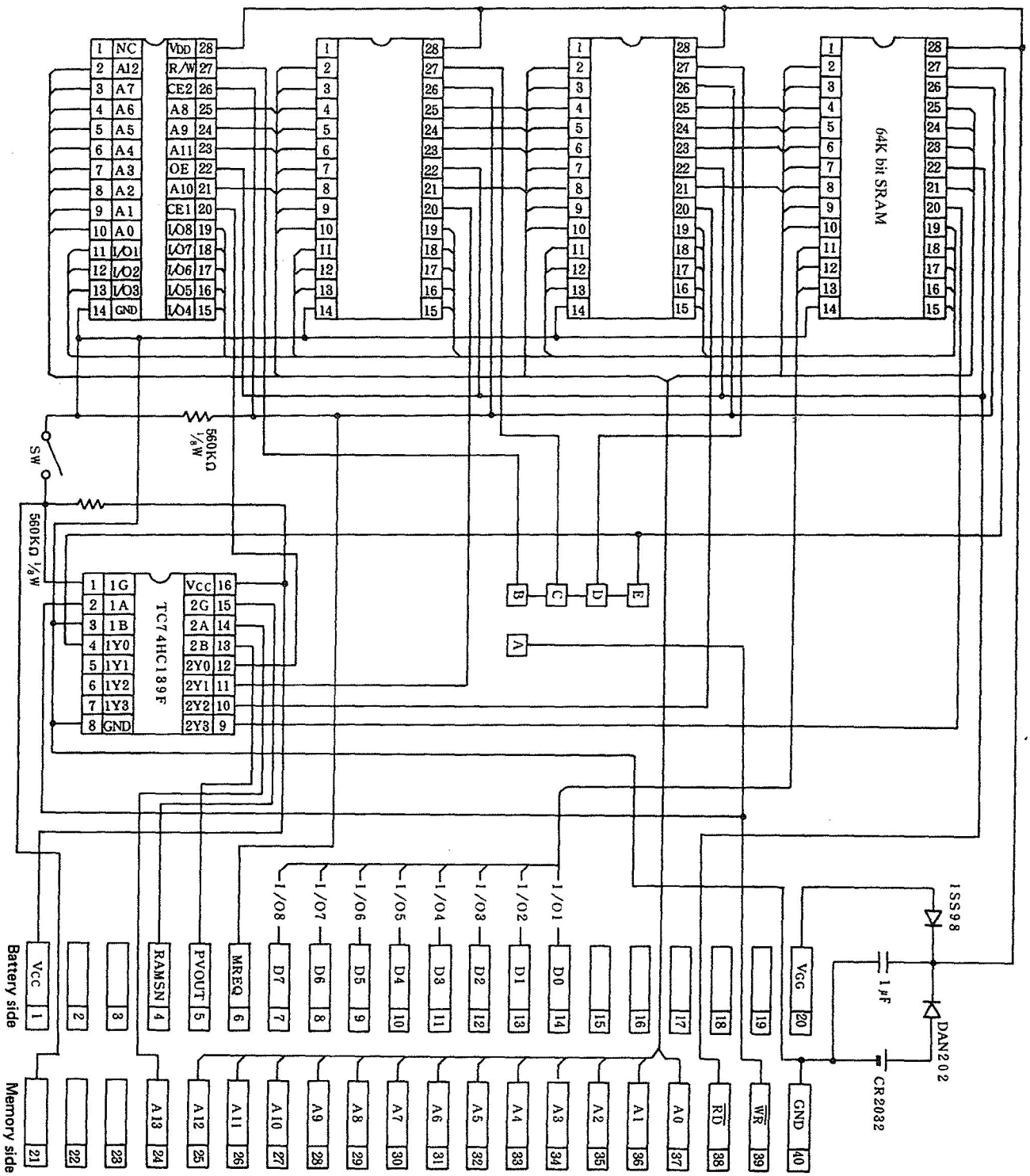
(1) CE-1600P Circuit Diagram



(2) CE-1600F Circuit Diagram

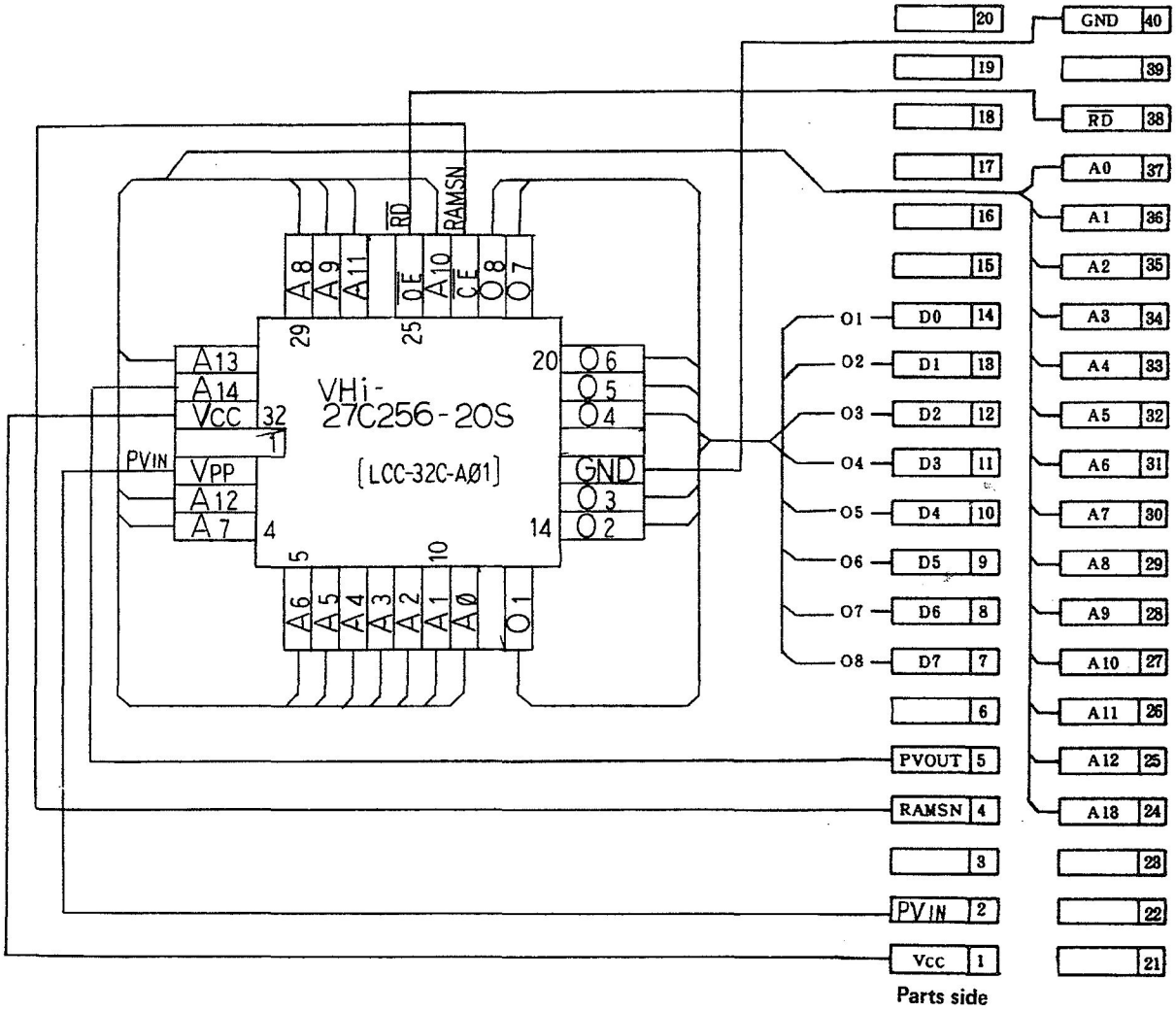


(3) CE-1600M Circuit Diagram

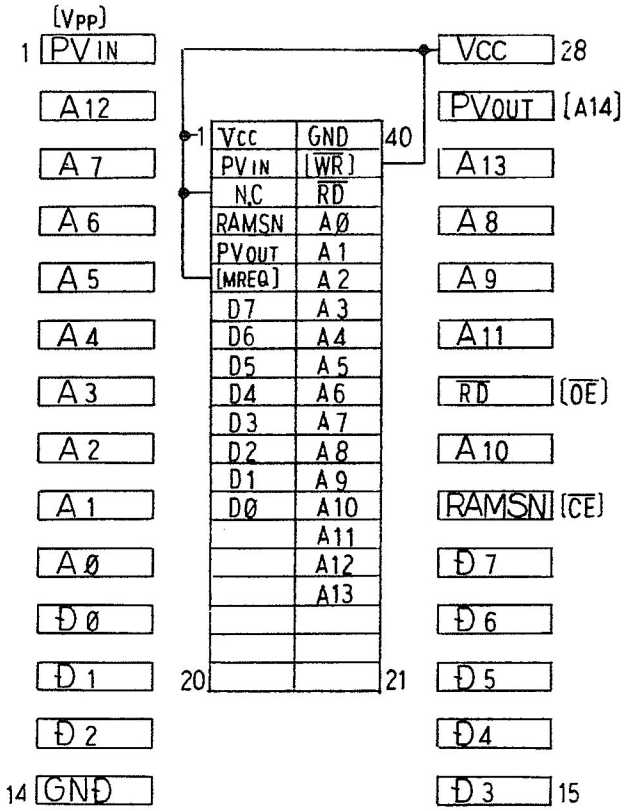


NOTE: Replacement is not permitted for the RAM chip as the wire bonding type RAM chip is used.

