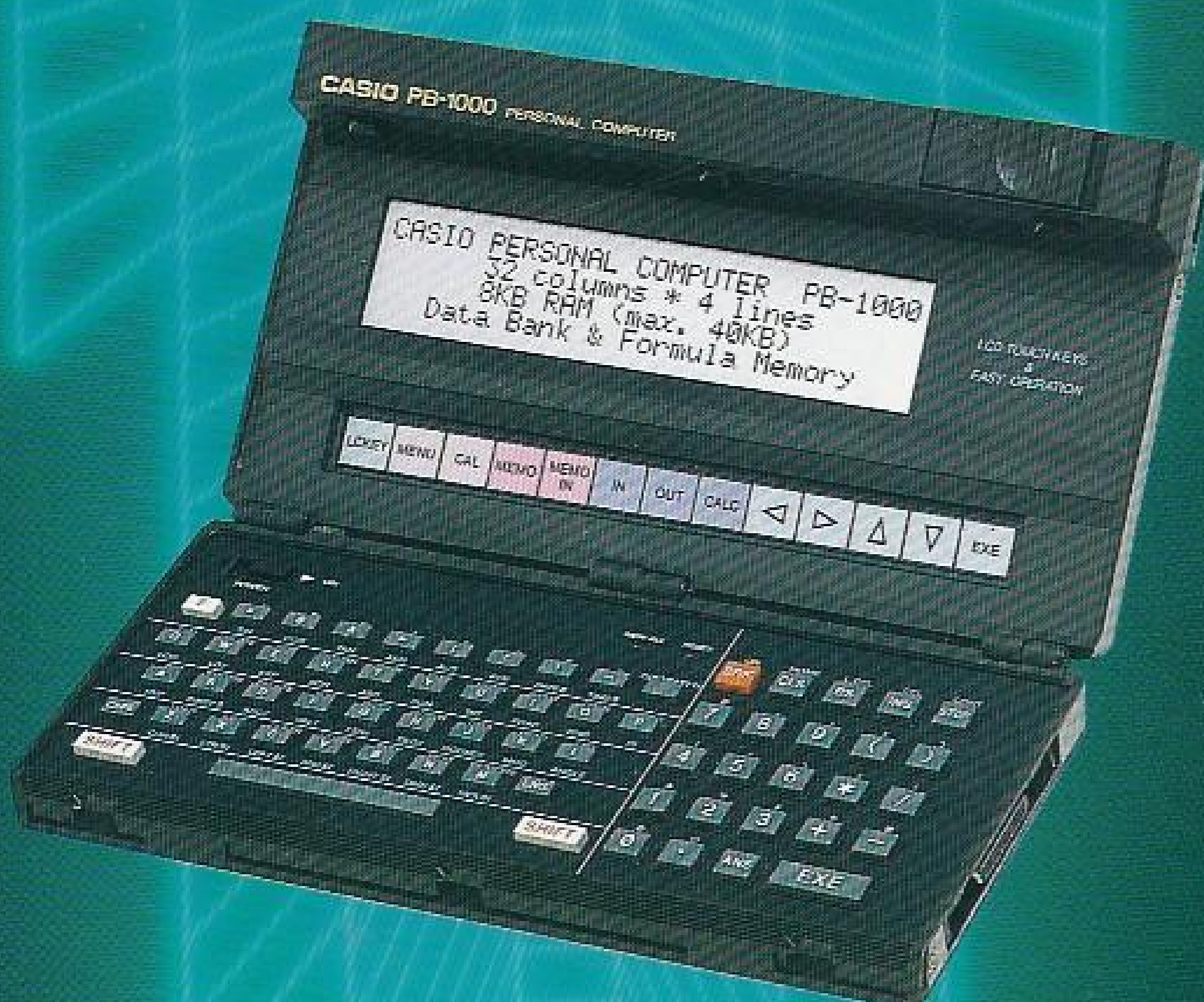
TECHNICAL REFERENCE PB-1000

CASIO PERSONAL COMPUTER



CASIO®

TECHNICAL REFERENCE

PB-1000

CASIO PERSONAL COMPUTER

The contents of this book may be subject to change without notice.
Unlawful copying of all or any portion of this book is strictly forbidden.
Please be aware that the use of this book for other than personal use without permission from CASIO is prohibited under the copyright law.
CASIO Computer Co., Ltd. shall not be held responsible for any damages or losses resulting from the use of this book.

CASIO

TECHNICAL REFERENCE

PB-1000

CASIO PERSONAL COMPUTER

The contents of this book may be subject to change without notice.
Unlawful copying of all or any portion of this book is strictly forbidden.
Please be aware that the use of this book for other than personal use without permission from CASIO is prohibited under the copyrighting law.
CASIO Computer Co., Ltd. shall not be held responsible for any damages or losses resulting from the use of this book.

CASIO®

FOREWORD

Many of the first pocket computers were nothing more than programmable calculators, with most programming limited to features which enhanced the performance of basic arithmetic functions. Advances in hardware, however, has caused the pocket computer to evolve into a miniaturized version of the popular desk top computer. Modern pocket computers have proven themselves equals to desk top computers with their BASIC language capabilities and memory capacity.

Despite successes at miniaturization and outstanding capacity, the BASIC language of pocket computers imposes limits upon their processing speed. This has led to an increasing demand from users for machine language programming capabilities for faster processing speed. Unfortunately, however, most pocket computer manufacturers do not make the specifications of the CPUs public, making machine language programming impossible.

With the CASIO PB-1000, both the CPU specifications and memory map are made public to provide users with direct access into the inner operation of the unit. The PB-1000 also features a built-in assembler, eliminating the need to create a simulator on a desk top computer for hand assembly. A 4KB machine language work area (with expanded RAM) provides enough area for virtually any pocket computer machine language application.

This manual has been written as a comprehensive guide of these and the many other functions of the PB-1000 to allow full application of its outstanding potential. Besides explanations of the CPU and memory map, a firmware entry point summary is provided, as well as actual programs which show practical applications of the entry. This manual puts the technological data of the PB-1000 into the hands of the user, in addition to the ability to develop customized programs which meet specific requirements.

Finally, this book has also been written to allow a more intimate knowledge of the inner workings of the pocket computer for users at virtually every level.

CONTENTS

CHAPTER 1

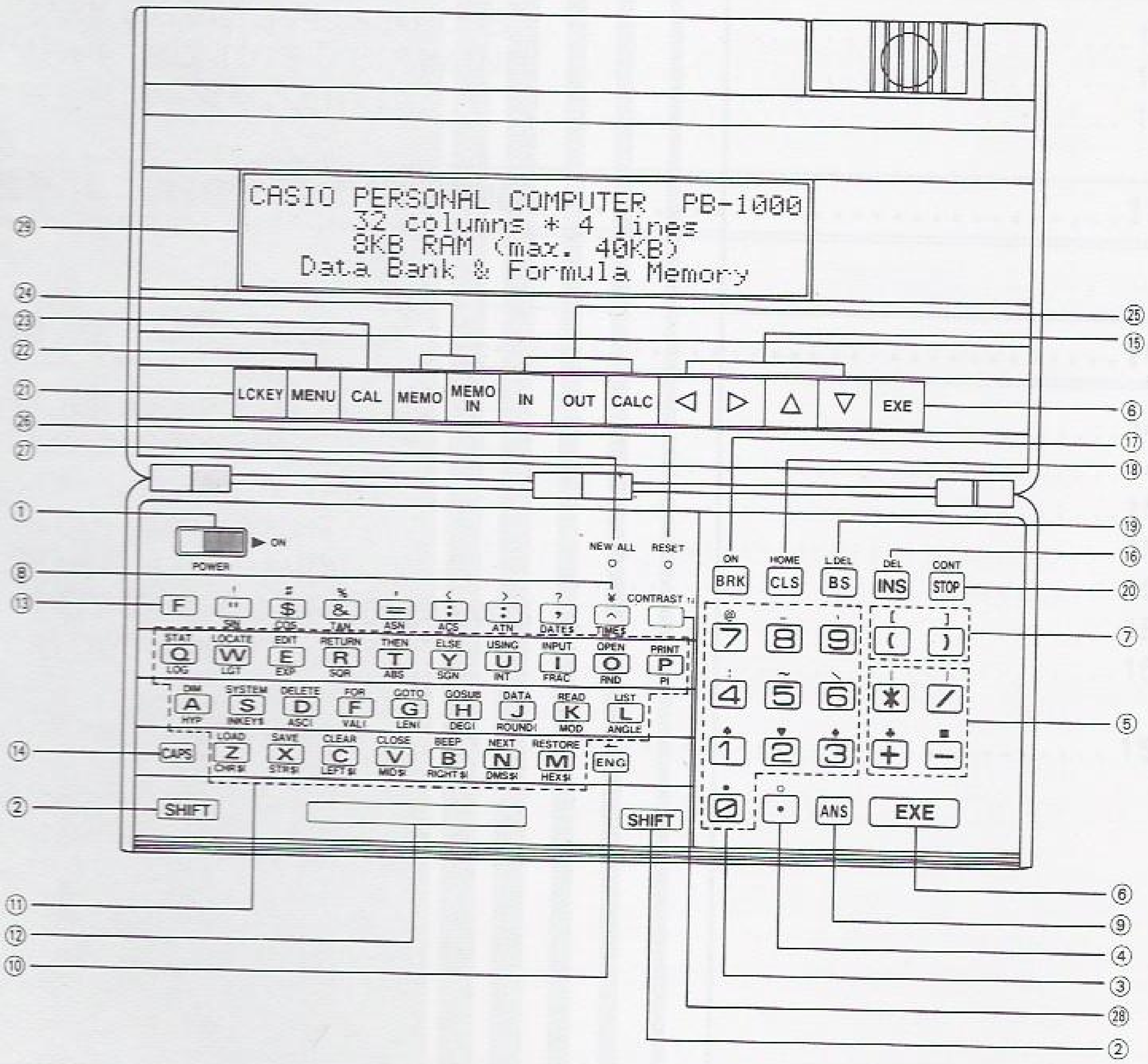
SYSTEM INTRODUCTION

1-1 UNIT SPECIFICATIONS

The PB-1000 changes the image usually associated with the pocket computer. Traditionally, the pocket computer carries a strong image of a BASIC device plus calculator. The PB-1000, on the other hand, adds a variety of other aspects to the functions of BASIC. One representative example is the touch key function which displays keys on the screen to execute functions at a touch. The touch keys team up with the sheet keys below the screen to make the following functions instantaneously available for simplified, convenient operation:

- Calculator function
- Memo function
- Formula storage function

1-1-1 GENERAL GUIDE



① Power Switch
② Shift Key
③ Numeric keys
④ Decimal Key
⑤ Arithmetic Operator Keys
⑥ Execute Key
⑦ Parentheses Keys
⑧ Power Key
⑨ Answer Key
⑩ Engineering Key
⑪ Alphabet Keys
⑫ Space Bar
⑬ Function Key
⑭ CAPS Key
⑮ Cursor Keys
⑯ Insert/Delete Key
⑰ Break Key
⑱ Clear Screen Key
⑲ Backspace Key
⑳ Stop Key
㉑ LC Display Key
㉒ Menu Key
㉓ CAL Mode Key
㉔ Memo Key/Memo In Key
㉕ Formula Storage Keys
㉖ Reset Button
㉗ New ALL Button
㉘ Contrast Key
㉙ Screen

1-1-2 SPECIFICATIONS

Model :

PB-1000

Basic calculation functions :

Negative numbers, exponents, parenthetical arithmetic operations (with priority sequence judgment function – true algebraic logic), integer division, integer division remainders, logical operators

Built-in functions :

Trigonometric/inverse trigonometric functions (angular units: degrees, radians, grads), logarithmic/exponential functions, square roots, powers, hyperbolic/inverse hyperbolic functions, conversion to integer, deletion of integer portion, absolute values, signs, rounding, random numbers, pi, decimal-sexagesimal conversions, decimal-hexadecimal conversion

Statistical calculation functions :

Standard deviation – number of data, sum of *x*, sum of squares, standard deviation (two types)

Linear regression – number of data, sum of *x*, sum of *y*, sum of squares of *x*, sum of squares of *y*, sum of products of data, standard deviation of *x* (two types), standard deviation of *y* (2 types), constant term, regression coefficient, correlation coefficient, estimated value of *x*, estimated value of *y*

Function calculation accuracy :

10th digit ±1 of the mantissa

Commands :

CLEAR, VARLIST, END, GOTO, ON GOSUB, REM ('), PRINT, ANGLE, INPUT, CLS, ON ERROR GOTO, DEFCHR$, NEW, POKE, SYSTEM, EDIT, TRON/TROFF, GOSUB/RETURN, IF/THEN/ELSE, LET, DRAW/DRAWC, LOCATE, DIM, MON, RESUME, PASS, STAT, STAT CLEAR, LIST, RUN, STOP, ON GOTO, FOR/NEXT, DATA/READ/RESTORE, PRINT USING, BEEP (ON/OFF), ERASE, CALL, DELETE, LLIST, FORMAT, OPEN, INPUT#, PUT/GET, VERIFY, LINE INPUT#, LPRINT, BSAVE, CLOSE, SAVE, FIELD, CHAIN, PRINT# USING, LPRINT USING, BLOAD, PRINT#, LOAD, RSET/LSET, MERGE

Functions :

INPUT$, INPUT#, ERR, ERL, EOF, LOF, REV, NORM, TIME$, DATE$

Program functions :

CHR$, ASC, STR$, VAL, MID$, RIGHT$, LEFT$, LEN, HEX$, INKEY$, DEG, DMS$, POINT, PEEK, TAB, &H

Calculation precision :

±1 at 10th digit

However, errors may be cumulative for internal consecutive calculations using ^, HYP, and statistical functions, and accuracy is sometime affected.

Accuracy is lessened when the following functions approach the values noted:

sinx $|x| = 90° \times 2n$

cosx $|x| = 90° \times (2n + 1)$

tanx $|x| = 90° \times n$

Program language :

C_{61}-BASIC, HD61700 assembler

Memory capacity (user area) :
4,096 bytes (when 8KB)
36,864 bytes (when 40KB)

Number of variables :
Limited by memory capacity only

Number of stacks :

System stack	255 bytes
User stack	249 bytes
FOR ~ NEXT	Limited by memory capacity or 255 levels
GOSUB	Limited by memory capacity

Numeric display :
10-digit mantissa + 2-digit exponent

Display element :
192 × 32 dot LCD (32 × 4 characters)

Main component :
C-MOS LSI

Power supply :
AA-size batteries × 3

Power consumption :
0.14W

Battery life:
1. Continuous program execution: Approx. 55 hours
2. Continuous display of 5555555555 at 20°C (68°F): Approx. 100 hours
   3 months when unit is used 1 hour per day.

* NOTE: 1 hour includes 10 minutes of condition 1 and 50 minutes of condition 2.

Auto power OFF :
Approximately 7 minutes after last key operation.

Ambient temperature range :
0°C ~ 40°C (32°F ~ 104°F)

Dimensions :

Folded:	24(H) × 187(W) × 97(D)mm
	1''(H) × $7^3/_8$''(W) × $3^7/_8$''(D)
Unfolded :	24(H) × 187(W) × 176.5(D)mm
	1''(H) × $7^3/_8$''(W) × 7''(D)

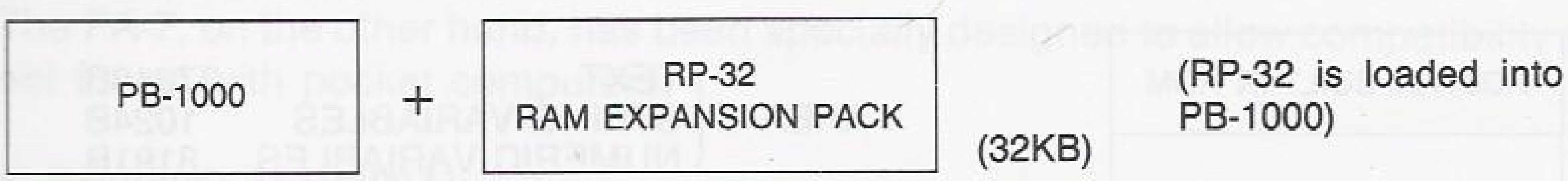
Weight :
435g (15.3 oz) including batteries

1-2 SYSTEM CONFIGURATION

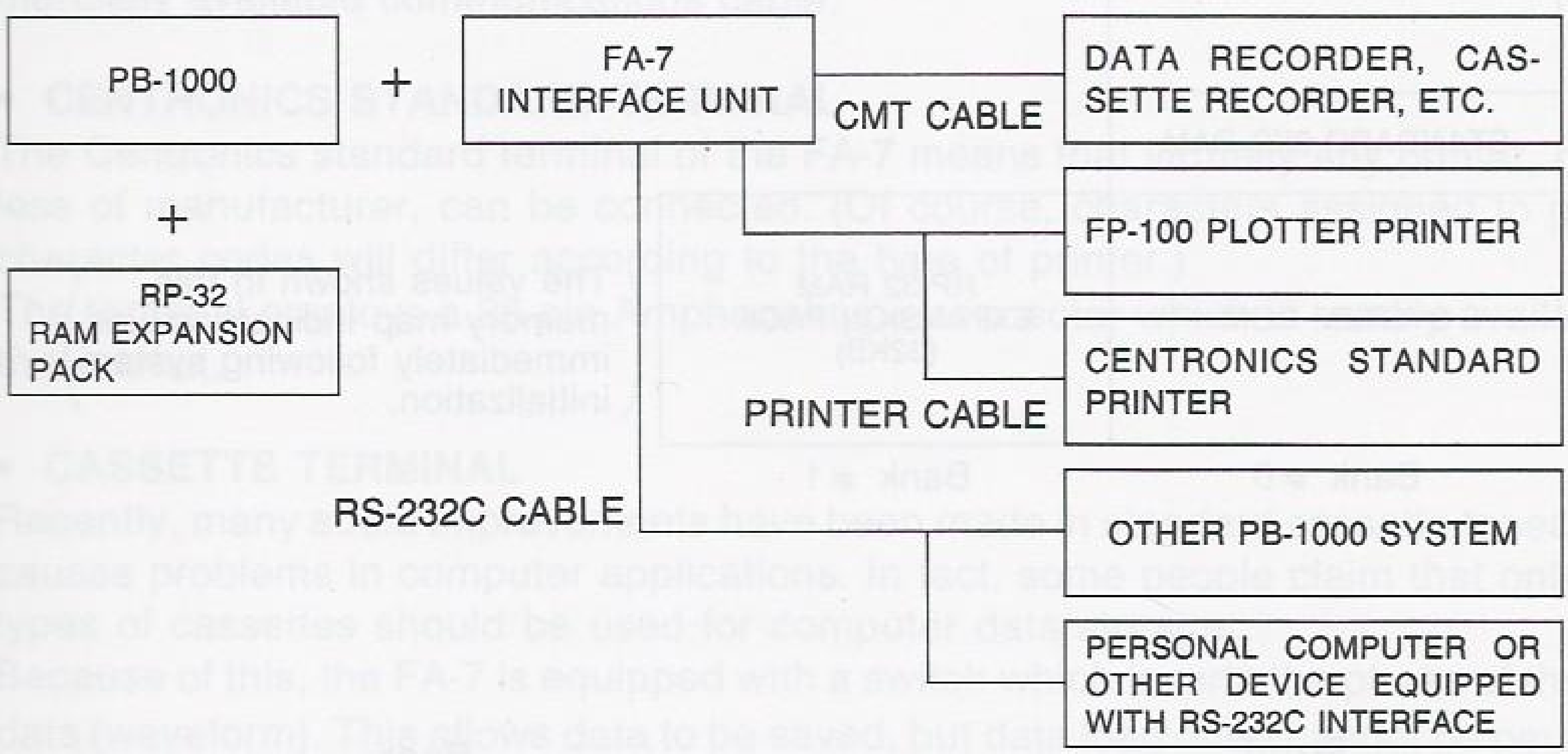
A host of peripheral devices are available to help further enhance the power of the PB-1000. Using these peripherals makes it possible to store larger volumes of programs and data, to print out programs and results, and to perform data communications.

1-2-1 SAMPLE CONFIGURATIONS

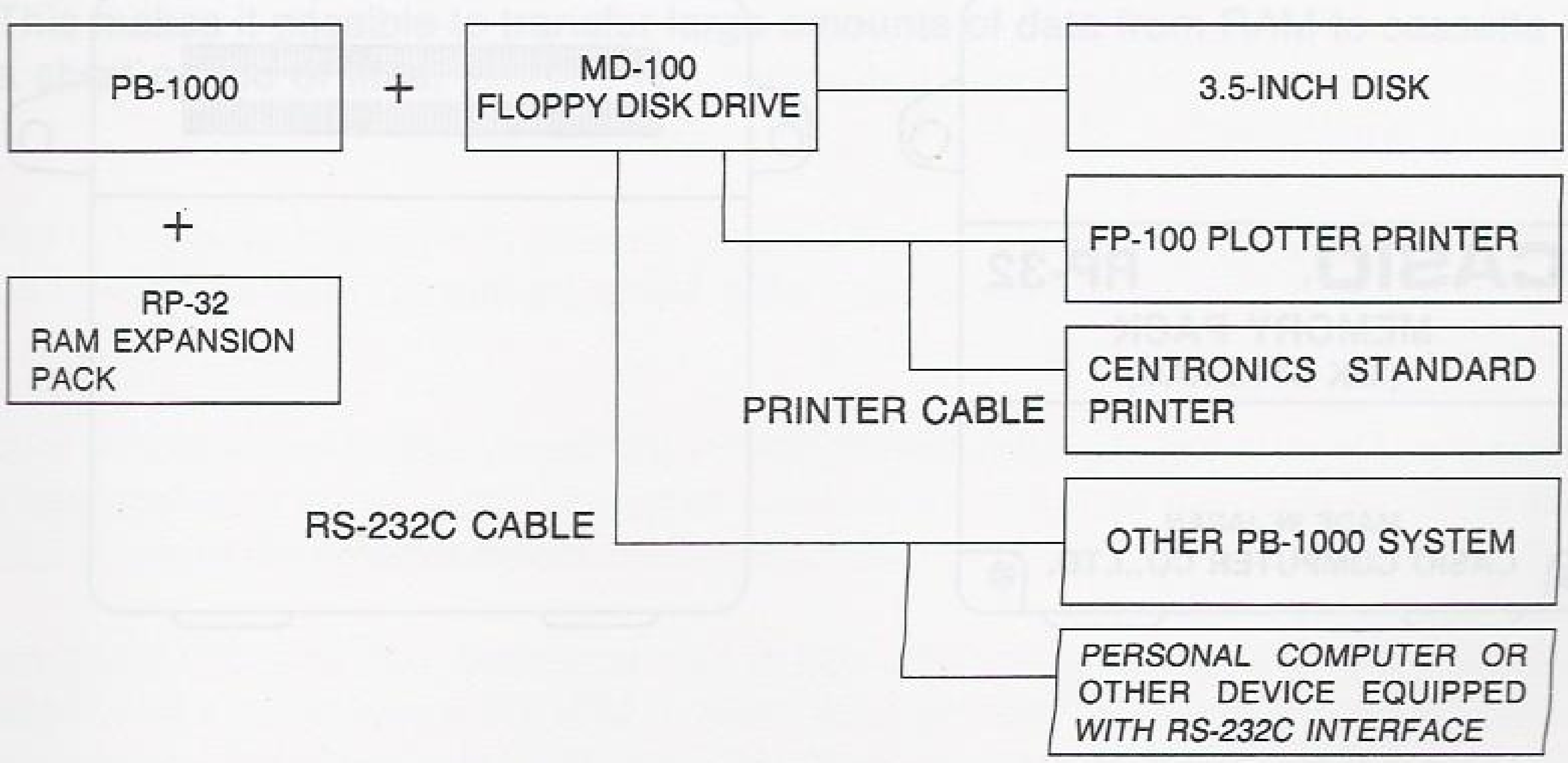
1) Memory Expansion



2) With Cassette Recorder

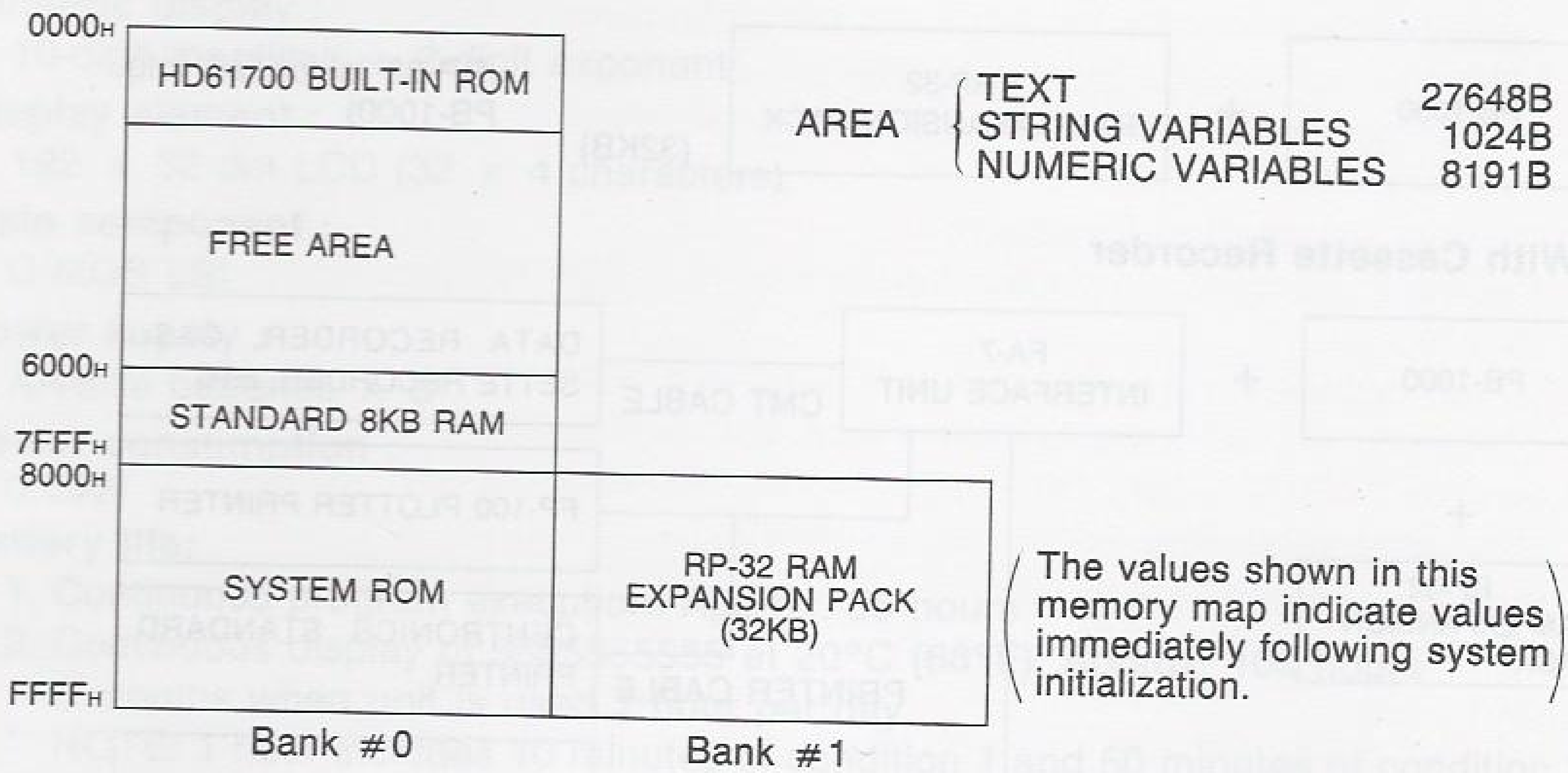


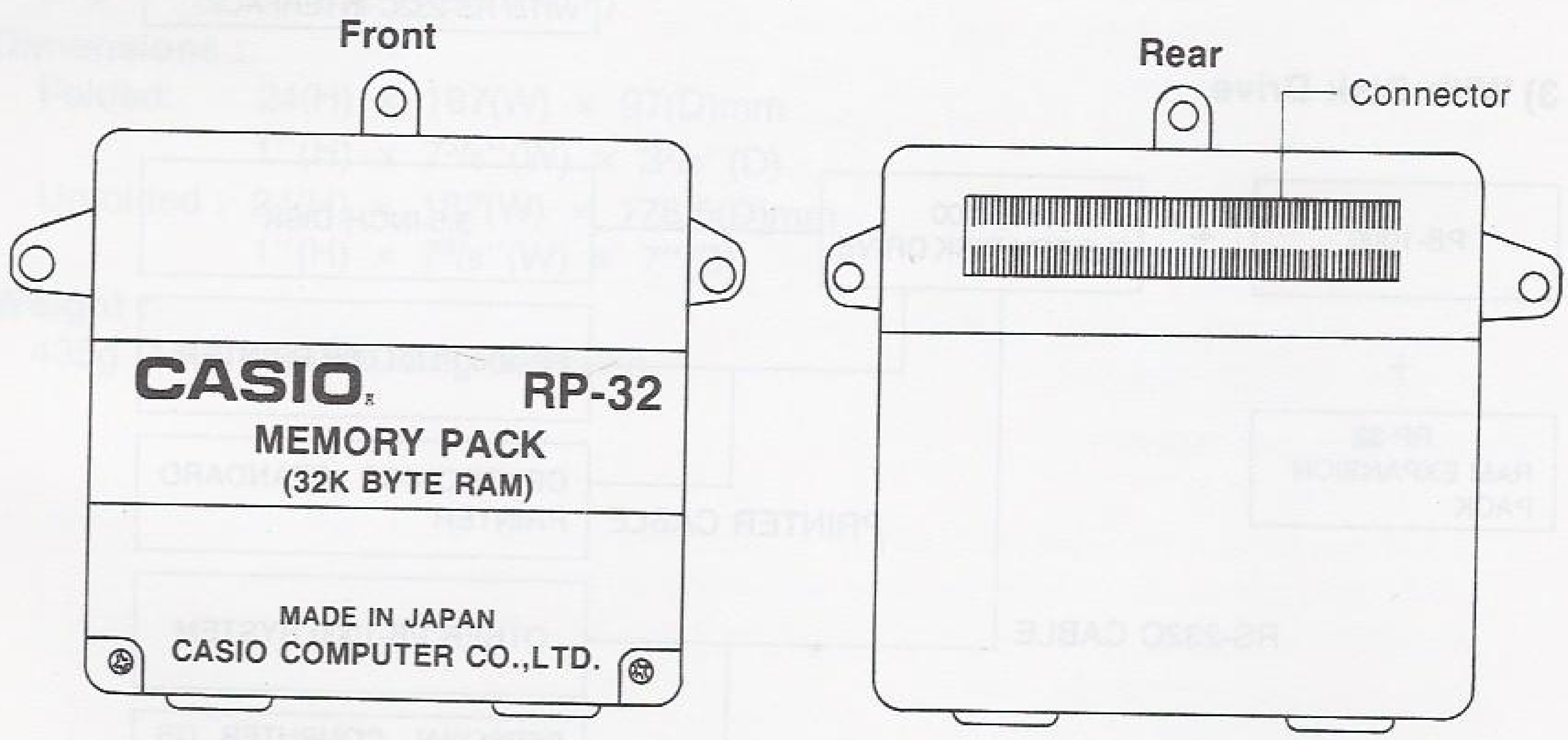
3) With Disk Drive



1-2-2 RAM EXPANSION PACK (RP-32)

The RP-32 RAM expansion pack expands the memory of the PB-1000 by 32KB, providing a total of 40KB (user area approximately 36KB). Memory expansion is especially useful when a disk drive (MD-100) has been added to the system. Assembly to machine language programs requires the entire built-in RAM of the PB-1000, so memory expansion also becomes necessary when handling long programs (especially source programs). The memory provided by the RAM pack is set in bank 1, so the memory map becomes as illustrated below:





1-2-3 INTERFACE UNIT (FA-7)

Connection of the FA-7 interface unit greatly expands the potential of the PB-1000. Pocket type computers of the past usually offered the following as peripheral devices:

- Device for connection to special printer
- Level converter for connection to communication circuit (RS-232C)
- Converter for load/save of data or programs using cassette recorder

These devices are seldom compatible with each other, and are generally devised for specific printers or recorders.
The FA-7, on the other hand, has been specially designed to allow compatibility usually not found with pocket computers.

- **RS-232C TERMINAL**

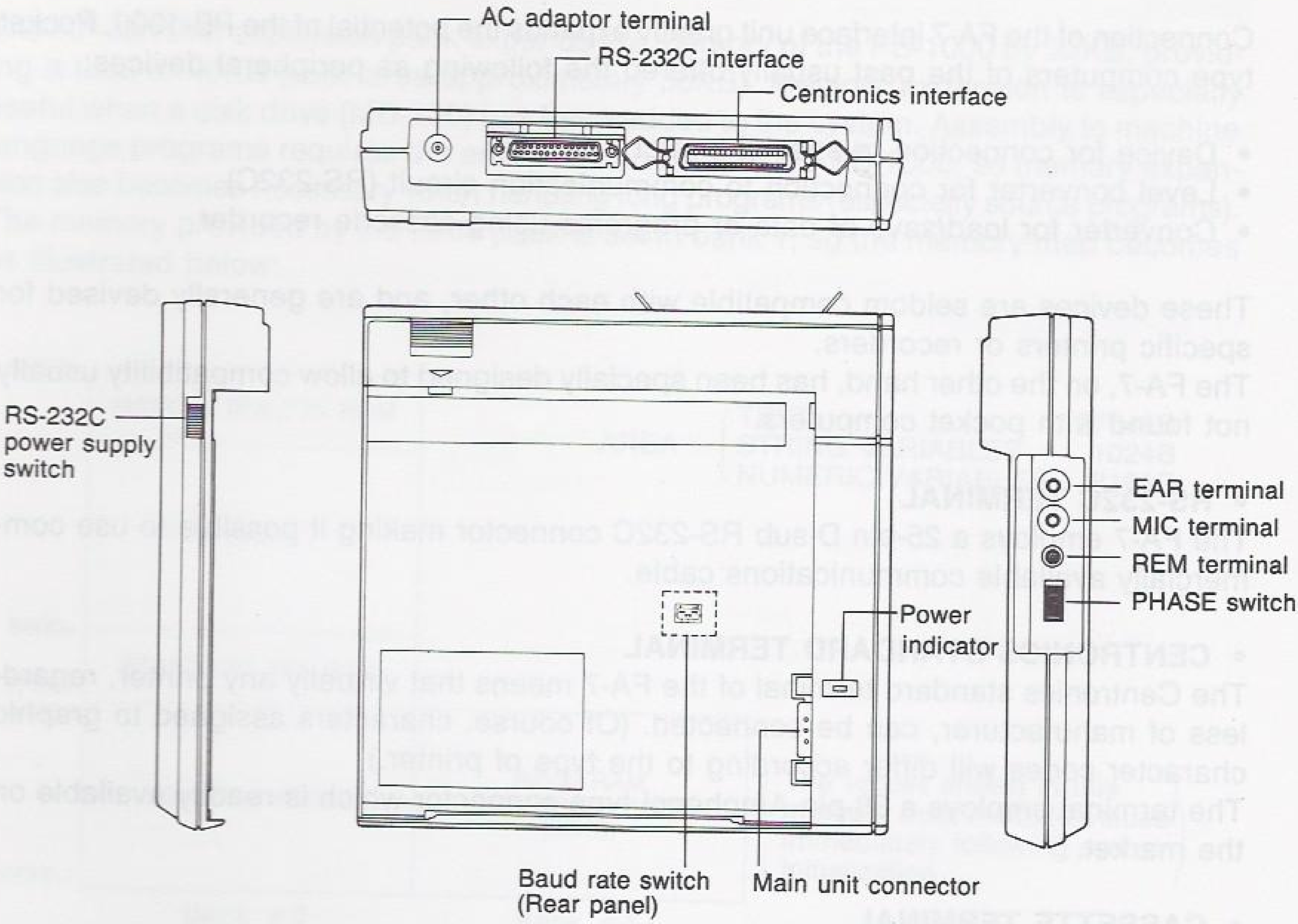
The FA-7 employs a 25-pin D-sub RS-232C connector making it possible to use commercially available communications cable.

- **CENTRONICS STANDARD TERMINAL**

The Centronics standard terminal of the FA-7 means that virtually any printer, regardless of manufacturer, can be connected. (Of course, characters assigned to graphic character codes will differ according to the type of printer.)
The terminal employs a 36-pin Amphenol type connector which is readily available on the market.

- **CASSETTE TERMINAL**

Recently, many audio improvements have been made in standard cassette tapes which causes problems in computer applications. In fact, some people claim that only older types of cassettes should be used for computer data storage.
Because of this, the FA-7 is equipped with a switch which inverts the phase of the input data (waveform). This allows data to be saved, but data load often cannot be performed correctly.
The FA-7 allows the load/save baud rate to be set just as with the RS-232C interface. This makes it possible to transfer large amounts of data from RAM to cassette tape in a short period of time.