(4) CE-1620M Circuit Diagram



(5) CE-1610E Circuit Diagram



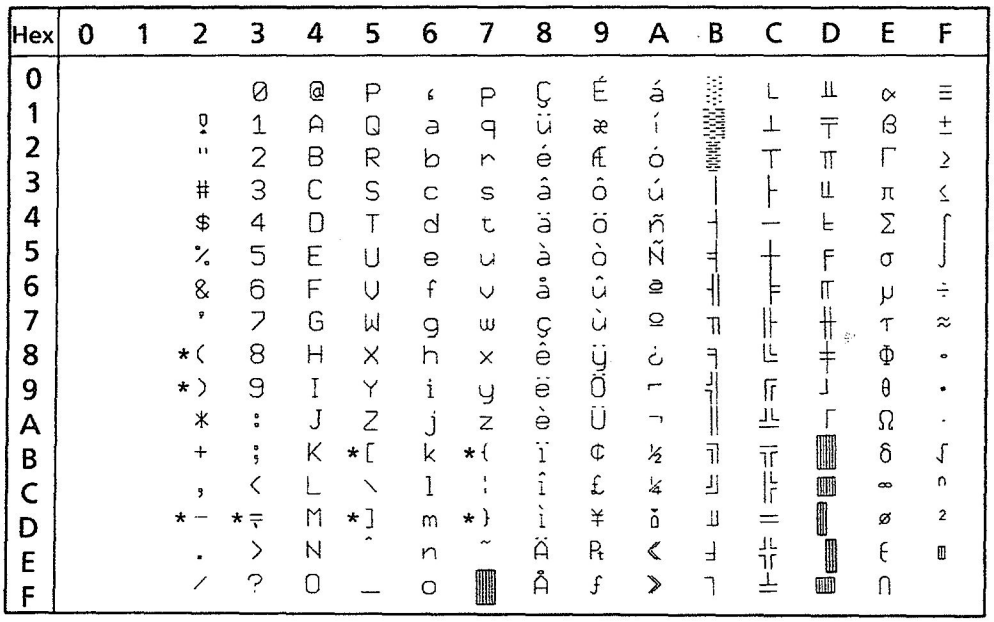
CHAPTER 10

APPENDICES

10.1 CHARACTER CODE TABLE

Mode Ø Character Code Table

In Mode Ø the PC-1600 character set includes graphic and Greek symbols and international characters in addition to the normal upper and lower case letters, numbers and symbols. This character set is similar to the IBM PC character set.



PC-1600 Character Code Table

*These characters are rotated through 90° during vertical printing set with the ROTATE command.

To look up the hexadecimal code corresponding to a character in the table, write down the number at the head of the column in which the character appears followed by the number on the left of the row in which the character is. The hexadecimal code for "a" is therefore &61.

Any character in the table can be printed out to the screen using the CHR$ (<character code>) function in BASIC. Refer to CHR$ in the Command Dictionary for details.

International Character Set

The international characters in the character code table from code value &80 to code value &A8 are assigned to the alphabetic keys on the keyboard on pressing the KBII button to the right of the six function keys. A template is provided with the international characters shown above the corresponding key positions.

Mode 1 Character Code Table

In Mode 1, the PC-1600 character set is modified to make it compatible with the more limited character set in the PC-1500. The table below gives the hexadecimal value of the character position, and the corresponding character for Mode 1.

HEX CODE	27	5B	5C	5D	5E	5F	60	7B	7C	7D	7E	7F
MODE 0	'	[	¥	]	^	_	`	{	|	}	~	
MODE 1		√	≠	π	∩						~	

10.2 KEY CODE TABLE

(1) Key codes to be retured from KEYDIRECT IOCS routine

High / Low	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	Not allocatted		SPACE	0		P										
1	SHIFT	F1		1	A	Q										
2	SMALL	F2		2	B	R										
3	CTRL	F3		3	C	S										
4	KBII	F4		4	D	T										
5	BS	F5		5	E	U										
6		F6		6	F	V										
7				7	G	W										
8	◀	CL	(	8	H	X										
9	◆	RCL	)	9	I	Y										
A	↓		*		J	Z										
B	↑	DEF	+		K											
C	▶				L											
D	ENTER		−	=	M											
E	ON		.		N											
F	OFF	MODE	/		O											

□ Keys added for PC-1600

(2) Key codes to be returned from KEYGET and KEYGETR IOCS routines

Low \ High	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NULL															
1	SHIFT	F1														
2	SMALL	F2														
3	CTRL	F3														
4	KBII	F4														
5	BS	F5														
6		F6														
7																
8	◀	CL														
9	⬍	RCL														
A	↓	CA														
B	↑	DEF														
C	▶	INS														
D	ENTER	DEL														
E	ON															
F	OFF	MODE														

- 20H to FFH are the same as the character codes.

(3) Key codes in key buffer

Low \ High	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NULL	Repeat code	SPACE	0	@	P	`	p								
1	SHIFT	F1	!	1	A	Q	a	q								
2	SMALL	F2	"	2	B	R	b	r								
3	CTRL	F3	#	3	C	S	c	s								
4	KBII	F4	$	4	D	T	d	t								
5	BS	F5	%	5	E	U	e	u								
6		F6	&	6	F	V	f	v								
7			'	7	G	W	g	w								
8	◀	CL	(	8	H	X	h	x								
9	◆	RCL	)	9	I	Y	i	y								
A	↓	CA	*	:	J	Z	j	z								
B	↑	DEF	+	;	K	[	k	{								
C	▶	INS	,	<	L	\	l	¦								
D	ENTER	DEL	−	=	M	]	m	}								
E	ON		.	>	N	^	n	~								
F	OFF	MODE	/	?	O	_	°	■								

- A code between 00H and FFH is stored to the key buffer.
- The blank squares between 00H and 7FH are assigned a space code.
- The codes from 80H to FFH are allocated to the international characters and special symbols.

10.3 CONNECTOR PIN CONFIGURATION

(1) 60-pin system bus connector

Pin No.	Signal name	Pin No.	Signal name	Pin No.	Signal name	Pin No.	Signal name
1	A7	16	PVOUT	31	A8	46	VBAT
2	A6	17	D7	32	A9	47	VP
3	A5	18	D6	33	A10	48	NC
4	A4	19	D5	34	A11	49	MREQ
5	A3	20	D4	35	A12	50	BFO
6	A2	21	D3	36	A13	51	ϕOS
7	A1	22	D2	37	A14	52	GND
8	A0	23	D1	38	A15	53	GND
9	$\overline{\text{INT1}}$	24	D0	39	VGG	54	GND
10	$\overline{\text{M1}}$	25	INH	40	NC	55	NC
11	VCC	26	IORQ	41	VCC	56	DMEO
12	NC	27	CMTIN	42	NC	57	$\overline{\text{WR}}$
13	RSTE	28	WAIT	43	FG	58	$\overline{\text{ELH}}$
14	PT	29	CMTOUT	44	FG	59	IOE
15	PU	30	IRQ	45	VBAT	60	$\overline{\text{RD}}$

(2) Memory slot 1 (S1) connector

Pin No.	Signal name	Pin No.	Signal name	Pin No.	Signal name	Pin No.	Signal name
1	VCC	11	D3	21	NC	31	A6
2	PVIN	12	D2	22	A15	32	A5
3	PU	13	D1	23	A14	33	A4
4	RAM1	14	D0	24	A13	34	A3
5	PVOUT	15	INH	25	A12	35	A2
6	MREQ	16	K0	26	A11	36	A1
7	D7	17	K1	27	A10	37	A0
8	D6	18	K2	28	A9	38	$\overline{\text{RD}}$
9	D5	19	PT	29	A8	39	$\overline{\text{WR}}$
10	D4	20	VGG	30	A7	40	GND

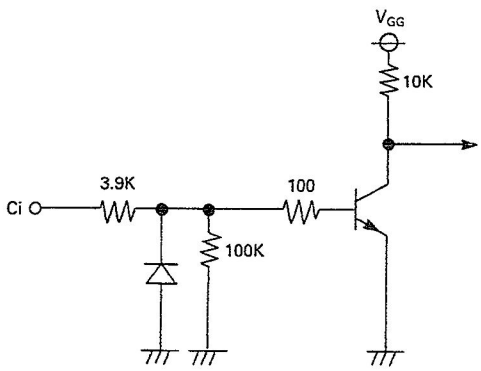
(3) Memory slot 2 (S2) connector

Pin No.	Signal name	Pin No.	Signal name	Pin No.	Signal name	Pin No.	Signal name
1	VCC	11	D3	21	NC	31	A6
2	PVIN	12	D2	22	A15	32	A5
3	PU	13	D1	23	A14	33	A4
4	RAM2	14	D0	24	A13	34	A3
5	PVOUT	15	INH	25	A12	35	A2
6	MREQ	16	S1	26	A11	36	A1
7	D7	17	S2	27	A10	37	A0
8	D6	18	S3	28	A9	38	$\overline{RD}$
9	D5	19	PT	29	A8	39	$\overline{WR}$
10	D4	20	VGG	30	A7	40	GND

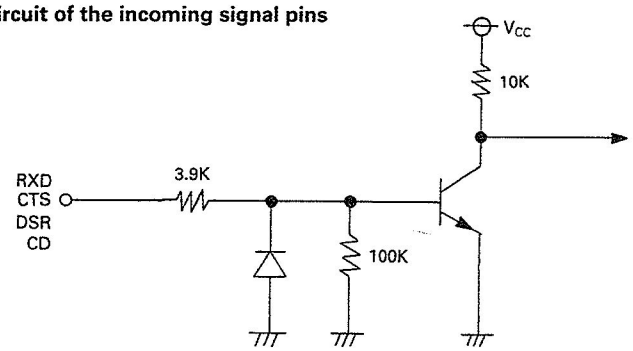
(4) RS-232C connector and input circuit

RS-232C interface connector pin configuration

Pin NO.	Signal name
1	FG
2	SD (TXD)
3	RD (RXD)
4	RS (RTS)
5	CS (CTS)
6	DS (DSR)
7	SG (GND)
8	CD
9	CI
10	VC1
11	NC
12	NC
13	NC
14	ER (DTR)
15	NC



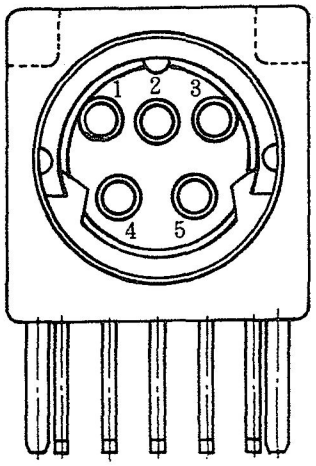
Input circuit of the incoming signal pins



(Any of the RS-232C, CMOS, and TTL levels may be accepted.)

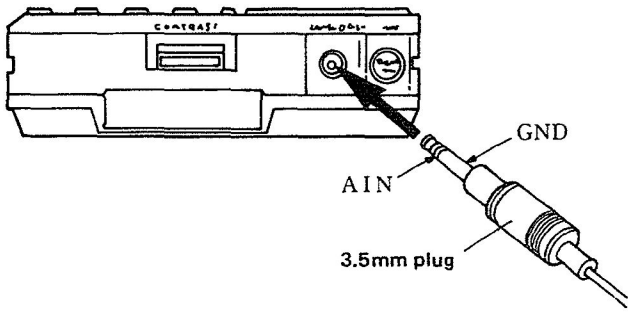
(5) SIO connector

Pin No.	Signal name
1	RD
2	GND
3	SD
4	VCC
5	VCC

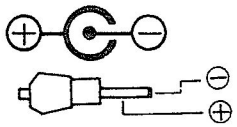


(6) Analog input plug

Pin No.	Signal name
1	GND
2	Not used.
3	AIN



(7) AC power adapter plug



10.4 Z-80 MNEMONIC CODES

(1) Flags of Z-80 CPU

The flag registers (F and F') are used to check the status of the CPU.
The flag bits are configured as follows.

7	6	5	4	3	2	1	0
S	Z	X	H	X	P/V	N	C

C : Carry flag
N : Add/subtract flag
P/V : Parity/overflow flag
H : Half-carry flag
Z : Zero flag
S : Sign flag
X : not used

These flag are set or reset depending on the CPU operation. You can check the C, P/V, Z and S flags by using an instruction such as a conditional jump instruction or a call instruction, however, you cannot directly check the H and N flags that are used for BCD operation.

Carry flag (C)
The way the carry flag is set or reset differs depending on an arithmetic operation to be performed.
The carry flag is set when a carry occurs during an ADD instruction or when a borrow occurs during a SUB instruction. If a carry or borrow does not occur, the carry flag is reset.
When the conditions for the decimal correction are satisfied, a DAA instruction sets the carry flag.
With RLA, RRA, RL and RR instructions, the carry flag is included as a bit within the bit loop.
With RLCA, RLC and SLA instructions, the content of bit 7 of the register memory location is shifted to the carry flag.
With RRCA, RRC, SRA and SRL instructions, the content of bit 0 of the register memory location is shifted to the carry flag.
With AND, OR and XOR instructions, the carry flag is reset.
The carry flag is set by SCF instruction and reset by CCF instruction.

Add/subtract flag (N)
This flag is used during a DAA instruction.
The flag is reset to "0" by an ADD instruction and is set to "1" by a SUB instruction.

Parity/overflow flag (P/V)
In an arithmetic operation, if the result to be stored in the accumulator is out of the range between −128 and +127, an overflow occurs and the parity/overflow flag is set.
The conditions in which the flag is set or reset are as follows:

1. If two numbers having different signs are added, the flag is set.
2. If two numbers having the same sign are added and the result will have the opposite sign, the flag is set.

Example	Decimal number		Binary number	
	+120	=	0111	1000
+)	+105	=	0110	1001
	− 95	=	1110	0001 (overflow)

3. If two numbers having the same sign are subtracted, the flag is reset.
4. In subtraction of two numbers having different signs, whether the flag is set or reset depends on the absolute values of these two numbers.
   For instance, the flag is set in the following case.

Example	Decimal number		Binary number	
	+127	=	0111	1111
−)	− 64	=	1100	0000
	− 65	=	1011	1111 (overflow)

The parity/overflow flag is also used to check the parity (the number of "1s" in a byte) of the result in a logical operation or in rotate instructions. If the total number of 1s is odd, it is the odd parity (P=0). If the total number of 1s is even, it is the even parity (P=1).
During a search instruction such as CPI or CPD or a block transfer instruction such as LDI or LDD, the state of the byte counter (BC) is monitored. If the byte counter is not "0", the P/V flag is set to "1", and if the byte counter becomes "0", the flag is reset to "0".
During an (LD A,I) instruction or an (LD A,R) instruction, the content of IFF2 (interrupt enable flip-flop 2) is copied to the P/V flag. Thus, you can save or test the content of IFF2.
When you read one byte from an I/O device by IN r,(C) instruction, the P/V flag is set or reset depending on the parity of the data read.

Half-carry flag (H)
The half-carry flag is set or reset depending on whether there is a carry or borrow between bit 3 and bit 4 during an 8-bit arithmetic operation.
The half-carry flag is used in a DAA instruction to correct the result of an add or subtract operation for packed BCD numbers.
The flag is set to "1" if a carry or borrow occurs, and reset to "0" if a carry or borrow does not occur.

Zero flag (Z)
The zero flag is set or reset depending on whether the result of an instruction is zero or not.
If the result (i.e., the content of the accumulator) of an 8-bit arithmetic or logical operation is zero, the zero flag is set to "1", and if the result is not zero, the flag is reset to "0".
During a search instruction, if the content of the accumulator coincides with the content of the memory location specified by HL register pair, the zero flag is set to "1".
During a bit test instruction, the complement of the specified bit is copied to the zero flag.
During an input/output instruction (INI, IND, OUTI or OUTD), if the byte counter value minus 1 (B-1) is zero, the zero flag is set to "1". If not zero, the zero flag is reset to "0".
During an IN r,(C) instruction, if the input data is zero, the zero flag is set to "1".

Sign flag (S)

The sign flag holds the content of bit 7 of the accumulator and is used for arithmetic operation. For a signed operation, a complement of 2 is used. If bit 7 is "0", the value is a positive number, and if bit 7 is "1", the value is a negative number.
The range of a positive number is that can be expressed by 7 bits (from 0 to 127), and the range of a negative number is also that can be expressed by 7 bits (from −1 to −128).

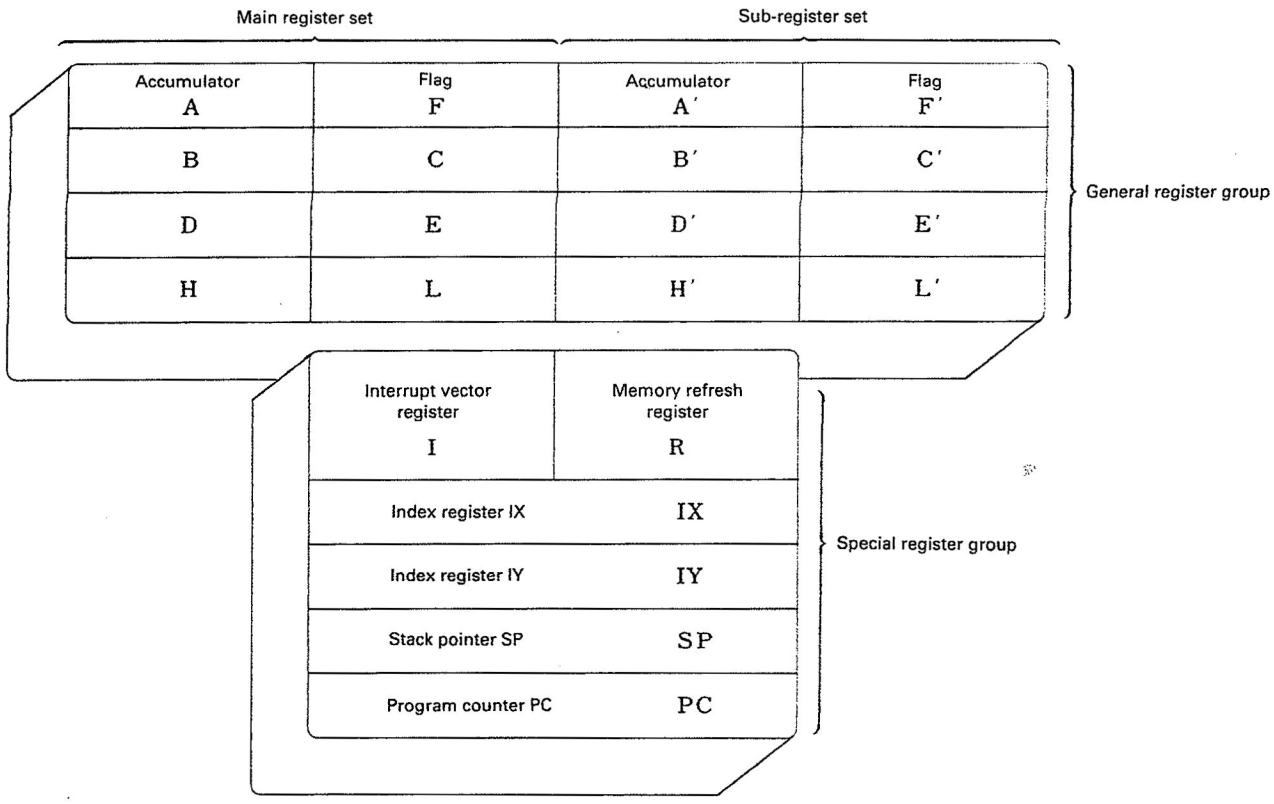
During an input instruction (IN r,(C)), the polarity of the data read is copied to the sign flag: if the data is positive, the sign flag is reset to "0", and if it is negative, the sign flag is set to "1".

Symbols indicating the state of the flags used for Z-80 instruction set

↕	The flag will be affected by the result.
•	The flag will not change.
0	The flag will be reset.
1	The flag will be set.
X	The flag content will be destroyed.
V	The flag will be set if an overflow occurs, otherwise, it will be reset.
P	The flag will be set if the parity is even, otherwise, it will be reset.

(2) Structure of internal registers

The internal registers of Z-80 CPU are composed of 207 bits of read/write memory, and have the following structure.



The CPU registers consist of the general register group and the special register group. The general register group consists of two sets of registers: the main register set and the sub-register set, and the content of a main register can be exchanged with the content of the corresponding sub-register by using an exchange instruction.

Each register set consists of one 8-bit accumulator, one 8-bit flag register and six 8-bit general registers. Two general registers can be paired like BC, DE or HL to be used as one 16-bit register.

The interrupt vector register I (8-bit register), upon interruption, specifies the upper 8 bits of the indirect address of an interrupt service routine, while the lower 8 bits of the address is given by the device which has issued the interrupt.

The memory refresh register R (7-bit register) automatically generates memory refresh addresses when a dynamic RAM is used as the external memory.