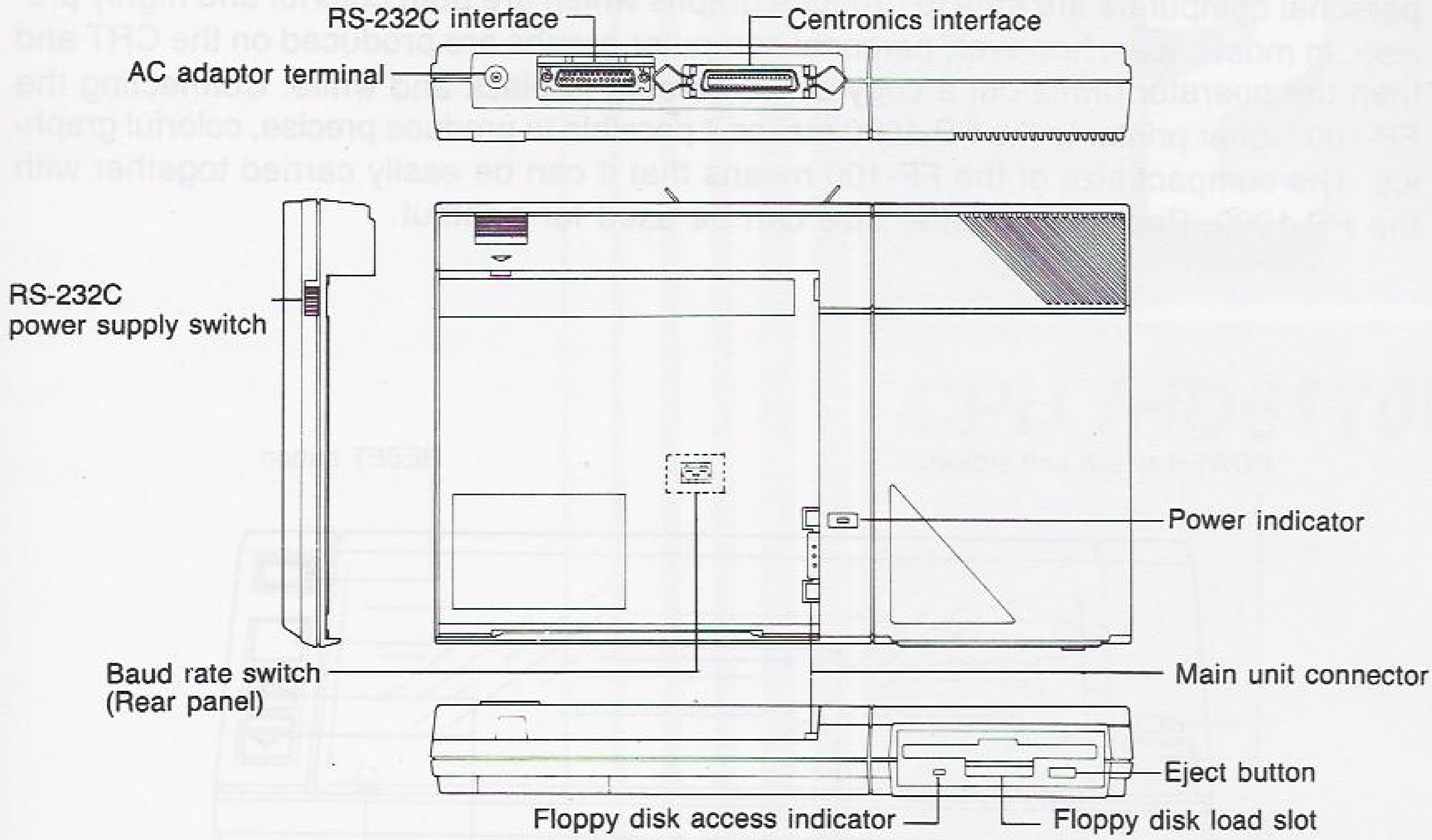


1-2-4 MICRO FLOPPY DISK DRIVE (MD-100)

Floppy disks are widely used because of their large capacity storage, high speed, and ease of handling. Because of this, the floppy disk is currently considered almost essential to any complete computer system. Until now, however, floppy disks were unavailable for use with such pocket computers as the PB-1000. Pocket type computers were limited to such auxiliary storage devices as cassette tape or RAM cartridges, which have the following drawbacks:

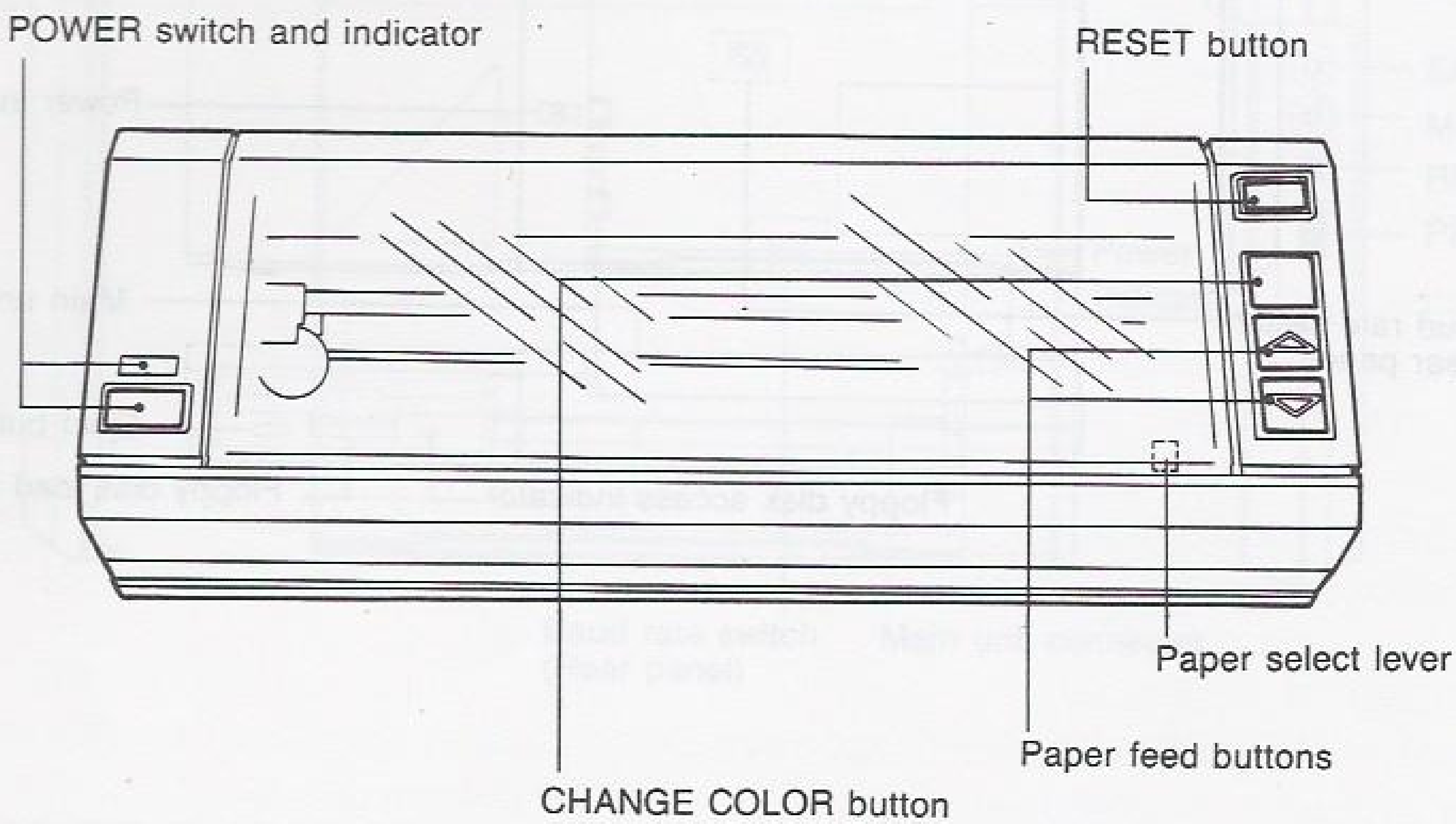
- Cassette tape requires a large amount of time for reading or writing data or programs
- RAM cartridges can be used immediately after being loaded into the computer, but they require special storage and handling

The demerits of both these media are overcome by the floppy disk. The protective case of the 3.5-inch floppy disk makes it even easier to handle than 5-inch or 8-inch disks. The MD-100 employs a 3.5-inch 1DD disk drive which stores data much more quickly than cassette tape.
As with the FA-7 interface unit, the MD-100 is also equipped with both RS-232C and Centronics standard terminals, making a PB-1000 + MD-100 combination a full-fledged personal computer system. Battery power capabilities allow portability for operation virtually anywhere, indoors and out.

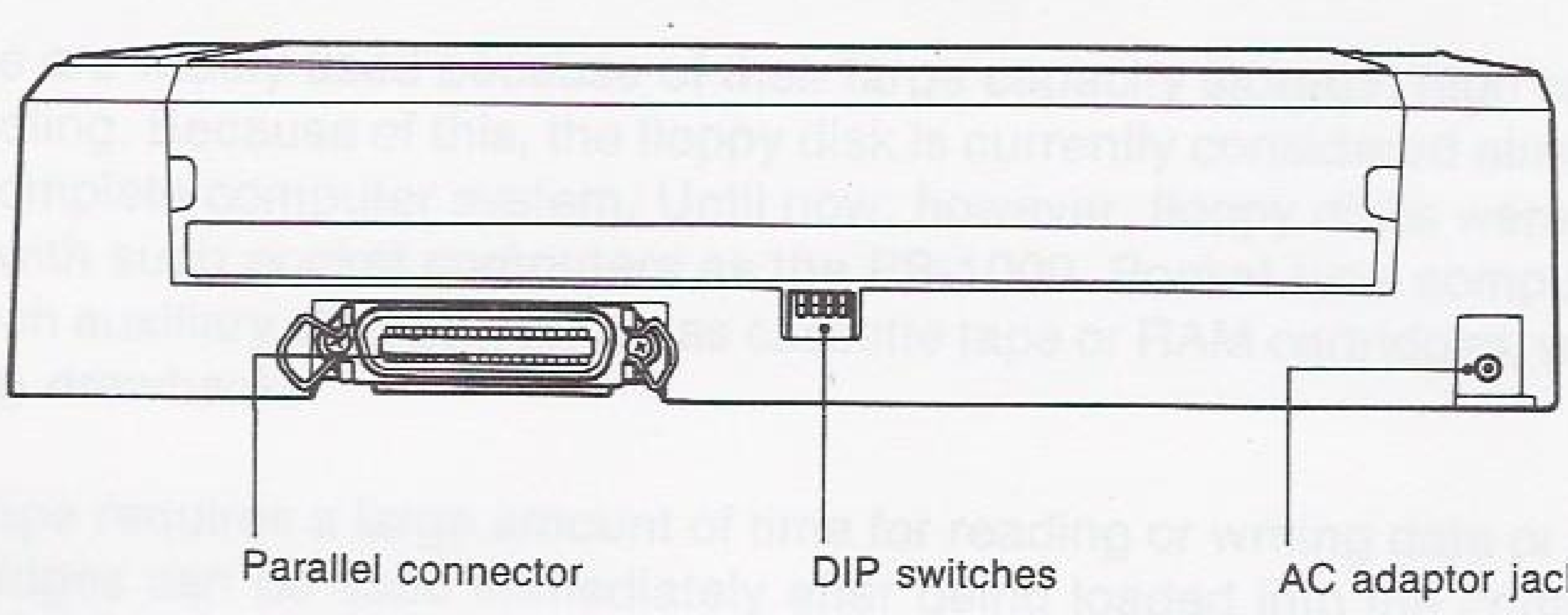

RS-232C interface
Centronics interface
AC adaptor terminal
RS-232C
power supply switch
Power indicator
Baud rate switch
(Rear panel)
Main unit connector
Eject button
Floppy disk access indicator
Floppy disk load slot

1-2-5 PLOTTER PRINTER (FP-100)

One popular application of the computer is graphing data or calculation results. Modern personal computers are able to produce graphs which are both colorful and highly precise. In most cases, however, personal computer graphs are produced on the CRT and then the operator prints out a copy of the screen, in black and white. Connecting the FP-100 plotter printer to the PB-1000 makes it possible to produce precise, colorful graphics. The compact size of the FP-100 means that it can be easily carried together with the PB-1000. Paper up to letter size can be used for printout.



BACK



CHAPTER 2

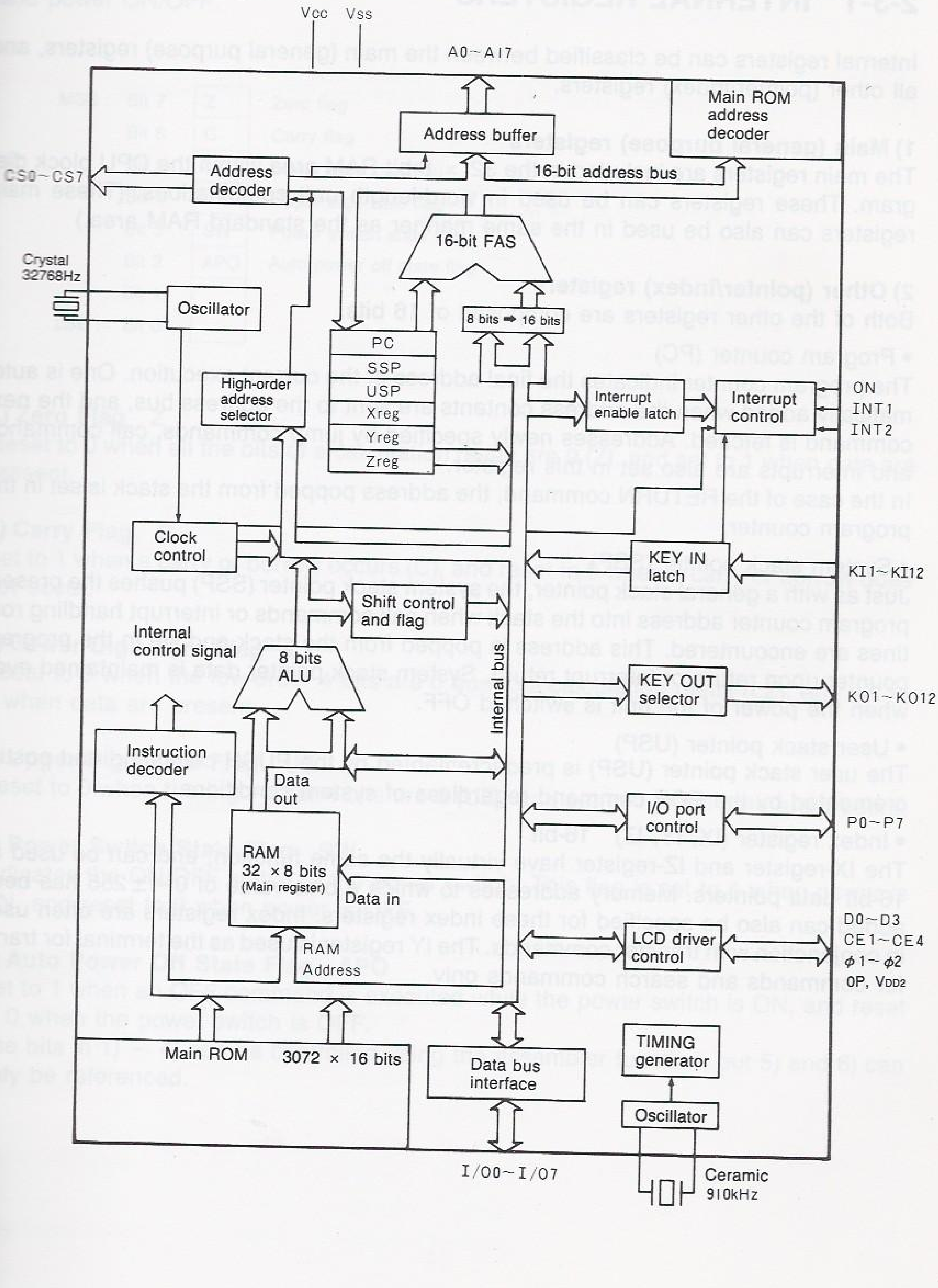
CPU (HD61700)

There are a large number of pocket computers available on the market today, each of which is powered by one of a number of different CPUs. The CPU (HD61700) used in the PB-1000 is one of the most powerful CPUs available.

2-1 HD61700 FEATURES

1) The HD61700 performs decimal based arithmetic computations, and it has its own 256K byte memory area. Though basically an 8-bit CPU, its address bus is 16-bit.
2) A 3072 × 16-bit built-in ROM makes high speed processing possible.
3) The static construction of the C-MOS provides operation with low power (maximum power consumption = 800μA).
4) A built-in 32 × 8-bit RAM can be used for both data storage and for registers. RAM may be accessed in word-length units.
5) Built-in clock function provides a stable time.
6) 5-level interrupt function.
7) 8-bit input/output ports.
8) Built-in special bus for LCD driver.

2-2 HD61700 BLOCK DIAGRAM



2-3 REGISTER CONFIGURATION

2-3-1 INTERNAL REGISTERS

Internal registers can be classified between the main (general purpose) registers, and all other (pointer/index) registers.

1) Main (general purpose) registers
The main registers are included in the 32 × 8-bit RAM area within the CPU block diagram. These registers can be used in word-length unit combinations. (These main registers can also be used in the same manner as the standard RAM area.)

2) Other (pointer/index) registers
Both of the other registers are composed of 16 bits.

• Program counter (PC)
The program counter indicates the final address of the current execution. One is automatically added when the address contents are sent to the address bus, and the next command is fetched. Addresses newly specified by jump commands, call commands and interrupts are also set in this register.
In the case of the RETURN command, the address popped from the stack is set in the program counter.

• System stack pointer (SSP)
Just as with a general stack pointer, the system stack pointer (SSP) pushes the present program counter address into the stack when call commands or interrupt handling routines are encountered. This address is popped from the stack and set in the program counter upon return or interrupt return. System stack pointer data is maintained even when the power of the unit is switched OFF.

• User stack pointer (USP)
The user stack pointer (USP) is predecremented by the PUSH command and postincremented by the POP command regardless of system conditions.

• Index register (IX, IY, IZ) 16-bit
The IX-register and IZ-register have virtually the same function, and can be used as 16-bit data pointers. Memory addresses to which a bias value of 0 ~ ±255 has been added can also be specified for these index registers. Index registers are often used in conjunction with transfer commands. The IY register is used as the terminal for transfer commands and search commands only.

2-3-2 FLAG REGISTERS

Flag registers are composed of 8 bits and indicate data related to calculation results and power ON/OFF.

MSB :	Bit 7	Z	Zero flag
	Bit 6	C	Carry flag
	Bit 5	LZ	Lower digit zero flag
	Bit 4	UZ	Upper digit zero flag
	Bit 3	SW	Power switch state flag
	Bit 2	APO	Auto power off state flag
	Bit 1		
LSB :	Bit 0		

1) Zero Flag Z
Reset to 0 when all the bits of a calculation result are 0 (Z), and set to 1 when data are present.

2) Carry Flag C
Set to 1 when a carry or borrow occurs (C), and reset to 0 when a carry or borrow does not occur.

3) Lower Digit Zero Flag LZ
Reset to 0 when the low-order 4 bits are 0 due to a calculation result (LZ), and set to 1 when data are present.

4) Upper Digit Zero Flag UZ
Reset to 0 when the high-order 4 bits are 0 (UZ), and set to 1 when data are present.

5) Power Switch State Flag SW
Indicates the ON/OFF status of the power switch. This flag is set to 1 when power is ON, and reset to 0 when power is OFF.

6) Auto Power Off State Flag APO
Set to 1 when an OFF command is executed while the power switch is ON, and reset to 0 when the power switch is OFF.
The bits in 1) ~ 4) can be controlled using the assembler function, but 5) and 6) can only be referenced.

2-3-3 STATUS REGISTERS

The status registers are used to determine the status of signal line being used internally by the computer, and the I/O status.

1) Interrupt Enable Register IE
Performs interrupt masking and sets the interrupt conditions (read/write).

MSB :	Bit 7	INT 1 interrupt enable	High priority
	Bit 6	KEY/Pulse interrupt enable	
	Bit 5	INT 2 interrupt enable	
	Bit 4	One-minute timer interrupt enable	
	Bit 3	$\overline{\text{ON}}$ terminal interrupt enable	
	Bit 2	Power ON enable from $\overline{\text{ON}}$ terminal	
	Bit 1	$\overline{\text{INT 1}}$ terminal interrupt edge specification	
LSB :	Bit 0	$\overline{\text{INT 2}}$ terminal interrupt level specification	

Bits 7 through 2 are set to 1 for enable and 0 for disable. Bit 1 is set to 0 for trailing edge and 1 for leading edge. Bit 0 is set to 0 for low level and 1 for high level. The RESET operation clears this register entirely, and bits 7, 6, 5, 1, and 0 are also cleared when power is switched OFF. The settings of bits 4 through 2 are maintained when power is switched OFF.

2) Interrupt Select and Key Output Register IA
Selects each interrupt and sets key output (read/write).

MSB :	Bit 7	Interrupt selection	1 = Key interrupt, 0 = pulse interrupt
	Bit 6	Pulse interrupt frequency	1 = 32Hz, 0 = 256Hz
	Bit 5	Key interrupt PIN specification	00 = NONE, 01 = ONE, 10 = TWO, 11 = ALL
	Bit 4		
	Bit 3	Key output specification	0000 (0) : Stop 0001 ~ 1100 (1 ~ 12) : ONE 1101 (13) : ALL 1110 ~ 1111 (14 ~ 15) : UNDEFINED
	Bit 2		
	Bit 1		
LSB :	Bit 0		

3) High-Order Address Specification Register UA

The 2-bit units of the contents of this register are added to the 16-bit registers (PC, IX, IY, IZ, SSP, USP) to allow specification of 18-bit addresses (read/write).

MSB :	Bit 7	IZ-register high-order address specification (00 ~ 11)
	Bit 6	
	Bit 5	IX-register/IY-register high-order address specification (00 ~ 11)
	Bit 4	
	Bit 3	SSP/USP register high-order address specification (00 ~ 11)
	Bit 2	
	Bit 1	PC high-order address specification (00 ~ 11)
LSB :	Bit 0	

The RESET operation clears this register entirely, and Z, X, Y and PC are also cleared when power is OFF. SSP and USP are maintained when power is switched OFF.

4) Display Driver Control Register

Outputs a control signal when display data or commands are sent to the display (write only).

MSB :	Bit 7	V_{DD2}
	Bit 6	$\phi1$, $\phi2$ clock 1 = ON, 0 = OFF
	Bit 5	Not used
	Bit 4	CE 4
	Bit 3	CE 3
	Bit 2	CE 2
	Bit 1	CE 1
LSB :	Bit 0	OP

Set values (except that set in bit 6) are output to the pin according to negative logic.

5) Port Status Specification Register PE

Specifies each bit of an I/O (8-bit) port for either input or output (read/write).

MSB :	Bit 7	Port 7	1 = Output,	0 = Input
	Bit 6	Port 6	1 = Output,	0 = Input
	Bit 5	Port 5	1 = Output,	0 = Input
	Bit 4	Port 4	1 = Output,	0 = Input
	Bit 3	Port 3	1 = Output,	0 = Input
	Bit 2	Port 2	1 = Output,	0 = Input
	Bit 1	Port 1	1 = Output,	0 = Input
LSB :	Bit 0	Port 0	1 = Output,	0 = Input

This register is cleared (reset to input status) when the RESET operation is performed or when power is switched OFF.

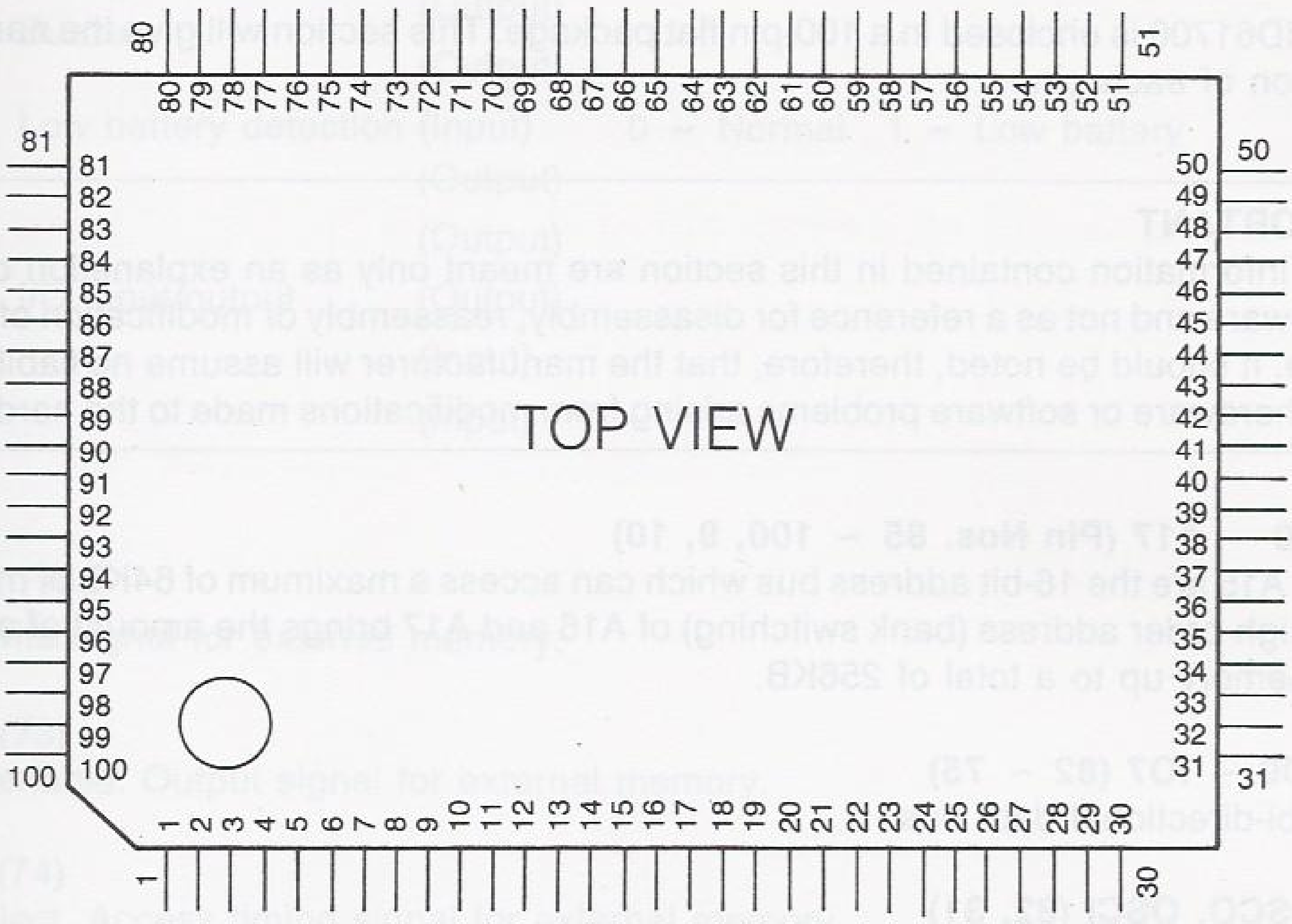
6) Port Data Register PD

Outputs contents through ports specified by port status specification register (read/write).

MSB :	Bit 7	Port 7 output data
	Bit 6	Port 6 output data
	Bit 5	Port 5 output data
	Bit 4	Port 4 output data
	Bit 3	Port 3 output data
	Bit 2	Port 2 output data
	Bit 1	Port 1 output data
LSB :	Bit 0	Port 0 output data

This register is not initialized by the RESET operation or when power is switched OFF (undefined).

2-4 HD61700 PIN LAYOUT



NO.	NAME	NO.	NAME	NO.	NAME	NO.	NAME	NO.	NAME
1	$\overline{\text{CS 0}}$	21	V_{DD} 2	41	KI 2	61	KO 10	81	IO 1
2	$\overline{\text{CS 1}}$	22	$\overline{\text{D 0}}$	42	KI 3	62	KO 11	82	IO 0
3	$\overline{\text{CS 2}}$	23	$\overline{\text{D 1}}$	43	KI 4	63	KO 12	83	Vcc
4	$\overline{\text{CS 3}}$	24	$\overline{\text{D 2}}$	44	KI 5	64	P 7	84	Vss
5	$\overline{\text{CS 4}}$	25	$\overline{\text{D 3}}$	45	KI 6	65	P 6	85	A 0
6	$\overline{\text{CS 5}}$	26	$\overline{\text{DB}}$	46	KI 7	66	P 5	86	A 1
7	$\overline{\text{CS 6}}$	27	Vx	47	KI 8	67	P 4	87	A 2
8	$\overline{\text{CS 7}}$	28	XO	48	KI 9	68	P 3	88	A 3
9	A 16	29	XI	49	KI 10	69	P 2	89	A 4
10	A 17	30	Vcc	50	KI 11	70	P 1	90	A 5
11	$\overline{\text{ON}}$	31	OSCI	51	KI 12	71	P 0	91	A 6
12	$\overline{\text{INT 1}}$	32	OSCO	52	KO 1	72	R/$\overline{\text{W}}$	92	A 7
13	$\overline{\text{INT 2}}$	33	V_{DD}	53	KO 2	73	$\overline{\text{OE}}$	93	A 8
14	ϕ 1	34	$\overline{\text{T}}$	54	KO 3	74	CS	94	A 9
15	ϕ 2	35	PST	55	KO 4	75	IO 7	95	A 10
16	$\overline{\text{OP}}$	36	$\overline{\text{SW}}$	56	KO 5	76	IO 6	96	A 11
17	$\overline{\text{CE 1}}$	37	F	57	KO 6	77	IO 5	97	A 12
18	$\overline{\text{CE 2}}$	38	$\overline{\text{M}}$	58	KO 7	78	IO 4	98	A 13
19	$\overline{\text{CE 3}}$	39	$\overline{\phi 3}$	59	KO 8	79	IO 3	99	A 14
20	$\overline{\text{CE 4}}$	40	$\overline{\text{KI 1}}$	60	KO 9	80	IO 2	100	A 15

2-5 HD61700 PIN CONFIGURATION

The HD61700 is enclosed in a 100-pin flat package. This section will give the name and function of each pin.

IMPORTANT
The information contained in this section are meant only as an explanation of unit hardware and not as a reference for disassembly, reassembly or modification of hardware. It should be noted, therefore, that the manufacturer will assume no liability for any hardware or software problems arising from modifications made to the hardware.

1) A0 ~ A17 (Pin Nos. 85 ~ 100, 9, 10)
A0 ~ A15 are the 16-bit address bus which can access a maximum of 64KB of memory. The high-order address (bank switching) of A16 and A17 brings the amount of accessible memory up to a total of 256KB.

2) IO0 ~ IO7 (82 ~ 75)
8-bit bi-directional data bus.

3) OSCO, OSCI (32, 31)
910KHz ceramic filter connection terminal used by the system clock. Signal divided by 2 or 3 internally.

4) $\overline{CS0}$ ~ $\overline{CS7}$ (1 ~ 8)
Chip select signal produced by decoding of address (negative logic).

$\overline{CS0}$	$08000_H \sim 0FFFF_H$	ROM	
$\overline{CS1}$	$04000_H \sim 05FFF_H$	FREE	
$\overline{CS2}$	$06000_H \sim 07FFF_H$	RAM	
$\overline{CS3}$	$18000_H \sim 19FFF_H$	RAM	Expanded RAM (option)
$\overline{CS4}$	$1A000_H \sim 1BFFF_H$	RAM	Expanded RAM (option)
$\overline{CS5}$	$1C000_H \sim 1DFFF_H$	RAM	Expanded RAM (option)
$\overline{CS6}$	$1E000_H \sim 1FFFF_H$	RAM	Expanded RAM (option)
$\overline{CS7}$	$00C00_H \sim 00C0F_H$	I/O	

NOTE: Some differences may be present with other versions.

5) KO1 ~ KO12 (52 ~ 63)
Key output terminals. Low level output is possible for all or individual KO terminals.

6) KI1 ~ KI12 (40 ~ 51)
Key input terminals. After latching of input from KI1 ~ KI12, input is divided into KI1 ~ 8 and KI9 ~ 12, and then introduced into the data bus (KI terminal pulled up).

7) P0 ~ P7 (71 ~ 64)
8-bit input/output port. Input/output can be selected by software.

P7	Buzzer	(Output)	
P6		(Output)	
P5	Low battery detection	(Input)	0 = Normal 1 = Low battery
P4	I/O input/output	(Output)	
P3		(Output)	
P2		(Output)	
P1		(Input)	
P0		(Input)	

8) $R/\overline{W}$ (72)
Read/Write signal for external memory.

9) $\overline{OE}$ (73)
Output enable. Output signal for external memory.

10) $\overline{CS}$ (74)
Chip select. Access timing signal for external memory.

11) $\overline{SW}$ (36)
Power switch terminal.
$\overline{SW}$ = LOW : Internal logic power ON.
$\overline{SW}$ = HIGH : Change in status flag only, without power OFF. Power OFF performed by software.

12) $\overline{ON}$, $\overline{INT1}$, $\overline{INT2}$ (11 ~ 13)
Maskable interrupt request terminals (all negative logic).

13) XO, XI (28, 29)
Quartz oscillator (32,768Hz) connection terminal.

14) $\overline{DB}$, $\overline{M}$, $\overline{\phi_3}$ (26, 38, 39)
Debug terminal. $\overline{M}$ and ϕ_3 output when $\overline{DB}$ is LOW ($\overline{DB}$ is normally HIGH). $\overline{M}$ outputs the operation code fetch timing (fetch cycle of operation code for next execution command).

15) $\overline{RST}$ (35)
Reset terminal. Low level is the reset status during which HD61700 is initialized. (Live even when power is OFF.)

16) $\overline{T}$ (34)
HD61700 test terminal (normally HIGH).

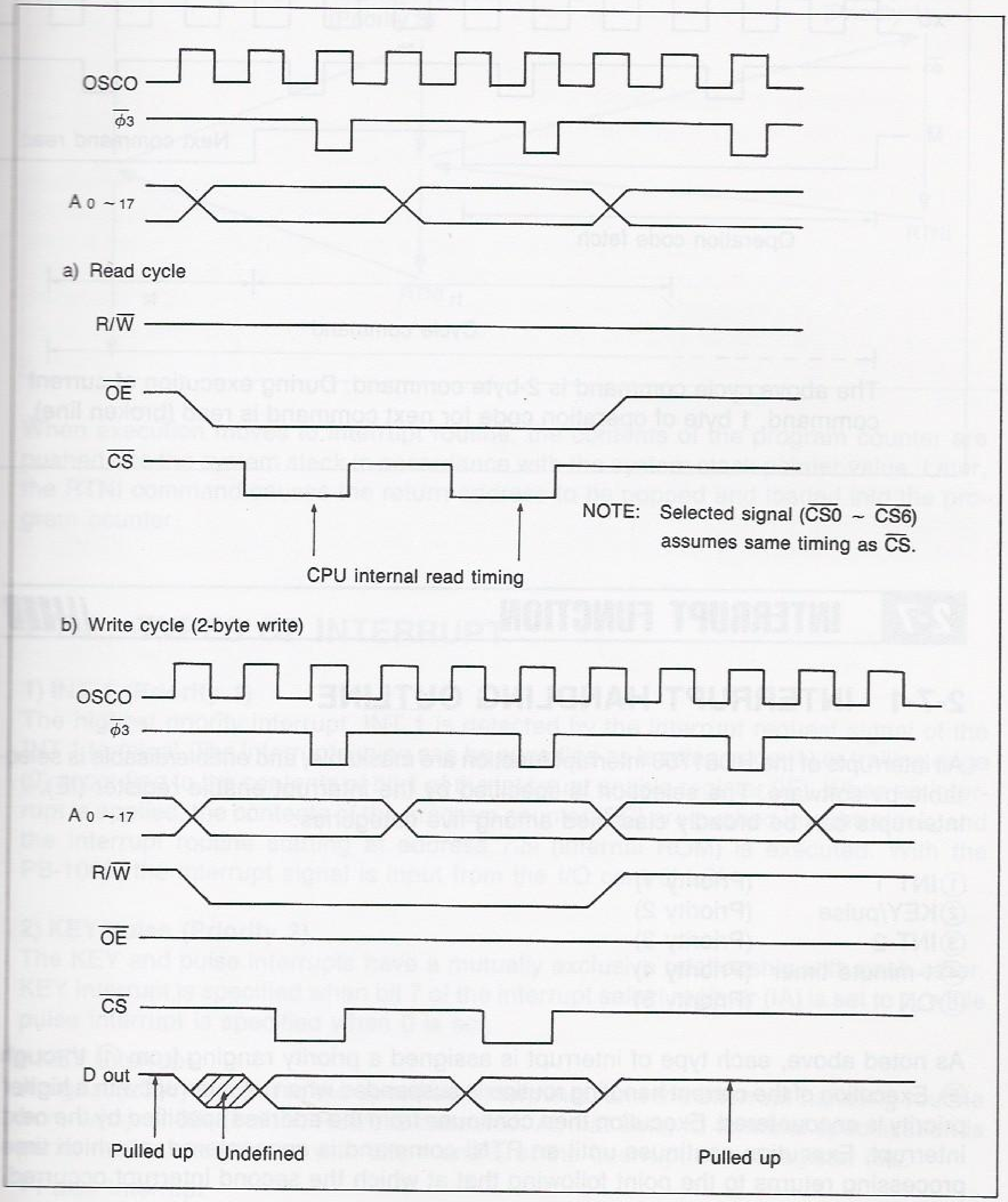
17) F (37)
Indicates whether or not dividing is being performed. The dividing command of the HD61700 (SLOW command) allows the system clock to be divided by 1/16, causing this terminal to switch to low level.

18) $\overline{\phi 1}$, $\overline{\phi 1}$, $\overline{OP}$, $\overline{CE1}$ ~ $\overline{CE4}$, $\overline{D0}$ ~ $\overline{D3}$, V_{DD2} (14 ~ 20, 22 ~ 26, 21)
Control signals to the display driver.

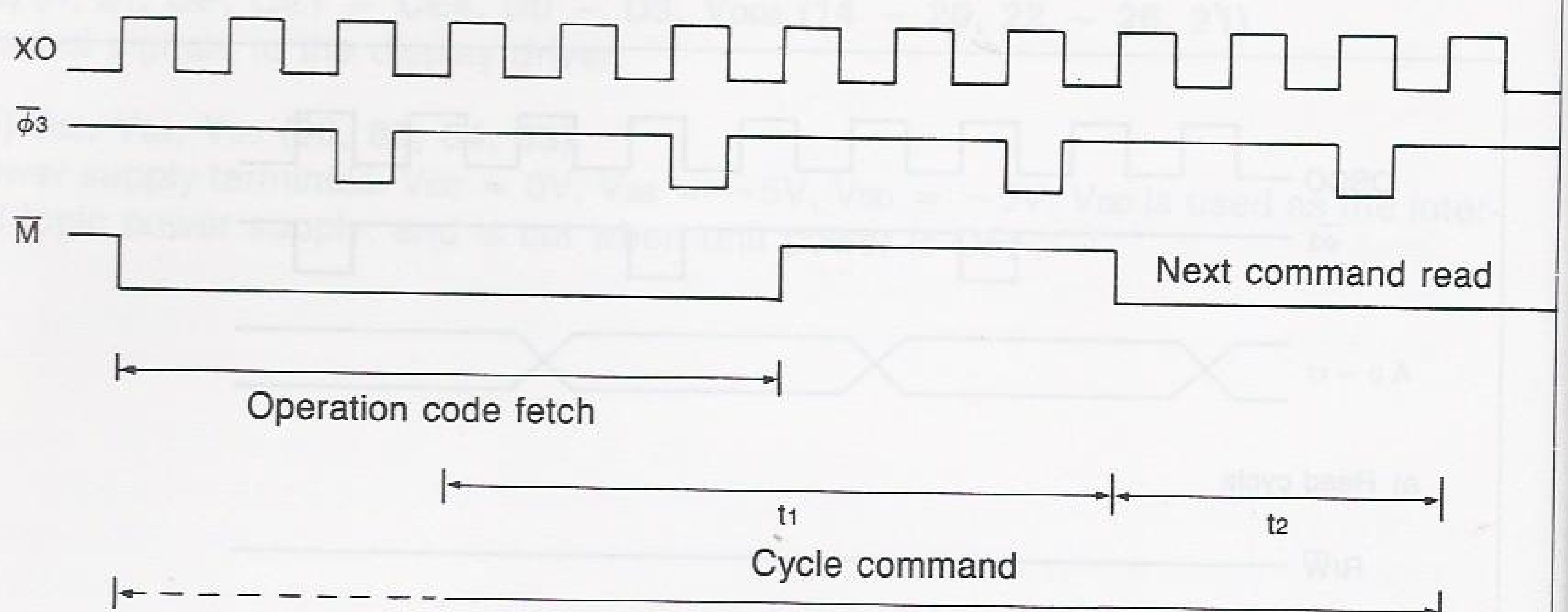
19) V_{CC}, V_{SS}, V_{DD} (30, 83, 84, 33)
Power supply terminals. V_{CC} = 0V, V_{SS} = −5V, V_{DD} = −5V. V_{DD} is used as the internal logic power supply, and is cut when unit power is OFF.

2-6 TIMING CHART

2-6-1 BUS TIMING CHART



2-6-2 COMMAND FETCH TIMING CHART



The above cycle command is 2-byte command. During execution of current command, 1 byte of operation code for next command is read (broken line).

2-7 INTERRUPT FUNCTION

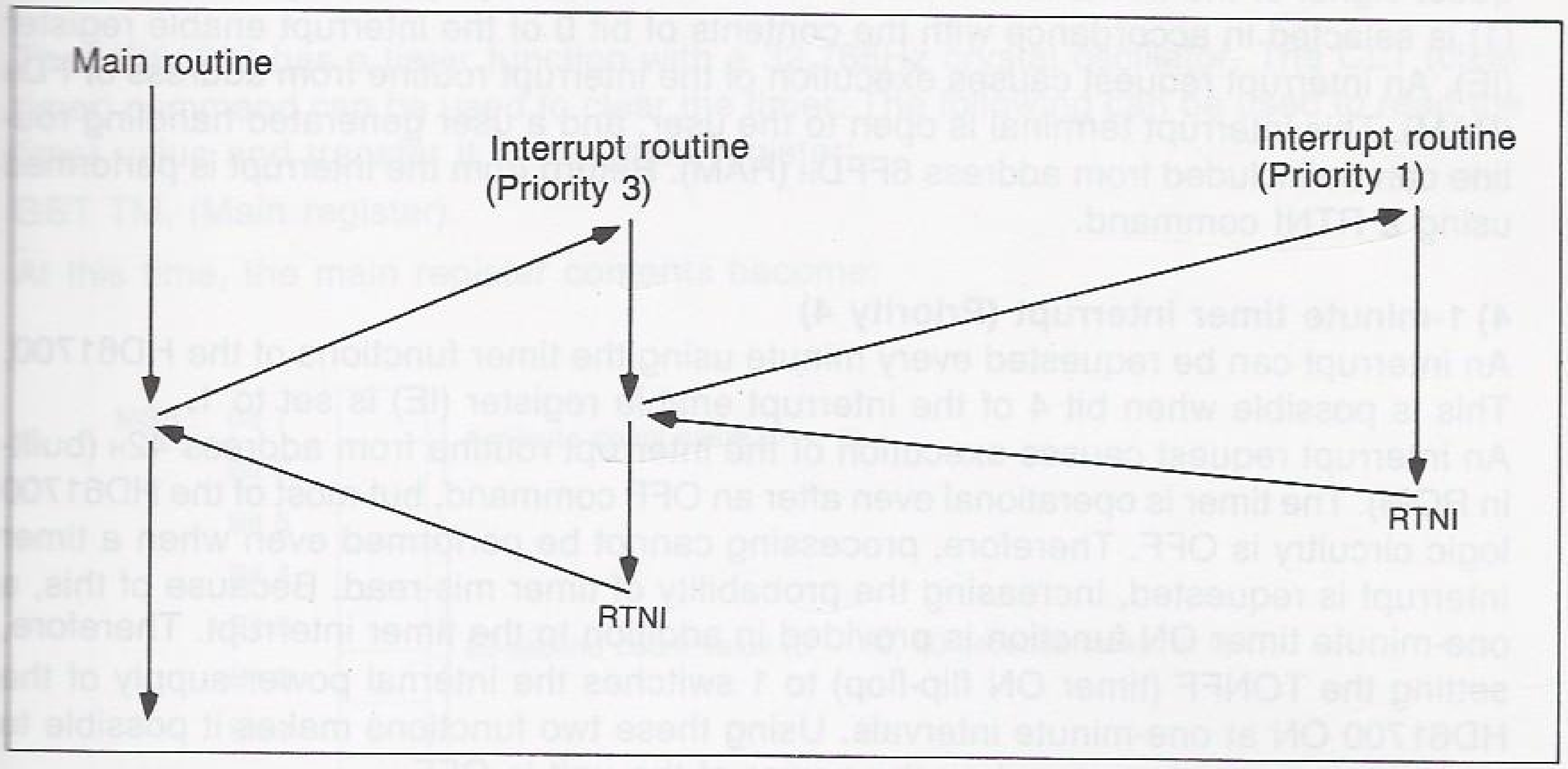
2-7-1 INTERRUPT HANDLING OUTLINE

All interrupts of the HD61700 interrupt function are maskable, and enable/disable is selectable by software. The selection is specified by the interrupt enable register (IE). Interrupts can be broadly classified among five categories:

①INT 1	(Priority 1)
②KEY/pulse	(Priority 2)
③INT 2	(Priority 3)
④1-minute timer	(Priority 4)
⑤ON	(Priority 5)

As noted above, each type of interrupt is assigned a priority ranging from ① through ⑤. Execution of the current handling routine is suspended when an interrupt with a higher priority is encountered. Execution then continues from the address specified by the next interrupt. Execution continues until an RTNI command is encountered, at which time processing returns to the point following that at which the second interrupt occurred.

Example



When execution moves to interrupt routine, the contents of the program counter are pushed into the system stack in accordance with the system stack pointer value. Later, the RTNI command causes the return address to be popped and loaded into the program counter.

2-7-2 TYPES OF INTERRUPT

1) INT 1 (Priority 1)

The highest priority interrupt, INT 1 is detected by the interrupt request signal of the INT 1 terminal. The interrupt timing can be specified as leading edge (1) or trailing edge (0) according to the contents of bit 1 of the interrupt enable register (IE). When an interrupt is applied, the contents of the program counter (PC) are pushed into the stack, and the interrupt routine starting at address 72_H (internal ROM) is executed. With the PB-1000, the interrupt signal is input from the I/O control LSI.

2) KEY/pulse (Priority 2)

The KEY and pulse interrupts have a mutually exclusive relationship with each other. KEY interrupt is specified when bit 7 of the interrupt select register (IA) is set to 1, while pulse interrupt is specified when 0 is set.

• KEY Interrupt

A signal (LOW) is input to the specified KI terminal, and the interrupt handling routine from address 62_H (built-in ROM) is executed. At that time, KI terminal specification is performed in accordance with bits 4 and 5 of the interrupt select register (IA).

• Pulse Interrupt

An interrupt is requested at each cycle of the specified frequency, and the interrupt handling routine from address 62_H (built-in ROM) is executed. Frequency selection is performed in accordance with bit 6 of the interrupt selection register (IA). The PB-1000 normally uses pulse interrupt, and bit 6 is set to 0 (256Hz, 3.9ms).

3) INT 2 (Priority 3)
$\overline{INT}$ 2 is an external pin interrupt which is detected in accordance with the interrupt request signal of the INT 2 terminal. Either low level interrupt (0) or high level interrupt (1) is selected in accordance with the contents of bit 0 of the interrupt enable register (IE). An interrupt request causes execution of the interrupt routine from address $6FFD_H$ (RAM). This interrupt terminal is open to the user, and a user generated handling routine can be included from address $6FFD_H$ (RAM). Return from the interrupt is performed using a RTNI command.

4) 1-minute timer interrupt (Priority 4)
An interrupt can be requested every minute using the timer functions of the HD61700. This is possible when bit 4 of the interrupt enable register (IE) is set to 1.
An interrupt request causes execution of the interrupt routine from address 42_H (built-in ROM). The timer is operational even after an OFF command, but most of the HD61700 logic circuitry is OFF. Therefore, processing cannot be performed even when a timer interrupt is requested, increasing the probability of timer mis-read. Because of this, a one-minute timer ON function is provided in addition to the timer interrupt. Therefore, setting the TONFF (timer ON flip-flop) to 1 switches the internal power supply of the HD61700 ON at one-minute intervals. Using these two functions makes it possible to perform time counts even when the power of the unit is OFF.

5) ON (Priority 5)
The lowest priority interrupt, ON is detected by the interrupt request signal of the $\overline{ON}$ terminal.
The five interrupts detailed above are handled by detection of signals from hardware. Besides these, the HD61700 is capable of software interrupts.

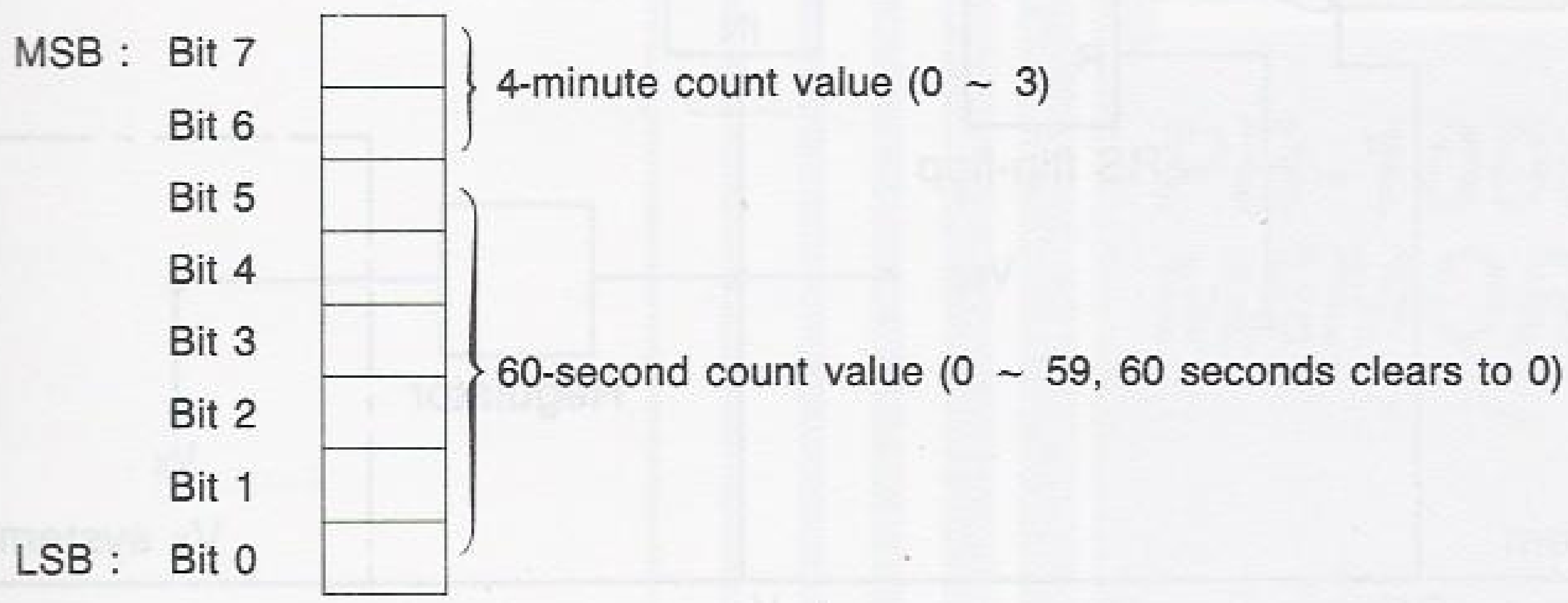
6) Software Interrupts
The operation code FF_H is assigned as the software interrupt code. The interrupt routine from address $6FFA_H$ is executed when this command is encountered. This interrupt is open to the user, and a user generated handling routine can be included from address $6FFA_H$.

2-8 TIMER FUNCTION

The HD61700 has a timer function with a 32,768Hz crystal oscillator. The CLT (clear timer) command can be used to clear the timer. The following can be used to read the timer value and transfer it to the main register:

GST TM, (Main register)

At this time, the main register contents become:



This internal register may be located anywhere within the range of \$0 ~ \$29, and this procedure is used to set the timer within the range of 0 minutes 59 seconds through 3 minutes 59 seconds. It should be noted that using this procedure in an actual program results in two readings to confirm whether or not the timer values are identical. Values may sometimes be different because the timer and internal clock are not synchronized. This timer can be used either with the 1-minute timer interrupt function or the pulse interrupt function.

2-9 INTERNAL POWER SUPPLY CONFIGURATION

The internal logic of the HD61700 is divided among three power supplies.

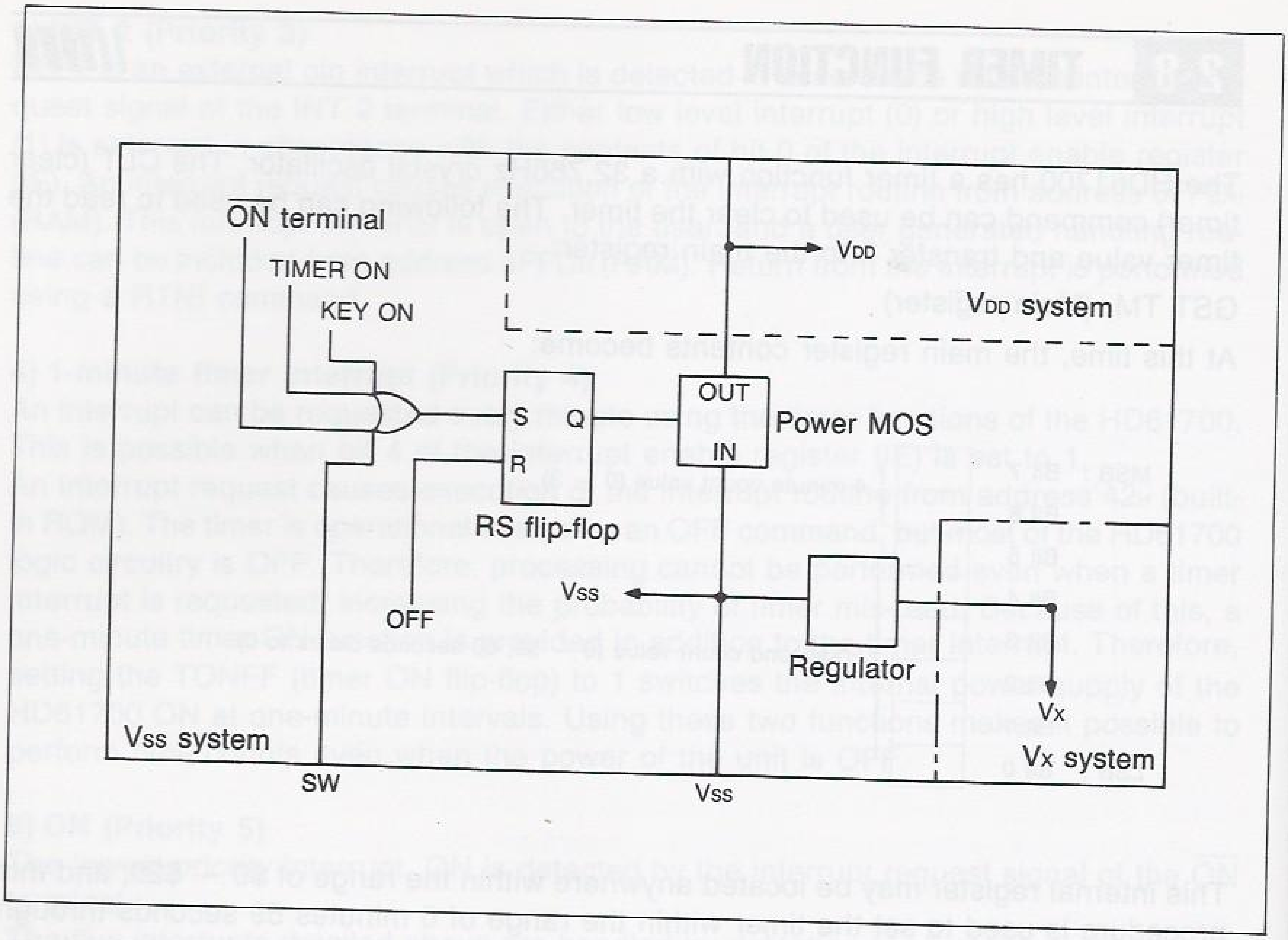
V_{SS} system : Power supply not interrupted even when OFF command is executed. Maintains memory contents as long as batteries are supplying power.

V_{DD} system : Power supply interrupted by OFF command.

V_X system : Power supply for timer crystal oscillator. As with V_{SS}, power supply not interrupted when OFF command is executed.

• PB-1000 ON/OFF

The power switch, timer, KEY, and $\overline{\text{ON}}$ terminal can all be used to switch power ON. Only software (OFF command) can be used to switch power OFF.

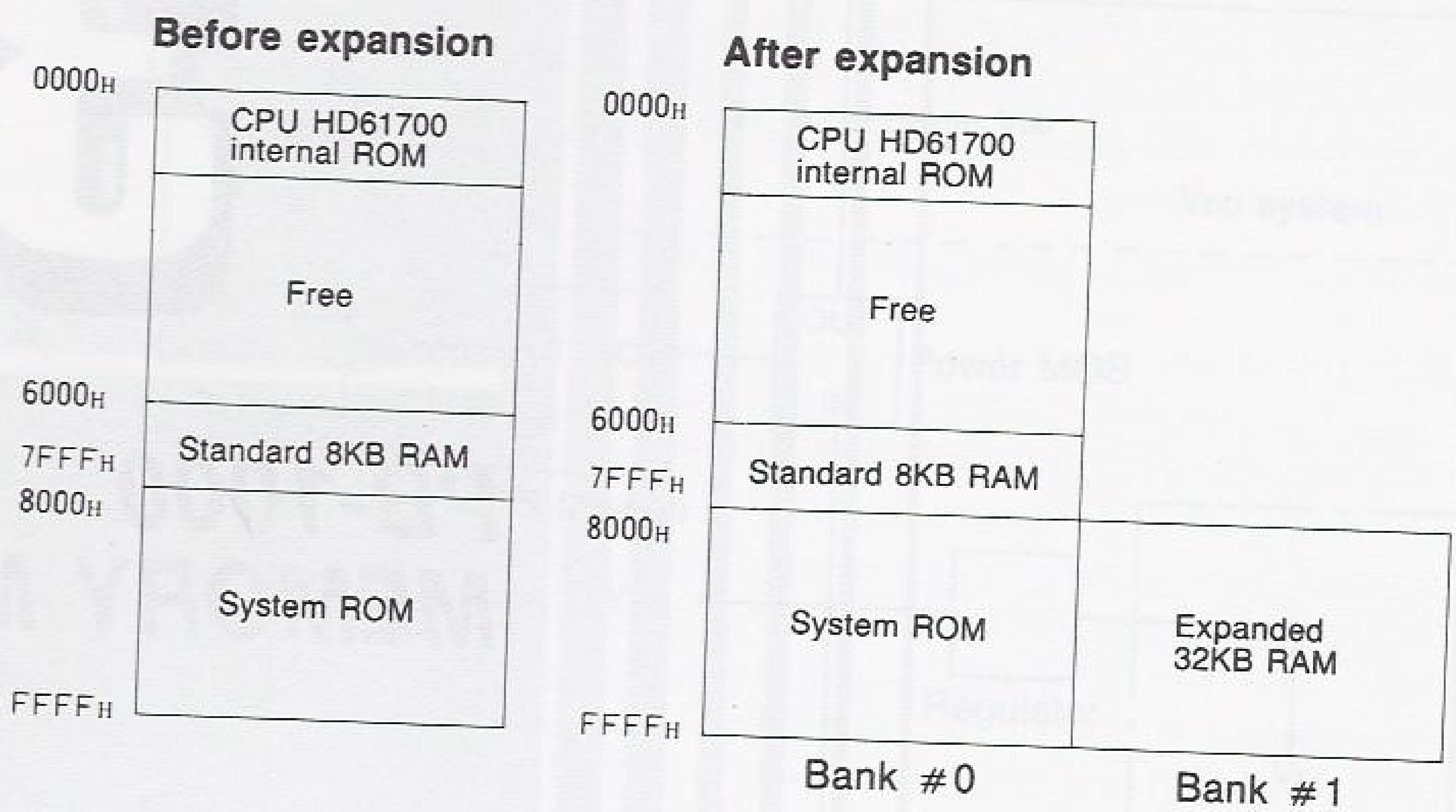


The above is a simplified layout diagram of the HD61700 power supply. Q output switches to HIGH when any ON signal is input at the RS flip-flop set side. Then the power MOS switches ON and V_{DD} power is supplied. Execution of an OFF command resets the RS flip-flop and cuts the V_{DD} power supply.

CHAPTER 3

PB-1000 MEMORY MAP

3-1 UNIT MEMORY LAYOUT



Before expansion:

Free area	TEXT	3072 bytes	(FREE)
	String variables	256 bytes	($)
	Numeric variables	768 bytes	(V)

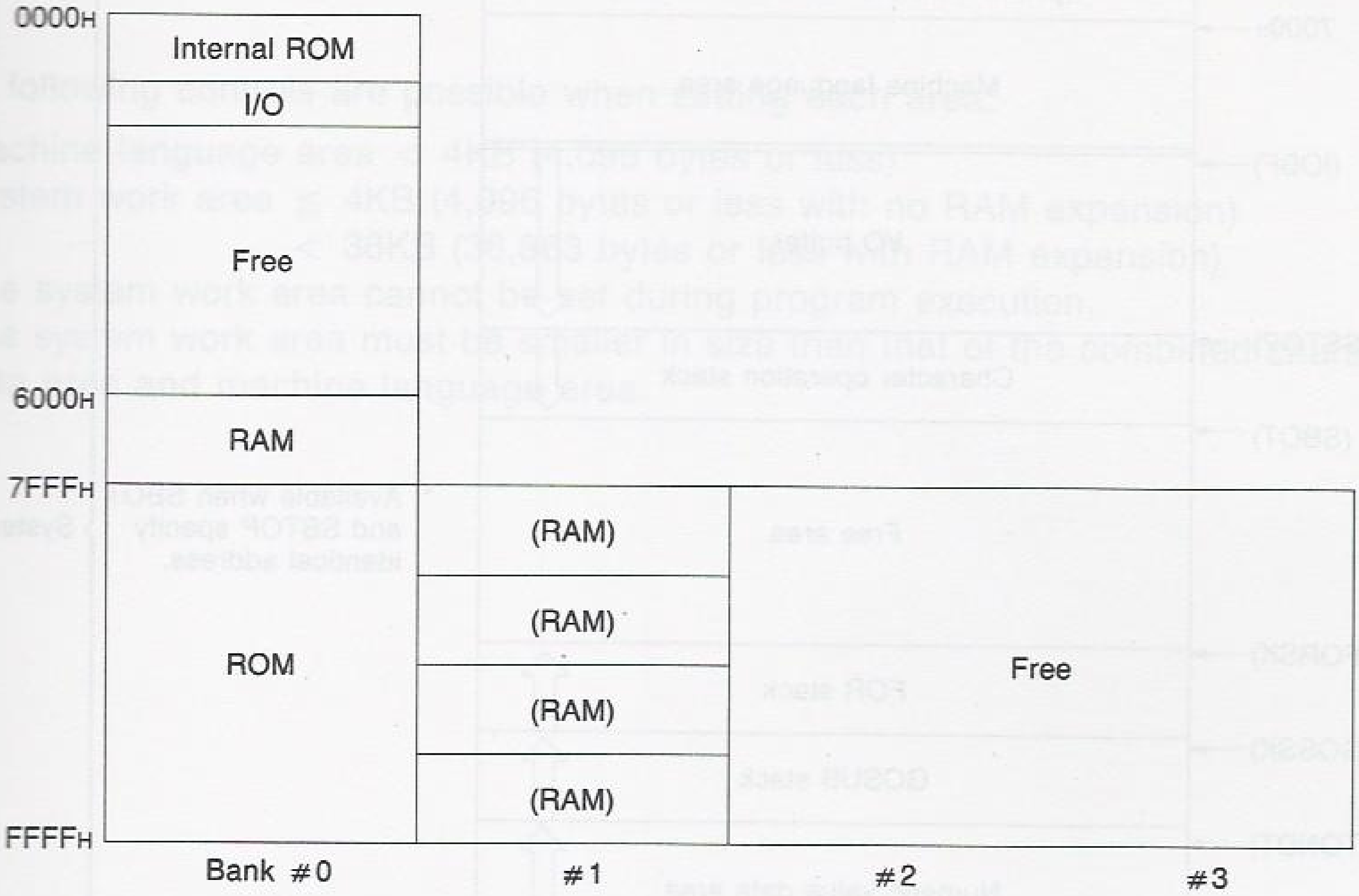
After expansion:

Free area	TEXT	27648 bytes
	String variables	1024 bytes
	Numeric variables	8191 bytes

1) Built-in ROM is CPU (HD61700) internal ROM (3072 × 16 bits).
2) Standard RAM is 8KB, but 4KB is available as the user area. Therefore, expansion RAM (RP-32) is recommended for larger files and data volumes.
3) A program is located in system ROM for BASIC, I/O control, etc.
4) Expanded memory is all allocated to Bank #1, not to the free area.
5) Banks 2 and 3 are free areas, and the bank cannot be specified (from the monitor).
6) 0000_H ~ $7FFE_H$ of Bank #0 are normally accessed regardless of the bank specified.

3-1-1 HD61700

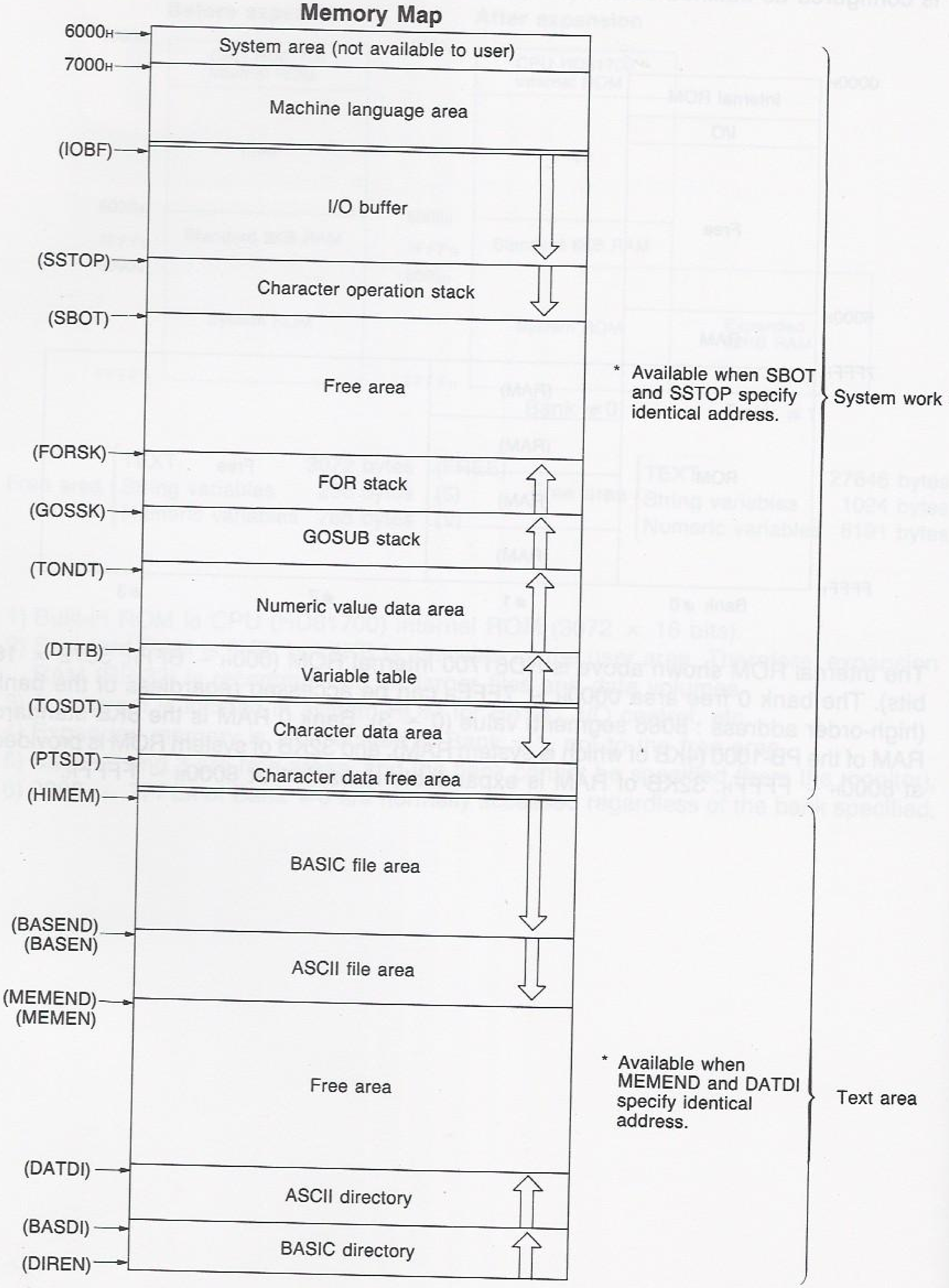
The free area of the HD61700 is at 00000H ~ 3FFFFH, but the free area of the PB-1000 is configured as outlined below. (18 external addresses, 8 data addresses.)



The internal ROM shown above is HD61700 internal ROM (000H ~ BFFH, 3072 × 16 bits). The bank 0 free area 0000H ~ 7FFEH can be accessed regardless of the bank (high-order address : 8086 segment) value (0 ~ 3). Bank 0 RAM is the 8KB standard RAM of the PB-1000 (4KB of which is system RAM), and 32KB of system ROM is provided at 8000H ~ FFFFH. 32KB of RAM is expandable in bank 1 at 8000H ~ FFFFH.

3-2 DETAILED MEMORY MAP

Memory Map



1) Names in parentheses indicate work area labels. See the work area table at the back of this manual for the address of each label.
2) The character data area, machine language area, and system work area sizes are determined by the CLEAR statement.
3) The NEW ALL statement results in the same layout as that produced by CLEAR 256, 0, 1024 (CLEAR 1024, 0, 9215 after RAM expansion).

The following controls are possible when setting each area:

- Machine language area < 4KB (4,095 bytes or less)
- System work area ≦ 4KB (4,096 bytes or less with no RAM expansion)
  < 36KB (36,863 bytes or less with RAM expansion)
- The system work area cannot be set during program execution.
- The system work area must be smaller in size than that of the combined character data area and machine language area.

3-3 FCB, I/O BUFFER MEMORY ALLOCATION

0	FCB + I/O buffer Length	COM only = 35 bytes, Other = 291 bytes
2	File No.	··· 0 ~ 15
3	Access type	··· 30 ··· OUTPUT, 31 ··· RANDOM, 32 ··· INPUT, 34 ··· APPEND
4	Device type	··· 0 ··· RAM, 1 ··· FDD, 2 ··· COM
5	Filename	
13	Extension	
16	I/O buffer top address	
18	I/O buffer end address	
20	Character counter	
21	Memory overflow flag	
22	EOF flag	··· 0 ··· NOT EOF, FF ··· EOF
23	INPUT # / PRINT # / FIELD	
25	Record No.	
27	Character pointer	··· Zone tab
28	Classification	··· WORK
29	RAM file top address	
31	RAM file end address	
33	RAM file access pointer	
35	I/O buffer	

CHAPTER 4

BASIC PROGRAM INTERNAL CONFIGURATION AND DATA FORMAT

4-1 VIEWING BASIC PROGRAMS

The monitor is used to see how BASIC programs are stored internally. The D monitor command is used to dump memory contents within a specified range onto the screen. The STOP key can be used to suspend scrolling of the memory contents on the screen, or the BRK key can be used to halt scrolling when the specified range exceeds the limitations of the display.

4-1-1 VIEWING PROGRAM CONTENTS

Assume that the following sample program is present in memory:

```
10 REM**********
20 REM ABCDEFG
30 REM**********
40 FOR I=1 TO 100
50 PRINT I
60 NEXT I
70 END
```

This BASIC program is stored in memory in the following format:

```
10  0A 00  20  04 A9   2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  00
V   10         REM     *  *  *  *  *  *  *  *  *  *   *

0E  14 00  20  04 A9   20  41 42 43 44 45 46 47  00  10  1E
V   20         REM         A  B  C  D  E  F  G   *   V

00  20  04 A9  2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  00  11  28
30      REM    *  *  *  *  *  *  *  *  *  *   *   V

00  20  04 81  20  49 30 31  20  07 C1  20  31 30 30  00  08
40      FOR        I  =  1       TO         1  0  0   *   V

32 00  20  04 A3  20  49  00  08  3C 00  20  04 82  20  49  00
50         PRINT      I   *   V   60         NEXT       I   *

06  46 00  20  04 87  00  00
V   70         END    *
```

*: 00_H codes used to indicate the end of statements

V: Number of characters in statements

As can be seen here, a PB-1000 BASIC program consists of the following:

1. Statement lengths (1 byte each)
2. Line numbers (2 bytes each)
3. Spaces (1 byte each)
4. Statements (n bytes each)
5. Null codes (1 byte each)

With PB-1000 BASIC, the program start address is not established, as it is with other personal computers. Because of this, multiple PB-1000 BASIC programs can be simultaneously stored in memory, each with its own individual start address. These BASIC programs are managed as files, and the file (BASIC file) directory is stored at a location beginning from the address located at $694B_H$, $694C_H$ (label name DIREN). The directory is stored upwards (towards address 0000_H). The main BASIC programs are stored at a location beginning from the address stored at 6941_H, 6942_H (label name HIMEM) downwards (towards $FFFF_H$), up to a location at the address stored at 6943_H, 6944_H (label name BASEN).

4-1-2 CHARACTER CREATION

The following command is another interesting feature of PB-1000 BASIC which makes it possible to freely define characters:

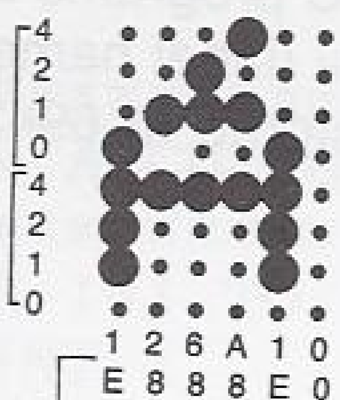
DEF CHR$

This command can be used to create graphic as well as alphabetic characters.

ALPHABETIC CHARACTERS

Á

```
10  DEF CHR$ (&HF0)="1E2868A81E00"
20  PRINT CHR$ (&HF0)
```



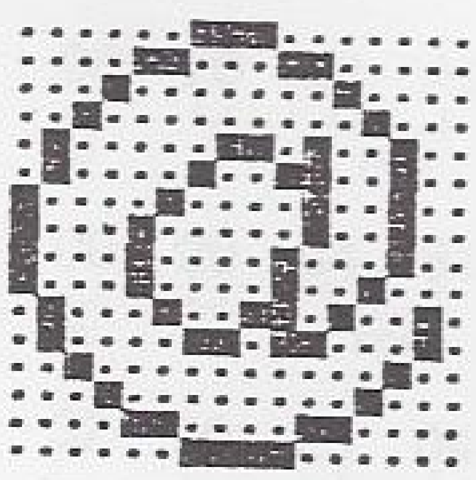
C_0 C_2 C_4 C_6 C_8 C_{10}
C_1 C_3 C_5 C_7 C_9 C_{11} → C_0 C_1 C_2 C_3 C_4 C_5 C_6 C_7 C_8 C_9 C_{10} C_{11}

||

1 E 2 8 6 8 A 8 1 E 0 0

• Characters are formed using an 8 × 6 dot matrix.

SYMBOLS



001244 888442 1000
3C0012 4884F0 0F00

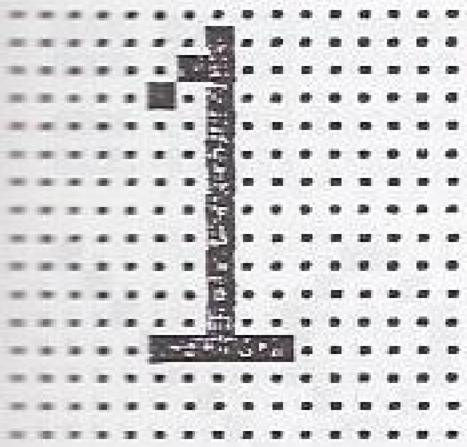
C300C2 112F12 4830
008422 111122 4800

```
10  FOR I=1 TO 6
20  READ A$
30  DEFCHR$ (&HF0)=A$
40  PRINTCHR$ (&HF0) ;
50  IF I=3 THEN PRINT
60  NEXT I
70  PRINT
80  DATA 030C10204142, 848888444F20, 100F00000000
90  DATA C0300804C222, 111121F11222, 448830000000
```

• Symbols are formed using a 16 × 16 dot matrix.