Symbols for registers used for Z-80 instruction set

r, r'	: Any one of the CPU internal registers A, B, C, D, E, H, and L
dd, pp, qq, rr, ss	: A register pair composed of CPU internal registers
ii	: One of the index registers IX and IY
R	: Memory refresh register
d	: 8-bit displacement used for memory location using an index register
e	: A signed complement of 2 (from −126 to 129) in the relative addressing mode
n	: 8-bit data (from 0 to 255)
nn	: 16-bit memory location (from 0 to 65535)

(3) 8-bit load instructions

Mnemonic code	Operation	Flags						OP code	No. of bytes	No. of M cycles	No. of T states	Remarks
		C	Z	P/V	S	N	H	76 543 210				
LD r,r′	r←r′	•	•	•	•	•	•	01 r r′	1	1	4	r, r′ / Register: 000 B; 001 C; 010 D; 011 E; 100 H; 101 L; 111 A
LD r,n	r←n	•	•	•	•	•	•	00 r 110 ← n →	2	2	7	
LD r,(HL)	r←(HL)	•	•	•	•	•	•	01 r 110	1	2	7	
LD r,(IX+d)	r←(IX+d)	•	•	•	•	•	•	11 011 101 01 r 110 ← d →	3	5	19	
LD r,(IY+d)	r←(IY+d)	•	•	•	•	•	•	11 111 101 01 r 110 ← d →	3	5	19	
LD (HL),r	(HL)←r	•	•	•	•	•	•	01 110 r	1	2	7	
LD (IX+d),r	(IX+d)←r	•	•	•	•	•	•	11 011 101 01 110 r ← d →	3	5	19	
LD (IY+d),r	(IY+d)←r	•	•	•	•	•	•	11 111 101 01 110 r ← d →	3	5	19	
LD (HL),n	(HL)←n	•	•	•	•	•	•	00 110 110 ← n →	2	3	10	
LD (IX+d),n	(IX+d)←n	•	•	•	•	•	•	11 011 101 00 110 110 ← d → ← n →	4	5	19	
LD (IY+d),n	(IY+d)←n	•	•	•	•	•	•	11 111 101 00 110 110 ← d → ← n →	4	5	19	
LD A,(BC)	A←(BC)	•	•	•	•	•	•	00 001 010	1	2	7	
LD A,(DE)	A←(DE)	•	•	•	•	•	•	00 011 010	1	2	7	
LD A,(nn)	A←(nn)	•	•	•	•	•	•	00 111 010 ← n → ← n →	3	4	13	
LD (BC),A	(BC)←A	•	•	•	•	•	•	00 000 010	1	2	7	
LD (DE),A	(DE)←A	•	•	•	•	•	•	00 010 010	1	2	7	
LD (nn),A	(nn)←A	•	•	•	•	•	•	00 110 010 ← n → ← n →	3	4	13	
LD A,I	A←I	•	↕	IFF2	↕	0	0	11 101 111 01 010 111	2	2	9	IFF2: Content of the interrupt enable flip-flop 2
LD A,R	A←R	•	↕	IFF2	↕	0	0	11 101 101 01 011 111	2	2	9	
LD I,A	I←A	•	•	•	•	•	•	11 101 101 01 000 111	2	2	9	
LD R,A	R←A	•	•	•	•	•	•	11 101 101 01 001 111	2	2	9	

(4) 16-bit load instructions

Mnemonic code	Operation	Flags						OP code	No. of bytes	No. of M cycles	No. of T states	Remarks	
		C	Z	P/V	S	N	H	76 543 210				dd	Register
LD dd,nn	dd←nn	•	•	•	•	•	•	00 dd0 001 ← n → ← n →	3	3	10	00	BC
LD IX,nn	IX←nn	•	•	•	•	•	•	11 011 101 00 100 001 ← n → ← n →	4	4	14	01	DE
LD IY,nn	IY←nn	•	•	•	•	•	•	11 111 101 00 100 001 ← n → ← n →	4	4	14	10	HL
LD HL,(nn)	$H \leftarrow (nn+1)$ $L \leftarrow (nn)$	•	•	•	•	•	•	00 101 010 ← n → ← n →	3	5	16	11	SP
LD dd,(nn)	$dd_H \leftarrow (nn+1)$ $dd_L \leftarrow (nn)$	•	•	•	•	•	•	11 101 101 01 dd1 011 ← n → ← n →	4	6	20		
LD IX,(nn)	$IX_H \leftarrow (nn+1)$ $IX_L \leftarrow (nn)$	•	•	•	•	•	•	11 011 101 00 101 010 ← n → ← n →	4	6	20		
LD IY,(nn)	$IY_H \leftarrow (nn+1)$ $IY_L \leftarrow (nn)$	•	•	•	•	•	•	11 111 101 00 101 010 ← n → ← n →	4	6	20		
LD (nn),HL	$(nn+1) \leftarrow H$ $(nn) \leftarrow L$	•	•	•	•	•	•	00 100 010 ← n → ← n →	3	5	16		
LD (nn),dd	$(nn+1) \leftarrow dd_H$ $(nn) \leftarrow dd_L$	•	•	•	•	•	•	11 101 101 01 dd0 011 ← n → ← n →	4	6	20		
LD (nn),IX	$(nn+1) \leftarrow IX_H$ $(nn) \leftarrow IX_L$	•	•	•	•	•	•	11 011 101 00 100 010 ← n → ← n →	4	6	20		
LD (nn),IY	$(nn+1) \leftarrow IY_H$ $(nn) \leftarrow IY_L$	•	•	•	•	•	•	11 111 101 00 100 010 ← n → ← n →	4	6	20		
LD SP,HL	SP←HL	•	•	•	•	•	•	11 111 001	1	1	6		
LD SP,IX	SP←IX	•	•	•	•	•	•	11 011 101 11 111 001	2	2	10		
LD SP,IY	SP←IY	•	•	•	•	•	•	11 111 101 11 111 001	2	2	10		

(5) Exchange, block transfer, and search instructions

Mnemonic code	Operation	Flags						OP code	No. of bytes	No. of M cycles	No. of T states	Remarks
		C	Z	P/V	S	N	H	76 543 210				
EX DE, HL	DE↔HL	•	•	•	•	•	•	11 101 011	1	1	4	
EX AF, AF'	AF↔AF'	•	•	•	•	•	•	00 001 000	1	1	4	
EXX	$\begin{pmatrix}BC\\DE\\HL\end{pmatrix}\leftrightarrow\begin{pmatrix}BC'\\DE'\\HL'\end{pmatrix}$	•	•	•	•	•	•	11 011 001	1	1	4	Exchange the content of a register pair with the content of the corresponding sub-register pair.
EX (SP), HL	H↔(SP+1) L↔(SP)	•	•	•	•	•	•	11 100 011	1	5	19	
EX (SP), IX	IX_H↔(SP+1) IX_L↔(SP)	•	•	•	•	•	•	11 011 101 11 100 011	2	6	23	
EX (SP), IY	IY_H↔(SP+1) IY_L↔(SP)	•	•	•	•	•	•	11 111 101 11 100 011	2	6	23	
LDI	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC−1	•	•	↕ ①	•	0	0	11 101 101 10 100 000	2	4	16	
LDIR	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC−1 Repeat until BC=0	•	•	0 ①	•	0	0	11 101 101 10 110 000	2 2	5 4	21 16	When BC≠0 When BC=0
LDD	(DE)←(HL) DE←DE−1 HL←HL−1 BC←BC−1	•	•	↕	•	0	0	11 101 101 10 101 000	2	4	16	
LDDR	(DE)←(HL) DE←DE-1 HL←HL-1 BC←BC-1 Repeat until BC=0	•	•	0 ①	•	0	0	11 101 101 10 111 000	2 2	5 4	21 16	When BC≠0 When BC=0
CPI	A−(HL) HL←HL+1 BC←BC−1	•	↕ ②	↕ ①	↕	1	↕	11 101 101 10 100 001	2	4	16	

Mnemonic code	Operation	Flags						OP code	No. of bytes	No. of M cycles	No. of T states	Remarks	
		C	Z	P/V	S	N	H	76 543 210					
PUSH qq	$(SP-2) \leftarrow qq_L$ $(SP-1) \leftarrow qq_H$	•	•	•	•	•	•	11 qq0 101	1	3	11	qq	Register
PUSH IX	$(SP-2) \leftarrow IX_L$ $(SP-1) \leftarrow IX_H$	•	•	•	•	•	•	11 011 101 11 100 101	2	4	15	00	BC
PUSH IY	$(SP-2) \leftarrow IY_L$ $(SP-1) \leftarrow IY_H$	•	•	•	•	•	•	11 111 101 11 100 101	2	4	15	01	DE
POP qq	$qq_H \leftarrow (SP+1)$ $qq_L \leftarrow (SP)$	•	•	•	•	•	•	11 qq0 001	1	3	10	10	HL
POP IX	$IX_H \leftarrow (SP+1)$ $IX_L \leftarrow (SP)$	•	•	•	•	•	•	11 011 101 11 100 001	2	4	14	11	AF
POP IY	$IY_H \leftarrow (SP+1)$ $IY_L \leftarrow (SP)$	•	•	•	•	•	•	11 111 101 11 100 001	2	4	14		

Mnemonic code	Operation	Flags C	Z	P/V	S	N	H	OP code 76 543 210	No. of bytes	No. of M cycles	No. of T states	Remarks
			②	①								
CPIR	A−(HL) HL←HL+1 BC←BC−1 Repeat until A=(HL) or BC=0	●	↕	↕	↕	1	↕	11 101 101 10 110 001	2 2	5 4	21 16	When BC≠0 and A≠(HL) When BC=0 or A=(HL)
			②	①								
CPD	A−(HL) HL←HL−1 BC←BC−1	●	↕	↕	↕	1	↕	11 101 101 10 101 001	2	4	16	
			②	①								
CPDR	A−(HL) HL←HL−1 BC←BC−1 Repeat until A=(HL) or BC=0	●	↕	↕	↕	1	↕	11 101 101 10 111 001	2 2	5 4	21 16	When BC≠0 and A≠(HL) When BC=0 or A=(HL)

Note The P/V flag indicated by ① in the table is set to "0" if the result is BC−1=0, otherwise the flag is set to "1".
The Z flag indicated by ② in the table is set to "1" if A=(HL), otherwise the flag is set to "0".

(6) Correction flag and CPU control instructions

Mnemonic code	Operation	Flags C	Z	P/V	S	N	H	OP code 76 543 210	No. of bytes	No. of M cycles	No. of T states	Remarks
DAA		↕	↕	P	↕	⊛	↕	00 100 111	1	1	4	Binary coded decimal number (BCD) correction
CPL	$A \leftarrow \bar{A}$	●	●	●	⊛	1	1	00 101 111	1	1	4	
NEG	A←0−A	↕	↕	V	↕	1	↕	11 101 101 01 000 100	2	2	8	
CCF	$CY \leftarrow \overline{CY}$	↕	●	⊛	⊛	0	X	00 111 111	1	1	4	
SCF	CY←1	↕	●	⊛	⊛	0	0	00 110 111	1	1	4	
NOP	No action, but PC←PC+1	⊛	⊛	⊛	⊛	⊛	⊛	00 000 000	1	1	4	
HALT	Halt the CPU.	⊛	●	⊛	⊛	⊛	⊛	01 110 110	1	1	4	
DI	IFF←0	⊛	⊛	⊛	⊛	⊛	⊛	11 110 011	1	1	4	
EI	IFF←1	⊛	⊛	⊛	⊛	⊛	⊛	11 111 011	1	1	4	
IM 0	Set in the interrupt mode 0.	⊛	⊛	⊛	⊛	⊛	⊛	11 101 101 01 000 110	2	2	8	
IM 1	Set in the interrupt mode 1.	⊛	⊛	⊛	⊛	⊛	⊛	11 101 101 01 010 110	2	2	8	
IM 2	Set in the interrupt mode 2.	⊛	⊛	⊛	⊛	⊛	⊛	11 101 101 01 011 110	2	2	8	