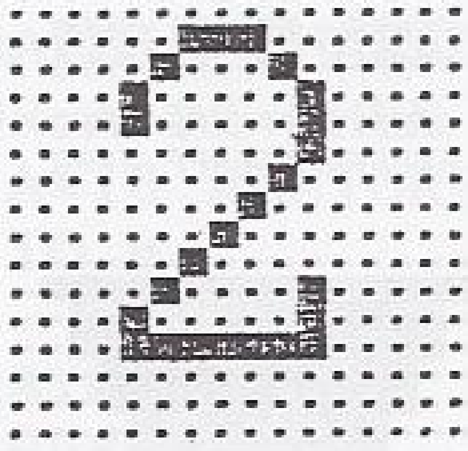
4-1-3 SAMPLE CHARACTERS

A program which produces the following 32 types of characters is presented as an example of the capabilities of the DEF CHR$ command:



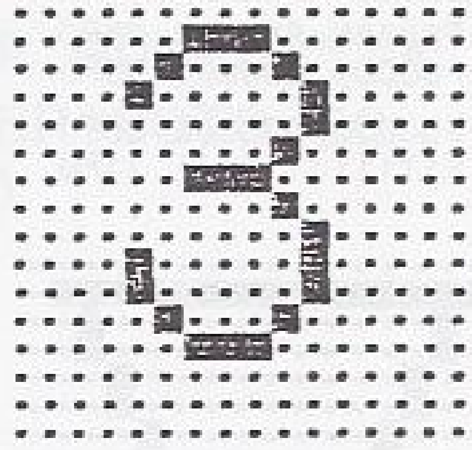
000001 270000 0000
000000 0F0000 0000

000000 0F0000 0000
000008 888800 0000



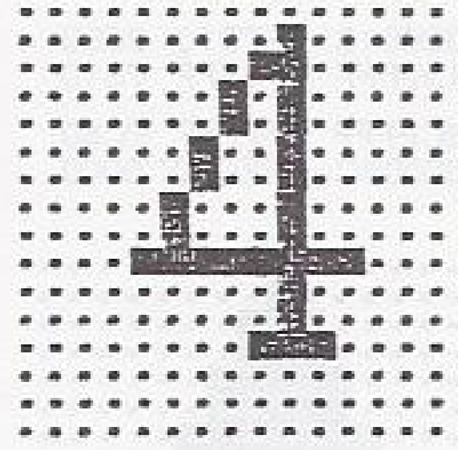
000012 444210 0000
000080 0012C0 0000

000012 480030 0000
000088 888880 0000



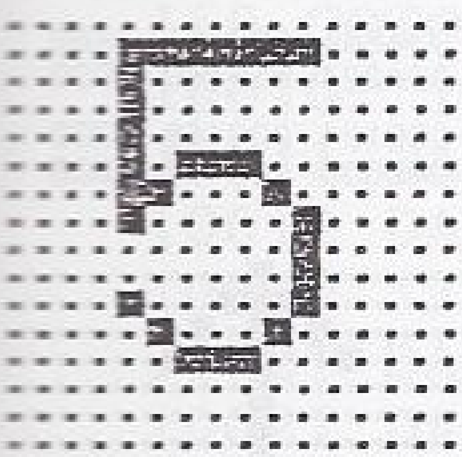
000012 444210 0000
000000 222580 0000

000061 0001E0 0000
000000 888000 0000



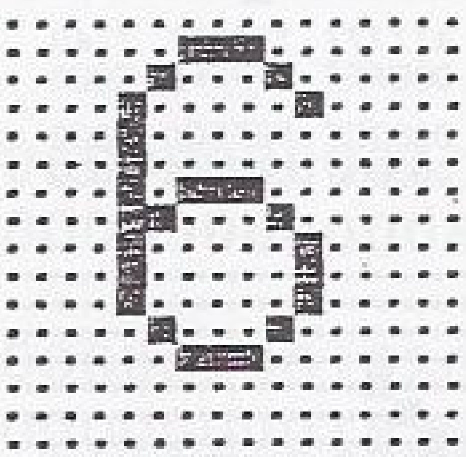
000000 012700 0000
000001 680F00 0000

00004C 444F44 0000
000000 008880 0000



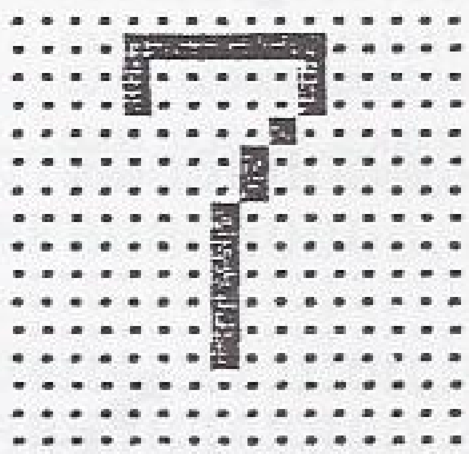
000074 444440 0000
0000F2 444210 0000

000021 0001E0 0000
000000 888000 0000



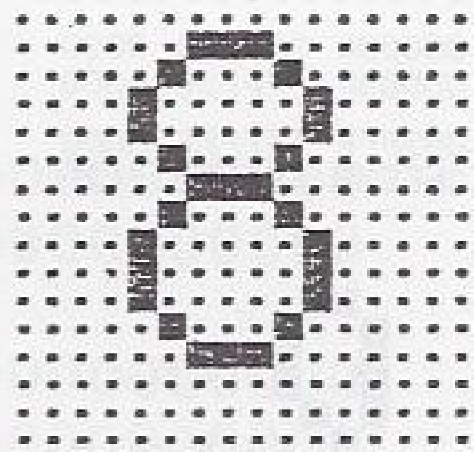
000012 444210 0000
0000F1 222100 0000

0000E1 0001E0 0000
000000 888000 0000



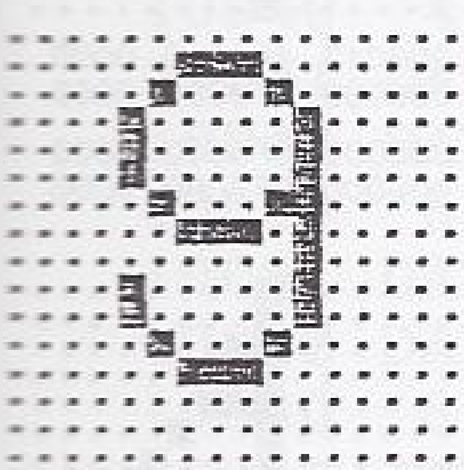
000074 444470 0000
000000 016800 0000

000000 0F0000 0000
000000 080000 0000



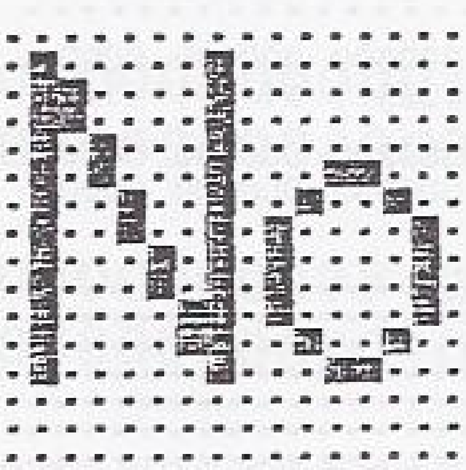
000012 444210 0000
000085 222580 0000

0000E1 0001E0 0000
000000 888000 0000



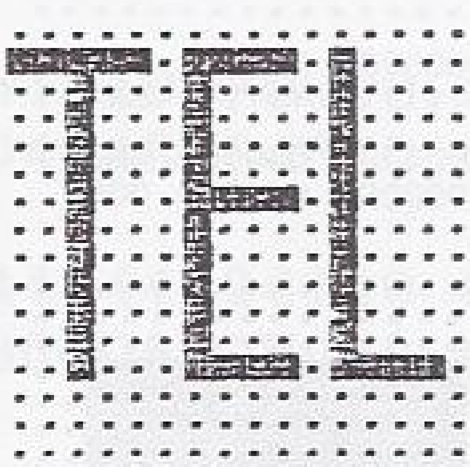
000012 444210 0000
0000C2 1112F0 0000

000061 0001E0 0000
000000 888000 0000



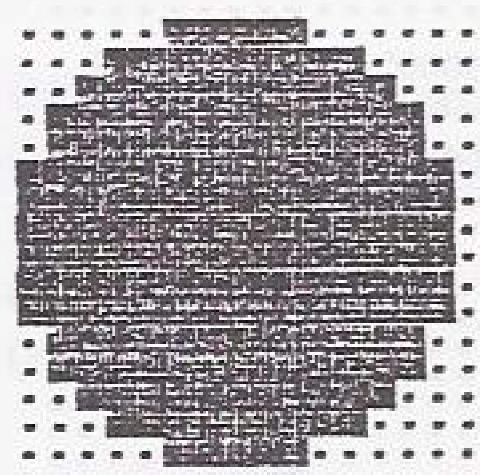
073000 070000 0000
0F0C30 0F0124 4210

0F000C 3F0E10 01E0
080000 080008 8000



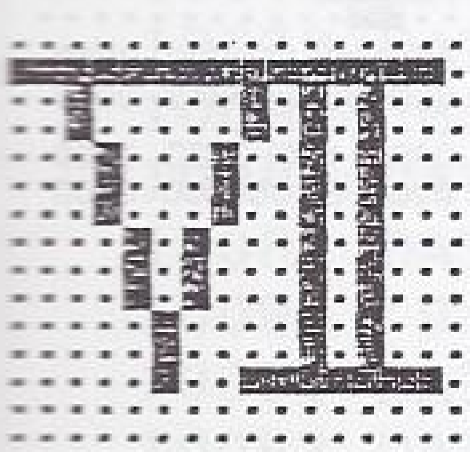
447440 744407 0000
00F000 F2220F 0000

00F000 F0000F 0000
008000 888808 8880



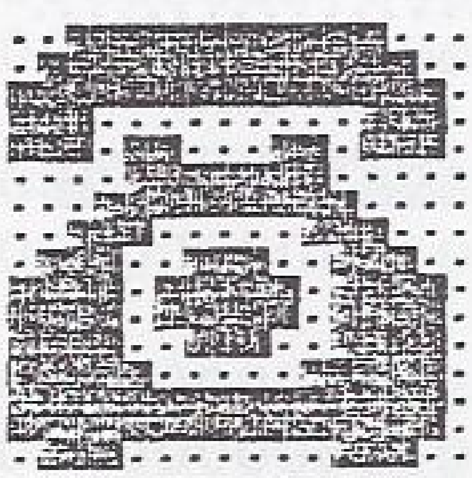
01377F FFFF77 3100
7FFFFF FFFFFF FF70

EFFFFF FFFFFF FFE0
08CEEF FFFFEE C800



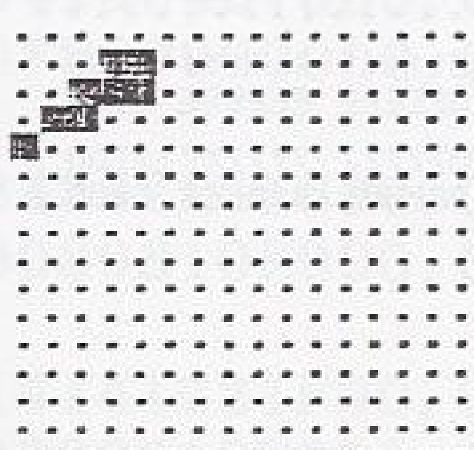
447444 447474 7440
000E10 1E00F0 F000

0000C3 C000F0 F000
000008 008888 8880



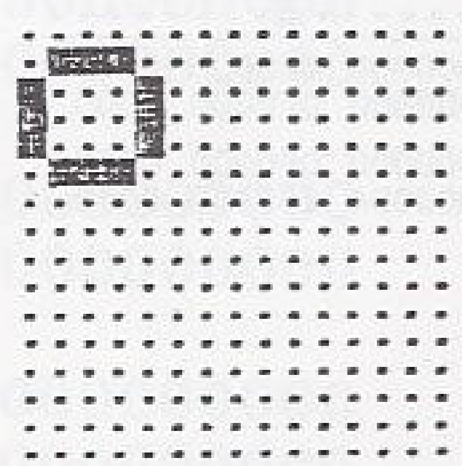
37FEEE EEEEEE F730
8893FE 666EF3 9880

7FFF06 FFF60F FF70
EFFFE6 6666EF FFE0



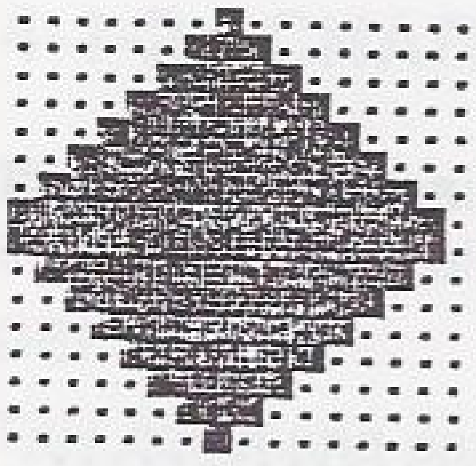
013660 000000 0000
800000 000000 0000

000000 000000 0000
000000 000000 0000

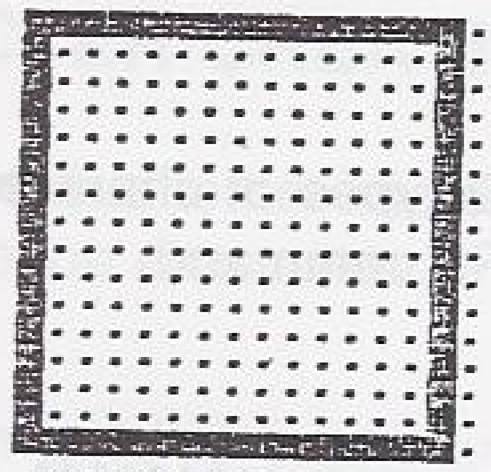


344430 000000 0000
844480 000000 0000

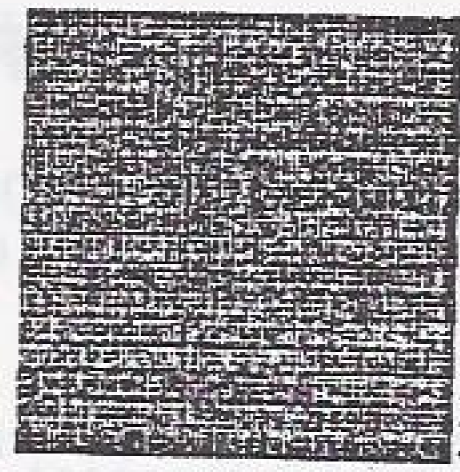
000000 000000 0000
000000 000000 0000



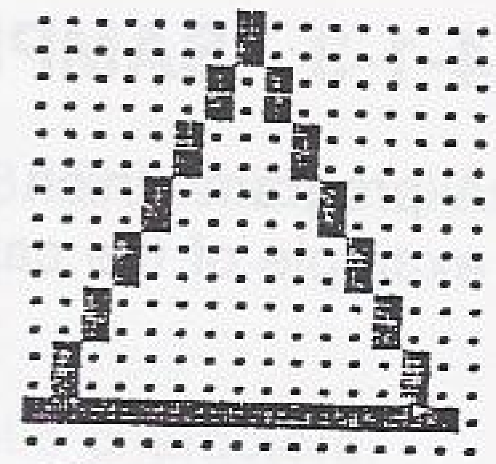
```
000013 7F7310 0000
137FFF FFFFFF 7310

8CEFFF FFFFFF EC80
00008C EFEC80 0000
```



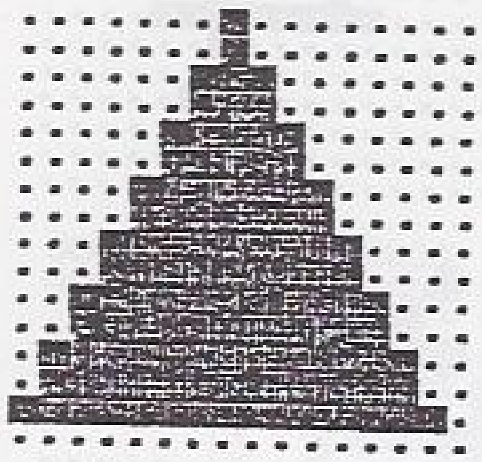
```
F88888 888888 88F0
F00000 000000 00F0

F00000 000000 00F0
F11111 111111 11F0
```



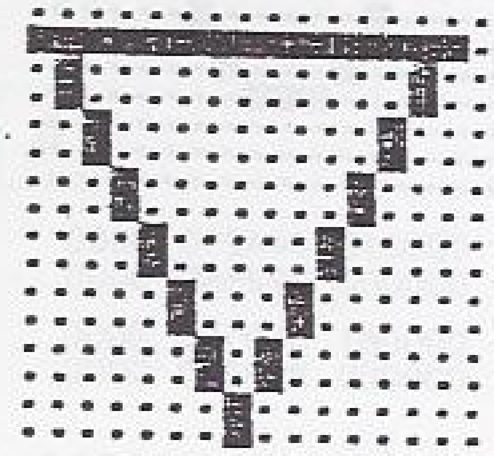
```
FFFFFF FFFFFF FFF0
FFFFFF FFFFFF FFF0

FFFFFF FFFFFF FFF0
FFFFFF FFFFFF FFF0
```



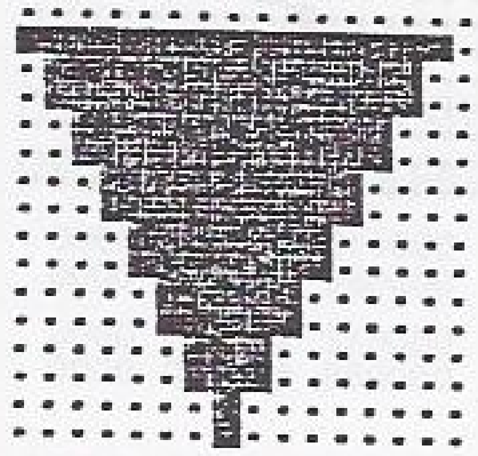
```
000000 3C3000 0000
00003C 000C30 0000

003C00 00000C 3000
2E2222 222222 2E20
```



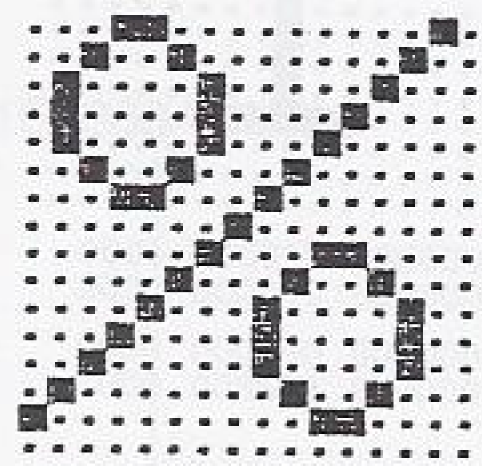
```
000000 3F3000 0000
00003F FFFF30 0000

003FFF FFFFFF 3000
2EEEEE EEEEEE EE20
```



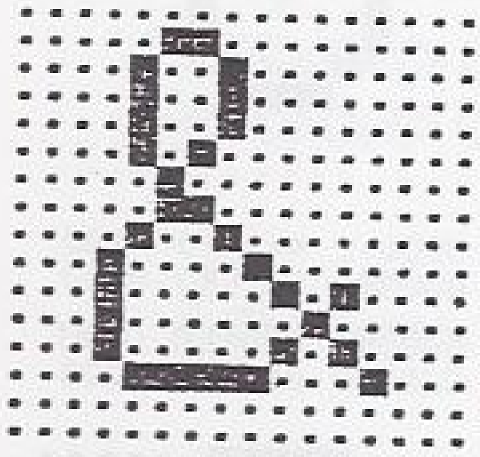
```
474444 444444 4740
00C300 000003 C000

0000C3 0003C0 0000
000000 C3C000 0000
```



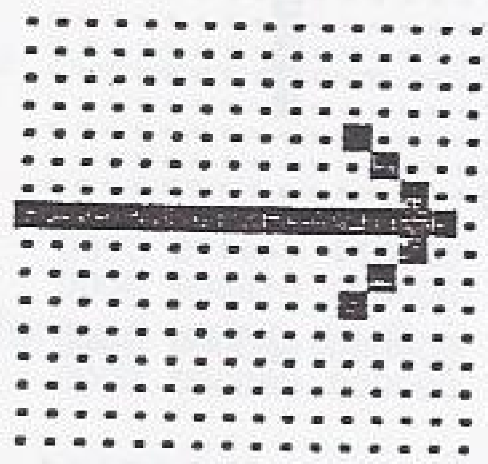
```
477777 777777 7740
00CFFF FFFFFF C000

0000CF FFFFC0 0000
000000 CFC000 0000
```



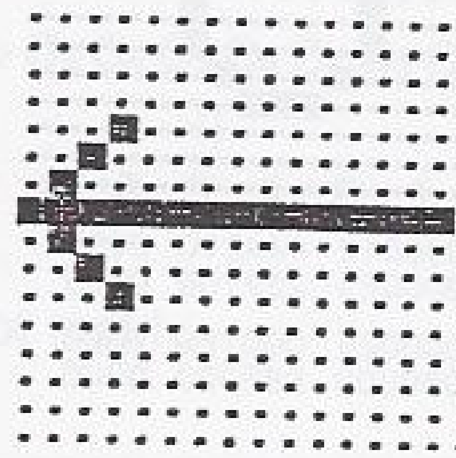
```
034884 300001 2480
084224 812480 0000

000124 803488 4300
248000 008422 4800
```



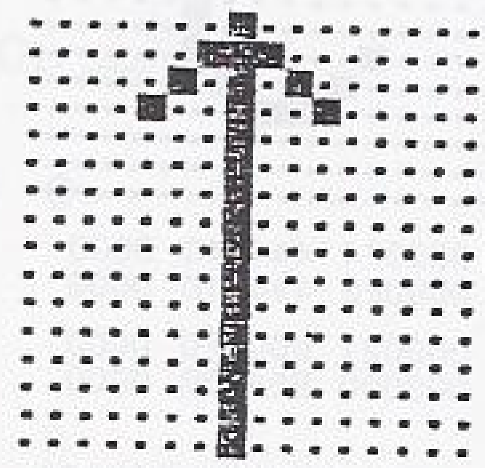
```
000034 430000 0000
0000C3 580000 0000

000780 084212 0000
000844 444808 4000
```



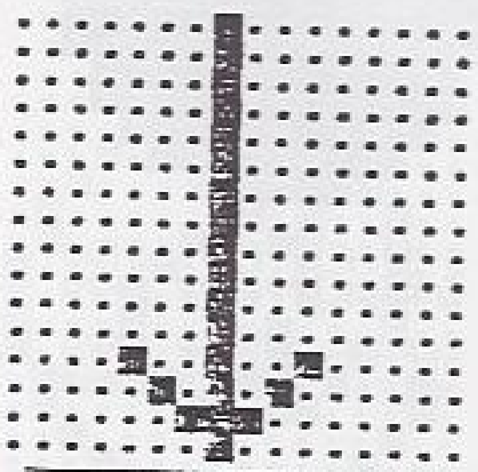
```
000000 000000 0000
111111 111119 5310

000000 000002 4800
000000 000000 0000
```



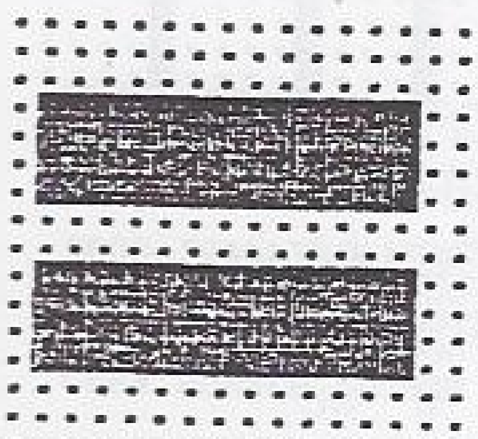
```
000000 000000 0000
135911 111111 1110

084200 000000 0000
000000 0C0000 0000
```



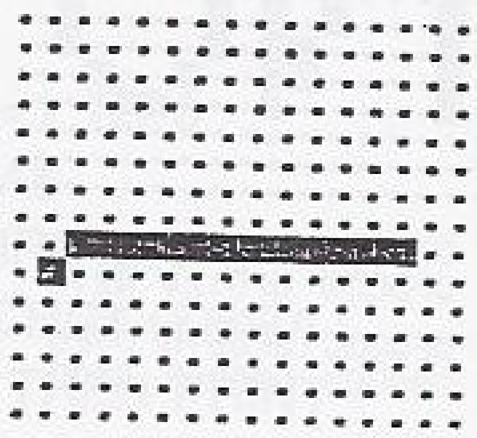
```
000012 4F4210 0000
000000 0F0000 0000

000000 0F0000 0000
000000 0F0000 0000
```



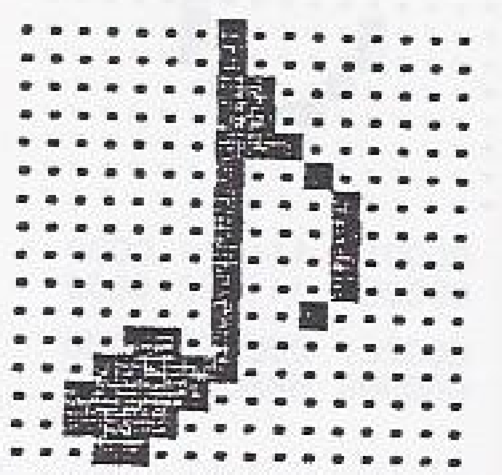
```
000000 0F0000 0000
000000 0F0000 0000

000000 0F0000 0000
000084 2F2480 0000
```



```
011111 111111 1100
0EEEEE EEEEEE EE00

077777 777777 7700
088888 888888 8800
```



```
000000 000000 0000
000000 000000 0000

048888 888888 8800
000000 000000 0000
```



```
000000 0F3000 0000
000000 0F8843 0000

000011 0F002C 0000
006FFE C80000 0000
```

CHARACTER DISPLAY PROGRAM

```
10 '
20 'DOT MATRIX GRAPHIC CHARACTERS
30 'GRAPHIC CHARACTER NUMBER
40 '
50 INPUT"GRAPHIC CHARACTER NUMBER";A
60 IF A<1 OR A>32 THEN BEEP:GOTO 50
70 A=10000+A*10:RESTORE(A)
80 FOR I=1 TO 6
90 READ A$
100 DEF CHR$(&HF0)=A$
110 PRINT CHR$(&HF0);
120 IF I=3 THEN PRINT
130 NEXT I
140 PRINT
150 GOTO 50
10010 '1
10011 DATA 000000000010,207F00000000,000000000000
10012 DATA 000000000008,08F808080000,000000000000
10020 '2
10021 DATA 000000001820,404041221C00,000000000000
10022 DATA 000000001828,488808083800,000000000000
10030 '3
10031 DATA 000000001020,424242251800,000000000000
10032 DATA 000000006010,08080810E000,000000000000
10040 '4
10041 DATA 000000000001,0618207F0000,000000000000
10042 DATA 0000000040C0,404048F84840,000000000000
10050 '5
10051 DATA 000000007F42,444444424100,000000000000
10052 DATA 000000002010,08080810E000,000000000000
10060 '6
10061 DATA 000000001F21,424242211000,000000000000
10062 DATA 00000000E010,08080810E000,000000000000
10070 '7
10071 DATA 000000007040,404146487000,000000000000
10072 DATA 000000000000,00F800000000,000000000000
10080 '8
10081 DATA 000000001825,424242251800,000000000000
10082 DATA 00000000E010,08080810E000,000000000000
10090 '9
10091 DATA 000000001C22,414141221F00,000000000000
10092 DATA 000000006010,08080810E000,000000000000
10100 'No
10101 DATA 007F300C0300,007F00010204,040201000000
10102 DATA 00F8000000C0,30F800E01008,0810E0000000
10110 'TEL
10111 DATA 40407F404000,7F424242007F,000000000000
10112 DATA 0000F8000000,F808080800F8,080808000000
10120 'FILLED CIRCLE
```

```
10121 DATA 071F3F7F7FFF,FFFFFFFF7F7F,3F1F07000000
10122 DATA E0F8FCFEFEFF,FFFFFFFFFEFE,FCF8E0000000
10130 'ROMAN 7
10131 DATA 4040704E4140,414E70407F40,7F4040000000
10132 DATA 00000000C038,C0000808F808,F80808000000
10140 'TELEPHONE MARK
10141 DATA 3878F9E3EFEE,E6E6E6EEEFE3,F97838000000
10142 DATA 7EFFFFFF0E66,F6F6F6660EFF,FFFF7E000000
10150 ' '
10151 DATA 081030606000,000000000000,000000000000
10152 DATA 000000000000,000000000000,000000000000
10160 '°
10161 DATA 384444443800,000000000000,000000000000
10162 DATA 000000000000,000000000000,000000000000
10170 'BLACK DIAMOND
10171 DATA 0103070F1F3F,7FFF7F3F1F0F,070301000000
10172 DATA 80C0E0F0F8FC,FEFFFEFCF8F0,E0C080000000
10180 'WHITE SQUARE
10181 DATA FF8080808080,808080808080,8080FF000000
10182 DATA FF0101010101,010101010101,0101FF000000
10190 'BLACK SQUARE
10191 DATA FFFFFFFFFFFF,FFFFFFFFFFFF,FFFFFF000000
10192 DATA FFFFFFFFFFFF,FFFFFFFFFFFF,FFFFFF000000
10200 'WHITE TRIANGLE
10201 DATA 00000000030C,30C0300C0300,000000000000
10202 DATA 020E32C20202,0202020202C2,320E02000000
10210 'BLACK TRIANGLE
10211 DATA 00000000030F,3FFF3F0F0300,000000000000
10212 DATA 020E3EFEFEFE,FEFEFEFEFEFE,3E0E02000000
10220 'INVERTED WHITE TRIANGLE
10221 DATA 40704C434040,404040404043,4C7040000000
10222 DATA 00000000C030,0C030C30C000,000000000000
10230 'INVERTED BLACK TRIANGLE
10231 DATA 40707C7F7F7F,7F7F7F7F7F7F,7C7040000000
10232 DATA 00000000C0F0,FCFFFCF0C000,000000000000
10240 '%
10241 DATA 003844828244,380102040810,204080000000
10242 DATA 020408102040,800038448282,443800000000
10250 '&
10251 DATA 000000003C43,453800000000,000000000000
10252 DATA 000000788404,048444281028,040000000000
10260 'RIGHT SYMBOL
10261 DATA 010101010101,010101010109,050301000000
10262 DATA 000000000000,000000000020,408000000000
10270 'LEFT SYMBOL
10271 DATA 010305090101,010101010101,010101000000
10272 DATA 008040200000,000000000000,000000000000
10280 'UP SYMBOL
10281 DATA 000000001020,40FF40201000,000000000000
10282 DATA 000000000000,00FF00000000,000000000000
10290 'DOWN SYMBOL
10291 DATA 000000000000,00FF00000000,000000000000
10292 DATA 000000000804,02FF02040800,000000000000
```

```
10300 'EQUAL SIGN
10301 DATA 001E1E1E1E1E,1E1E1E1E1E1E,1E1E00000000
10302 DATA 007878787878,787878787878,787800000000
10310 'DASH
10311 DATA 000000000000,000000000000,000000000000
10312 DATA 004080808080,808080808080,808000000000
10320 'MUSICAL NOTE
10321 DATA 000000000000,00FF38080403,000000000000
10322 DATA 0000060F1F1E,0CF8000020C0,000000000000
```

Execution Example

```
RUN
GRAPHIC CHARACTER NUMBER?32
```

4-2 BASIC COMMAND TABLE

• **Commands**

· CLEAR	©	· SYSTEM	©	· LIST	Ⓜ
· VARLIST	©	· EDIT	Ⓜ	· DELETE	Ⓜ
· RUN	Ⓜ	· TRON/TROFF	©	· END	
· STOP		· GOTO		· GOSUB/RETURN	
· ON GOTO		· ON GOSUB		· IF/THEN/ELSE	
· FOR/NEXT		· REM(')		· LET	©
· DATA/READ/RESTORE		· INPUT		· PRINT	©
· PRINT USING	©	· LOCATE	©	· ANGLE	©
· BEEP(ON/OFF)	©	· CLS	©	· DIM	©
· ERASE	©	· DRAW/DRAWC	©	· MON	©
· CALL	©	· ON ERROR GOTO		· RESUME	
· DEFCHR$	©	· PASS	Ⓜ	· NEW	Ⓜ
· STAT	©	· STAT CLEAR	©	· POKE	©

• **Input/Output commands**

· LLIST	Ⓜ	· LPRINT	©	· LPRINT USING	©
· FORMAT	©	· BSAVE	©	· BLOAD	©
· OPEN		· CLOSE	©	· PRINT#	
· INPUT#		· SAVE	Ⓜ	· LOAD	Ⓜ
· PUT/GET		· FIELD		· RSET/LSET	
· VERIFY	©	· CHAIN	Ⓜ	· MERGE	Ⓜ
· LINEINPUT#		· PRINT# USING			

Ⓜ : manual execution only

© : manual or CAL mode execution

▪ Functions

· CHR $	· ASC	· STR $
· VAL	· MID $	· RIGHT $
· LFFT $	· LEN	· HEX $
· &H	· INKEY $	· INPUT $
· INPUT #	· DEG	· DMS $
· POINT		
· SIN	· COS	· TAN
· ASN	· ACS	· ATN
· HYP SIN	· HYP COS	· HYP TAN
· HYP ASN	· HYP ACS	· HYP ATN
· EXP	· LOG	· LGT
· SQR	· ABS	· SGN
· INT	· FRAC	· ROUND
· PI	· RND	· PEEK
· TAB	· FIX	
· CNT	· SUMX	· SUMY
· SUMXY	· SUMX2	· SUMY2
· MEANX	· MEANY	· SDX
· SDY	· SDXN	· SDYN
· LRA	· LRB	· COR
· EOX	· EOY	
· EOF	· ERR	· ERL
· LOF	· REV	· NORM
· TIME $	· DATE $	

CHAPTER 5

ASSEMBLER

5-1 USING THE BUILT-IN ASSEMBLER

The easy-to-use assembler built into the PB-1000 makes it possible to develop simple machine language programs after only a little practice. This assembler contains only a minimal number of essential functions, making it different from the disk base assembler usually found in other personal computers. This chapter contains points which should be observed when creating a source program, based on actual examples. Refer to the owner's manual for procedures on editing the source program.

5-1-1 LABELS

Labels can be used when creating a source program with the PB-1000, subjected to the following restrictions:

- Label names may be up to 5 characters long and may include alphabetic characters, numbers and the symbols @ and _ (underline). The first character of a label name must be an alphabetic character (either upper case or lower case).
- A label name is declared by a colon following the name. As any spaces between the last character of the label name and the colon are included as part of the label name, an error is generated when the label name is later referenced.
- Using the EQU command to declare labels for CALL destinations which are ROM routines makes programs easier to read.
- Labels can be freely assigned, but only JP, JR, and CAL commands may be used as operands with label names. Therefore, usage is rather limited when compared to general use label names. Details on general use label names are given in a following section.

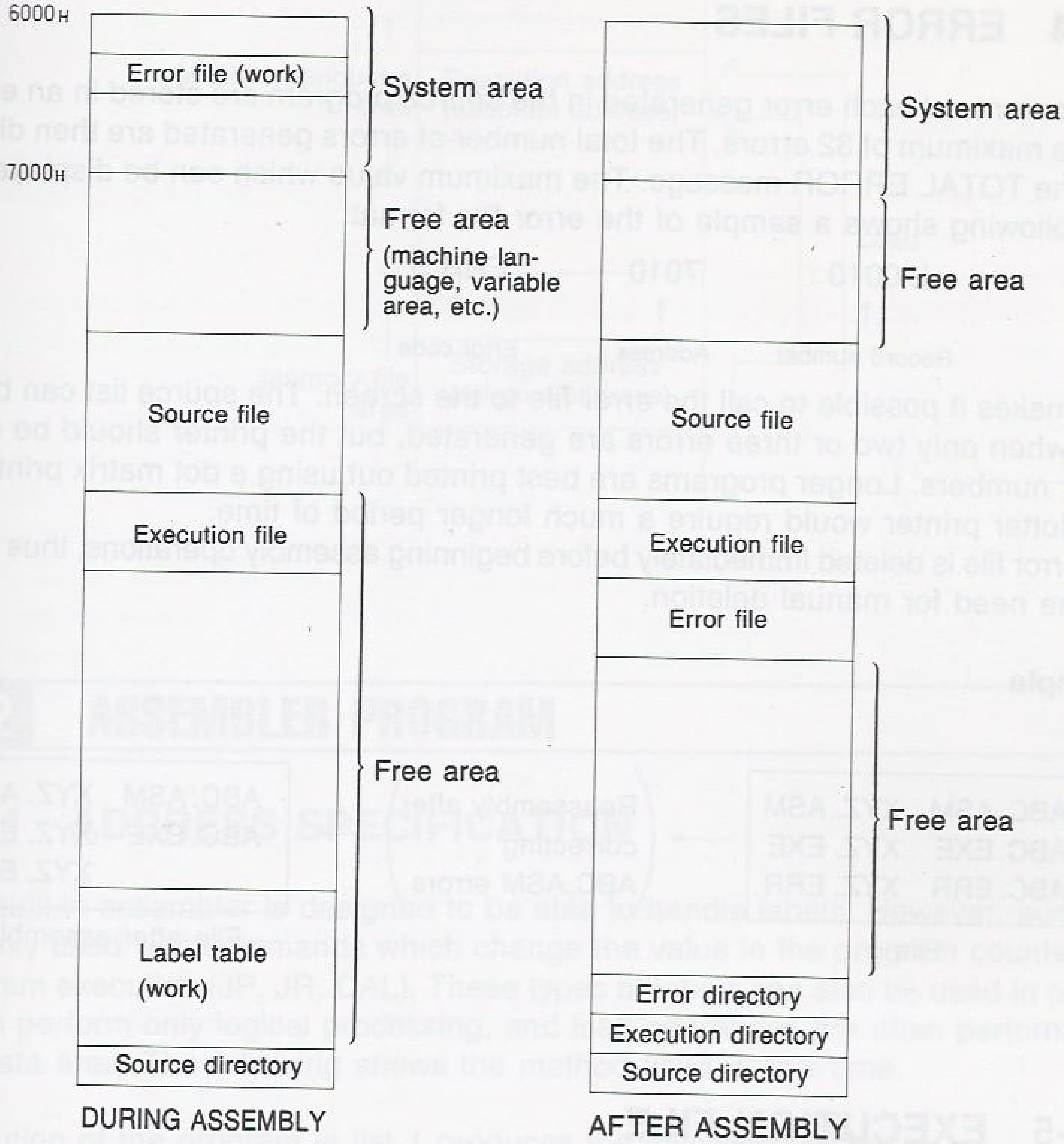
5-1-2 MACHINE LANGUAGE AREA

7000_H through $7FFE_H$ are assigned as the area for the loading of machine language, and this area is subjected to a variety of restrictions.

- Machine language may be loaded anywhere within the area 7000_H ~ $7FFE_H$ (4,095 bytes), but an attempt to load machine language anywhere else results in an OM error.
- The use of memory for the source program and object program is complex, and so the memory map should be carefully referenced.

5-1-3 ASSEMBLER MEMORY MAP

The built-in assembler performs assembly only in the built-in RAM. Therefore, assembly cannot be performed when RAM area sufficient for creation of the work file is not available. The following shows how RAM is used by the assembler:



The following three files are created during execution of the assembler:

- Execution file
- Label table (deleted after assembly is complete)
- Error file (transferred to file area after assembly)

Therefore, the capacity of the unit must carefully be taken into consideration when working with long assembler programs containing a large number of label names, and large data data area. Most problems concerning insufficient memory capacity can be solved by using an optional RP-32 RAM expansion pack to increase memory by 32KB.

5-1-4 ERROR FILES

The contents of each error generated in the source program are stored in an error file, up to a maximum of 32 errors. The total number of errors generated are then displayed with the TOTAL ERROR message. The maximum value which can be displayed is 99. The following shows a sample of the error file format:

```
    L 0010 :          7010          ERR 5
       ↑                ↑              ↑
 Record number       Address      Error code
```

This makes it possible to call the error file to the screen. The source list can be easily read when only two or three errors are generated, but the printer should be used for larger numbers. Longer programs are best printed out using a dot matrix printer since the plotter printer would require a much longer period of time.
The error file is deleted immediately before beginning assembly operations, thus eliminating the need for manual deletion.