(7) 8-bit arithmetic and logical operation instructions

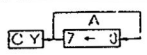
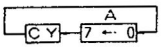
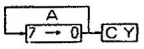
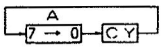
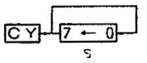
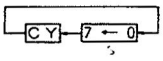
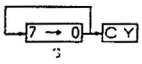
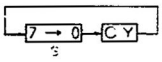
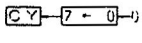
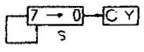
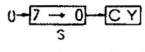
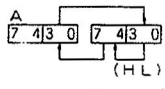
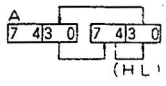
Mnemonic code	Operation	Flags						OP code	No. of bytes	No. of M cycles	No. of T states	Remarks
		C	Z	P/V	S	N	H	76 543 210				
ADD A,r	A←A+r	↕	↕	V	↕	0	↕	10 [000] r	1	1	4	
ADD A,n	A←A+n	↕	↕	V	↕	0	↕	11 [000] 110 ← n →	2	2	7	
ADD A,(HL)	A←A+(HL)	↕	↕	V	↕	0	↕	10 [000] 110	1	2	7	
ADD A,(IX+d)	A←A+(IX+d)	↕	↕	V	↕	0	↕	11 011 101 10 [000] 110 ← d →	3	5	19	
ADD A,(IY+d)	A←A+(IY+d)	↕	↕	V	↕	0	↕	11 111 101 10 [000] 110 ← d →	3	5	19	
ADC A,s	A←A+s+CY	↕	↕	V	↕	0	↕	[001]				Similar to ADD instructions, the operand s may be any of r, n, (HL), (IX+d), and (IY+d). The OP code is the same as that of the corresponding ADD instruction, with the [000] part replaced with those bits enclosed in a box.
SUB s	A←A−s	↕	↕	V	↕	1	↕	[010]				
SBC A,s	A←A−s−CY	↕	↕	V	↕	1	↕	[011]				
AND s	A←A∧s	0	↕	P	↕	0	1	[100]				
OR s	A←A∨s	0	↕	P	↕	0	0	[110]				
XOR s	A←A⊕s	0	↕	P	↕	0	0	[101]				
CP s	A—s	↕	↕	V	↕	1	↕	[111]				
INC r	r←r+1	•	↕	V	↕	0	↕	00 r [100]	1	1	4	
INC (HL)	(HL)←(HL)+1	•	↕	V	↕	0	↕	00 110 [100]	1	3	11	
INC (IX+d)	(IX+d)←(IX+d)+1	•	↕	V	↕	0	↕	11 011 101 00 110 [100] ← d →	3	6	23	
INC (IY+d)	(IY+d)←(IY+d)+1	•	↕	V	↕	0	↕	11 111 101 00 110 [100] ← d →	3	6	23	
DEC m	m←m−1	•	↕	V	↕	1	↕	[101]				Similar to INC instructions, the operand m may be any of r, (HL), (IX+d), and (IY+d). The OP code is the same as that of the corresponding INC instruction, with the [100] replaced with [101].

r	Register
000	B
001	C
010	D
011	E
100	H
101	L
111	A

(8) 16-bit arithmetic operation instructions

Mnemonic code	Operation	Flags						OP code	No. of bytes	No. of Mi cycles	No. of T states	Remarks	
		C	Z	P/V	S	N	H	76 543 210					
ADD HL,ss	HL←HL+ss	↕	●	●	●	0	X	00 ss1 001	1	3	11	ss	Register
ADC HL,ss	HL←HL+ss+CY	↕	↕	V	↕	0	X	11 101 101	2	4	15	00	BC
								01 ss1 010				01	DE
SBC HL,ss	HL←HL−ss−CY	↕	↕	V	↕	1	X	11 101 101	2	4	15	10	HL
								01 ss0 010				11	SP
ADD IX,pp	IX←IX+pp	↕	●	●	●	0	X	11 011 101	2	4	15	pp	Register
								00 pp1 001				00	BC
												01	DE
												10	IX
												11	SP
ADD IY,rr	IY←IY+rr	↕	●	●	●	0	X	11 111 101	2	4	15	rr	Register
								00 rr1 001				00	BC
												01	DE
												10	IY
												11	SP
INC ss	ss←ss+1	●	●	●	●	●	●	00 ss0 011	1	1	6		
INC IX	IX←IX+1	●	●	●	●	●	●	11 011 101	2	2	10		
								00 100 011					
INC IY	IY←IY+I	●	●	●	●	●	●	11 111 101	2	2	10		
								00 100 011					
DEC ss	ss←ss−1	●	●	●	●	●	●	00 ss1 011	1	1	6		
DEC IX	IX←IX−1	●	●	●	●	●	●	11 011 101	2	2	10		
								00 101 011					
DEC IY	IY←IY−1	●	●	●	●	●	●	11 111 101	2	2	10		
								00 101 011					

(9) Rotate and shift instructions

Mnemonic code	Operation	Flags C	Z	P/V	S	N	H	OP code 76 543 210	No. of bytes	No. of M cycles	No. of T states	Remarks
RLC A		‡	•	•	•	0	0	00 000 111	1	1	4	Rotate the content of the accumulator to the left.
RL A		‡	•	•	•	0	0	00 010 111	1	1	4	
RRC A		‡	•	•	•	0	0	00 001 111	1	1	4	Rotate the content of the accumulator to the right.
RR A		‡	•	•	•	0	0	00 011 111	1	1	4	
RLC r		‡	‡	P	‡	0	0	11 001 011 00 [000] r	2	2	8	Rotate the content of register r to the left.
RLC (HL)		‡	‡	P	‡	0	0	11 001 011 00 [000] 110	2	4	15	
RLC (IX+d)		‡	‡	P	‡	0	0	11 011 101 11 001 011 ← d → 00 [000] 110	4	6	23	
RLC (IY+d)		‡	‡	P	‡	0	0	11 111 101 11 001 011 ← d → 00 [000] 110	4	6	23	
RL s		‡	‡	P	‡	0	0	[010]				The operand s may be any of r, (HL), (IX+d) and (IY+d).
RRC s		‡	‡	P	‡	0	0	[001]				
RR s		‡	‡	P	‡	0	0	[011]				
SLA s		‡	‡	P	‡	0	0	[100]				
SRA s		‡	‡	P	‡	0	0	[101]				
SRL s		‡	‡	P	‡	0	0	[111]				
RLD		•	‡	P	‡	0	0	11 101 101 01 101 111	2	5	18	
RRD		•	‡	P	‡	0	0	11 101 101 01 100 111	2	5	18	

r	Register
000	B
001	C
010	D
011	E
100	H
101	L
111	A

(10) Bit manipulation (set, reset, test) instructions

Mnemonic code	Operation	Flags						OP code	No. of bytes	No. of M cycles	No. of T states	Remarks
		C	Z	P/V	S	N	H	76 543 210				
BIT b,r	$Z \leftarrow \overline{r}_b$	•	‡	X	X	0	1	11 001 011 01 b r	2	2	8	
BIT b,(HL)	$Z \leftarrow \overline{(HL)}_b$	•	‡	X	X	0	1	11 001 011 01 b 110	2	3	12	
BIT b,(IX+d)	$Z \leftarrow \overline{(IX+d)}_b$	•	‡	X	X	0	1	11 011 101 11 001 011 ← d → 01 b 110	4	5	20	
BIT b,(IY+d)	$Z \leftarrow \overline{(IY+d)}_b$	•	‡	X	X	0	1	11 111 101 11 001 011 ← d → 01 b 110	4	5	20	
SET b,r	$r_b \leftarrow 1$	•	•	•	•	•	•	11 001 011 11 b r	2	2	8	
SET b,(HL)	$(HL)_b \leftarrow 1$	•	•	•	•	•	•	11 001 011 11 b 110	2	4	15	
SET b,(IX+d)	$(IX+d)_b \leftarrow 1$	•	•	•	•	•	•	11 011 101 11 001 011 ← d → 11 b 110	4	6	23	
SET b,(IY+d)	$(IY+d)_b \leftarrow 1$	•	•	•	•	•	•	11 111 101 11 001 011 ← d → 11 b 110	4	6	23	
RES b,s	$s_b \leftarrow 0$ s ≡ r,(HL), (IX+d), (IY+d)							10				Reset bit b of the operand s.

r	Register
000	B
001	C
010	D
011	E
100	H
101	L
111	A

b	Test bit
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

(11) Jump instructions

Mnemonic code	Operation	Flags C	Z	P/V	S	N	H	OP code 76 543 210	No. of bytes	No. of M cycles	No. of T states	Remarks
JP nn	PC←nn	●	●	●	●	●	●	11 000 011 ← n → ← n →	3	3	10	
JP cc, nn	If cc is true, then PC←nn. If cc is false, move to the next instruction.	●	●	●	●	●	●	11 cc 010 ← n → ← n →	3	3	10	cc: 000 NZ non zero; 001 Z zero; 010 NC non carry; 011 C carry; 100 PO parity odd; 101 PE parity even; 110 P sign positive; 111 M sign negative
JR e	PC←PC+e	●	●	●	●	●	●	00 011 000 ← e-2 →	2	3	12	
JR C, e	If C=0, then move to the next instruction.	●	●	●	●	●	●	00 111 000 ← e-2 →	2	2	7	
	If C=1, then PC←PC+e								2	3	12	
JR NC, e	If C=1, then move to the next instruction.	●	●	●	●	●	●	00 110 000 ← e-2 →	2	2	7	
	If C=0, then PC←PC+e								2	3	12	
JR Z, e	If Z=0, then move to the next instruction.	●	●	●	●	●	●	00 101 000 ← e-2 →	2	2	7	
	If Z=1, then PC←PC+e								2	3	12	
JR NZ, e	If Z=1, then move to the next instruction.	●	●	●	●	●	●	00 100 000 ← e-2 →	2	2	7	
	If Z=0, then PC←PC+e								2	3	12	
JP (HL)	PC←HL	●	●	●	●	●	●	11 101 001	1	1	4	
JP (IX)	PC←IX	●	●	●	●	●	●	11 011 101 11 101 001	2	2	8	
JP (IY)	PC←IY	●	●	●	●	●	●	11 111 101 11 101 001	2	2	8	
DJNZ, e	B←B−1 If B=0, then move to the next instruction.	●	●	●	●	●	●	00 010 000 ← e-2 →	2	2	8	When B=0
	If B≠0, then PC←PC+e								2	3	13	When B≠0

Note The allowable range of displacement e is from −126 to +129. The OP code must be given by a binary number equivalent to e-2.

(12) Call and return instructions

Mnemonic code	Operation	Flags						OP code	No. of bytes	No. of M cycles	No. of T states	Remarks
		C	Z	P/V	S	N	H	76 543 210				
CALL nn	$(SP-1)\leftarrow PC_H$ $(SP-2)\leftarrow PC_L$ $PC\leftarrow nn$	•	•	•	•	•	•	11 001 101 ← n → ← n →	3	5	17	
CALL cc, nn	If cc is true, the operation is the same as CALL nn. If cc is false, then move to the next instruction.	•	•	•	•	•	•	11 cc 100 ← n → ← n →	3 3	3 5	10 17	When cc is false When cc is true
RET	$PC_L\leftarrow(SP)$ $PC_H\leftarrow(SP+1)$	•	•	•	•	•	•	11 001 001	1	3	10	
RET cc	If cc is true, the operation is the same as RET. If cc is false, then move to the next instruction.	•	•	•	•	•	•	11 cc 000	1 1	1 3	5 11	When cc is false When cc is true
RETI	Return from the interrupt.	•	•	•	•	•	•	11 101 101 01 001 101	2	4	14	
RETN	Return from タ－ NMI.	•	•	•	•	•	•	11 101 101 01 000 101	2	4	14	
RST p	$(SP-1)\leftarrow PC_H$ $(SP-2)\leftarrow PC_L$ $PC_H\leftarrow 0$ $PC_L\leftarrow P$	•	•	•	•	•	•	11 t 111	1	3	11	

cc	Conditions
000	NZ non zero
001	Z zero
010	NC non carry
011	C carry
100	PO parity odd
101	PE parity even
110	P sign positive
111	M sign negative

t	P
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

(13) Input and output instructions

Mnemonic code	Operation	Flags						OP code	No. of bytes	No. of M cycles	No. of T states	Remarks
		C	Z	P/V	S	N	H	76 543 210				
IN A,(n)	A←(n)	•	•	•	•	•	•	11 011 011 ← n →	2	3	11	n to A_0~A_7 Acc to A_8~A_{15}
IN r,(C)	r←(C) If r=110, only the flags are affected.	•	↕ ①	P	↕	0	0	11 101 101 01 r 000	2	3	12	C to A_0~A_7 B to A_8~A_{15}
INI	(HL)←(C) B←B−1 HL←HL+1	X	↕	X	X	1	X	11 101 101 10 100 010	2	4	16	C to A_0~A_7 B to A_8~A_{15}
INIR	(HL)←(C) B←B−1 HL←HL+1 Repeat until B=0	X	1 ①	X	X	1	X	11 101 101 10 110 010	2 2	5 (When B≠0) 4 (When B=0)	21 16	C to A_0~A_7 B to A_8~A_{15}
IND	(HL)←(C) B←B−1 HL←HL−1	X	↕	X	X	1	X	11 101 101 10 101 010	2	4	16	C to A_0~A_7 B to A_8~A_{15}
INDR	(HL)←(C) B←B−1 HL←HL−1 Repeat until B=0	X	1	X	X	1	X	11 101 101 10 111 010	2 2	5 (When B≠0) 4 (When B=0)	21 16	C to A_0~A_7 B to A_8~A_{15}
OUT (n),A	(n)←A	•	•	•	•	•	•	11 010 011 ← n →	2	3	11	n to A_0~A_7 Acc to A_8~A_{15}
OUT (C),r	(C)←r	•	•	•	•	•	•	11 101 101 01 r 001	2	3	12	C to A_0~A_7 B to A_8~A_{15}
OUTI	(C)←(HL) B←B−1 HL←HL+1	X	↕	X	X	1	X	11 101 101 10 100 011	2	4	16	C to A_0~A_7 B to A_8~A_{15}
OTIR	(C)←(HL) B←B−1 HL←HL+1 Repeat until B=0	X	1 ①	X	X	1	X	11 101 101 10 110 011	2 2	5 (When B≠0) 4 (When B=0)	21 16	C to A_0~A_7 B to A_8~A_{15}
OUTD	(C)←(HL) B←B−1 HL←HL−1	X	↕	X	X	1	X	11 101 101 10 101 011	2	4	16	C to A_0~A_7 B to A_8~A_{15}
OTDR	(C)←(HL) B←B−1 HL←HL−1 Repeat until B=0	X	1	X	X	1	X	11 101 101 10 111 011	2 2	5 (When B≠0) 4 (When B=0)	21 16	C to A_0~A_7 B to A_8~A_{15}

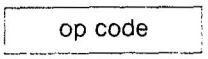
Note The Z flag indicated by ① in the table is set to "1" if the result is BC−1=0, otherwise the flag is set to "0".

10.5 MNEMONIC CODES OF LH-5803

List of LH5801 Microprocessor will be shown in pages to follow. There are following nine types of commands.

Single byte command

(1)



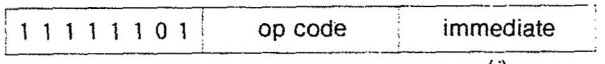
Two-byte command

(2)

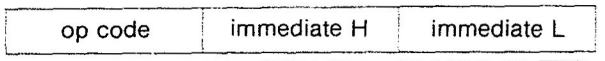
(3)

Three-byte command

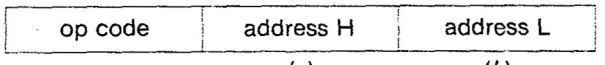
(4)



(5)



(6)



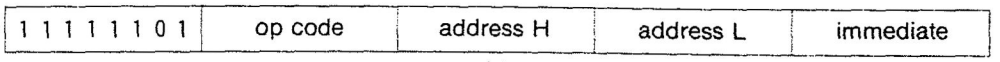
Four-byte command

(7)

(8)

Five-byte command

(9)



8-bit CPU command list (1)

Arithmetic/logical

MNEMONIC		SYMBOLIC OPERATION	STATUS	MACHINE LANGUAGE		BYTE	CYCLE	COMMENT
			C V H Z I E	7 6 5 4 3 2 1 0				
ADC	R_L	$A+R_L+C\rightarrow A$	○○○○−	0 0 R_L 0 0 1 0		1	6	● B5 B4 | R_L R_H R 0 0 | X_L X_H X 0 1 | Y_L Y_H Y 1 0 | U_L U_H U 1 1 | * * *
	R_H	$A+R_H+C\rightarrow A$		1 0 R_H 0 0 1 0		1	6	
	(R)	$A+(R)+C\rightarrow A$		0 0 R 0 0 1 1		1	7	
	(a,b)	$A+(a,b)+C\rightarrow A$		1 0 1 0 0 0 1 1		3	13	
	#(R)	$A+\#(R)+C\rightarrow A$		FD 0 0 R 0 0 1 1		2	11	
	$\#(a,b)$	$A+\#(a,b)+C\rightarrow A$		FD 1 0 1 0 0 0 1 1		4	17	● Address of (a,b) 15 ····· 87 ····· 0 a | b (High order) (Low order)
ADI	A,i	$A+i+C\rightarrow A$		1 0 1 1 0 0 1 1		2	7	
	(R),i	$(R)+i\rightarrow(R)$		0 1 R 1 1 1 1		2	13	
	$(a,b),i$	$(a,b)+i\rightarrow(a,b)$		1 1 1 0 1 1 1 1		4	19	● (R)···ME0 accessed #(R)···ME1 accessed
	#(R),i	$\#(R)+i\rightarrow\#(R)$		FD 0 1 R 1 1 1 1		3	17	
	$\#(a,b),i$	$\#(a,b)+i\rightarrow\#(a,b)$		FD 1 1 1 0 1 1 1 1		5	23	
DCA	(R)	$A+(R)+C\rightarrow A$(BCD)		1 0 R 1 1 0 0		1	15	
	#(R)	$A+\#(R)+C\rightarrow A$(BCD)		FD 1 0 R 1 1 0 0		2	19	
ADR		$R_L+A\rightarrow R_L$ (16-bit register operation) RH+1→RH · *if* C7		FD 1 1 R 1 0 1 0		2	11	
SBC	R_L	$A-R_L-\overline{C}\rightarrow A$	○○○○−	0 0 R_L 0 0 0 0		1	6	
	R_H	$A-R_H-\overline{C}\rightarrow A$		1 0 R_H 0 0 0 0		1	6	
	(R)	$A-(R)-\overline{C}\rightarrow A$		0 0 R 0 0 0 1		1	7	
	(a,b)	$A-(a,b)-\overline{C}\rightarrow A$		1 0 1 0 0 0 0 1		3	13	
	#(R)	$A-\#(R)-\overline{C}\rightarrow A$		FD 0 0 R 0 0 0 1		2	11	
	$\#(a,b)$	$A-\#(a,b)-\overline{C}\rightarrow A$		FD 1 0 1 0 0 0 0 1		4	17	
SBI	A,i	$A-i-\overline{C}\rightarrow A$		1 0 1 1 0 0 0 1		2	7	
DCS	(R)	$A-(R)-\overline{C}\rightarrow A$(BCD)		0 0 R 1 1 0 0		1	13	
	#(R)	$A-\#(R)-\overline{C}\rightarrow A$(BCD)		FD 0 0 R 1 1 0 0		2	17	
AND	(R)	$A\wedge(R)\rightarrow A$	−−−○−	0 0 R 1 0 0 1		1	7	
	(a,b)	$A\wedge(a,b)\rightarrow A$		1 0 1 0 1 0 0 1		3	13	
	#(R)	$A\wedge\#(R)\rightarrow A$		FD 0 0 R 1 0 0 1		2	11	
	$\#(a,b)$	$A\wedge\#(a,b)\rightarrow A$		FD 1 0 1 0 1 0 0 1		4	17	
ANI	A,i	$A\wedge i\rightarrow A$		1 0 1 1 1 0 0 1		2	7	
	(R),i	$(R)\wedge i\rightarrow(R)$		0 1 R 1 0 0 1		2	13	
	$(a,b),i$	$(a,b)\wedge i\rightarrow(a,b)$		1 1 1 0 1 0 0 1		4	19	
	#(R),i	$\#(R)\wedge i\rightarrow\#(R)$		FD 0 1 R 1 0 0 1		3	17	
	$\#(a,b),i$	$\#(a,b)\wedge i\rightarrow\#(a,b)$		FD 1 1 1 0 1 0 0 1		5	23	

8-bit CPU command list (2)