Example

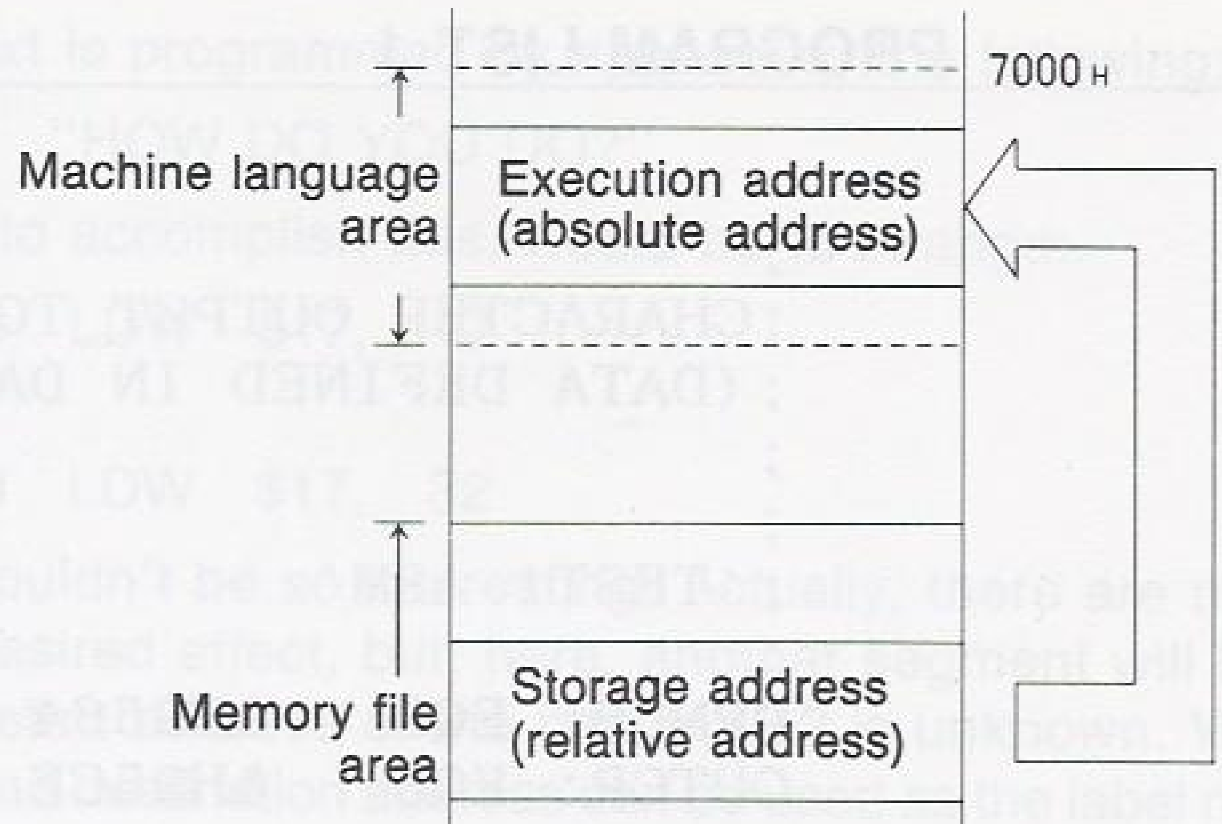
```
ABC.ASM  XYZ.ASM        Reassembly after        ABC.ASM  XYZ.ASM
ABC.EXE  XYZ.EXE  ──→   correcting        ──→   ABC.EXE  XYZ.EXE
ABC.ERR  XYZ.ERR        ABC.ASM errors                   XYZ.ERR

      File                                   File after assembly
```

5-1-5 EXECUTION FILE

Once assembly is successful, the execution file (object program) is stored in the same free area as the source program. The following procedures are performed when the object program is executed:

1) The execution file stored in the free area is transferred to the machine language area at 7000H ~ 7FFEH. At this time, the destination address becomes the same as that specified by ORG. Multiple ORG commands cause the file to be transferred one address at a time.
2) Program execution is performed from the address specified by the START command.

All programs or data previously existing in the machine language area are lost when the transfer takes place, so important data should be moved to another area before execution of the new program. Changing the contents of a program in the machine language area (using the BASIC POKE command) does not affect the program in the free area. Also note that execution files are deleted immediately before creation of the object file, just as error files.



5-2 ASSEMBLER PROGRAM

5-2-1 ADDRESS SPECIFICATION

The built-in assembler is designed to be able to handle labels. However, such labels are only used with commands which change the value in the program counter during program execution (JP, JR, CAL). These types of labels can also be used in programs which perform only logical processing, and load operations are often performed from the data area. The following shows the method used at this time.

Execution of the program in list 1 produces the following display:

```
I AM THE PB-1000!
```

The following example will show how to modify this program to produce another display:

```
I AM THE PB-1000!
HOW DO YOU DO?
```

PROGRAM LIST 1

```
0001:0000                    ;
0002:0000                    ;CHARACTER OUTPUT TO DISPLAY
0003:0000                    ;(DATA DEFINED IN DATA AREA)
0004:0000                    ;
0005:0000                    ;
0006:0000                    ;'TEST1.ASM'
0007:0000                    ;
0008:0000                    PRNLB:  EQU    &H9664
0009:0000                    OUTCR:  EQU    &H95CE
0010:0000                    ;
0011:0000                    OUTDV:  EQU    &H690C
0012:0000                    ;
0013:7000                            ORG    &H7000
0014:7000                            START  LCD
0015:7000                    ;
0016:7000                    ;SET LCD AS OUTPUT AREA.
0017:7000                    ;
0018:7000   D10F0C69         LCD:    LDW    $15,&H690C
0019:7004   421100                   LD     $17,&H00
0020:7007   10710F                   ST     $17,($15)
0021:700A                    ;
0022:700A                    ;SET DATA(MSG1) START ADDRESS.
0023:700A                    ;OUTPUT OF 17 CHARACTERS
0024:700A                    ;
0025:700A   D10F1970                 LDW    $15,&H7019
0026:700E   D1111100                 LDW    $17,17
0027:7012   776496                   CAL    PRNLB
0028:7015                    ;
0029:7015                    ;CR/LF
0030:7015                    ;
0031:7015   77CE95                   CAL    OUTCR
0032:7018   F7                       RTN
0033:7019                    ;
0034:7019                    ;TEST DATA
0035:7019                    ;
0036:7019   4920414D         MSG1:   DB     "I AM "
            20
0037:701E   54484520                 DB     "THE PB-1000."
            50422D31
            3030302E
0038:702A                    ;
```

The additional text is programmed by appending the following:

MSG 2: DB 13, "HOW DO YOU DO?"

The easiest way to accomplish this, would be to change:

700E D1111100 LDW $17, 17

to the following:

700E D1112000 LDW $17, 32

This, however, wouldn't be so interesting. Actually, there are many different ways to accomplish the desired effect, but, here, another segment will be created which outputs MSG1. The start address of the new MSG2 is unknown. With high-performance assemblers, the load destination address can be used as the label name. However, built-in assemblers require that the load destination and store destination addresses normally be absolute addresses. Therefore, the address is unknown until the program is assembled. Because of this, it is suggested that all load destination addresses be set to 0000_H in such cases. 0000_H is used as a dummy address during assembly. The address 0000_H has no special significance other than it is easier to spot within a program list than other addresses. Any other address (such as 8888_H or $FFFF_H$) could be used with no problem at all.

Program list 2 shows the resulting program once the new lines are added. It is evident that the address of MSG2 is 7035_H, and that the dummy address (0000_H has been changed accordingly (7035_H)). Also note that the address of MSG1 has been changed from 7019_H to 7024_H, necessitating a change in the addresses referencing MSG1. The completed program is shown in program list 3.

PROGRAM LIST 2

```
0001:0000           ;
0002:0000           ;CHARACTER OUTPUT TO DISPLAY
0003:0000           ;(DATA DEFINED IN DATA AREA)
0004:0000           ;
0005:0000           ;
0006:0000           ;'TEST2.ASM'
0007:0000           ;
0008:0000           PRNLB:  EQU     &H9664
0009:0000           OUTCR:  EQU     &H95CE
0010:0000           ;
0011:0000           OUTDV:  EQU     &H690C
0012:0000           ;
0013:7000                   ORG     &H7000
0014:7000                   START LCD
0015:7000           ;
0016:7000           ;SET LCD AS OUTPUT AREA.
0017:7000           ;
0018:7000 D10F0C69  LCD:    LDW     $15,&H690C
0019:7004 421100            LD      $17,&H00
0020:7007 10710F            ST      $17,($15)
0021:700A           ;
0022:700A           ;SET DATA(MSG1) START ADDRESS.
0023:700A           ;OUTPUT OF 17 CHARACTERS.
```

```
0024:700A                     ;
0025:700A  D10F1970                  LDW    $15,&H7019
0026:700E  D1111100                  LDW    $17,17
0027:7012  776496                    CAL    PRNLB
0028:7015                     ;
0029:7015                     ;SET DATA(MSG2) START ADDRESS.
0030:7015                     ;OUTPUT OF 15 CHARACTERS.
0031:7015                     ;
0032:7015  D10F0000                  LDW    $15,&H0000
0033:7019  D1110F00                  LDW    $17,15
0034:701D  776496                    CAL    PRNLB
0035:7020                     ;
0036:7020                     ;CR/LF
0037:7020                     ;
0038:7020  77CE95                    CAL    OUTCR
0039:7023  F7                        RTN
0040:7024                     ;
0041:7024                     ;TEST DATA
0042:7024                     ;
0043:7024  4920414D       MSG1:   DB      "I AM "
           20
0044:7029  54484520               DB      "THE PB-1000."
           50422D31
           3030302E
0045:7035                     ;
0046:7035  0D484F57       MSG2:   DB      13,"HOW DO YOU DO?"
           20444F20
           594F5520
           444F3F
0047:7044                     ;
```

PROGRAM LIST 3

```
0001:0000
0002:0000                     ;CHARACTER OUTPUT TO DISPLAY
0003:0000                     ;(DATA DEFINED IN DATA AREA)
0004:0000                     ;
0005:0000                     ;
0006:0000                     ;'TEST3.ASM'
0007:0000                     ;
0008:0000                 PRNLB:  EQU    &H9664
0009:0000                 OUTCR:  EQU    &H95CE
0010:0000                     ;
0011:0000                 OUTDV:  EQU    &H690C
0012:0000                     ;
0013:7000                         ORG    &H7000
0014:7000                         START LCD
0015:7000                     ;
0016:7000                     ;SET LCD AS OUTPUT AREA.
0017:7000                     ;
0018:7000  D10F0C69       LCD:    LDW    $15,&H690C
```

```
0019:7004  421100           LD    $17,&H00
0020:7007  10710F           ST    $17,($15)
0021:700A            ;
0022:700A            ;SET DATA(MSG1) START ADDRESS.
0023:700A            ;OUTPUT OF 17 CHARACTERS.
0024:700A            ;
0025:700A  D10F2470         LDW   $15,&H7024
0026:700E  D1111100         LDW   $17,17
0027:7012  776496           CAL   PRNLB
0028:7015            ;
0029:7015            ;SET DATA(MSG2) START ADDRESS.
0030:7015            ;OUTPUT OF 15 CHARACTERS.
0031:7015            ;
0032:7015  D10F3570         LDW   $15,&H7035
0033:7019  D1110F00         LDW   $17,15
0034:701D  776496           CAL   PRNLB
0035:7020            ;
0036:7020            ;CR/LF
0037:7020            ;
0038:7020  77CE95           CAL   OUTCR
0039:7023  F7               RTN
0040:7024            ;
0041:7024            ;TEST DATA
0042:7024            ;
0043:7024  4920414D  MSG1:  DB    "I AM "
           20
0044:7029  54485220         DB    "THE PB-1000."
           50422D31
           3030302E
0045:7035            ;
0046:7035  0D484F57  MSG2:  DB    13,"HOW DO YOU DO?"
           20444F20
           594F5520
           444F3F
0047:7044            ;
```

5-2-2 ERROR FREE PROGRAMS

Machine language programs must be in the area within 7000H through 7FFEH. During the assembly process, however, errors are not generated no matter what the address. Program list 4 shows a program in which only the data section is specified from the free area 9000H. This program is successfully assembled without generating an error. An OM error is generated, however, when the program is actually executed, and load cannot be performed because data falls outside of the data area. This example shows a special case, but in actual applications, programs starting in the vicinity of 7F00H may surpass 7FFEH resulting in an error.

PROGRAM LIST 4

```
0001:0000                    ;
0002:0000                    ;CHARACTER OUTPUT TO DISPLAY
0003:0000                    ;(DATA DEFINED IN DATA AREA)
0004:0000                    ;
0005:0000                    ;
0006:0000                    ;'TEST4.ASM'
0007:0000                    ;
0008:0000                    PRNLB:  EQU    &H9664
0009:0000                    OUTCR:  EQU    &H95CE
0010:0000                    ;
0011:0000                    OUTDV:  EQU    &H690C
0012:0000                    ;
0013:7000                            ORG    &H7000
0014:7000                            START  LCD
0015:7000                    ;
0016:7000                    ;SET LCD AS OUTPUT AREA.
0017:7000                    ;
0018:7000   D10F0C69         LCD:    LDW    $15,&H690C
0019:7004   421100                   LD     $17,&H00
0020:7007   10710F                   ST     $17,($15)
0021:700A                    ;
0022:700A                    ;SET DATA(MSG1) START ADDRESS.
0023:700A                    ;OUTPUT OF 17 CHARACTERS
0024:700A                    ;
0025:700A   D10F0090                 LDW    $15,&H9000
0026:700E   D1111100                 LDW    $17,17
0027:7012   776496                   CAL    PRNLB
0028:7015                    ;
0029:7015                    ;CR/LF
0030:7015                    ;
0031:7015   77CE95                   CAL    OUTCR
0032:7018   F7                       RTN
0033:7019                    ;
0034:7019                    ;TEST DATA
0035:7019                    ;
0036:9000                            ORG    &H9000
0037:9000                    ;
0038:9000   4920414D         MSG1:   DB     "I AM "
            20
0039:9005   54484520                 DB     "THE PB-1000."
            50422D31
            3030302E
0040:9011                    ;
```

CHARACTER DISPLAY PROGRAM (APPLICATION PROGRAM)

```
0001:0000                    ;
0002:0000                    ;CHARACTER OUTPUT TO DISPLAY
0003:0000                    ;(DATA DEFINED IN DATA AREA)
0004:0000                    ;(APPLICATION PROGRAM)
0005:0000                    ;
0006:0000                    ;'TEST5.ASM'
0007:0000                    ;
0008:0000                    PRNLB: EQU   &H9664
0009:0000                    OUTCR: EQU   &H95CE
0010:0000                    ;
0011:0000                    OUTDV: EQU   &H690C
0012:0000                    ;
0013:7000                           ORG   &H7000
0014:7000                           START LCD
0015:7000                    ;
0016:7000                    ;SET LCD AS OUTPUT AREA.
0017:7000                    ;
0018:7000  D10F0C69          LCD:   LDW   $15,&H690C
0019:7004  421100                   LD    $17,&H00
0020:7007  10710F                   ST    $17,($15)
0021:700A                    ;
0022:700A                    ;SET DATA(MSG1) START ADDRESS.
0023:700A                    ;OUTPUT OF 32 CHARACTERS
0024:700A                    ;
0025:700A  D10F1970                 LDW   $15,&H7019
0026:700E  D1112000                 LDW   $17,32
0027:7012  776496                   CAL   PRNLB
0028:7015                    ;
0029:7015                    ;CR/LF
0030:7015                    ;
0031:7015  77CE95                   CAL   OUTCR
0032:7018  F7                       RTN
0033:7019                    ;
0034:7019                    ;TEST DATA
0035:7019                    ;
0036:7019  4920414D          MSG1:  DB    "I AM "
           20
0037:701E  54484520                 DB    "THE PB-1000."
           50422D31
           3030302E
0038:702A  0D                       DB    &H0D
0039:702B  484F5720                 DB    "HOW DO YOU DO?"
           444F2059
           4F552044
           4F3F
0040:7039                    ;
```

5-3 ASSEMBLER FORMAT AND PSEUDO-INSTRUCTIONS

5-3-1 ASSEMBLER FORMAT

Assembler programs are written according to the following format:

△<filename> : △ [mnemonic] ⊔ <operand 1> △, △ <operand 2> △ ; <comment>

1) Symbols

△	Spaces which may be omitted if desired.
⊔	One or more spaces.
:	Colon required immediately following a label. Assembly will not be successful if a space is inserted.
,	Commas are required to delimit operands.
;	Comments are written immediately following a semicolon. Anything following a semicolon is disregarded during assembly.
< >	Contents may be included or omitted as required.
[ ]	Contents must be included.

2) Label Names

Label names may be up to five characters long, and must begin with an alphabetic character (A ~ Z, a ~ z). Any alphanumeric character, @ or — (underline) may be used for the other four characters.

Labels can be stored in another format using the EQU command.

The number of labels stored is only limited by the amount of memory capacity available.

3) Mnemonics

Details of mnemonics can be found in the Chapter 13. Mnemonics are followed by a space or return.

4) Operands

Multiple operations are separated by commas. The following operand symbols have special significance:

$0 ~ $31 ($&H0 ~ $&H1F)   Main register addresses

&H   Following characters are hexadecimal

All other values (not preceded by $ or &H) are handled as decimal values.

5) Comments

All characters following a semicolon to the end of the line are treated as a comment. Comments are disregarded during assembly.

5-3-2 PSEUDO-INSTRUCTIONS

ORG

Format: ORG⊔[address]

Purpose: Specifies the object program start address. Multiple occurrences may be present in a single program, but a newly specified address must be larger than the address specified by the ORG command immediately preceding. An OR error is generated and assembly is terminated when an erroneous ORG specification is made.

START

Format: START⊔[address or filename]

Purpose: Specifies the object program execution start address. Normally, the address specified by this command is the same as that specified by the ORG command.

EQU

Format: [label name]⊔EQU⊔[address or numeric value]

Purpose: Gives a numeric value to a label.

DS

Format: <label name:> ΔDS⊔[numeric value]

Purpose: Maintains memory from the assembled address up to the number of bytes specified by the numeric value. The numeric value may be specified using either a decimal or hexadecimal value.

DB

Format: <label name:> ΔDB⊔[numeric value or string]

Purpose: Outputs the ASCII code of the numeric value or string as object code from the assembled address. Strings must be enclosed in quotation marks, and multiple numeric values or strings are separated by commas.

5-3-3 ASSEMBLER ERRORS

The following are errors which may be generated during assembly:

ERR 1 . . . . . . Double definition of label
ERR 2 . . . . . . Assembler system error
ERR 3 . . . . . . Operand label syntax error
ERR 4 . . . . . . Mnemonic syntax error
ERR 5 . . . . . . Labeling of command which cannot be labeled
OR error . . . . Erroneous ORG specification
OM error . . . . Insufficient memory for assembly

• Object program execution error
OM error . . . . Address set outside of machine language area

CHAPTER 6

KEYBOARD

6-1 KEYBOARD SPECIFICATIONS

The most outstanding feature of the PB-1000 keyboard is its touch keys which uses a sensor function built into a liquid crystal display. Just as standard keyboard keys, each touch key is assigned its own individual key code, meaning that they can be sensed during BASIC programs.
The cursor UP/DOWN keys have a scroll function which allows full use of the virtual screen (8 lines × 32 columns).

6-2 USING THE KEYBOARD IN BASIC

The following two programs are fundamental routines which allow the use of touch keys and cursor keys. These programs can be used to create easy-to-read displays.

TOUCH : Pressing a touch key causes the key code for that key to appear in reverse field.

SCROLL : Pressing the UP/DOWN cursor keys causes the display to scroll in the respective direction.

```
10 '
20 'TOUCH KEY
30 '
40 '  "TOUCH."
50 '
60 CLS:BEEP OFF
70 A$=INKEY$:IF A$="" THEN 70
80 KY=ASC(A$)
90 IF KY<240 THEN 70
100 K=KY-240
110 Y=INT(K/4):X=(K-Y*4)*8
120 M$="  "+STR$(KY)+"  "
130 LOCATE X,Y:PRINT REV;M$;
140 FOR I=1 TO 100:NEXT I
150 LOCATE X,Y:PRINT NORM;"          ";
160 GOTO 70
```

```
10 '
20 'PROGRAMMED CONTROL OF
30 'DISPLAY SCROLL
40 '
50 ' "SCROLL."
60 '
70 CLS
80 PRINT 1;"THIS IS THE 1ST LINE."
90 PRINT 2;"THIS IS THE 2ND LINE."
100 PRINT 3;"●●●●SCROLL●●●●"
110 PRINT 4;"♣♣♣♣SAMPLE♣♣♣♣"
120 PRINT 5;"♥♥♥♥PROGRAM♥♥♥♥"
130 PRINT 6;"THIS IS THE 6TH LINE."
140 PRINT 7;"♦♦♦SCROLL SAMPLE♦♦♦"
150 PRINT 8;"THIS IS THE FINAL LINE.";
160 A$=INKEY$:IF A$="" THEN 160
170 ' SCROLL DOWN
180 IF A$=CHR$(31) THEN PRINT CHR$(1);:GOTO 160
190 ' SCROLL UP
200 IF A$=CHR$(30) THEN PRINT CHR$(16);:GOTO 160
210 GOTO 160
```

6-3 USING THE KEYBOARD WITH THE ASSEMBLER

This program causes a beep to sound with each key pressed, and the character for the key pressed to be displayed. The EXE key is pressed to terminate the program.

INKEY. ASM

```
0001:0000                 ;
0002:0000                 ;SINGLE CHARACTER KEYBOARD INPUT
0003:0000                 ;SAME AS INKEY$.
0004:0000                 ;
0005:0000                 ; "INKEY.ASM"
0006:0000                 ;
0007:0000                 INKEY:  EQU    &H9E3B
0008:0000                 BEEP:   EQU    &HFFE5
0009:0000                 PRINT:  EQU    &H9664
0010:0000                 ;
0011:7000                         ORG    &H7000
0012:7000                         START  KEY
0013:7000                 ;
0014:7000   773B9E        KEY:    CAL    INKEY
0015:7003                 ;
0016:7003                 ;WAIT FOR KEY INPUT
0017:7003                 ;
0018:7003   491100                SB     $17,0
0019:7006   B087                  JR     Z,KEY
```

```
7008                 ;
7008                 ;EXE KEY TO QUIT
7008                 ;
7008   11630F              LD    $03,($15)
700B   49030D              SB    $03,13
700E   B01C                JR    Z,EXIT
7010                 ;
7010                 ;KEY INPUT CONFIRMATION TONE
7010                 ;
7010   77E5FF              CAL   BEEP
7013                 ;
7013                 ;DISPLAY OF KEY INPUT CHARACTER
7013                 ;
7013   421200              LD    $18,0
7016   776496              CAL   PRINT
7019                 ;
7019                 ;KEY INPUT TIMING
7019                 ;
7019   420110              LD    $01,&H10
701C   4202FF        TIM1: LD    $02,&HFF
701F                 ;
701F   490201        TIM2: SB    $02,1
7022   B484                JR    NZ,TIM2
7024                 ;
7024   490101              SB    $01,1
7027   B48C                JR    NZ,TIM1
7029                 ;
7029                 ;RETURN TO KEY INPUT
7029                 ;
7029   B7AA                JR    KEY
702B                 ;
702B                 ;RETURN TO SYSTEM
702B                 ;
702B   F7            EXIT: RTN
```

6-4 KEY MODE CHANGE

Upper case/Lower case characters are generally selected from the keyboard. However, the program can also specify upper/lower case, making keyboard specification unnecessary. Character specification can be accomplished by changing the value at address 68D7H.

- 68D7H Bit 3
  - 1 : Lower case characters
  - 0 : Upper case characters

BASIC control example

POKE &H68D7, 0. . . Upper case
POKE &H68D7, 8. . . Lower case

The following is used to input both BASIC program execution examples:

CASIO EXE

KEY MODE SELECTION PROGRAM

```
10 '
20 'AUTOMATIC SETTING OF
30 'INPUT CHARACTER MODE
40 '
50 'KEY MODE=68D7H
60 '76543210(bit)
70 '    !
80 '    +----CAPS(1=ON,0=OFF)
90 '
100 'CAPS=0 UPPER CASE(DEFAULT)
110 'CAPS=1 LOWER CASE
120 '
130 'SAMPLE
140 KEYMD=&H68D7
150 'UPPER CASE (ABC)
160 POKE KEYMD,0
170 INPUT "YOUR NAME=";A$
180 'LOWER CASE (abc)
190 POKE KEYMD,8
200 INPUT "your name=";A$
210 'RETURN TO ORIGINAL SETTING
220 POKE KEYMD,0
230 END
```

Execution Example

```
YOUR NAME=?CASIO
your name=?casio
```

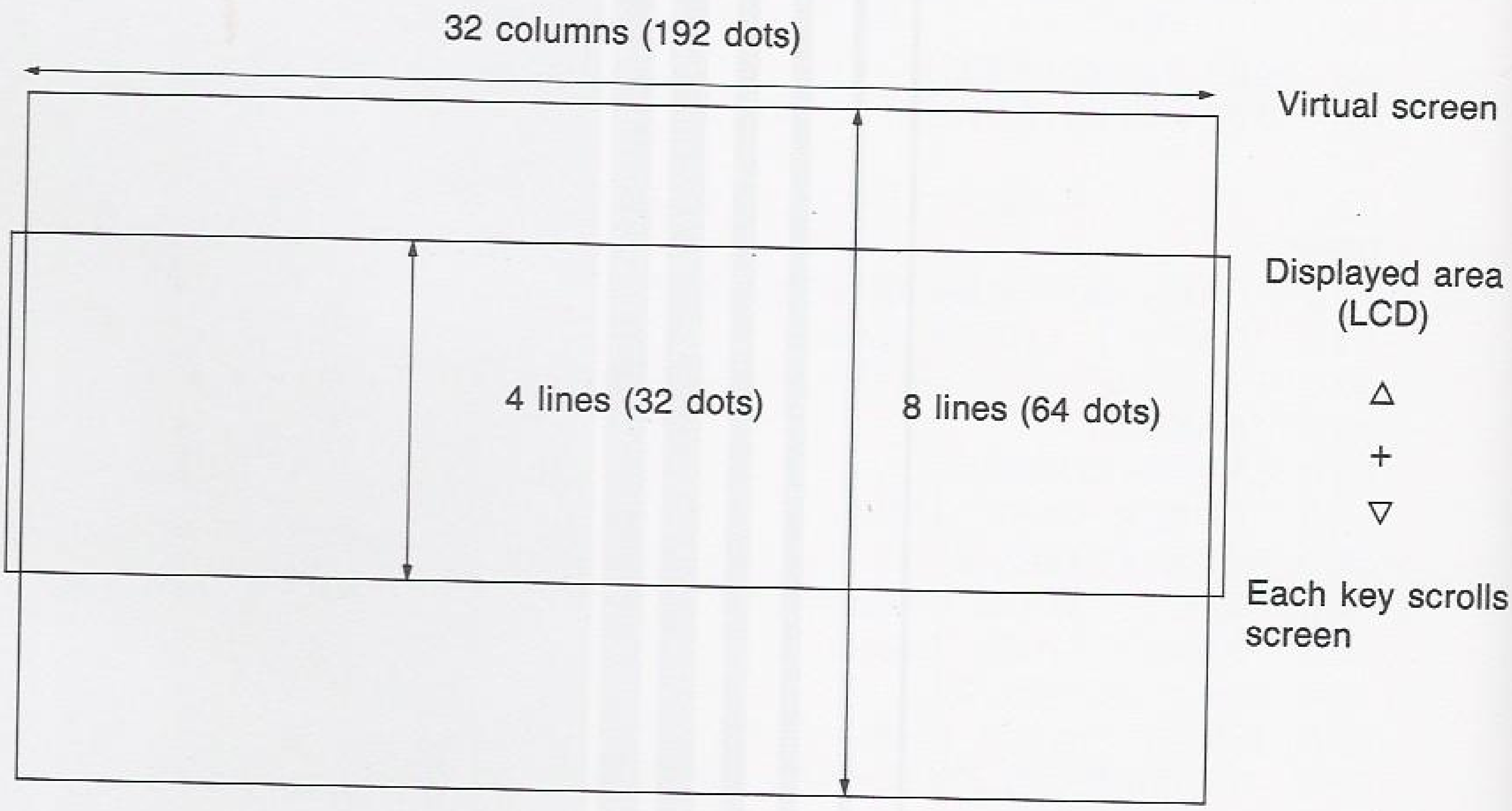
CHAPTER 7

GRAPHICS

7-1 GRAPHIC SCREEN SPECIFICATIONS

The LCD of the PB-1000 screen has an area of 192 × 32 dots (32 columns × 4 lines). This, however, is actually only one part of the PB-1000's virtual screen which measures 32 columns × 8 lines. Any part of the virtual screen can be called onto the display using BASIC commands and the cursor keys. This screen also has two types of VRAM (video RAM): a VRAM which stores each alphabetic characters stored as ASCII codes, and a VRAM which stores each character dot pattern as a bit image. The following are the commands which are used for drawing graphics on the screen:

COMMAND	FUNCTION
PRINT	Displays characters
DRAW	Draws dots and straight lines
DRAWC	Erases dots and straight lines
POINT	Checks whether dot is present at specific point on virtual screen
REV	Displays character in reverse field
NORM	Returns reverse field character to normal display
CLS	Clears screen



7-2 VRAM MAP

7-2-1 DISPLAY BUFFER

EDTOP (257 bytes) : The contents of the 8-line virtual screen are stored as they are in ASCII code. The 257th byte is used for detection of character input end.
For screen editor

LEDTP
(256 × 6 bytes) : The contents of the 8-line virtual screen are stored as 6 dots per character.
For graphics

Example

ABC

The contents of EDTOP and LEDTP are as noted below when "ABC" is displayed on the first line of the virtual screen.

EDTOP (6100H)

	MSB						LSB	
0	1	0	0	0	0	0	1	41H
0	1	0	0	0	0	1	0	42H
0	1	0	0	0	0	1	1	43H

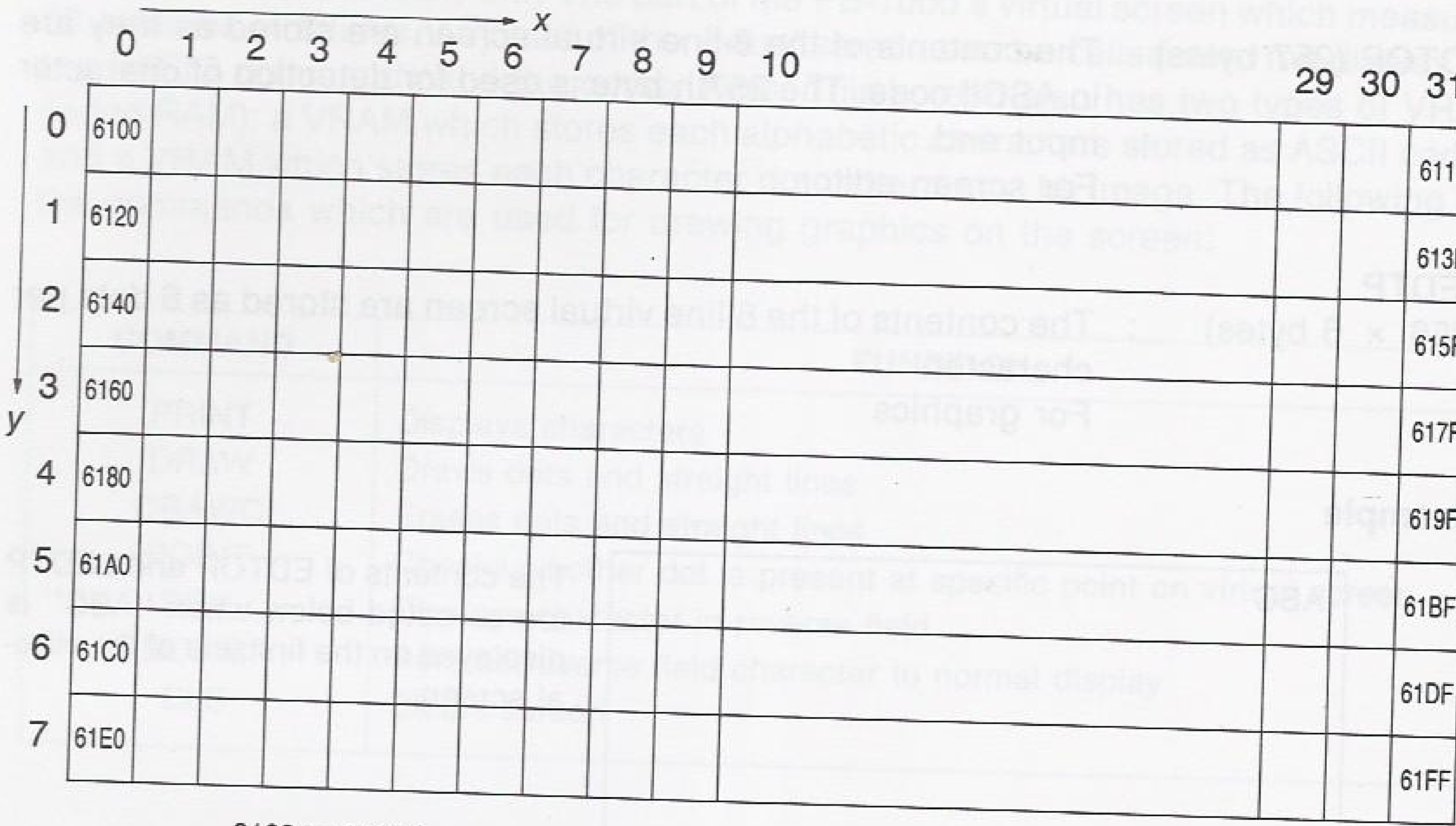
LEDTP (6201H)

MSB							LSB	
0	1	1	1	1	1	1	0	7EH
1	0	0	1	0	0	0	0	90H
1	0	0	1	0	0	0	0	90H
1	0	0	1	0	0	0	0	90H
0	1	1	1	1	1	1	0	7EH
0	0	0	0	0	0	0	0	00H
1	1	1	1	1	1	1	0	FEH
1	0	0	1	0	0	1	0	92H
1	0	0	1	0	0	1	0	92H
1	0	0	1	0	0	1	0	92H
0	1	1	0	1	1	0	0	6CH
0	0	0	0	0	0	0	0	00H
0	1	1	1	1	1	0	0	7CH
1	0	0	0	0	0	1	0	82H
1	0	0	0	0	0	1	0	82H
1	0	0	0	0	0	1	0	82H
0	1	0	0	0	1	0	0	44H
0	0	0	0	0	0	0	0	00H

The contents of LEDTP are displayed as they are, so data are written directly to the LEDTP area. The desired pattern or character is produced by calling the appropriate subroutine.

7-2-2 EDTOP MAP

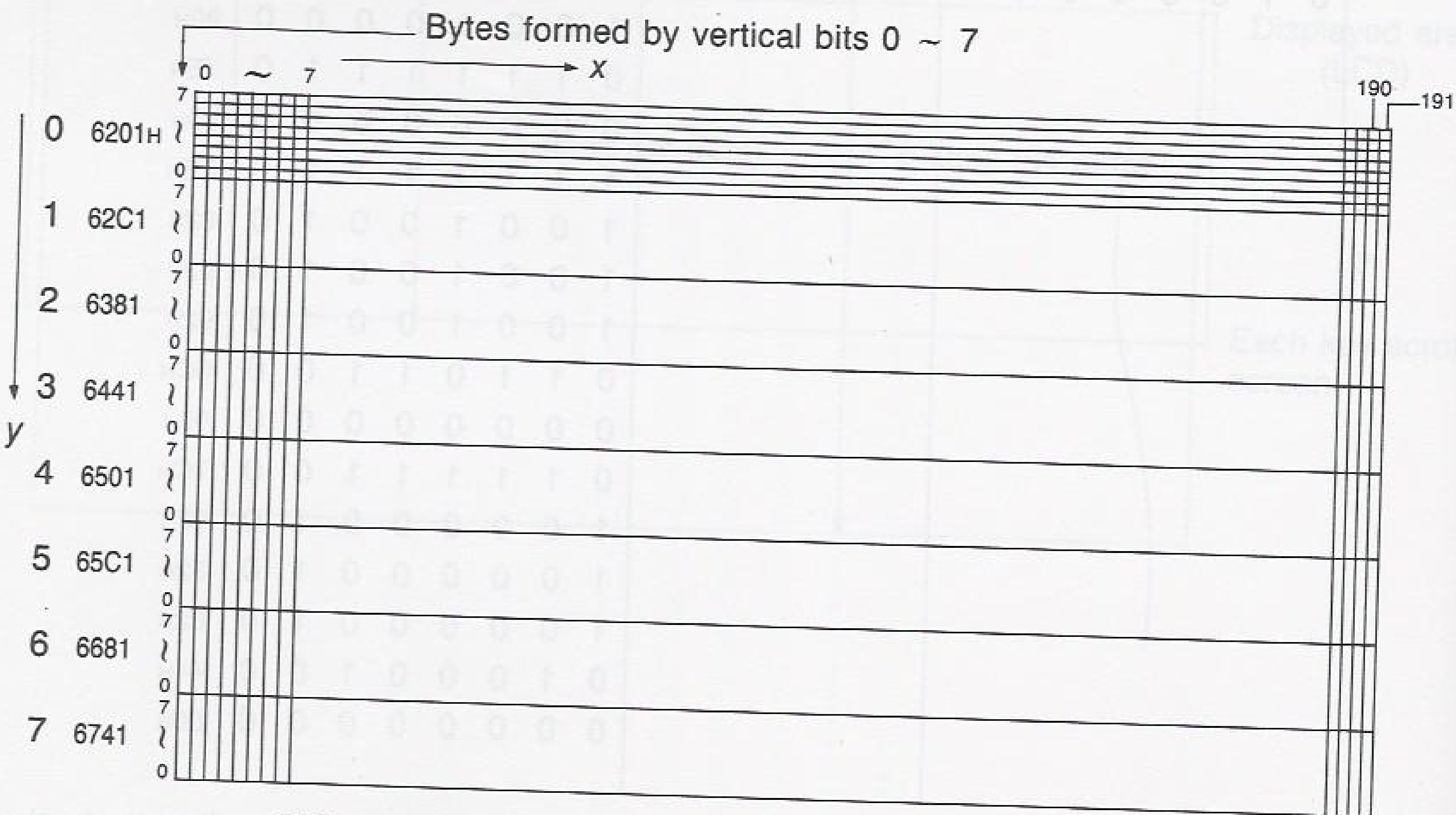
This map is used as the layout sheet and address calculation for character display. (See page 154 for details.)



6100H = 24832
Address = 24832 + 32 · y + x

7-2-3 LEDTP MAP

This map is used as the layout sheet and address calculation for graphics display. (See page 155 for details.)



6201H = 25089
Address = 25089 + 192 · y + x

7-3 ASSEMBLER GRAPHICS

The following two commands are very powerful in that they are used to transfer blocks in the CPU (HD61700):

BUP
BDN

These commands make it possible to reduce the usual 10 or more steps required in the standard Z80 and 6809 CPU's for block transfer to a mere 5 steps. In the sample program, these commands are used for horizontal scroll of the third line on the display. This application can be used for horizontal scroll of the entire display.

PROGRAM LIST 1

```
10 '
20 'PARTIAL SCROLL
30 '
40 ' "ANIME."
50 '
60 A$=""
70 FOR I=128 TO 135
80 A$=A$+CHR$(I)
90 NEXT I
100 FOR I=134 TO 129 STEP -1
110 A$=A$+CHR$(I)
120 NEXT I
130 CLS
140 LOCATE 0,0
150 PRINT "*****  3RD LINE SCROLLS  *****";
160 LOCATE 1,1
170 PRINT "(SCROLL WHILE KEY PRESSED)"
180 LOCATE 0,3
190 FOR I=1 TO 14:PRINT "■";:NEXT I
200 LOCATE 0,2:PRINT A$;A$;
210 A$=INKEY$:IF A$="" THEN 210
220 CALL "SHIFT.EXE"
230 GOTO 210
```

PROGRAM LIST 2

```
0001:0000          ;
0002:0000          ;SCROLL 3RD LINE
0003:0000          ;(EXCEPT FOR RIGHTMOST PART)
0004:0000          ;
0005:0000          ; "SHIFT.ASM"
0006:0000          ;
0007:7000                 ORG    &H7000
0008:7000                 START SHIFT
0009:7000          ;
```

```
0010:7000                DOTDI: EQU   &H022C
0011:7000                ;
0012:7000                ;IX:=TRANSFER DATA START ADDRESS
0013:7000                ;IY:=TRANSFER DATA END ADDRESS
0014:7000                ;IZ:=TRANSFER DESTINATION ADDRESS
0015:7000                ;
0016:7000   D6008263     SHIFT: PRE   IX,&H6382
0017:7004   D6202964            PRE   IY,&H6429
0018:7008   D6408163            PRE   IZ,&H6381
0019:700C                ;
0020:700C                ;BLOCK TRANSFER BY BUP COMMAND
0021:700C                ;
0022:700C   690000              LD    $0,(IZ+0)
0023:700F   D8                  BUP
0024:7010                ;
0025:7010   610000              ST    $0,(IZ+0)
0026:7013   772C02              CAL   DOTDI
0027:7016                ;
0028:7016   F7                  RTN
```

SAMPLE PROGRAM USING BDN COMMAND

```
0001:0000                ;
0002:0000                ;SCROLL 3RD LINE
0003:0000                ;(EXCEPT FOR RIGHTMOST PART)
0004:0000                ;
0005:0000                ; "SHIFT2.ASM"
0006:0000                ;
0007:7000                       ORG   &H7000
0008:7000                       START SHIFT
0009:7000                ;
0010:7000                DOTDI: EQU   &H022C
0011:7000                ;
0012:7000                ;IX:=TRANSFER DATA START ADDRESS
0013:7000                ;IY:=TRANSFER DATA END ADDRESS
0014:7000                ;IZ:=TRANSFER DESTINATION ADDRESS
0015:7000                ;
0016:7000   D6002764     SHIFT: PRE   IX,&H6427
0017:7004   D6208063            PRE   IY,&H6380
0018:7008   D6402864            PRE   IZ,&H6428
0019:700C                ;
0020:700C                ;BLOCK TRANSFER BY BDN COMMAND
0021:700C                ;
0022:700C   690000              LD    $0,(IZ+0)
0023:700F   D9                  BDN
0024:7010                ;
0025:7010   610000              ST    $0,(IZ+0)
0026:7013   772C02              CAL   DOTDI
0027:7016                ;
0028:7016   F7                  RTN
```

7-4 SCREEN HARD COPY

Since there is no COPY command, the data displayed on the 8-line × 32-column virtual screen of the PB-1000 cannot be directly printed out. This section contains two programs which can respectively be used to produce a hardcopy of the text screen and of the graphics screen.