MNEMONIC		SYMBOLIC OPERATION	STATUS	MACHINE LANGUAGE		BYTE	CYCLE	COMMENT
			C V H Z IE	7 6 5 4 3 2 1 0				
ORA	(R)	A∨(R)→A	---○-	0 0 R 1 0 1 1		1	7	
	(*a*,*b*)	A∨(*a*,*b*)→A		1 0 1 0 1 0 1 1 FD		3	13	
	#(R)	A∨#(R)→A		0 0 R 1 0 1 1 FD		2	11	
	#(*a*,*b*)	A∨#(*a*,*b*)→A		1 0 1 0 1 0 1 1		4	17	
ORI	A,*i*	A∨*i*→A		1 0 1 1 1 0 1 1		2	7	
	(R),*i*	(R)∨*i*→(R)		0 1 R 1 0 1 1		2	13	
	(*a*,*b*),*i*	(*a*,*b*)∨*i*→(*a*,*b*)		1 1 1 0 1 0 1 1 FD		4	19	
	#(R),*i*	#(R)∨*i*→#(R)		0 1 R 1 0 1 1 FD		3	17	
	#(*a*,*b*),*i*	#(*a*,*b*)∨*i*→#(*a*,*b*)		1 1 1 0 1 0 1 1		5	23	
EOR	(R)	A⊕(R)→A	---○-	0 0 R 1 1 0 1		1	7	
	(*a*,*b*)	A⊕(*a*,*b*)→A		1 0 1 0 1 1 0 1 FD		3	13	
	#(R)	A⊕#(R)→A		0 0 R 1 1 0 1 FD		2	11	
	#(*a*,*b*)	A⊕#(*a*,*b*)→A		1 0 1 0 1 1 0 1		4	17	
EAI	*i*	A⊕*i*→A		1 0 1 1 1 1 0 1		2	7	
INC	A	A+1→A	○○○○-	1 1 0 1 1 1 0 1		1	5	
	RL	RL+1→RL		0 1 RL 0 0 0 0 FD		1	5	
	RH	RH+1→RH		0 1 RH 0 0 0 0		2	9	
	R	R+1→R	-----	0 1 R 0 1 0 0		1	5	
DEC	A	A−1→A	○○○○-	1 1 0 1 1 1 1 1		1	5	
	RL	RL−1→RL		0 1 RL 0 0 1 0 FD		1	5	
	RH	RH−1→RH		0 1 RH 0 0 1 0		2	9	
	R	R−1→R	-----	0 1 R 0 1 1 0		1	5	

Compare and bit test

CPA	RL	A−RL	○○○○-	0 0 RL 0 1 1 0		1	6	
	RH	A−RH		1 0 RH 0 1 1 0		1	6	
	(R)	A−(R)		0 0 R 0 1 1 1		1	7	
	(*a*,*b*)	A−(*a*,*b*)		1 0 1 0 0 1 1 1 FD		3	13	
	#(R)	A−#(R)		0 0 R 0 1 1 1 FD		2	11	
	#(*a*,*b*)	A−#(*a*,*b*)		1 0 1 0 0 1 1 1		4	17	
CPI	RL,*i*	RL−*i*		0 1 RL 1 1 1 0		2	7	
	RH,*i*	RH−*i*		0 1 RH 1 1 0 0		2	7	
	A,*i*	A−*i*		1 0 1 1 0 1 1 1		2	7	
BIT	(R)	A∧(R)→Z	---○-	0 0 R 1 1 1 1		1	7	
	(*a*,*b*)	A∧(*a*,*b*)→Z		1 0 1 0 1 1 1 1 FD		3	13	
	#(R)	A∧#(R)→Z		0 0 R 1 1 1 1 FD		2	11	
	#(*a*,*b*)	A∧#(*a*,*b*)→Z		1 0 1 0 1 1 1 1		4	17	
BII	A,*i*	A∧*i*→Z		1 0 1 1 1 1 1 1		2	7	
	(R),*i*	(R)∧*i*→Z		0 1 R 1 1 0 1		2	10	
	(*a*,*b*),*i*	(*a*,*b*)∧*i*→Z		1 1 1 0 1 1 0 1 FD		4	16	
	#(R),*i*	#(R)∧*i*→Z		0 1 R 1 1 0 1 FD		3	14	
	#(*a*,*b*),*i*	#(*a*,*b*)∧*i*→Z		1 1 1 0 1 1 0 1		5	20	

8-bit CPU command list (3)

Load and store

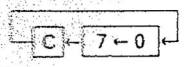
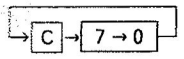
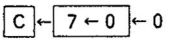
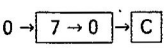
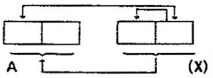
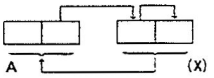
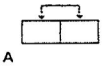
MNEMONIC	SYMBOLIC OPERATION	STATUS	MACHINE LANGUAGE	BYTE	CYTLE	COMMENT
		C V H Z IE	7 6 5 4 3 2 1 0			
LDA R_L	$R_L \rightarrow A$	---○-	00 R_L 0100	1	5	
R_H	$R_H \rightarrow A$		10 R_H 0100	1	5	
(R)	*(R)→A*		*00 R 0101*	*1*	*6*	
(*a,b*)	(*a,b*)→A		10100101	3	12	
#(R)	#(R)→A		00 R (FD) 0101	2	10	
#(*a,b*)	#(*a,b*)→A		10100101 (FD)	4	16	
LDE R	(R)→A,R−1→R		01 R 0111	1	6	
LIN R	(R)→A,R+1→R		01 R 0101	1	6	
LDI R_L,*i*	*i*→R_L	-----	01 R_L 1010	2	6	
R_H,*i*	*i*→R_H		01 R_H 1000	2	6	
A,*i*	*i*→A	---○-	10110101	2	6	
S,*i,j*	*ij*→S	-----	10101010	3	12	
LDX R	R→X		00 R (FD) 1000	2	11	
S	S→X		01001000 (FD)	2	11	
P	P→X		01011000 (FD)	2	11	
STA R_L	A→R_L	-----	00 R_L 1010	1	5	
R_H	A→R_H		00 R_H 1000	1	5	
(R)	A→(R)		00 R 1110	1	6	
(*a,b*)	A→(*a,b*)		10101110	3	12	
#(R)	A→#(R)		00 R (FD) 1110	2	10	
#(*a,b*)	A→#(*a,b*)		10101110 (FD)	4	16	
SDE R	A→(R),R−1→R		01 R 0011	1	6	
SIN R	A→(R),R+1→R		01 R 0001	1	6	
STX R	X→R		01 R (FD) 1010	2	11	
S	X→S		01001110 (FD)	2	11	
P	X→P		01011110 (FD)	2	11	
PSH A	A→(S),S−1→S		11001000 (FD)	2	11	
R	R_L→(S),RH→(S−1), S−2→S		10 R (FD) 1000	2	14	
POP A	(S+1)→A,S+1→S	---○-	10001010 (FD)	2	12	
R	(S+1)→RH,(S+2)→RL, S+2→S	-----	00 R (FD) 1010	2	15	
ATT	A→T(STATUS)	○○○○○	11101100 (FD)	2	9	
TTA	T(STATUS)→A	---○-	10101010 (FD)	2	9	

Block transfer and search

TIN	(X)→(Y), X+1→X, Y+1→Y	-----	11110101	1	7	
CIN	A→(X),X+1→X	○○○○-	11110111	1	7	

8-bit CPU command list (4)

Rotate and shift

MNEMONIC	SYMBOLIC OPERATION	STATUS	MACHINE LANGUAGE		BYTE	CYCLE	COMMENT
		C V H Z IE	7 6 5 4 3 2 1 0				
ROL		OOOO–	1 1 0 1 1 0 1 1		1	6	
ROR			1 1 0 1 0 0 0 1		1	9	
SHL			1 1 0 1 1 0 0 1		1	6	
SHR			1 1 0 1 0 1 0 1		1	9	
DRL		–––––	1 1 0 1 0 1 1 1		1	12	
DRL #	ME1 Area		FD 1 1 0 1 0 1 1 1		2	16	
DRR			1 1 0 1 0 0 1 1		1	12	
DRR #	ME1 Area		FD 1 1 0 1 0 0 1 1		2	16	
AEX			1 1 1 1 0 0 0 1		1	6	

CUP control

AM0	A→TIMER(T0~T7),0→T8	–––––	FD 1 1 0 0 1 1 1 0		2	9	
AM1	〃 1→T8		FD 1 1 0 1 1 1 1 0		2	9	
CDV	divider clear		FD 1 0 0 0 1 1 1 0		2	8	
ATP	A→Output port (Clock output)		FD 1 1 0 0 1 1 0 0		2	9	
SDP	1→Disp		FD 1 1 0 0 0 0 0 1		2	8	
RDP	0→Disp		FD 1 1 0 0 0 0 0 0		2	8	
SPU	1→PU		1 1 1 0 0 0 0 1		1	4	
RPU	0→PU		1 1 1 0 0 0 1 1		1	4	
SPV	1→PV		1 0 1 0 1 0 0 0		1	4	
RPV	0→PV		1 0 1 1 1 0 0 0		1	4	
ITA	IN→A	––– O–	FD 1 0 1 1 1 0 1 0		2	9	
RIE	0→IE	–––– O	FD 1 0 1 1 1 1 1 0		2	8	
SIE	1→IE	–––– O	FD 1 0 0 0 0 0 0 1		2	8	
HLT		–––––	FD 1 0 1 1 0 0 0 1		2	9	
OFF			FD 0 1 0 0 1 1 0 0		2	8	
NOP			0 0 1 1 1 0 0 0		1	5	
SEC	1→C	O––––	1 1 1 1 1 0 1 1		1	4	
REC	0→C	O––––	1 1 1 1 1 0 0 1		1	4	

8-bit CPU command list (5)

Jump

MNEMONIC	CYMBOLIC OPERATION	STATUS C V H Z IE	MACHINE LANGUAGE 7 6 5 4 3 2 1 0		BYTE	CYCLE	COMMENT
JMP	$ij \rightarrow P$	-----	1 0 1 1 1 0 1 0		3	12	$s=0$: $+i$
BCH	$s=0$: $P+i \rightarrow P$		1 0 0 s 1 1 1 0		2	8	$s=1$: $-i$
	$s=1$: $P-i \rightarrow P$					9	(Includes on more cycle)
BCS	*if* C=1, $P \pm i \rightarrow P$		1 0 0 s 0 0 1 1		2	8/10/11	
	if C=0. continue						
BCR	*if* C=0, $P \pm i \rightarrow P$		1 0 0 s 0 0 0 1		2	8/10/11	
	if C=1. continue						
BVS	*if* V=1, $P \pm i \rightarrow P$		1 0 0 s 1 1 1 1		2	8/10/11	
	if V=0. continue						
BVR	*if* V=0, $P \pm i \rightarrow P$		1 0 0 s 1 1 0 1		2	8/10/11	
	if V=1. continue						
BHS	*if* H=1, $P \pm i \rightarrow P$		1 0 0 s 0 1 1 1		2	8/10/11	
	if H=0. continue						
BHR	*if* H=0, $P \pm i \rightarrow P$		1 0 0 s 0 1 0 1		2	8/10/11	
	if H=1. continue						
BZS	*if* Z=1, $P \pm i \rightarrow P$		1 0 0 s 1 0 1 1		2	8/10/11	
	if Z=0. continue						
BZR	*if* Z=0, $P \pm i \rightarrow P$		1 0 0 s 1 0 0 1		2	8/10/11	
	if Z=1. continue						
LOP UL,i	UL−1→UL		1 0 0 0 1 0 0 0		2	8/11	
	if Borrow=0, $P-i \rightarrow P$						
	if Borrow=1,continue						

B3 B2 B1	Condition
0 0 0	NC: non carry
0 0 1	C: carry
0 1 0	NH: non harf
0 1 1	H: half
1 0 0	NZ: non zero
1 0 1	NV: zero
1 1 0	NV: non overflow
1 1 1	V: overflow

Call

SJP	$P_L \rightarrow (S), P_H \rightarrow (S-1)$,	-----	1 0 1 1 1 1 1 0		3	19	• Vector address (q)
	S−2→S, $ij \rightarrow P$						VEJ: FF→q_H
VEJ	$P_L \rightarrow (S), PH \rightarrow (S-1)$	---○-	1 1 ← i → 0		1	17	11i0→q_L
	$S-2 \rightarrow S, (q) \rightarrow P_H, (q+1) \rightarrow P_L$						VMJ: FF→q_H
VCS	*if* C=1,(q)→P_H→(S−1)		1 1 0 0 0 0 1 1		2	8/21	etc.
	(q+1)→P_L→(S),S−2→S						$i \rightarrow q_L$
VCR	*if* C=0, 〃		1 1 0 0 0 0 0 1		2	8/21	
VHS	*if* H=1, 〃		1 1 0 0 0 1 1 1		2	8/21	
VHR	*if* H=0, 〃		1 1 0 0 0 1 0 1		2	8/21	
VZS	*if* Z=1, 〃		1 1 0 0 1 0 1 1		2	8/21	
VZR	*if* Z=0, 〃		1 1 0 0 1 0 0 1		2	8/21	
VVS	*if* V=1, 〃		1 1 0 0 1 1 1 1		2	8/21	
VMJ	(q)→P_H→(S−1), S−2→S		1 1 0 0 1 1 0 1		2	20	
	(q+1)→P_L→(S)						

Return

RTN	(S+1)→PH,(S+2)→PL,	-----	1 0 0 1 1 0 1 0		1	11	
	S+2→S						
RTI	(S+1)→PH,(S+2)→PL	○○○○○	1 0 0 0 1 0 1 0		1	14	
	(S+3)→T,S+3→S						

NOTE: P in above list indicate a succeeding byte. For a command accompanying the immediate value, it indicates the byte that follows to the immediate value.

LH5801

MNEMONIC		MACHINE LANGUAGE	MNEMONIC		MACHINE LANGUAGE	MNEMONIC		MACHINE LANGUAGE
ADC	XL	02	ANI	(*ab*)	E9 *a b i*	BVR	−	9D *i*
	YL	12		#(X)	FD 49 *i*	BZS	+	8B *i*
	UL	22		#(Y)	FD 59 *i*		−	9B *i*
	XH	82		#(U)	FD 69 *i*	BZR	+	89 *i*
	YH	92		#(*ab*)	FD E9 *a b i*		−	99 *i*
	UH	A2	AM0		FD CE	CDV		FD 8E
	(X)	03	AM1		FD DE	CIN		F7
	(Y)	13	ATP		FD CC	CPA	XL	06
	(U)	23	ATT		FD EC		YL	16
	(*ab*)	A3 *a b*	BCH	+	8E *i*		UL	26
	#(X)	FD 03		−	9E *i*		XH	86
	#(Y)	FD 13	BCS	+	83 *i*		YH	96
	#(U)	FD 23		−	93 *i*		UH	A6
	#(*ab*)	FD A3 *a b*	BCR	+	81 *i*		(X)	07
ADI	A	B3 *i*		−	91 *i*		(Y)	17
	(X)	4F *i*	BHS	+	87 *i*		(U)	27
	(Y)	5F *i*		−	97 *i*		(*ab*)	A7 *a b*
	(U)	6F *i*	BHR	+	85 *i*		#(X)	FD 07
	(*ab*)	EF *a b i*		−	95 *i*		#(Y)	FD 17
	#(X)	FD 4F *i*	BII	A	BF *i*		#(U)	FD 27
	#(Y)	FD 5F *i*		(X)	4D *i*		#(*ab*)	FD A7 *a b*
	#(U)	FD 6F *i*		(Y)	5D *i*	CPI	A	B7 *i*
	#(*ab*)	FD EF *a b i*		(U)	6D *i*		XL	4E *i*
ADR	X	FD CA		(*ab*)	ED *a b i*		YL	5E *i*
	Y	FD DA		#(X)	FD 4D *i*		UL	6E *i*
	U	FD EA		#(Y)	FD 5D *i*		XH	4C *i*
AEX		F1		#(U)	FD 6D *i*		YH	5C *i*
AND	(X)	09		#(*ab*)	FD ED *a b i*		UH	6C *i*
	(Y)	19	BIT	(X)	0F	DCA	(X)	8C
	(U)	29		(Y)	1F		(Y)	9C
	(*ab*)	A9 *a b*		(U)	2F		(U)	AC
	#(X)	FD 09		(*ab*)	AF *a b*		#(X)	FD 8C
	#(Y)	FD 19		#(X)	FD 0F		#(Y)	FD 9C
	#(U)	FD 29		#(Y)	FD 1F		#(U)	FD AC
	#(*ab*)	FD A9 *a b*		#(U)	FD 2F	DCS	(X)	0C
ANI	A	B9 *i*		#(*ab*)	FD AF *a b*		(Y)	1C
	(X)	49 *i*	BVS	+	8F *i*		(U)	2C
	(Y)	59 *i*		−	9F *i*		#(X)	FD 0C
	(U)	69 *i*	BVR	+	8D *i*		#(Y)	FD 1C

APPENDICES

MNEMONIC		MACHINE LANGUAGE
DCS	#(U)	FD 2C
DEC	A	DF
	XL	42
	YL	52
	UL	62
	XH	FD 42
	YH	FD 52
	UH	FD 62
	X	46
	Y	56
	U	66
DRL	(X)	D7
	#(X)	FD D7
DRR	(X)	D3
	#(X)	FD D3
EAI		BD *i*
EOR	(X)	0D
	(Y)	1D
	(U)	2D
	(*ab*)	AD *a b*
	#(X)	FD 0D
	#(Y)	FD 1D
	#(U)	FD 2D
	#(*ab*)	FD AD *a b*
HLT		FD B1
INC	A	DD
	XL	40
	YL	50
	UL	60
	XH	FD 40
	YH	FD 50
	UH	FD 60
	X	44
	Y	54
	U	64
ITA		FD BA
JMP		BA *i j*
LDA	XL	04
	YL	14

MNEMONIC		MACHINE LANGUAGE
LDA	UL	24
	XH	84
	YH	94
	UH	A4
	(X)	05
	(Y)	15
	(U)	25
	(*ab*)	A5 *a b*
	#(X)	FD 05
	#(Y)	FD 15
	#(U)	FD 25
	#(*ab*)	FD A5 *a b*
LDI	A	B5 *i*
	XL	4A *i*
	YL	5A *i*
	UL	6A *i*
	XH	48 *i*
	YH	58 *i*
	UH	68 *i*
	S	AA *i j*
LDE	X	47
	Y	57
	U	67
LDX	X	FD 08
	Y	FD 18
	U	FD 28
	S	FD 48
	P	FD 58
LIN	X	45
	Y	55
	U	65
LOP	UL	88 *i*
NOP		38
OFF		FD 4C
ORA	(X)	0B
	(Y)	1B
	(U)	2B
	(*ab*)	AB *a b*
	#(X)	FD 0B

MNEMONIC		MACHINE LANGUAGE
ORA	#(Y)	FD 1B
	#(U)	FD 2B
	#(*ab*)	FD AB *a b*
ORI	A	BB *i*
	(X)	4B *i*
	(Y)	5B *i*
	(U)	6B *i*
	(*ab*)	EB *a b i*
	#(X)	FD 4B
	#(Y)	FD 5B
	#(U)	FD 6B
	#(*ab*)	FD EB *a b i*
POP	A	FD 8A
	X	FD 0A
	Y	FD 1A
	U	FD 2A
PSH	A	FD C8
	X	FD 88
	Y	FD 98
	U	FD A8
RDP		FD C0
REC		F9
RIE		FD BE
ROL		DB
ROR		D1
RPU		E3
RPV		B8
RTI		8A
RTN		9A
SBC	XL	00
	YL	10
	UL	20
	XH	80
	YH	90
	UH	AO
	(X)	01
	(Y)	11
	(U)	21
	(*ab*)	A1 *a b*

MNEMONIC		MACHINE LANGUAGE	MNEMONIC		MACHINE LANGUAGE	MNEMONIC		MACHINE LANGUAGE
SBC	#(X)	FD 01	TTA		FD AA			
	#(Y)	FD 11	VCS		C3 *i*			
	#(U)	FD 21	VCR		C1 *i*			
	#(*ab*)	FD A1 *a b*	VEJ	C0	C0			
SBI		B1 *i*		C2	C2			
SDE	X	43		C4	C4			
	Y	53		C6	C6			
	U	63		C8	C8			
SDP		FD C1		CA	CA			
SEC		FB		CC	CC			
SHL		D9		CE	CE			
SHR		D5		DO	D0			
SIE		FD 81		D2	D2			
SIN	X	41		D4	D4			
	Y	51		D6	D6			
	U	61		D8	D8			
SJP		BE *i j*		DA	DA			
SPU		E1		DC	DC			
SPV		A8		DE	DE			
STA	XL	0A		E0	EO			
	YL	1A		E2	E2			
	UL	2A		E4	E4			
	XH	08		E6	E6			
	YH	18		E8	E8			
	UH	28		EA	EA			
	(X)	0E		EC	EC			
	(Y)	1E		EE	EE			
	(U)	2E		FO	F0			
	(*ab*)	AE *a b*		F2	F2			
	#(X)	FD 0E		F4	F4			
	#(Y)	FD 1E		F6	F6			
	#(U)	FD 2E	VMJ		CD *i*			
	#(*ab*)	FD AE *a b*	VVS		CF *i*			
STX	X	FD 4A	VZS		CB *i*			
	Y	FD 5A	VZR		C9 *i*			
	U	FD 6A	VHR		C5 *i*			
	S	FD 4E	VHS		C7 *i*			
	P	FD 5E						
TIN		F5						