TCOPY. ASM:
Copies the text screen (8 lines × 32 columns) to the printer. The characters displayed on the text screen are loaded in an area starting from address 6100H up to 256 bytes. All characters are processed in ASCII code, so the contents of this area can be output directly to the printer. However, graphic characters cannot be output exactly when a printer other than the model recommended is used (because of differences in character fonts).

GCOPY:
Copies the graphics screen (64 × 192 dots) to the plotter printer (FP-100). Graphics data are loaded in an area from address 6201H up to 1536 bytes. This program was developed for the FP-100, so rather extensive modification is required for use with other printers (especially dot printers). This is because it is very easy to print dots using a plotter, but the number of pins in the head must be considered when using a dot printer.

Processing is extremely slow when producing graphics copies in BASIC, so the program should first be completed in BASIC and then translated into assembly language. This greatly reduces the amount of time required for debugging.

TCOPY TEST PROGRAM

```
100 CLS
110 PRINT "********************************";
120 PRINT "* FULL SCREEN MESSAGE.         *";
130 PRINT "* THIS IS A SCREEN COPY TEST   *";
140 PRINT "* TEXT SCREEN IS 8x32 COLUMNS  *";
150 PRINT "*                              *";
160 PRINT "* I AM THE CASIO PB-1000       *";
170 PRINT "* POCKET COMPUTER!             *";
180 PRINT "********************************";
190 CALL "TCOPY.EXE"
200 LPRINT
210 END
```

Execution Example

```
**********************************
* FULL SCREEN MESSAGE.           *
* THIS IS A SCREEN COPY TEST     *
* TEXT SCREEN IS 8x32 COLUMNS    *
*                                *
* I AM THE CASIO PB-1000         *
* POCKET COMPUTER!               *
**********************************
```

TCOPY. ASM

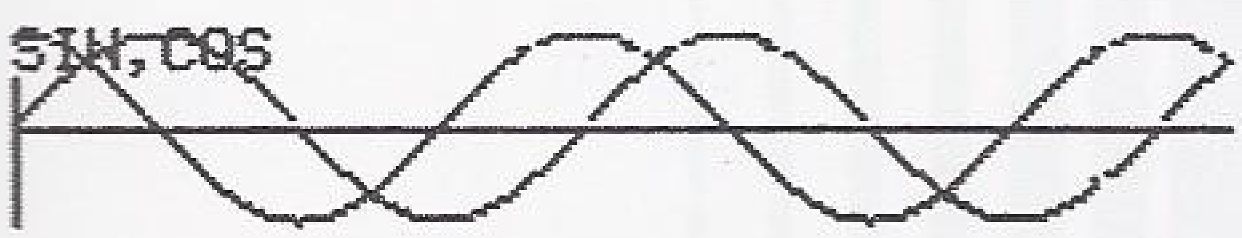
```
0001:0000                   ;
0002:0000                   ;COPY PB-1000 TEXT SCREEN
0003:0000                   ;
0004:0000                   ;GRAPHICS CANNOT BE COPIED
0005:0000                   ;USING THIS PROGRAM
0006:0000                   ;
0007:0000                   ;'TCOPY.ASM'
0008:0000                   ;
0009:0000                   PRINT: EQU   &H961F
0010:0000                   OUTCR: EQU   &H95CE
0011:0000                   ;
0012:7000                          ORG   &H7000
0013:7000                          START TCOPY
0014:7000                   ;
0015:7000                   ;TEXT RAM AREA=256 BYTES
0016:7000                   ;FROM &H6100
0017:7000                   ;
0018:7000  D6000061         TCOPY: PRE   IX,&H6100
0019:7004  D10A0000                LDW   $10,0
0020:7008                   ;
0021:7008                   ;CHARACTER ON SCREEN
0022:7008                   ;OUTPUT TO PRINTER
0023:7008                   ;
0024:7008  28700A           COPY1: LD    $16,(IX+$10)
0025:700B  771F96                  CAL   PRINT
0026:700E                   ;
0027:700E                   ;RETURN TO PREVIOUS ROUTINE
0028:700E                   ;WHEN SCREEN COPY COMPLETE
0029:700E                   ;
0030:700E  480A01                  AD    $10,1
0031:7011  341770                  JP NZ,COPY2
0032:7014  372C70                  JP    EXIT
0033:7017                   ;
0034:7017                   ;FETCH NEXT CHARACTER.
0035:7017                   ;CR/LF PERFORMED IF 32
0036:7017                   ;CHARACTERS ALREADY OUTPUT.
0037:7017                   ;
```

```
0038:7017  480B01   COPY2: AD    $11,1
0039:701A  026C0B          LD    $12,$11
0040:701D  490C20          SB    $12,32
0041:7020  340870          JP NZ,COPY1
0042:7023          ;LINEFEED AND RESET OF
0043:7023          ;CHARACTER COUNTER($11)
0044:7023          ;
0045:7023  77CE95          CAL   OUTCR
0046:7026  420B00          LD    $11,0
0047:7029  370870          JP    COPY1
0048:702C          ;
0049:702C  F7       EXIT:  RTN
```

GCOPY TEST PROGRAM

```
10 CLS
20 DRAW(0,16)-(191,16)
30 DRAW(0,0)-(0,31)
40 ANGLE 0
50 PRINT "SIN,COS"
60 FOR I=0 TO 191
70 DRAW(I,SIN(180+I*4)*15+16)
80 DRAW(I,COS(180+I*4)*15+16)
90 NEXT I
100 GOSUB 1000
999 END
```

Execution Example



GCOPY

```
1000 '
1010 'GRAPHIC SCREEN COPY
1020 ' "GCOPY."
1030 'BE CAREFUL OF USING TOO MUCH INK
1040 '
1050 '(THIS PROGRAM IS DESIGNED
1060 'FOR USE WITH A PLOTTER)
1070 '
1080 'PLOTTER INITIALIZATION
1090 '
1100 'SET TO TEXT MODE
```

```
1110 LPRINT CHR$(&H1C)CHR$(&H2E)
1120 'SET CHARACTER SCALE
1130 LPRINT CHR$(&H1B);"S1,1"
1140 'CHARACTER SPACING AND LINE SPACING
1150 LPRINT CHR$(&H1B);"Z-4,-5"
1160 'TAKE DATA FROM GRAPHICS AREA
1170 FOR Y=0 TO 63
1180 FOR X=0 TO 191
1190 D=POINT(X,Y)
1200 'DRAW DOT
1210 IF D=1 THEN LPRINT "."; ELSE LPRINT " ";
1220 NEXT X:LPRINT
1230 NEXT Y
1240 'RETURN PLOTTER TO ORIGINAL MODE
1250 LPRINT CHR$(&H1B);"S1,1"
1260 LPRINT CHR$(&H1B);"Z1,6"
1270 RETURN
```

CHAPTER 8

PRINTER INTERFACE

8-1 PRINTER INTERFACE SPECIFICATIONS

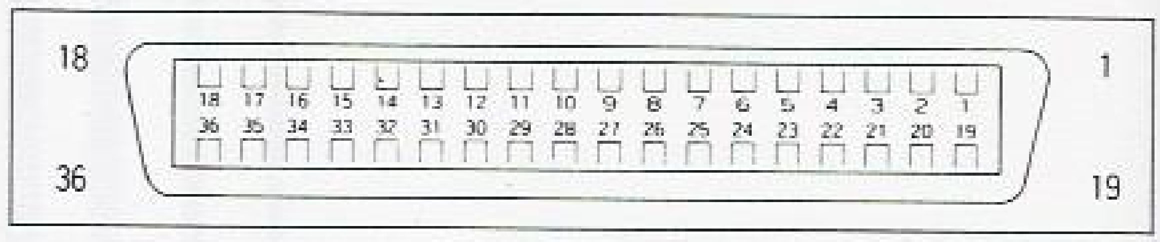
The FA-7 or MD-100 allows connection of a Centronics standard printer to the PB-1000. The following BASIC commands are available for use with these Centronics interfaces.

COMMAND	FUNCTION
LLIST	Program contents to printer
LPRINT	Specified characters to printer
TAB	Spaces up to specified location to printer
LPRINT USING	Printing according to specified format

Sequential files which have been prepared on a disk or in RAM can be printed out by pressing the LLIST touch key. This command is very convenient when trying to determine whether or not data have been correctly written into a file.
The Centronics interface terminal is a 36-pin Amphenol type connector which is commercially readily available. The pin configuration and signal timing for the signals are shown below.

• Pin Configuration

Number		Terminal Name	Number		Terminal Name
1	Output	$\overline{\mathrm{STROBE}}$	19		GND
2	Output	DATA 1	20		GND
3	Output	DATA 2	21		GND
4	Output	DATA 3	22		GND
5	Output	DATA 4	23		GND
6	Output	DATA 5	24		GND
7	Output	DATA 6	25		GND
8	Output	DATA 7	26		GND
9	Output	DATA 8	27		GND
10	Input	$\overline{\mathrm{ACKNLG}}$	28		GND
11	Input	BUSY	29		GND
12		NC	30		GND
13		NC	31	Output	$\overline{\mathrm{INIT}}$
14		NC	32	Input	$\overline{\mathrm{ERROR}}$
15		NC	33		GND
16		NC	34		NC
17		NC	35		NC
18		NC	36		NC



Control Lines

1. $\overline{\mathrm{STROBE}}$
2. $\overline{\mathrm{INIT}}$

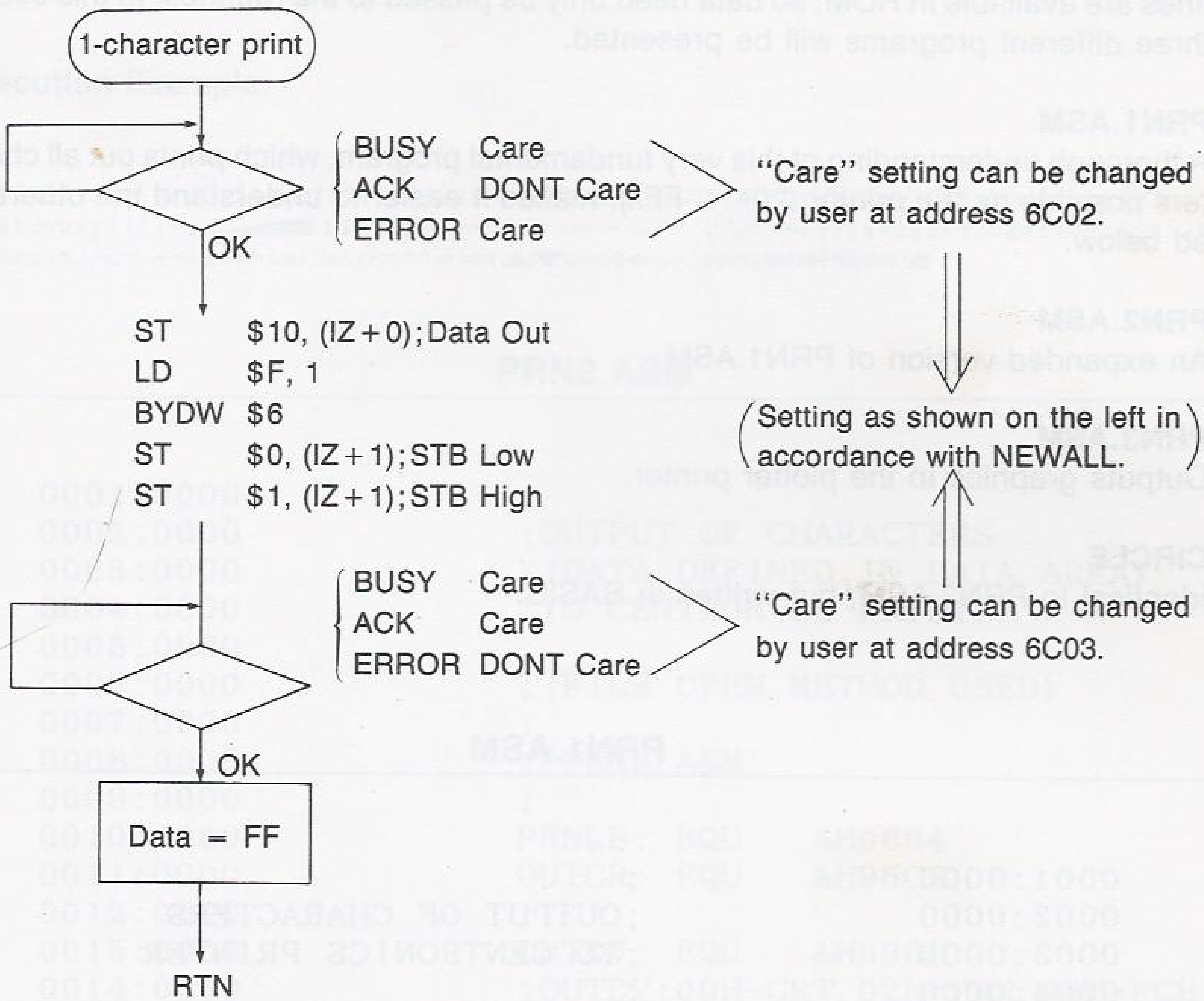
} Output control

3. $\overline{\mathrm{ACKNLG}}$
4. BUSY
5. $\overline{\mathrm{ERROR}}$

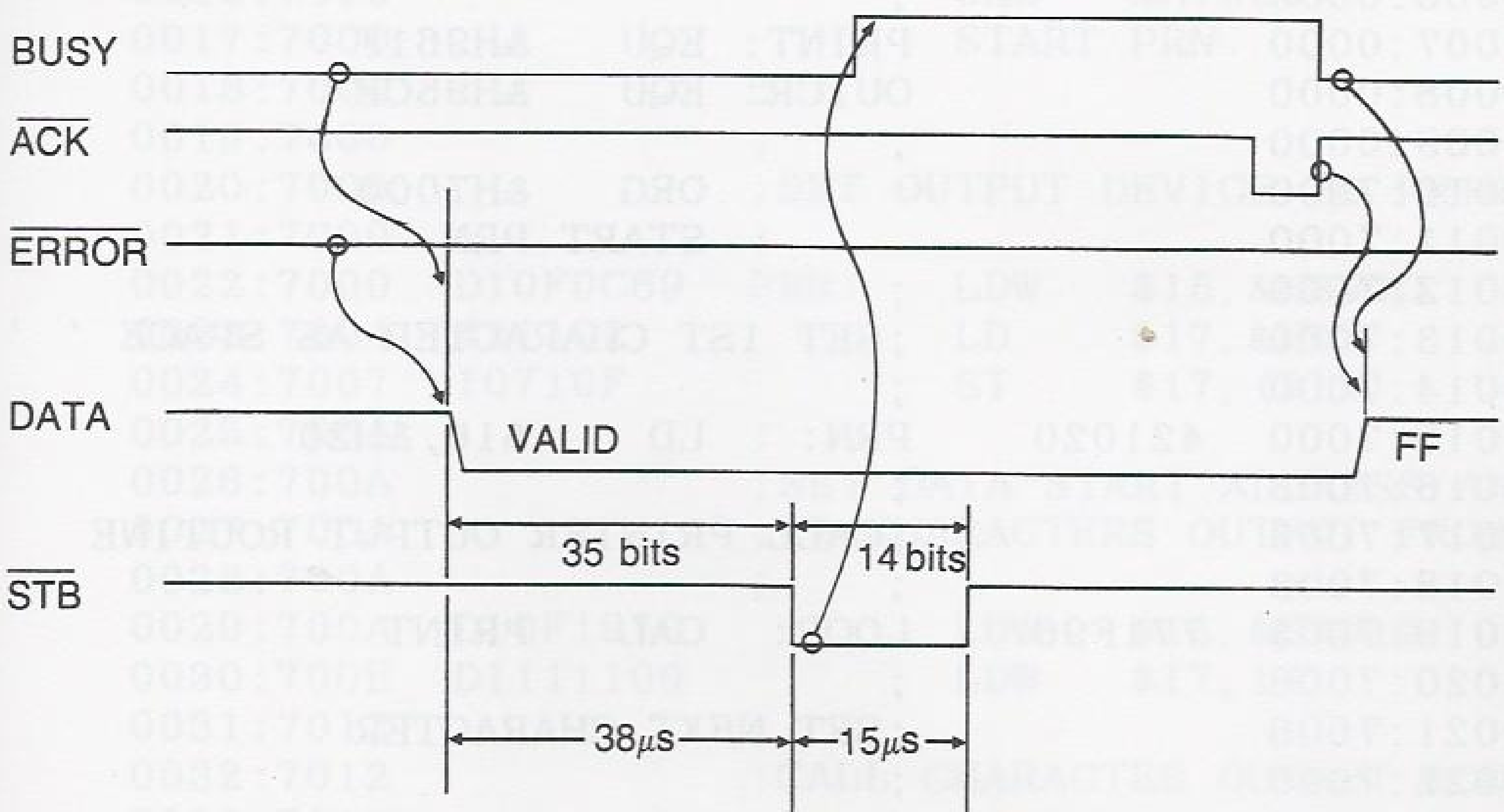
} Input control

• Data Transfer Sequence and Timing Chart

Centronics Data transfer sequence



Timing chart



Note: 1 bit = 1/910K[sec]

8-2 PRINTER CONTROL USING MACHINE LANGUAGE

Using machine language to control the printer is not very difficult. General purpose routines are available in ROM, so data need only be passed to the routines. In this section, three different programs will be presented.

PRN1.ASM
A thorough understanding of this very fundamental program, which prints out all characters possible on the printer (20_H ~ FF_H), makes it easier to understand the others listed below.

PRN2.ASM
An expanded version of PRN1.ASM.

PRN3.ASM
Outputs graphics to the plotter printer.

CIRCLE
Identical to PRN3.ASM, but written in BASIC.

PRN1.ASM

```
0000                ;
0000                ;OUTPUT OF CHARACTERS
0000                ;TO CENTRONICS PRINTER
0000                ;
0000                ;'PRN1.ASM'
0000                ;
0000                PRINT:  EQU    &H961F
0000                OUTCR:  EQU    &H95CE
0000                ;
7000                        ORG    &H7000
7000                        START PRN
7000                ;
7000                ;SET 1ST CHARACTER AS SPACE ' '
7000                ;
7000  421020        PRN:    LD     $16,&H20
7003                ;
7003                ;CALL PRINTER OUTPUT ROUTINE
7003                ;
7003  771F96        LOOP:   CAL    PRINT
7006                ;
7006                ;SET NEXT CHARACTER
7006                ;
7006  481001                AD     $16,1
7009  B187                  JR     NC,LOOP
700B                ;
```

```
0026:700B                  ;IF END,PUT CR/LF CODE TO PRINTER
0027:700B                  ;
0028:700B  77CE95                CAL    OUTCR
0029:700E  F7                    RTN
0030:700F                  ;
```

Execution Example

```
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[¥]^_'abcdefghijklmno
pqrstuvwxyz{¦}~▁▂▃▄▅▆▇█▏▎▍▌▋▊▉┼┴┬┤├▔─│▕┌┐└┘╭╮╰╯。「」、・ヲァィゥェォャュョッーアイウエオカキクケコサシスセソタ
チツテトナニヌネノハヒフヘホマミムメモヤユヨラリルレロワン゛゜═╪╡◢◣◥◤♠♥♦♣●○╱╲╳円年月日時分秒〒市区町村人▒
```

PRN2.ASM

```
0001:0000                  ;
0002:0000                  ;OUTPUT OF CHARACTERS
0003:0000                  ;(DATA DEFINED IN DATA AREA)
0004:0000                  ;TO CENTRONICS PRINTER
0005:0000                  ;
0006:0000                  ;(FILE OPEN METHOD USED)
0007:0000                  ;
0008:0000                  ;'PRN2.ASM'
0009:0000                  ;
0010:0000                  PRNLB: EQU    &H9664
0011:0000                  OUTCR: EQU    &H95CE
0012:0000                  ;
0013:0000                  OUTDV: EQU    &H690C
0014:0000                  ;OUTDV:00H=CRT,02H=PRN,04H=FCB
0015:0000                  ;
0016:7000                         ORG    &H7000
0017:7000                         START PRN
0018:7000                  ;
0019:7000                  ;
0020:7000                  ;SET OUTPUT DEVICE TO PRINTER
0021:7000                  ;
0022:7000  D10F0C69        PRN:   LDW    $15,&H690C
0023:7004  421102                 LD     $17,&H02
0024:7007  10710F                 ST     $17,($15)
0025:700A                  ;
0026:700A                  ;SET DATA START ADDRESS,AND
0027:700A                  ;17 CHARACTERS OUTPUT FROM THERE
0028:700A                  ;
0029:700A  D10F1970               LDW    $15,&H7019
0030:700E  D1111100               LDW    $17,17
0031:7012                  ;
0032:7012                  ;CALL CHARACTER OUTPUT ROUTINE
0033:7012                  ;
0034:7012  776496                 CAL    PRNLB
0035:7015                  ;
```

```
0036:7015                    ;SEND CR/LF TO PRINTER
0037:7015                    ;
0038:7015  77CE95                  CAL    OUTCR
0039:7018  F7                      RTN
0040:7019                    ;
0041:7019                    ;TEST DATA
0042:7019                    ;
0043:7019  4920414D          MSG:  DB     "I AM "
           20
0044:701E  54484520                DB     "THE PB-1000."
           50422D31
           3030302E
0045:702A                    ;
```

Execution Example

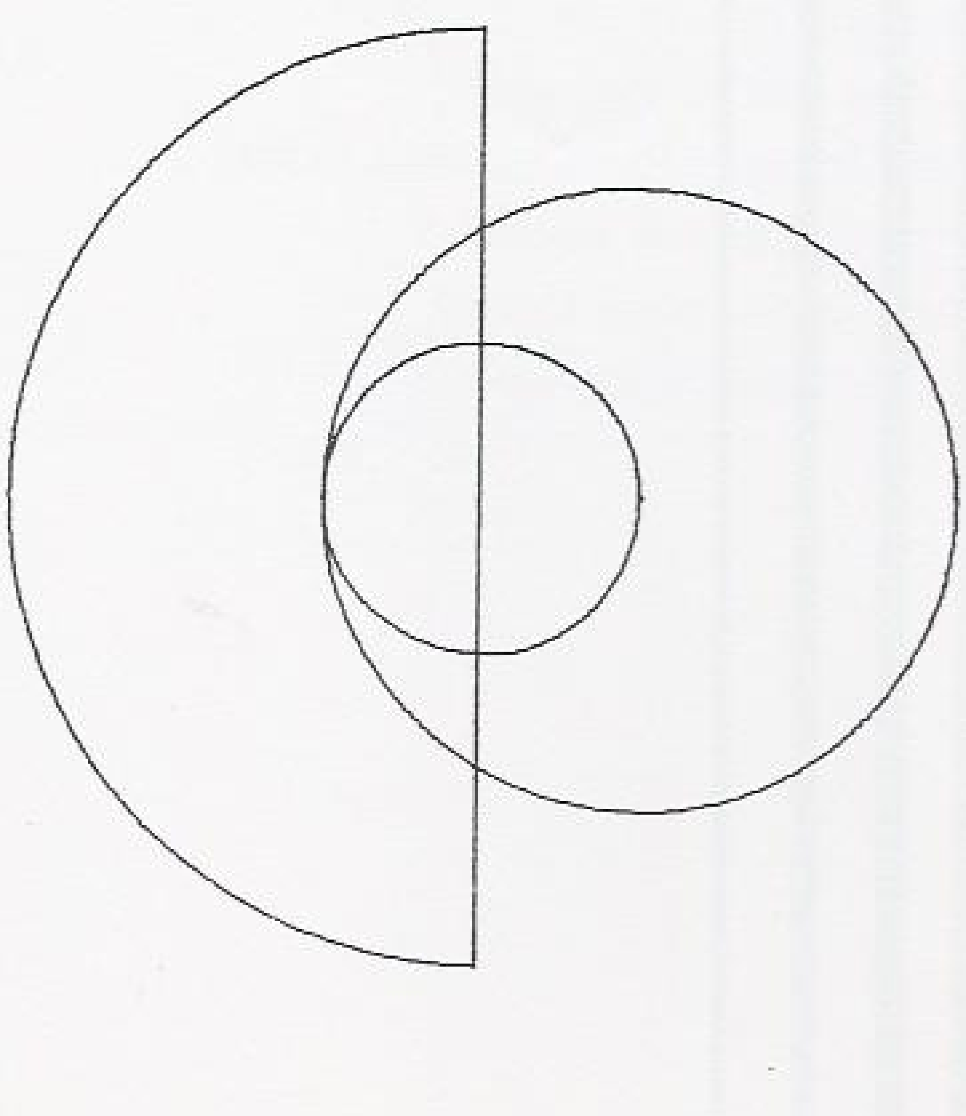
```
I AM THE PB-1000.
```

PRN3.ASM

```
0001:0000                    ;OUTPUT OF GRAPHICS
0002:0000                    ;TO PLOTTER.
0003:0000                    ;(PARAMETERS DEFINED IN
0004:0000                    ; DATA AREA)
0005:0000                    ;(FILE OPEN METHOD)
0006:0000                    ;
0007:0000                    ;'PRN3.ASM'
0008:0000                    ;
0009:0000                    PRNLB: EQU    &H9664
0010:0000                    OUTCR: EQU    &H95CE
0011:0000                    OUTDV: EQU    &H690C
0012:0000                    ;
0013:0000                    ;OUTDV:00H=CRT,02H=PRN
0014:0000                    ;      04H=FCB
0015:0000                    ;
0016:7000                          ORG    &H7000
0017:7000                          START PRN
0018:7000                    ;
0019:7000                    ;SET OUTPUT DEVICE
0020:7000                    ;TO PLOTTER
0021:7000  D10F0C69          PRN:  LDW    $15,&H690C
0022:7004  421102                  LD     $17,&H02
0023:7007  10710F                  ST     $17,($15)
0024:700A                    ;
0025:700A                    ;SET DATA START ADDRESS
0026:700A                    ;AND 52 CHARACTERS
0027:700A                    ;OUTPUT FROM THERE
0028:700A  D10F1670                LDW    $15,&H7016
0029:700E  D1113400                LDW    $17,52
0030:7012                    ;
```

```
0031:7012                   ;CALL CHARACTER
0032:7012                   ;OUTPUT ROUTINE
0033:7012  776496                  CAL    PRNLB
0034:7015  F7                      RTN
0035:7016                   ;
0036:7016                   ;GRAPHICS DATA
0037:7016                   ;
0038:7016                   ;SET PLOTTER TO
0039:7016                   ;GRAPHICS MODE
0040:7016                   ;
0041:7016  1C250D           MSG:   DB     &H1C, &H25, &H0D
0042:7019                   ;
0043:7019                   ;DRAW TWO CIRCLES
0044:7019  4333302C                DB     "C30, -30, 10", &H0D
           2D33302C
           31300D
0045:7024  432C3230                DB     "C, 20", &H0D
           0D
0046:7029                   ;DRAW HALF-CIRCLE
0047:7029  4333302C                DB     "C30, -30, 30, 90, 270"
           2D33302C
           33302C39
           302C3237
           30
0048:703A  0D                      DB     &H0D
0049:703B  49302C36                DB     "I0, 60", &H0D
           300D
0050:7041                   ;RETURN TO TEXT MODE
0051:7041  1C2E0D                  DB     &H1C, &H2E, &H0D
0052:7044  454E442E                DB     "END. ", &H0D, &H0A
           0D0A
```

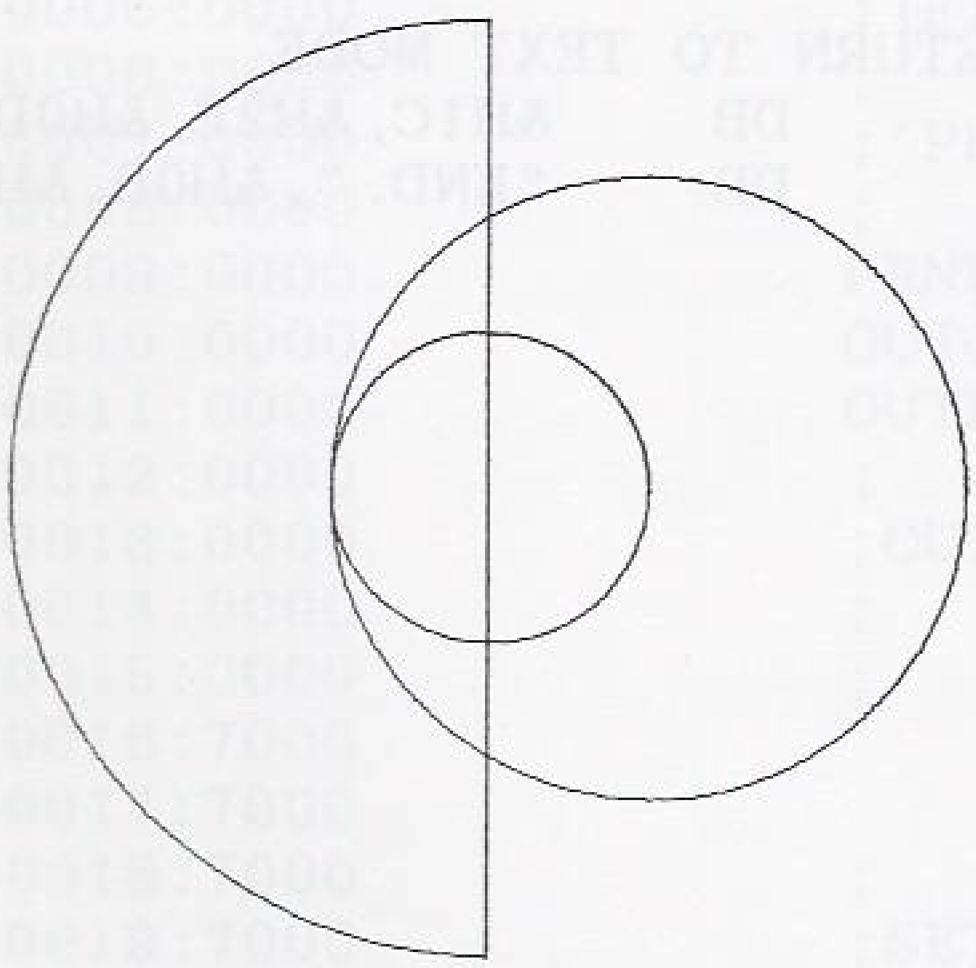
Execution Example



CIRCLE

```
10 '
20 'DRAW GRAPHICS ON PLOTTER.
30 '
40 ' "CIRCLE."
50 '
60 'SET TO GRAPHICS MODE
70 LPRINT CHR$(&H1C);CHR$(&H25)
80 'DRAW TWO CIRCLES
90 LPRINT "C30,-30,10"
100 LPRINT "C,20"
110 'DRAW HALF-CIRCLE
120 LPRINT "C30,-30,30,90,270"
130 LPRINT "I0,60"
140 'RETURN TO TEXT MODE
150 LPRINT CHR$(&H1C);CHR$(&H2E)
160 LPRINT "END."
170 END
```

Execution Example



CHAPTER 9

RS-232C INTERFACE

9-1 RS-232C SPECIFICATIONS

Using the PB-1000 in combination with the FA-7 or MD-100 provides RS-232C operations. Until now, data input to this class of computer was always limited in its applicable scope because the RS-232C interface was not available. The FA-7 and MD-100 provides all the functions of the RS-232C interface in a compact configuration. This means that the system provides most of the operation of a stand-alone personal computer. The following are the parameters for the PB-1000 RS-232C interface:

- Communication system: Start-stop (asynchronous), full duplex mode only
- Baud rate: 75, 150, 300, 600, 1200, 2400, 4800, 9600bps
- Parity bit: Odd, Even, None
- Word length: 7 bits, 8 bits
- Stop bit: 1 bit, 2 bits

However, the following combination is impossible:

Baud rate: 9600bps
Parity: None
Word length: 7 bits
Stop bit: 1

- **Communication BASIC Commands**

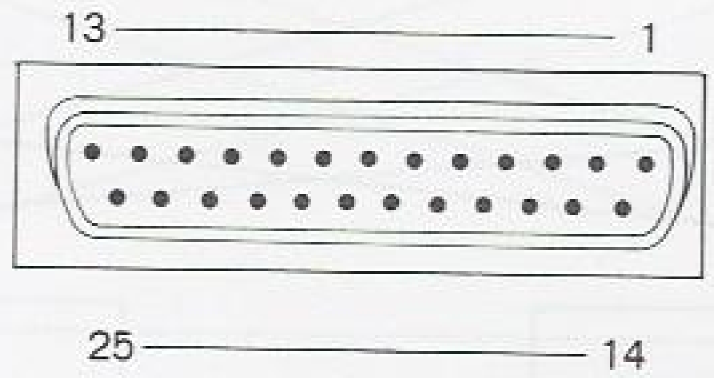
The following BASIC commands are available for use with this function:

COMMAND	FUNCTION
OPEN	Declares use of communications circuit
CLOSE	Closes open communications circuit
PRINT#	Outputs data to communications circuit
PRINT#USING	Outputs data to communications circuit
INPUT#	Reads data from communications circuit
LINE INPUT#	Reads data from communications circuit
INPUT$	Reads data from communications circuit
EOF	Function to indicate receive buffer status
LOF	Function to indicate bytes remaining in receive buffer
SAVE	Outputs program to communications circuit
LOAD	Reads program from communications circuit
MERGE	Merges program from communications circuit

- **Pin Assignments**

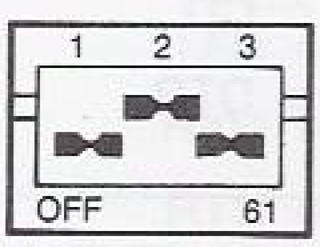
Terminal #	Signal name
1	FG
2	TXD
3	RXD
4	RTS
5	CTS
6	DSR
7	SG
8	CD
9	NC
10	NC
11	NC
12	NC
13	NC
14	NC
15	NC
16	NC
17	NC
18	NC
19	NC
20	DTR
21	NC
22	NC
23	NC
24	NC
25	NC

Pin connection



9-1-1 BAUD RATE

The baud rate can be set by software, or by the dip switches on the back of the FA-7/MD-100.



BPS	1	2	3
75	OFF	OFF	OFF
150	ON	OFF	OFF
300	OFF	ON	OFF
600	ON	ON	OFF
1200	OFF	OFF	ON
2400	ON	OFF	ON
4800	OFF	ON	ON
9600	ON	ON	ON

9-1-2 CABLE CONNECTING

A variety of connecting patterns is possible with the RS-232C (except for modems). The illustrations below show some typical cable connections. Select the connection method which matches the specific device being connected.

1) The pattern shown here is known as the null modem. This is the simplest pattern, but it disregards the status of the connected device. Therefore, both devices must be operated manually, and a large buffer is required.

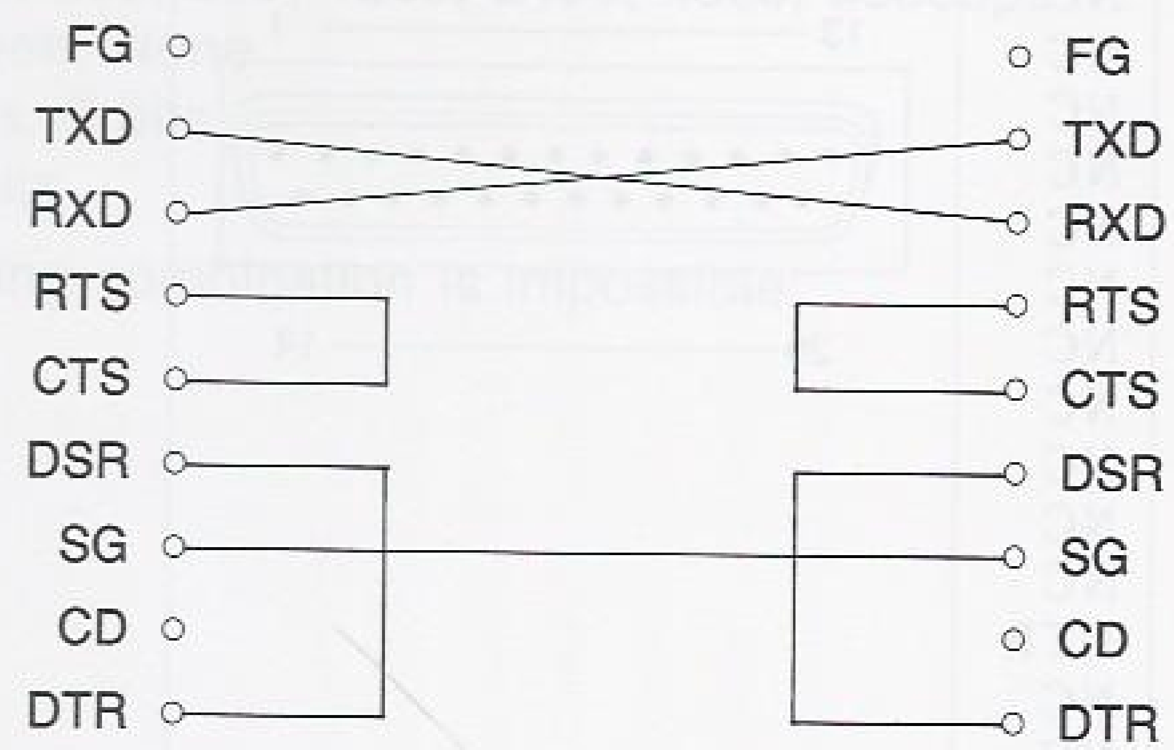


Fig. 1

2) This pattern is applicable in most cases, allowing communications with most devices. The pattern of Fig. 3 should be used, however, when CD terminal is required.

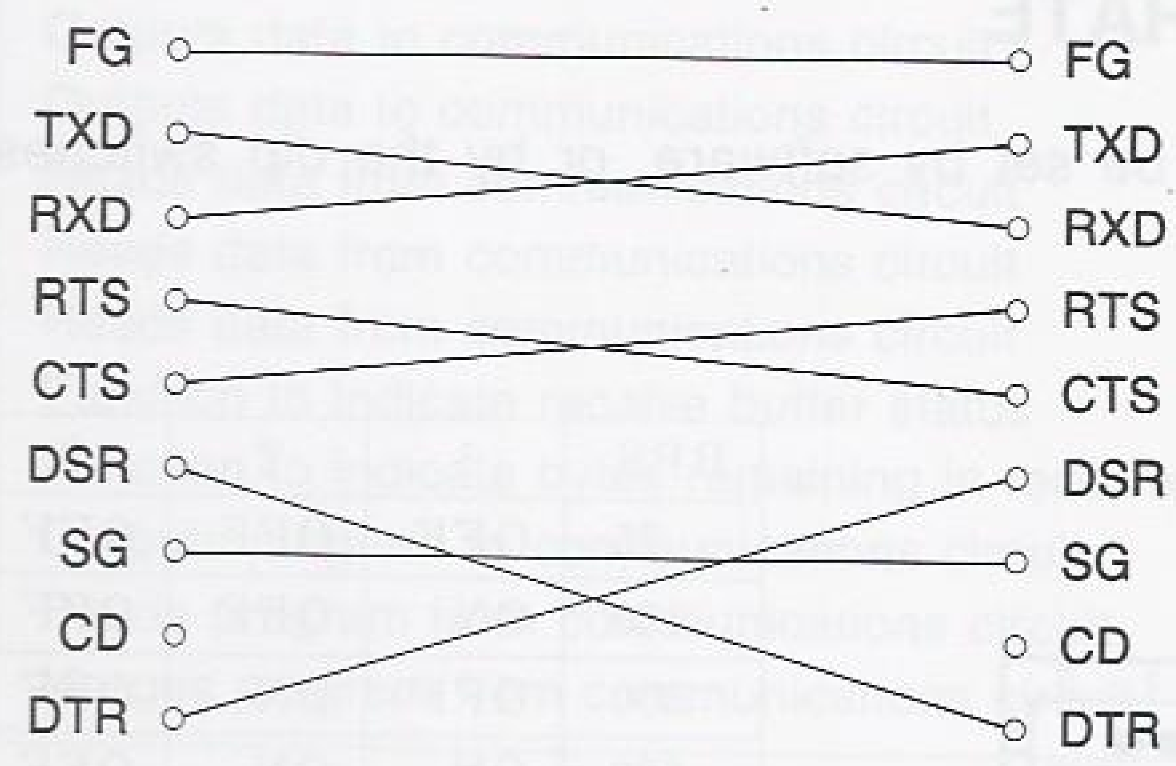


Fig. 2

3) This pattern should be used when CD terminal is required. In this case, RTS terminal contents are sent to CD to inform the send status. The carrier is generally ON during send.

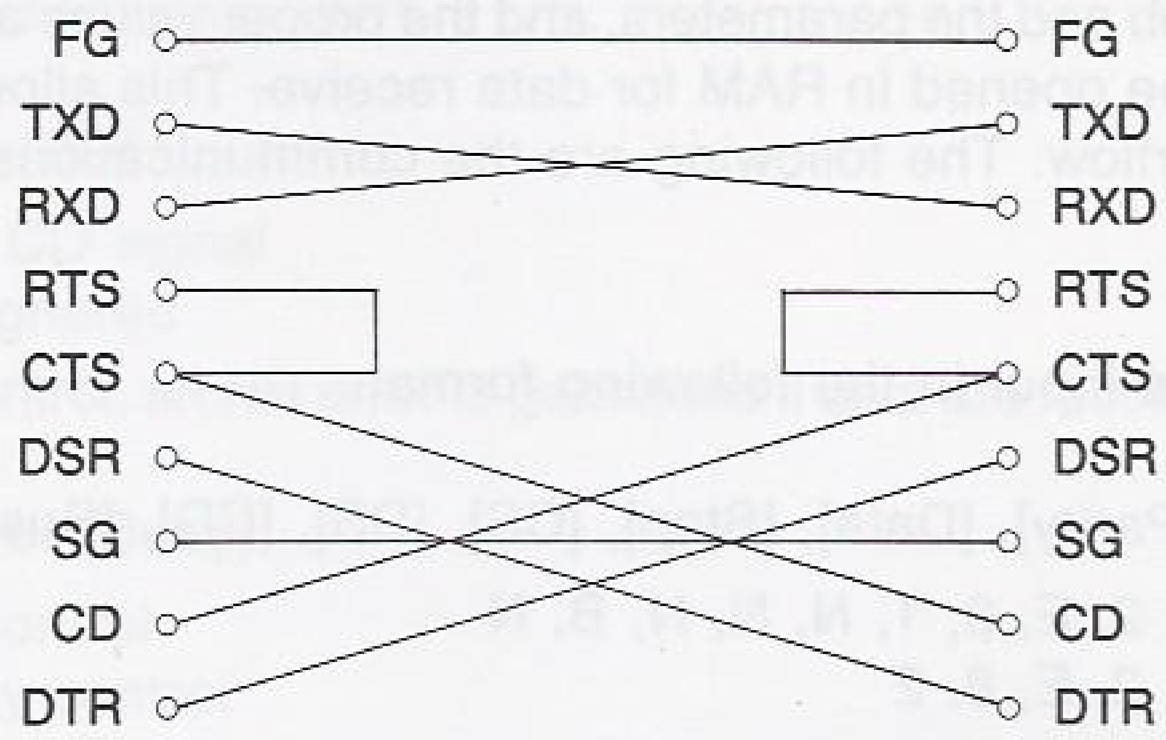


Fig. 3

4) This pattern is a rather special pattern, but it should be noted that successful data transfer is sometimes possible only with this pattern.

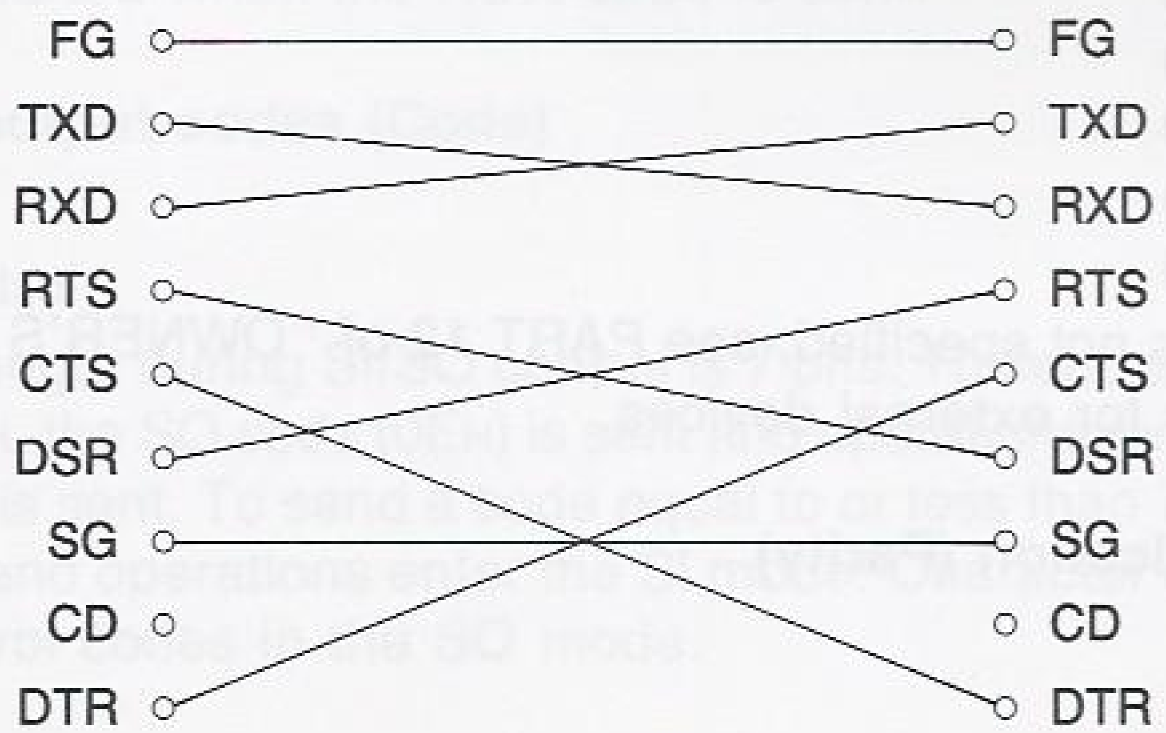


Fig. 4

5) This pattern is used for connection with a modem. A modem performs signal exchange within itself, so communications are possible by parallel connection of each terminal.

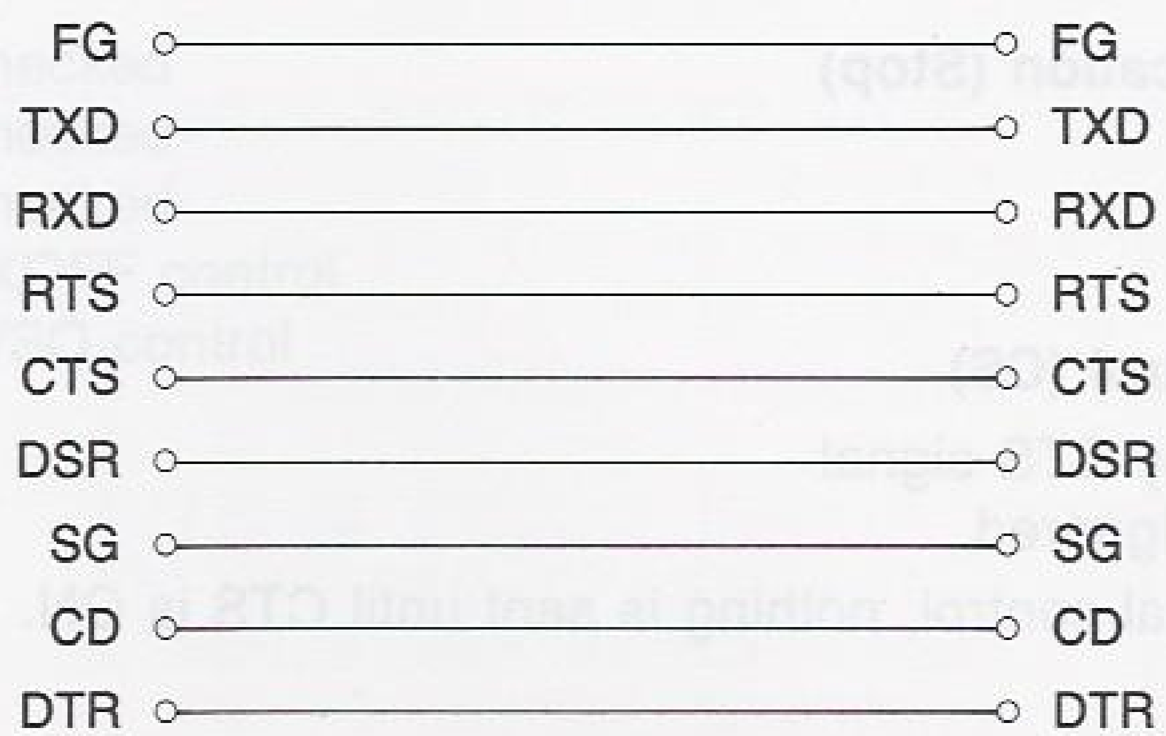


Fig. 5

9-2 BASIC COMMUNICATIONS PROGRAM

Program list 1 allows RS-232C communications using BASIC. Once executed, it asks the contents of the job and the parameters, and the proper values are entered. It is recommended that a file be opened in RAM for data receive. This allows quick access, thus avoiding buffer overflow. The following are the communications parameters:

The file descriptor is input in the following format.

COM0 : [[Speed], [Parity], [Data], [Stop], [CS], [DS], [CD], [Busy], [Code]]

Example: COM0 : 2, E, 8, 1, N, N, N, B, N
COM0 : 2, E, 8, 2

a) Baud rate specification (Speed)

7 : 9600 (BPS)
6 : 4800 (BPS)
5 : 2400 (BPS)
4 : 1200 (BPS)
3 : 600 (BPS)
2 : 300 (BPS)
1 : 150 (BPS)
0 : 75 (BPS)

If the baud rate is not specified, see PART 12 of ''OWNER'S MANUAL'' for setting the DIP switches for external devices.

b) Parity bit specification (Parity)

N : No parity
E : Even parity
O : Odd parity

c) Character bit length specification (Data)

7 : 7 bits
8 : 8 bits

d) Stop bit specification (Stop)

1 : 1 bit
2 : 2 bits

e) CTS signal control (CS)

C : Controlled by CTS signal
N : CTS signal ignored

During CTS signal control, nothing is sent until CTS is ON.

f) DSR signal control (DS)

D : Controlled by DSR signal
N : DSR signal ignored

During DSR signal control, an NR error is generated when data is received while DSR is OFF. Nothing is sent until DSR is ON.

g) CD signal control (CD)

C : Controlled by CD signal
N : CD signal is ignored

During CD signal control, an NR error is generated if data are received while CD is OFF.

h) Buffer busy control (Busy)

B: Buffer busy control
N : No buffer busy control

An XOFF code (13H) is sent and data transfer from the host is temporarily halted when busy control is invoked and the empty area of the buffer is 64 characters or less. After the XOFF signal is sent, the data in the buffer is read. If there are 32 or fewer characters remaining in the buffer, the XON (11H) signal is sent and a send request is issued to the host. If the XOFF code is received from the host, data transfer is halted. Transfer is restarted when the XON code is sent.

i) System of input/output codes (Code)

S : SI/SO control
N : No SI/SO control

The character bit length during SI/SO control is 7 bits. To send a code that is greater than or equal to 80H, the SO code (0EH) is sent and operations remain in the SO mode until a code of 7FH is sent. To send a code equal to or less than 7FH in the SO mode, an SI code is sent and operations enter the SI mode. Character codes 80H ~ 9FH are processed as control codes in the SO mode.

- **Parameter default values (initial values)**

COM0	:	2, E, 8, 1, N, N, N, B, N
Speed	:	300 BPS
Parity	:	Even
Data bits	:	8
Stop bits	:	1
CS	:	Not checked
DS	:	Not checked
CD	:	Not checked
Busy	:	XON/XOFF control
Code	:	No SI/SO control

PROGRAM LIST 1

```
100 PARA$="2,E,8,1,N,N,N,B,N"
110 ON ERROR GOTO 5000
120 CLS
130 PRINT "****PB-1000 COMMUNICATIONS****"
140 PRINT " [  1  ] SEND"
142 PRINT " [  2  ] RECEIVE"
144 PRINT " [  3  ] END";
150 IN=ASC(INKEY$)
160 IF IN=244 OR IN=49 THEN 1000
170 IF IN=248 OR IN=50 THEN 2000
180 IF IN=252 OR IN=51 THEN 200
190 GOTO 150
200 PRINT:PRINT "**END**";
210 END
1000 ' SEND
1010 CLS
1020 PRINT "ENTER FILE NAME TO BE SENT"
1030 INPUT FIL$
1040 OPEN FIL$ FOR INPUT AS #1
1050 CLS
1060 PRINT "CURRENT PARAMETERS"
1070 PRINT PARA$
1080 PRINT "","[EXE] STARTS SEND";
1090 LOCATE 0,1:INPUT "",PARA$
1100 PARA$=LEFT$(PARA$,17)
1110 OPEN "COM0:"+PARA$ FOR OUTPUT AS #2
1120 LOCATE 0,3
1125 PRINT "···   SENDING    ...";
1130 LINE INPUT #1,A$
1140 PRINT #2,A$
1150 IF EOF(1)=0 THEN 1130
1160 CLS :CLOSE
1170 PRINT "END OF SEND"
1180 PRINT " [EXE] RETURN TO MENU"
1190 IF INKEY$=CHR$(&HD) THEN 120 ELSE 1190
2000 'RECEIVE
2010 CLS
2020 PRINT "ENTER SEQUENTIAL FILE NAME"
2030 INPUT "",FIL$
2040 OPEN FIL$ FOR OUTPUT AS #1
2050 CLS
2060 PRINT "CURRENT PARAMETERS"
2070 PRINT PARA$
2080 PRINT "","[EXE] STARTS RECEIVE      ";
2090 LOCATE 0,1:INPUT "",PARA$
2100 OPEN "COM0:"+PARA$ FOR INPUT AS #2
2110 IF EOF(2)=0 THEN 2110
2120 LOCATE 0,3
2125 PRINT "···   RECEIVING  ···";
2130 LINE INPUT #2,A$
```

```
2140 PRINT #1,A$
2150 IF EOF(2) THEN 2130
2160 CLS :CLOSE
2170 PRINT "END OF RECEIVE"
2180 PRINT " [EXE] RETURN TO MENU"
2190 IF INKEY$=CHR$(&HD) THEN 120 ELSE 2190
5000 'ERROR HANDLING
5010 CLS :BEEP1
5020 IF ERR=10 THEN 5080
5030 PRINT "ERROR GENERATED"
5040 PRINT "ERROR CODE";ERR
5050 LOCATE 15,1:PRINT "GENERATED";ERL
5060 END
5070 'FILE NOT FOUND
5080 PRINT "FILE NOT FOUND"
5090 FOR I=1 TO 150:NEXT I:RESUME 1000
```

9-3 PROGRAM TO TRANSFER TEXT DATA BETWEEN PC AND PB-1000

The following two programs transfer text data between an IBM-compatible PC and PB-1000 using the RS-232C. Executing the program displays a menu on the CRT and operation is in accordance with subsequent screens.
The baud rate of the PC should be previously specified using the memory switch. This program allows data transfer regardless of the cable connection pattern. Because of this, the data "//END" should be included at the end of the text. Program or cable connection pattern modification may be required for the handling of data other than BASIC data.

* IBM is a registered trademark of International Business Machines Corporation.

• PB-1000 PROGRAM

```
10 '
20 ' PB-1000 <=> PC
30 '  (FOR PB-1000)
40 ' "RSPBTOPC"
50 '
60 CLEAR
100 PARA$="2,E,8,1,N,N,N,B,N"
110 ON ERROR GOTO 5000
120 CLS
130 PRINT "****PB-1000 COMMUNICATIONS****"
140 PRINT " [  1  ] SEND"
142 PRINT " [  2  ] RECEIVE"
144 PRINT " [  3  ] END";
150 IN=ASC(INKEY$)
160 IF IN=244 OR IN=49 THEN 1000
170 IF IN=248 OR IN=50 THEN 2000
180 IF IN=252 OR IN=51 THEN 200
190 GOTO 150
200 PRINT:PRINT "**END**";
210 END
```

```
1000 ' SEND
1010 CLS
1020 PRINT "ENTER FILE NAME TO BE SENT"
1030 INPUT FIL$
1040 OPEN FIL$ FOR INPUT AS #1
1050 CLS
1060 PRINT "CURRENT PARAMETERS"
1070 PRINT PARA$
1080 PRINT "","[EXE] TO STARTS SEND";
1090 LOCATE 0,1:INPUT"",PARA$
1100 PARA$=LEFT$(PARA$,17)
1110 OPEN "COM0:"+PARA$ FOR OUTPUT AS #2
1120 LOCATE 0,3
1125 PRINT "...          SENDING          ...";
1130 LINE INPUT #1,A$
1140 PRINT#2,A$
1150 IF EOF(1)=0 THEN 1130
1160 CLS :PRINT#2,"//END":CLOSE
1170 PRINT "END OF SEND"
1180 PRINT " [EXE] RETURN TO MENU"
1190 IF INKEY$=CHR$(&HD) THEN 120 ELSE 1190
2000 ' RECEIVE
2010 CLS
2020 PRINT "ENTER SEQUENTIAL FILE NAME"
2030 INPUT "",FIL$
2040 OPEN FIL$ FOR OUTPUT AS #1
2050 CLS
2060 PRINT "CURRENT PARAMETERS"
2070 PRINT PARA$
2080 PRINT "","[EXE] START RECEIVE";
2090 LOCATE 0,1:INPUT "",PARA$
2100 OPEN "COM0:"+PARA$ FOR INPUT AS #2
2110 IF EOF(2)=0 THEN 2110
2120 LOCATE 0,3
2125 PRINT "*****   RECEIVING    *****";
2130 LINE INPUT #2,A$
2135 IF A$="//END" THEN 2160
2140 PRINT #1,A$
2150 GOTO 2130
2160 CLS :CLOSE
2170 PRINT "END OF RECEIVE"
2180 PRINT " [EXE] RETURN TO MENU"
2190 IF INKEY$=CHR$(&HD) THEN 120 ELSE 2190
5000 ' ERROR HANDLING
5010 CLS :BEEP1
5020 IF ERR=10 THEN 5080
5030 PRINT "ERROR GENERATED"
5040 PRINT "ERROR CODE";ERR
5050 LOCATE 15,1:PRINT "ERROR GENERATED";ERL
5060 END
5070 'FILE NOT FOUND
5080 PRINT "FILE NOT FOUND."
5090 FOR I=1 TO 150:NEXT I:RESUME 1000
```

• PC PROGRAM

```
100 '
110 ' PB-1000<=> PC
120 ' (FOR PC)
130 ' "RSPCTOPB"
140 '
150 M=1
160 PARA$="300,N,8,1"
170 ON ERROR GOTO 660
180 CLS
190 PRINT "**** PC COMMUNICATION ****"
200 PRINT "[1] SEND"
210 PRINT "[2] RECEIVE"
220 PRINT "[3] END"
230 INPUT "NUMBER=";IN
240 ON IN GOTO 300,460,260
250 GOTO 180
260 PRINT:PRINT "**END**";
270 ON ERROR GOTO 0
280 END
290 ' SEND
300 PRINT "ENTER FILE NAME TO BE SENT";
310 INPUT FIL$
320 OPEN FIL$ FOR INPUT AS #1
330 PRINT "PRESS SPACE TO START"
340 K$=INKEY$:IF K$=" " THEN 350 ELSE 340
350 OPEN "COM1:"+PARA$ FOR OUTPUT AS #2
360 PRINT "SENDING...."
370 LINE INPUT #1,A$
380 PRINT #2,A$
390 IF M=1 THEN PRINT A$
400 IF EOF(1)=0 THEN 370
410 PRINT#2,"//END":CLOSE
420 PRINT "END OF SEND"
430 PRINT "PRESS SAPCE KEY"
440 K$=INKEY$:IF K$=" " THEN 180 ELSE 440
450 ' RECEIVE
460 PRINT "ENTER SEQUENTIAL FILE NAME";
470 INPUT FIL$
480 OPEN FIL$ FOR OUTPUT AS #1
490 PRINT "PRESS SPACE TO START"
500 K$=INKEY$:IF K$=" " THEN 510 ELSE 500
510 OPEN "COM1:"+PARA$ FOR INPUT AS #2
520 IF EOF(2)=0 THEN 520
530 PRINT "RECEIVING..."
540 LINE INPUT #2,A$
550 IF A$="//END" THEN 590
```

```
560 PRINT #1,A$
570 IF M=1 THEN PRINT A$
580 GOTO 540
590 CLOSE
600 PRINT "END OF RECEIVE"
610 PRINT "PRESS ANY KEY"
620 K$=INKEY$:IF K$=" " THEN 180 ELSE 620
630 ' ERROR HANDLING
640 ON ERROR GOTO 0
650 CLOSE
660 PRINT "ERROR GENERATED"
670 PRINT "ERROR CODE";ERR
680 PRINT "GENERATED";ERL
690 END
```

CHAPTER 10

DISK DRIVE

10-1 DISK DRIVE SPECIFICATIONS

The 3.5″ disk drive of the MD-100 has the following specifications:

- Type: 3.5″ single-sided, double-density, double-track
- Recording method: MFM
- Storage capacity: 500KB when unformatted/320KB when formatted
- Number of tracks: 80 (0 ~ 79)
- Number of sectors: 16 (1 ~ 16)
- Transfer rate: 250K bits/sec
- Rotational speed: 300 rpm

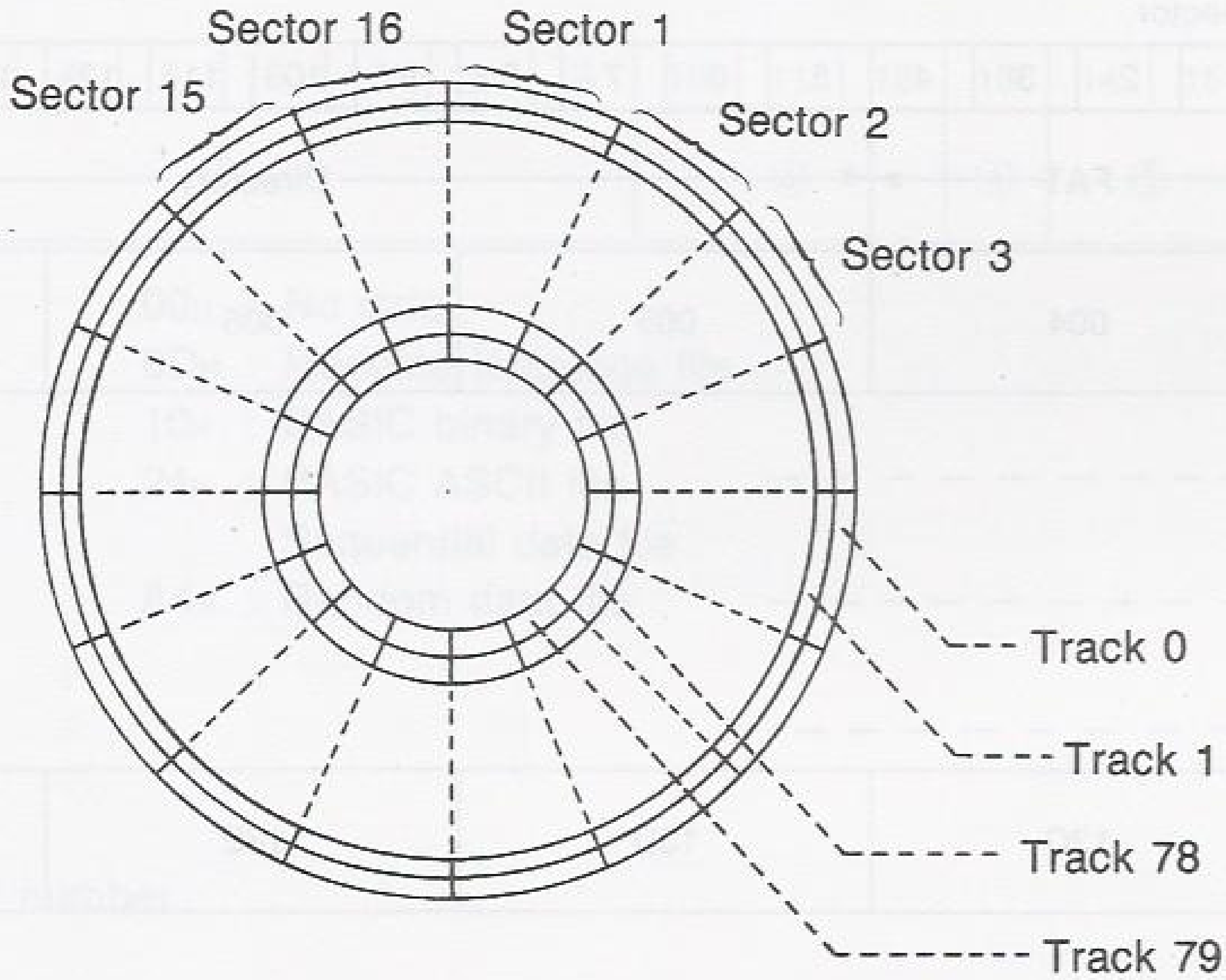
The following BASIC commands are available for use with the disk drive.

Command	Purpose
FORMAT	Initializes floppy disk
OPEN	Declares file use
CLOSE	Closes open file
PRINT#	Outputs data to sequential file
PRINT# USING	Outputs data to sequential file
INPUT#	Reads data from sequential file
LINE INPUT#	Reads data from sequential file up to CR code
INPUT$	Reads data from sequential file up to specified character
GET	Reads data from file to I/O buffer
PUT	Writes data from I/O buffer to file
LOF	Returns number of records in file
EOF	Returns end of file read
SAVE	Saves program to specified file
LOAD	Loads program contents
BSAVE	Saves memory contents to specified file
BLOAD	Loads file to specified memory in file
MERGE	Merges program with program in file
CHAIN	Loads program contents and executes

10-2 DISK CONFIGURATION

The disks used with the MD-100 are 3.5″ 1DD (single-sided, double-density, double-track) disks. Each disk has tracks numbered from 0 through 79, and each track consists of sectors from 1 through 16. Each sector is capable of storing up to 256 bytes of data.

• 3.5″ single-sided, double-density, double-track disk



Disks do not contain any tracks or sectors when purchased. Tracks and sectors must be written onto blank disks using a procedure known as "formatting" (initializing). This is accomplished with the PB-1000 using the FORMAT command.

10-3 FORMATTING AND INITIALIZATION

A formatted disk is managed as shown below.

- **3.5″ single-sided, double-density, double-track (320KB)**

256 bytes/sector

Track	Sector 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	←FAT→				←Directory→											
1	004				005				006				007			
⋮																
79	13C				13D				13E				13F			

NOTE:
- 004 ~ 13F are cluster numbers (HEX)
- FAT = file allocation table

File data are managed as single clusters of 1024 bytes which are formed by collections of four physical processing units called sectors (256 bytes/sector).
A single disk uses four clusters (16 sectors = 4096 bytes) for a control area, so the data area consists of 316 clusters (1264 sectors = 323,584 bytes). The maximum number of files possible is 192.

1 sector = 256 bytes
1 cluster = 4 sectors (1,024 bytes)
1 track = 4 clusters (4,096 bytes)
1 disk = 80 tracks (327,680 bytes)

The formatting procedure outlined above is required for newly purchased disks and for disks that cannot be accessed for some reason. It should be noted that formatting a disk deletes all data contained on it.

10-4 DIRECTORY

Disk files are written on the disk using one of the tracks (1 ~ 79) or any number of tracks. The filename is written in a directory area provided in track 0. Besides the filename, other data related to each file are also stored in the directory.
The directory is located in an area in track 0, sectors 5 through 16. 16 bytes of control data are required for each file.

16 bytes/directory

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
①	②								③			④	⑤		⑥

① Classification 00H : No data
0DH : Machine language file
10H : BASIC binary file
24H : BASIC ASCII file
Sequential data file
A4H : Random data file

② Filename
③ Extension
④ Not used
⑤ Starting cluster number
⑥ Attribute

The first sector of BASIC and machine language program files are also used as sub-directories.

10-5 SUBDIRECTORY

Subdirectories are created when BASIC and machine language program files are written. Each subdirectory consists of 32 bytes and contains control data for built-in RAM file management. (BASIC files, binary files, machine language files)
An area of one sector (256 bytes) is reserved, but only the first 32 bytes are used.

32 bytes/subdirectory

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
①	②		③		④	⑤								⑥	
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
	⑦								⑧		⑨		⑩		⑪

① Timer boot — Program start flag
- 00H : Normal file
- 01H : Program start file
- 02H : Timer boot file

② File top address
③ File end address
④ Classification (Same as directory classification)
⑤ Filename
⑥ Extension
⑦ Password
⑧ Machine language file top address
⑨ Machine language file end address
⑩ Machine language file execution start address
⑪ Not used

10-6 FILE ALLOCATION TABLE (FAT)

Disks are managed by software in cluster units. The file allocation table (FAT) shows the status of each cluster. This table uses two bytes per cluster.

2 bytes (16 bits)/cluster

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Bit position
①	②	③		④	⑤		⑥									

① 0 : No data
  1 : Data present
② 0 : Sequential cluster
  1 : Last cluster
③ Number of sectors used for last cluster (0 ~ 3)
④ 0 : Random file data present
  1 : No data
⑤ Not used
⑥ Cluster number (Sequential file → Next cluster number; Last file → Present cluster number)

10-7 FILES USED BY DISKS

The files used by disks can be broadly classified as illustrated below:

- Disk files
  - Program files
    - BASIC program files
    - Machine language program files
  - Data files
    - Sequential files
    - Random files

All of these files are called and stored according to their filenames. The format of filenames is as follows:

0 : <Filename>.<Extension> <Attribute>

0 : ....Disk drive device name (usually 0)
Filename.......Up to eight characters
Extension......Up to three characters
Attribute........Only indicated when displayed on screen
B : BASIC program file
S : Sequential file
M : Machine language file
R : Random file
Attributes are appended automatically.

10-7-1 PROGRAM FILE

The program file is created for the storage of BASIC programs and machine language programs on disk. The commands used for saving to the disk are:

SAVE...........BASIC programs
BSAVE.........Machine language programs

"A" can be appended to the SAVE command for ASCII save, which handles files in the same way as sequential files. Loading program files saved using the above commands is accomplished using:

LOAD..........BASIC programs
BLOAD.........Machine language programs

These commands simply read programs from the disk and load them into the memory of the unit. The following commands can be used for automatic direct execution after the programs are loaded into the unit memory:

CHAIN.........BASIC programs
BLOAD "~", R....Machine language programs

These commands can either be used as they are or they can be included inside BASIC programs.

10-7-2 DATA FILES

The data files of the PB-1000 can be either sequential files or random files. Of these, sequential files are taken as having a rather broad meaning, and can be further broken down into the following types:

- Numeric or string data files created in BASIC
- BASIC program files saved as ASCII save
- Assembler source files (source programs)

All of these files are appended with the attribute S when displayed.
Of the files noted above, the most complex are random files, and so will be covered in more detail below.

10-7-3 RANDOM FILE OVERVIEW

Unlike sequential files, each data item (record) in a random file is a fixed size of 256 bytes. Therefore, the number of characters contained in each data item is limited to a size of 256 bytes. These 256 bytes (characters) can be divided into several fields, and data items are managed according to record numbers.
The following sample program has been created to manage the data listed below to serve as an example of random files.

• Data

Name	10 characters	A$, NM$
Telephone	12 characters	B$, TL$
Company	20 characters	C$, CO$
Remarks	30 characters	D$, NT$

• Main variables

J$	Job
RN	Current record number
MX	Total number of records in file

PROGRAM LIST 1

```
10 OPEN "0:ADDRESS" AS #1
20 FIELD #1,10 AS A$,12 AS B$,20 AS C$
30 FIELD #1,@,30 AS D$
40 MX=LOF(1)
50 'SELECT JOB
60 CLS
70 PRINT "Read/Add/Edit/Quit"
80 INPUT "JOB=";J$
90 IF J$="R" OR J$="E" THEN 130
100 IF J$="A" THEN 270
110 IF J$="Q" THEN CLOSE #1:END
120 BEEP:GOTO 50
130 ' READ DATA
140 INPUT "RECORD NUMBER:";RN
150 IF RN<1 OR RN>MX THEN BEEP:GOTO 140
160 GET #1,RN
170 CLS
180 PRINT "RECORD#:";RN;
190 PRINT ",NAME:";A$
200 PRINT "TEL :";B$
210 PRINT "COMPANY:";C$
220 PRINT "REMARKS:";D$;
230 LOCATE 0,0
240 Z$=INPUT$(1):'IF Z$="" THEN 290
250 IF J$="E" THEN 320
260 GOTO 50
270 ' WRITE DATA
280 CLS
290 IF RN<1264 THEN RN=MX+1:GOTO 320
300 BEEP:PRINT "NO SPACE!"
310 FOR I=1 TO 100:NEXT I:GOTO 50
320 CLS:PRINT "RECORD#";RN
330 INPUT "NAME=";NM$
340 IF NM$="" THEN NM$=A$
350 INPUT "TEL:";TL$
360 IF TL$="" THEN TL$=B$
370 INPUT "COMPANY:";CO$
380 IF CO$="" THEN CO$=C$
390 INPUT "REMARKS:";NT$
400 IF NT$="" THEN NT$=D$
410 INPUT "OK(Y/N)";YN$
420 IF YN$="N" THEN 320
430 LSET A$=NM$:LSET B$=TL$
440 LSET C$=CO$:LSET D$=NT$
450 PUT #1,RN
460 GOTO 40
```

10-8 DISK CONTROL BY ASSEMBLER

A thorough understanding of disks and a high level of assembler programming experience are required for machine language control of the disk. A delicate sensitivity is required for including disk programs which differ for each I/O device (i.e. printer, CRT). Even if data are written to the disk, contents are often spoiled, causing a need to reformat the disk (and possibly the loss of important data). Therefore, it is recommended that all important data be saved to another disk before primary attempts at disk programming. Backup copies of disks should also be made to further protect against unforeseen data loss.
Here, sample programs which use assembler to read and write disk sectors will be presented.

• DSKI.ASM
This program reads the contents from a specified track and sector and inputs them into the buffer. Parameters are as follows:

$6 .... Track number ($00_H$ ~ $4F_H$, 0 ~ 79)
$7 .... Sector number (01_H ~ 10_H, 1 ~ 16)

In the example shown here, track 0 sector 1 will be read.

DSKI.ASM

```
0001:0000                ;
0002:0000                ;DSKI Command program.
0003:0000                ;
0004:0000                ;'DSKI.ASM'
0005:0000                ;
0006:7000                        ORG    &H7000
0007:7000                        START  &H7000
0008:7000                ;
0009:7000                FDCOC:  EQU    &HD87C
0010:7000                FDNXT:  EQU    &HD902
0011:7000                ;
0012:7000                ;SET TRACK NUMBER
0013:7000                ;
0014:7000  420600        DSKI:   LD     $6,0
0015:7003                ;
0016:7003                ;SET SECTOR NUMBER
0017:7003                ;
0018:7003  420701                LD     $7,1
0019:7006                ;
0020:7006                ;READ BUFFER ADDRESS TOP
0021:7006                ;
0022:7006  D640406E              PRE    IZ,&H6E40
0023:700A  420180                LD     $1,&H80
0024:700D  777CD8                CAL    FDCOC
0025:7010  026106                LD     $1,$6
0026:7013  777CD8                CAL    FDCOC
```

```
0027:7016  026107          LD     $1,$7
0028:7019  777CD8          CAL    FDCOC
0029:701C  420500          LD     $5,0
0030:701F          ;
0031:701F  7702D9  DSIO1:  CAL    FDNXT
0032:7022  630000          STI    $0,(IZ+0)
0033:7025  490501          SB     $5,1
0034:7028  B48A            JR     NZ,DSIO1
0035:702A          ;
0036:702A  F7              RTN
```

• DSKO.ASM

This program writes the buffer contents to the sector of a specified track. Parameters are as follows:

$6.......Track number

$7.......Sector number

In the example shown here, track 0 sector 1 will be written.

DSKO.ASM

```
0001:0000                  ;
0002:0000                  ;DSKO Command program.
0003:0000                  ;
0004:0000                  ;'DSKO.ASM'
0005:0000                  ;
0006:7000                          ORG    &H7000
0007:7000                          START  &H7000
0008:7000                  ;
0009:7000                  FDCOC:  EQU    &HD87C
0010:7000                  FDLU:   EQU    &HD852
0011:7000                  FDCOM:  EQU    &HD904
0012:7000                  ;
0013:7000                  ;SET TRACK NUMBER
0014:7000                  ;
0015:7000  420600          DSKO:   LD     $6,0
0016:7003                  ;
0017:7003                  ;SET SECTOR NUMBER
0018:7003                  ;
0019:7003  420701                  LD     $7,1
0020:7006                  ;
0021:7006                  ;OUTPUT DATA START ADDRESS
0022:7006                  ;
0023:7006  D640406E                PRE    IZ,&H6E40
0024:700A  420170                  LD     $1,&H70
0025:700D  777CD8                  CAL    FDCOC
0026:7010  D1010301                LDW    $1,259
0027:7014  7752D8                  CAL    FDLU
0028:7017  026106                  LD     $1,$6
0029:701A  777CD8                  CAL    FDCOC
```

```
0030:701D  026107              LD    $1,$7
0031:7020  777CD8              CAL   FDCOC
0032:7023  420300              LD    $3,0
0033:7026            ;
0034:7026  6B0100    DSKO1:    LDI   $1,(IZ+0)
0035:7029  7704D9              CAL   FDCOM
0036:702C  490301              SB    $3,1
0037:702F  B48A                JR    NZ,DSKO1
0038:7031            ;
0039:7031  F7                  RTN
```

• **DSKF.ASM**

This program displays the remaining (unused) capacity of a disk.

DSKF.ASM

```
0001:0000            ;
0002:0000            ;DSKF Command program.
0003:0000            ;
0004:0000            ;'DSKF.ASM'
0005:0000            ;
0006:0000            ;REMAINING NUMBER OF CLUSTERS
0007:0000            ;1 CLUSTER=4 SECTORS
0008:0000            ;
0009:7000                      ORG   &H7000
0010:7000                      START &H7000
0011:7000            ;
0012:7000            FDCOC:    EQU   &HD87C
0013:7000            FDNXT:    EQU   &HD902
0014:7000            CONVR:    EQU   &H9AF4
0015:7000            STR:      EQU   &HA555
0016:7000            OUTAC:    EQU   &HFF9E
0017:7000            ;
0018:7000  4201D0    DSKF:     LD    $1,&HD0
0019:7003  777CD8              CAL   FDCOC
0020:7006  7702D9              CAL   FDNXT
0021:7009  7702D9              CAL   FDNXT
0022:700C  026F00              LD    $15,$0 ;Lu
0023:700F  7702D9              CAL   FDNXT
0024:7012  027000              LD    $16,$0 ;Up
0025:7015            ;
0026:7015            ;DISPLAY FORMATION
0027:7015            ;
0028:7015  77F49A              CAL   CONVR
0029:7018  7755A5              CAL   STR
0030:701B  960F                PRE   IX,$15
0031:701D  D6400C69            PRE   IZ,&H690C
0032:7021  611F00              ST    $31,(IZ+0)
0033:7024            ;
```

```
0034:7024  6A1000      DSKF1:  LDI    $16,(IX+0)
0035:7027  779EFF              CAL    OUTAC
0036:702A  491101              SB     $17,1
0037:702D  B48A                JR     NZ,DSKF1
0038:702F              ;
0039:702F  F7                  RTN
```

10-8-1 APPLICATION EXAMPLES

The DSKI.ASM program given in the previous section can be used to read the disk directory. The programs in this section are used to produce a hardcopy of the directory on the printer. These programs are:

LFILES-N..........BASIC program for standard Centronics printer
LFILES-P..........BASIC program for plotter printer (FP-100)
LFILES.ASM.....Assembler program for reading directory

The programs LFILES-N + LFILES.ASM (.EXE) are used for the standard Centronics printer, while LFILES-P + LFILES.ASM (.EXE) are used for the plotter printer (FP-100).

LFILES-N

```
10 '
20 'PRINTOUT OF DIRECTORY ON
30 'STANDARD PRINTER
40 '
50 'LFILES-N.
60 '
70 'RESERVE MACHINE LANGUAGE AREA
80 CLEAR ,200
90 POKE &H7050,0
100 '
110 FOR S=5 TO 16
120  POKE &H7051,S
130  CALL "LFILES.EXE"
140  FOR I=&H6E40 TO &H6E40+255 STEP16
150   A=PEEK(I)
160 ' M:MACHINE LANGUAGE FILE
170   IF A=&H0D THEN S$="M":GOTO 260
180 ' B:BASIC PROGRAM FILE
190   IF A=&H10 THEN S$="B":GOTO 260
200 ' S:SEQUENTIAL FILE OR SOURCE PROGRAM FILE
210   IF A=&H24 THEN S$="S":GOTO 260
220 ' R:RANDOM FILE
230   IF A=&HA4 THEN S$="R":GOTO 260
240   GOTO 330
250 '
260   FOR J=I+1 TO I+11
270    D$=D$+CHR$(PEEK(J))
280    IF(J-I)=8 THEN D$=D$+" "
```

```
290  NEXT J
300   LPRINT D$;" ";S$"     ";
310   X=X+1:IF X=2 THEN X=0:LPRINT
320   D$="":S$=""
330  NEXT I
340 NEXT S
350 LPRINT
360 END
```

LFILES-P

```
10 '
20 'PRINTOUT OF DIRECTORY ON
30 'PLOTTER PRINTER
40 '
50 'LFILES-P.
60 '
70 'SET PLOTTER TO TEXT MODE
80 LPRINT CHR$(28);CHR$(37)
90 'RESERVE MACHINE LANGUAGE AREA
100 CLEAR ,300
110 POKE &H7050,0
120 '
130 FOR S=5 TO 16
140  POKE &H7051,S
150  CALL "LFILES.EXE"
160  FOR I=&H6E40 TO &H6E40+255 STEP16
170   A=PEEK(I)
180 ' M:MACHINE LANGUAGE FILE
190   IF A=&H0D THEN S$="M":GOTO 270
200 ' B:BASIC PROGRAM FILE
210   IF A=&H10 THEN S$="B":GOTO 270
220 ' S:SEQUENTIAL OR SOURCE PROGRAM FILE
230   IF A=&H24 THEN S$="S":GOTO 270
240 ' R:RANDOM FILE
250   IF A=&HA4 THEN S$="R":GOTO 270
260   GOTO 420
270 '
280   FOR J=I+1 TO I+11
290    D$=D$+CHR$(PEEK(J))
300    IF(J-I)=8 THEN D$=D$+"."
310   NEXT J
320 'FILE 'B'=BLACK
330   IF RIGHT$(S$,1)="B" THEN LPRINT "J0"
340 'FILE 'S'=BLUE
350   IF RIGHT$(S$,1)="S" THEN LPRINT "J1"
360 'FILE 'M'=GREEN
370   IF RIGHT$(S$,1)="M" THEN LPRINT "J2"
380 'FILE 'R'=RED
390   IF RIGHT$(S$,1)="R" THEN LPRINT "J3"
```

```
400   LPRINT "P";D$;" ";S$;"     "
405   X=X+1:IF X=2 THEN X=0:LPRINT "R-91.2,-5"
410   D$="":S$=""
420  NEXT I
430 NEXT S
440 '
450 LPRINT CHR$(28);CHR$(46)
460 '
470 END
```

LFILES.ASM

```
0001:0000                 ;
0002:0000                 ;LFILES Command program.
0003:0000                 ;
0004:0000                 ;'LFILES.ASM'
0005:0000                 ;
0006:7000                         ORG    &H7000
0007:7000                         START  LFILE
0008:7000                 ;
0009:7000                 FDCOC:  EQU    &HD87C
0010:7000                 FDNXT:  EQU    &HD902
0011:7000                 ;
0012:7000   D6005070      LFILE:  PRE    IX,&H7050
0013:7004                 ;
0014:7004                 ;LOAD TRACK # FROM &H7050
0015:7004                 ;
0016:7004   680600                LD     $6,(IX+0)
0017:7007                 ;
0018:7007                 ;LOAD SECTOR # FROM &H7051
0019:7007                 ;
0020:7007   680701                LD     $7,(IX+1)
0021:700A                 ;
0022:700A                 ;READ BUFFER TOP ADDRESS
0023:700A                 ;
0024:700A   D640406E              PRE    IZ,&H6E40
0025:700E   420180                LD     $1,&H80
0026:7011   777CD8                CAL    FDCOC
0027:7014   026106                LD     $1,$6
0028:7017   777CD8                CAL    FDCOC
0029:701A   026107                LD     $1,$7
0030:701D   777CD8                CAL    FDCOC
0031:7020   420500                LD     $5,0
0032:7023                 ;
0033:7023   7702D9        DSIO1:  CAL    FDNXT
0034:7026   630000                STI    $0,(IZ+0)
0035:7029   490501                SB     $5,1
0036:702C   B48A                  JR     NZ,DSIO1
0037:702E                 ;
0038:702E   F7                    RTN
```

Execution Example

```
RS-232C .    B       SAMPLE54.ASM S
PRN1    .ASM S       PRN1    .EXE M
PRN2    .ASM S       PRN2    .EXE M
CIRCLE  .    B       SAMPLE52.ASM S
SAMPLE51.ASM S       SAMPLE53.ASM S
ANIME   .    B       INKEY   .ASM S
INKEY   .EXE M       SHIFT   .ASM S
SHIFT   .    B       SHIFT   .EXE M
PRN3    .ASM S       PRN3    .EXE M
BANK    .ASM S       BANK    .EXE M
```

CHAPTER 11

BANKING

11-1 BANK SELECT APPLICATIONS

The external ROM and expansion RAM (RP-32) of the PB-1000 use the same free area on the memory map. The CPU (HD61700) utilizes a technique known as "banking" to switch between ROM and RAM and to prevent errors caused by using the same address. Banking can also be performed by the user, using one of two banking methods:

- Using the monitor B command
- Using the assembler

11-1-1 MONITOR B COMMAND

Pressing the CAL key of the PB-1000 causes the screen to clear and the cursor to appear. Pressing MON EXE at this time enters the monitor mode. Here we will use the B command to view the ROM area 9000_H through $900F_H$.
Entering the B command in response to the monitor prompt (>) displays the current bank area (1 to indicate RAM).

```
>B
1 -
```

Enter 0 to switch to the ROM area, followed by D9000, 900F EXE to display the contents of 9000_H through $900F_H$. At this time, the following should appear on the display.

```
>D9000, 900F
9000  00  C7  68  22  1F  9E  02  77
9008  2A  90  D1  00  80  0A  64  00
>
```

Pressing the CAL key again causes the unit to return to its original status with the bank automatically switching back to RAM.

11-1-2 ASSEMBLER

The bank can be switched using the assembler by changing the contents of the UA registers. Of the UA registers, the high-order bit of the IX register has the greatest significance. There are two such bits, but only a combination of 00 and 01 is permitted. 00 is used for ROM, and 01 is used for RAM.
The following sample program BANK.ASM transfers the contents of the ROM area 9000_H through $900F_H$ to the RAM area 7100_H through $710F_H$. This method can be used to disassemble ROM data.

This program masks the other bits in order to clearly show the significance of the high-order bit of the UA register (lines 23 through 28). Accomplished assembler programmers may also wish to directly perform an AND operation for each of the bit patterns. Eliminating line 28 causes automatic banking of the ROM area.

BANK.ASM

```
0001:0000                  ;
0002:0000                  ;BANK SELECT SAMPLE PROGRAM
0003:0000                  ;
0004:0000                  ; "BANK.ASM"
0005:0000                  ;
0006:0000                  ;TRANSFER ROM AREA(9000H-900FH)
0007:0000                  ;TO RAM AREA(7100H-710FH)
0008:0000                  ;
0009:7000                        ORG   &H7000
0010:7000                        START BNK
0011:7000                  ;
0012:7000                  ;UA REGISTER BANK SWITCHING
0013:7000                  ;
0014:7000                  ;PREPARATION FOR TRANSFER
0015:7000                  ;OF 15 BYTES
0016:7000                  ;
0017:7000  42010F          BNK:  LD    $01,&H0F
0018:7003                  ;
0019:7003                  ;SET HIGH-ORDER OF X REGISTER
0020:7003                  ;(SWITCH UA REGISTER)
0021:7003                  ;
0022:7003  1E60                  GST   UA,$00
0023:7005  4C00CF                AN    $00,&HCF
0024:7008                  ;
0025:7008                  ;DETERMINE AREA
0026:7008                  ;&H00=ROM AREA,&H10=RAM AREA
0027:7008                  ;
0028:7008  4E0000                OR    $00,&H00
0029:700B                  ;
0030:700B                  ;RESET UA REGISTER
0031:700B                  ;
0032:700B  1660                  PST   UA,$00
0033:700D                  ;
0034:700D                  ;SET LOAD ADDRESS
0035:700D                  ;
0036:700D  D6000090              PRE   IX,&H9000
0037:7011  D6400071              PRE   IZ,&H7100
0038:7015                  ;
0039:7015                  ;START TRANSFER
0040:7015                  ;
0041:7015  286201          LOOP: LD    $02,(IX+$01)
0042:7018  216201                ST    $02,(IZ+$01)
0043:701B  490101                SB    $01,1
0044:701E  B48A                  JR    NZ,LOOP
```

```
0045:7020                  ;
0046:7020                  ;SET UA REGISTER IN RAM AREA
0047:7020                  ;
0048:7020   1E60           GST    UA,$00
0049:7022   4C00DF         AN     $00,&HDF
0050:7025   1660           PST    UA,$00
0051:7027                  ;
0052:7027   F7             RTN
```

DUMP PROGRAM

```
100 '
110 'DUMP MEMORY CONTENTS
120 '"DUMP."
130 '
140 'ENTER ADDRESS IN DECIMAL OR HEX
150 '
160 'ENTER ADDRESS EXCEEDING &H8000
170 'IN DECIMAL
180 '
190 '
200 INPUT "START ADDRESS=";A1
210 INPUT "END   ADDRESS=";A2
220 IF A1>=0 AND A2>=0 THEN 260
230 BEEP
240 PRINT "NEGATIVE ADDRESS!"
250 GOTO 200
260 IF A1<=A2 THEN 300
270 BEEP
280 PRINT "REVERSED ADDRESS SEQUENCE!"
290 GOTO 200
300 INPUT "DEVICE (1)CRT,(2)PRINTER";M
310 IF M=1 OR M=2 THEN 350
320 BEEP
330 PRINT "SELECT 1 OR 2"
340 GOTO 300
350 AD=A1:C=0:D=8
360 FOR I=A1 TO A2
370 IF C<>0 THEN 400
380 IF M=1 THEN PRINT HEX$(AD);":";
390 IF M=2 THEN LPRINT HEX$(AD);":";
400 A=PEEK(I)
410 IF M=1 THEN PRINT RIGHT$(HEX$(A),2);" ";
420 IF M=2 THEN LPRINT RIGHT$(HEX$(A),2);" ";
430 C=C+1
440 IF C<>D THEN 480
450 C=0:AD=AD+D
460 IF M=1 THEN PRINT
470 IF M=2 THEN LPRINT
480 A$=INKEY$:IF M=2 THEN 500
```

```
490 IF A$="" THEN 480
500 NEXT I
510 IF M=1 THEN PRINT
520 IF M=2 THEN LPRINT
530 END
```

Operation Example

```
RUN
START ADDRESS=?&H7000
END  ADDRESS=?&H70FF
DEVICE (1)CRT,(2)PRINTER?1
```

Output Example when D = 8

```
7000:3A 41 34 34 43 45 20 28
7008:31 29 43 52 54 2C 28 32
7010:29 50 52 49 4E 54 45 52
7018:45 52 2A 2A 2A 2A 9F B2
7020:A5 DE 9F 30 20 4C 49 4E
7028:45 20 49 4E 50 55 54 20
7030:23 32 2C 41 24 0D 0A 36
7038:33 30 20 49 46 20 41 24
7040:3D 22 2F 2F 45 4E 44 22
7048:20 54 48 45 4E 20 36 37
7050:30 3A 52 45 4D 20 45 4E
7058:44 20 4F 46 20 52 58 0D
7060:0A 36 34 30 20 50 52 49
7068:4E 54 20 23 31 2C 41 24
7070:0D 0A 36 35 30 20 49 46
7078:20 4D 3D 31 20 54 48 45
7080:4E 20 50 52 49 4E 54 20
7088:41 24 0D 0A 36 36 30 20
7090:47 4F 54 4F 20 36 32 30
7098:0D 0A 36 37 30 20 43 4C
70A0:4F 53 45 0D 0A 36 38 30
70A8:20 50 52 49 4E 54 20 43
70B0:48 52 24 28 37 29 3B 22
70B8:50 52 45 53 53 20 41 4E
70C0:59 20 4B 45 59 2E 22 0D
70C8:0A 36 39 30 20 58 24 3D
70D0:49 4E 4B 45 59 24 3A 49
70D8:46 20 58 24 3D 22 22 20
70E0:54 48 45 4E 20 36 39 30
70E8:0D 0A 37 30 30 20 52 45
70F0:54 55 52 4E 0D 0A 3A 22
70F8:2B 50 52 4D 24 20 41 53
```

NOTE: Resultant output data not limited to output shown here.

Output Example when D = 16

```
7000:3A 41 34 34 43 45 20 28 31 29 43 52 54 2C 28 32
7010:29 50 52 49 4E 54 45 52 45 52 2A 2A 2A 2A 9F B2
7020:A5 DE 9F 30 20 4C 49 4E 45 20 49 4E 50 55 54 20
7030:23 32 2C 41 24 0D 0A 36 33 30 20 49 46 20 41 24
7040:3D 22 2F 2F 45 4E 44 22 20 54 48 45 4E 20 36 37
7050:30 3A 52 45 4D 20 45 4E 44 20 4F 46 20 52 58 0D
7060:0A 36 34 30 20 50 52 49 4E 54 20 23 31 2C 41 24
7070:0D 0A 36 35 30 20 49 46 20 4D 3D 31 20 54 48 45
7080:4E 20 50 52 49 4E 54 20 41 24 0D 0A 36 36 30 20
7090:47 4F 54 4F 20 36 32 30 0D 0A 36 37 30 20 43 4C
70A0:4F 53 45 0D 0A 36 38 30 20 50 52 49 4E 54 20 43
70B0:48 52 24 28 37 29 3B 22 50 52 45 53 53 20 41 4E
70C0:59 20 4B 45 59 2E 22 0D 0A 36 39 30 20 58 24 3D
70D0:49 4E 4B 45 59 24 3A 49 46 20 58 24 3D 22 22 20
70E0:54 48 45 4E 20 36 39 30 0D 0A 37 30 30 20 52 45
70F0:54 55 52 4E 0D 0A 3A 22 2B 50 52 4D 24 20 41 53
```

NOTE: Resultant output data not limited to output shown here.

In the previous program, the MON command was used to confirm on the monitor whether ROM contents were correctly transferred to RAM. Here, RAM contents are output to the printer for confirmation. This program includes a BASIC routine, so modifications can be made easily.

• Variable D

This variable controls the number of columns. A value of 8 should be used for the display, and 16 for the printer (plotter or 80-column printer).

• Important

Address input can be performed using either decimal or hexadecimal values. Values exceeding 8000_H are handled as negative values, and so should be entered in decimal only. Execution is suspended by pressing any key during printout/display. Pressing a key again resumes execution.

CHAPTER 12

SYSTEM CALL REFERENCE

LABEL: NEXTC ADDRESS: 00F9H (249)

Purpose: Starts search from program address specified by IX. Assigns any code detected, except for space (20H), to $0.

Input parameters: IX......Program pointer

Output parameters: IX......Address of code assigned to $0
$0......Found code

LABEL: OKNM1 ADDRESS: 010AH (266)

Purpose: Sets flag register carry (to 1) when $0 contents are numeral (ASCII code 30H ~ 39H, 48H ~ 57H).

Input parameters: $0......Code being checked

Output parameters: $0......Code
FLG...Carry flag 1 = Numeral

LABEL: OKHX1 ADDRESS: 0116H (278)

Purpose: Sets flag register carry (to 1) when $0 contents are characters 0 ~ 9, A ~ F (30H ~ 39H, 41H ~ 46H).

Input parameters: $0......Code being checked

Output parameters: $0......Code
FLG...Carry flag 1 = 0 ~ 9, A ~ F

LABEL: OKAM1 ADDRESS: 011DH (285)

Purpose: Sets flag register carry (to 1) when $0 contents are upper case alphabetic characters (A ~ Z).

Input parameters: $0......Code being checked

Output parameters: $0......Code
FLG...Carry flag 1 = A ~ Z

LABEL: TCAPS ADDRESS: 015BH (347)

Purpose: Converts lower case alphabetic character codes contained in $0 to upper case codes. Other codes not converted.

Input parameters: $0......Lower case alphabetic character code

Output parameters: $0......Upper case alphabetic character code

LABEL: CHEX1 ADDRESS: 0160H (352)

Purpose: Converts $0 contents to binary data (00H ~ 0FH) when it contains characters 0 ~ 9, A ~ F, a ~ f (30H ~ 39H, 41H ~ 46H, 61H ~ 66H).

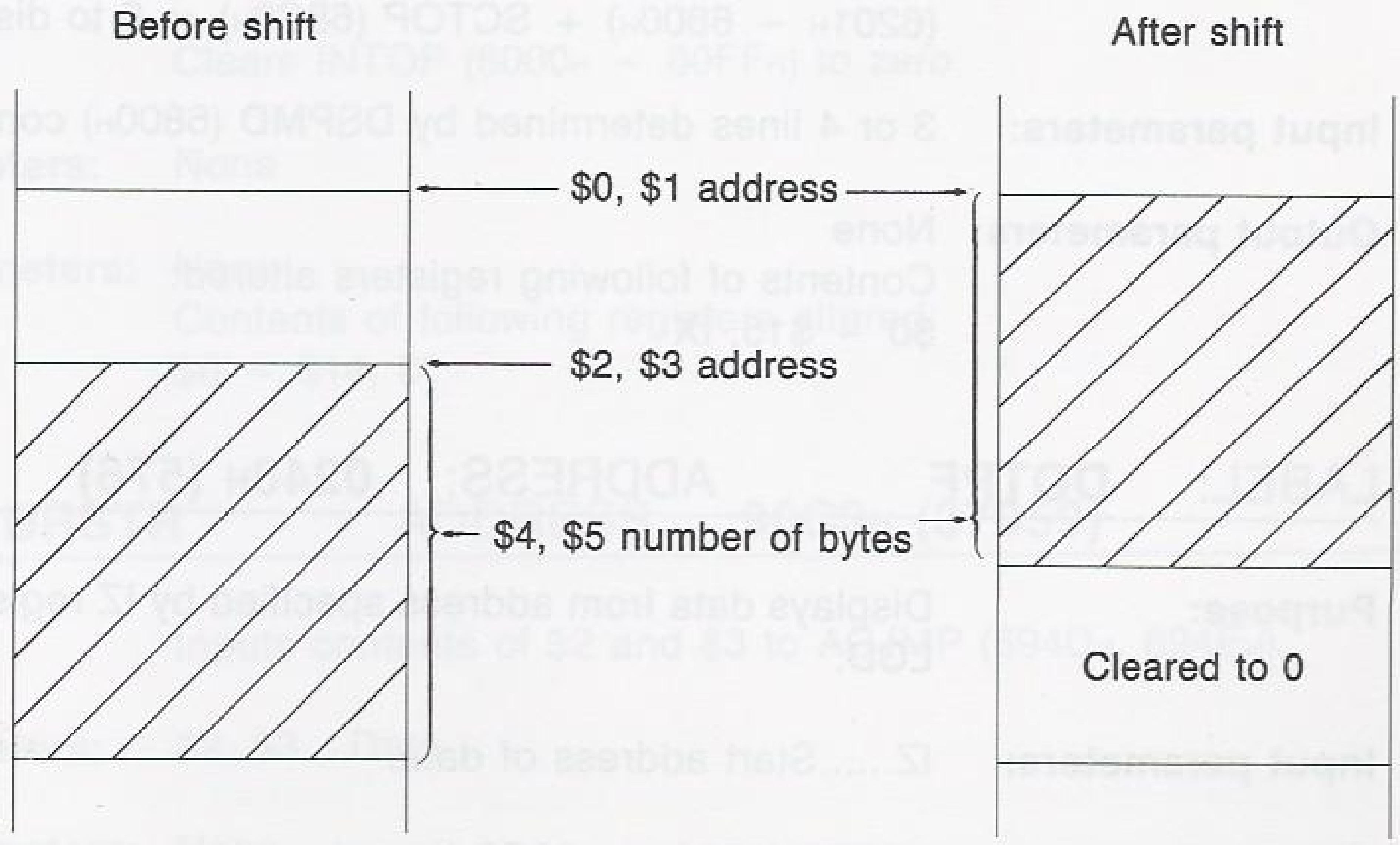
Input parameters: $0......Code to be converted

Output parameters: $0......Converted binary data
FLG...Flag register carry set (to 1) after conversion

LABEL: SHFTM ADDRESS: 016CH (364)

Purpose: Shifts the number of bytes specified by $4 and $5 from address specified by $2 and $3 to address specified by $0 and $1. Original area cleared to 0.

Example



Input parameters: \$0, \$1....Shift destination address
\$2, \$3....Shift origin start address
\$4, \$5....Number of bytes

Output parameters: None
Contents of following registers altered:
\$0 ~ \$14, IX, IZ

LABEL: CLRME ADDRESS: 016EH (366)

Purpose: Clears the number of bytes specified by \$4 and \$5 starting from the address specified by \$2 and \$3. No execution when contents of \$4 and \$5 equal zero.

Input parameters: \$2, \$3....Start address
\$4, \$5....Number of bytes

Output parameters: IX......Last address cleared + 1
\$5 ~ \$13...All cleared to 0
Contents of following registers altered:
\$2 ~ \$4, \$14

LABEL: DOTDS ADDRESS: 022CH (556)

Purpose: Full-screen display. Transfers 3 or 4 line contents from LEDTP (6201H ~ 6800H) + SCTOP (68C9H) × 6 to display driver.

Input parameters: 3 or 4 lines determined by DSPMD (6800H) contents.

Output parameters: None
Contents of following registers altered:
\$0 ~ \$15, IX

LABEL: DOTPF ADDRESS: 0240H (576)

Purpose: Displays data from address specified by IZ register on line 4 of LCD.

Input parameters: IZ......Start address of data.

Output parameters: LPFTP (6801H ~ 68C0H) used
Contents of following registers altered:
\$0 ~ \$15, IX, IZ

LABEL: CLEDB ADDRESS: 04E7H (1255)

Purpose: Clears contents of EDTOP (6100H ~ 6200H) and LEDTP (6201H ~ 6800H) to zero and sets each pointer at CLS.

Output parameters: IX......EDCSR (68C8H) contents
IZ......MOEDB (68CCH) contents
$5 ~ $14....All cleared to 0
Contents of following registers altered:
$0 ~ $4

LABEL: KBM16 ADDRESS: 0BB1H (2993)

Purpose: Produces product of integer values *x* and *y*.
However: $0 \leqq x, y \leqq 65535$

Input parameters: $5, $6.........Integer value *x*
$15, $16.....Integer value *y*

Output parameters: $0, $1........Product ($x \times y$)
FLG...........Flag register carry set (to 1) when product exceeds 65535.
Contents of following registers altered:
$5, $6, $15, $16

LABEL: INCLR ADDRESS: 901FH (36895)

Purpose: Clears INTOP (6000H ~ 60FFH) to zero.

Input parameters: None

Output parameters: None
Contents of following registers altered:
$0 ~ $14, IX

LABEL: BRSTR ADDRESS: 90C3H (37059)

Purpose: Inputs contents of $2 and $3 to ACJMP (694DH, 694EH).

Input parameters: $2, $3....Data

Output parameters: None
Contents of following registers altered:
IX, IZ

LABEL: **CRTKY** ADDRESS: **9158H (37208)**

Purpose: Key sample flow of contrast key execution.
Break key jumps to address specified by ACJMP (694D, 694E).
Touch keys 0 ~ 12 disregarded.

Input parameters: None

Output parameters: $0......Key code input. See following chart for key codes.
Contents of following registers altered:
$1 ~ $11, IX, IZ

PRINT CODE TABLE

HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		(ROLL DOWN)	SPACE	0	@	P	`	p		MEMO			タ	ミ		
1	(ROLL UP)	(DEL)	!	1	A	Q	a	q				ア	チ	ム		
2	(LINE TOP)	(INS)	"	2	B	R	b	r		ANS		イ	ツ	メ		
3			#	3	C	S	c	s		OUT		ウ	テ	モ		
4			$	4	D	T	d	t		IN		エ	ト	ヤ		
5	(LINE DEL)		%	5	E	U	e	u		LC		オ	ナ	ユ		
6	(LINE END)		&	6	F	V	f	v		STOP	ヲ	カ	ニ	ヨ		
7	(SEL)		'	7	G	W	g	w	■	CONT	ァ	キ	ヌ	ラ		
8	(BS)	(LINE C)	(	8	H	X	h	x		ENG	ィ	ク	ネ	リ	♠	
9	(TAB)		)	9	I	Y	i	y		←ENG	ゥ	ケ	ノ	ル	♥	
A			*	:	J	Z	j	z		NEW ALL	ェ	コ	ハ	レ	♦	
B	(HOME)		+	;	K	[	k	{		CALC	ォ	サ	ヒ	ロ	♣	
C	(CLS)	→	,	<	L	¥	l	|		MENU	ャ	シ	フ	ワ	●	TK13
D	(CR LF)	←	–	=	M	]	m	}		CAL	ュ	ス	ヘ	ン	○	TK14
E		↑	.	>	N	^	n	~		MEMO IN	ョ	セ	ホ	゛		TK15
F		↓	/	?	O	_	o				ッ	ソ	マ	゜	＼	TK16

Blanks indicate no output.
Indefinite character output by FF code when power is ON.
Configuration of characters output by E0 ~ FF are defined by the DEFCHR$ statement.

TK1	TK2	TK3	TK4
TK5	TK6	TK7	TK8
TK9	TK10	TK11	TK12
TK13	TK14	TK15	TK16

DISP

Call &H9429 to display one-key commands, &H01D9 for NOP. Contents of $00 checked after call of CRTKY. Key entered on keyboard treated as one-key command when $00 contents equal 0. Then calling &H9429 display one-key command (GOTO, INPUT, etc.). One-key command not displayed when &H01D9 is called after CRTKY call. Doing this causes key commands in key buffer to be disregarded.

LABEL: KYCHK ADDRESS: 91E2H (37346)

Purpose: Checks OFF, C.BOOT, BREAK and STOP keys.

Input parameters: None

Output parameters: FLG....Flag register Z set (to 1) when STOP key is pressed.
Contents of following registers altered:
$0 ~ $4

LABEL: BKCK ADDRESS: 9207H (37383)

Purpose: Checks OFF and C.BOOT keys, and samples BREAK key.

Input parameters: None

Output parameters: None
Contents of following registers altered:
$0 ~ $4

LABEL: OUTCR ADDRESS: 95CEH (38350)

Purpose: Outputs ODH, OAH to a device.

Input parameters: Device determined by OUTDV (690CH) contents.

Output parameters: None
Contents of following registers altered:
$0 ~ $13, $16, IZ

LABEL: PROUT ADDRESS: 961FH (38431)

Purpose: Outputs contents of $16 to printer. NR error generated if printer not connected. Checks for busy signal before and after printer output.

Input parameters: $16......Output data

Output parameters: None
Contents of following registers altered:
$0 ~ 6, IZ

LABEL: PRLB1 ADDRESS: 9664H (38500)

Purpose: Outputs to printer number of bytes specified by \$17 and \$18 starting from address specified by \$15 and \$16.

Input parameters: \$15, \$16....Output data (string) start address
\$17, \$18....Number of bytes of output data

Output parameters: None
Contents of following registers altered:
\$0 ~ \$19, IX, IZ

LABEL: DTBIN ADDRESS: 9828H (38952)

Purpose: Converts ASCII code presently located at address specified by IX register to binary.
OV error generated when conversion result exceeds 65536.
Immediate return to main routine and zero returned when numeric value (30H ~ 39H) not present at specified address. At this time, spaces are skipped.

Input parameters: IX............Address containing data to be converted

Output parameters: IX............Address containing data other than 30H ~ 39H
\$17, \$18...Converted binary data
Contents of following registers altered:
\$0 ~ \$3, \$16

LABEL: BIN01 ADDRESS: 98B1H (39089)

Purpose: Converts real number data x in \$10 ~ \$18 to integer value when $0 \leqq x < 256$. BS error generated when x is outside range.

Input parameters: \$10 ~ \$18....Real number value

Output parameters: \$15, \$16........Integer value
Contents of following registers altered:
\$10 ~ \$14, \$17, \$18, IZ

LABEL: BIN02 ADDRESS: 98CDH (39117)

Purpose: Converts real number data x in \$10 ~ \$18 to integer value when $0 \leqq x < 65536$. BS error generated when x is outside range.

Input parameters: \$10 ~ \$18...Real number value

Output parameters: \$15, \$16.......Integer value
Contents of following registers altered:
\$10 ~ \$14, \$17, \$18, IZ

LABEL: BIN11 ADDRESS: 98B9H (39097)

Purpose: Converts real number data x in \$10 ~ \$18 to integer value when $1 \leqq x < 256$. BS error generated when x is outside range.

Input parameters: \$10 ~ \$18....Real number value

Output parameters: \$15, \$16........Integer value
Contents of following registers altered:
\$10 ~ \$14, \$17, \$18, IZ

LABEL: BIN12 ADDRESS: 98D3H (39123)

Purpose: Converts real number data x in \$10 ~ \$18 to integer value when $1 \leqq x < 65536$. BS error generated when x is outside range.

Input parameters: \$10 ~ \$18....Real number value

Output parameters: \$15, \$16........Integer value
Contents of following registers altered:
\$10 ~ \$14, \$17, \$18, IZ

LABEL: BINM2 ADDRESS: 98A6H (39078)

Purpose: Converts real number data x in \$10 ~ \$18 to integer value when $-32769 < x < 65536$. BS error generated when x is outside range.

Input parameters: \$10 ~ \$18....Real number value

Output parameters: \$15, \$16..........Integer value
Contents of following registers altered:
\$10 ~ \$14, \$17, \$18, IZ

LABEL: SIKI ADDRESS: 98E5H (39141)

Purpose: Executes expression (including character expressions) and returns result.
Numeric value results stored in $10 ~ $18 as real numbers.
Character results stored in RAM free area. $15 and $16 hold string start address, and $17 holds string length.

Input parameters: IX......Start address in RAM where expression is located.
Reserved words in expression must be converted to internal code.

Output parameters: IX......Expression end address + 1
• Numeric value results
$10 ~ $18....Real number value
FLG..............Flag register carry cleared (to 0).
• String results
$15, $16.......Result string start address
$17...............String length
FLG..............Flag register carry set (to 1).
Contents of following registers altered:
$0 ~ $9, $19 ~ $29

LABEL: EXPRW ADDRESS: 9989H (39305)

Purpose: Executes numeric expression and returns result. Numeric value results stored in $10 ~ $18 as real numbers.

Input parameters: IX......Start address in RAM where numeric expression is located.
Reserved words in expression must be converted to internal code.

Output parameters: IX......Expression end address + 1
$10 ~ $18....Real number value
Contents of following registers altered:
$0 ~ $9, $19 ~ $29, IZ

LABEL: NISIN ADDRESS: 9A3FH (39487)

Purpose: Converts BCD value in $17 to binary.

Input parameters: $17......BCD value

Output parameters: $17......Binary
Contents of following register altered:
$19

LABEL: CNVR ADDRESS: 9AF4H (39668)

Purpose: Converts integer in \$15 and \$16 to real number.

Input parameters: \$15, \$16....Integer value

Output parameters: \$10 ~ \$18....Real number
Contents of following registers altered:
\$0 ~ \$4

LABEL: NCP ADDRESS: 9BB7H(39863)

Purpose: Compares real numbers *y* in \$0 ~ \$8 with real numbers *x* in \$10 ~ \$18.

Input parameters: \$0 ~ \$8.......Real numbers *y*
\$10 ~ \$18....Real numbers *x*

Output parameters: FLG

$x = y$	Flag register Z set (to 1)
$x < y$	Flag register carry set (to 1)
$x > y$	Flag register carry cleared (to 0)

LABEL: MCP ADDRESS: 9BD1H (39889)

Purpose: Compares string *x* with string *y*.

Input parameters: \$5, \$6.......Start address of string *y*
\$7.............String *y* length
\$15, \$16...Start address of string *x*
\$17...........String *x* length

Output parameters: FLG

$x = y$	Flag register Z set (to 1)
$x < y$	Flag register carry set (to 1)
$x > y$	Flag register carry cleared (to 0)

Contents of following registers altered:
\$0 ~ \$3, \$7, \$17, IZ

LABEL: **BXWY** ADDRESS: **9C99H (40089)**

Purpose: Returns quotient for $x \div y$, cutting off remainder. x and y are integers in range of $-32768 \leqq (x, y) \leqq 32767$

Input parameters: $5, $6...........Integer value *y*
$15, $16.......Integer value *x*

Output parameters: $15, $16....Quotient
FLG...........Flag register set (to 1) and quotient returned as 32768 when calculation results in overflow.
Contents of following registers altered:
$0 ~ $2, $4 ~ $6

LABEL: **INKEY** ADDRESS: **9E3BH (40507)**

Purpose: INKEY$ subroutine.

Input parameters: None

Output parameters: $15, $16....Address of key input data (same as WORK1 68F4H value).
$17...........0 = No key input
1 = Key input present
Contents of following registers altered:
$0 ~ $4, $18, IZ

The following table shows the key and the corresponding key codes which can be fetched using this routine.

PRINT CODE TABLE

HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		(ROLL DOWN)	SPACE	0	@	P		p								TK1
1	(ROLL UP)	(DEL)	!	1	A	Q	a	q								TK2
2	(LINE TOP)	(INS)	"	2	B	R	b	r								TK3
3			#	3	C	S	c	s								TK4
4			$	4	D	T	d	t								TK5
5	(LINE DEL)		%	5	E	U	e	u								TK6
6	(LINE END)		&	6	F	V	f	v								TK7
7	(SEL)		'	7	G	W	g	w	■							TK8
8	(BS)	(LINE C)	(	8	H	X	h	x							♠	TK9
9	(TAB)		)	9	I	Y	i	y							♥	TK10
A			*	:	J	Z	j	z							♦	TK11
B	(HOME)		+	;	K	[	k	{							♣	TK12
C	(CLS)	→	,	<	L	¥	l	|							●	TK13
D	(CR LF)	←	−	=	M	]	m	}							○	TK14
E		↑	.	>	N	^	n	~								TK15
F		↓	/	?	O	_	o								╲	TK16

Blanks indicate no output.
Indefinite character output by FF code when power is ON.
Configuration of characters output by E0 ~ FF are defined by the DEFCHR$ statement.

TK1	TK2	TK3	TK4
TK5	TK6	TK7	TK8
TK9	TK10	TK11	TK12
TK13	TK14	TK15	TK16

DISP

LABEL: ENLST ADDRESS: B0C5H (45253)

Purpose: Converts one line of BASIC program stored in internal code to ASCII code starting from address specified by IX, and stores result in INTOP (6000H ~ 60FFH).

Input parameters: IX......Start address of BASIC program line to be converted

Output parameters: IX......Next line start address or program end
Contents of following registers altered:
$0 ~ $16, IZ

LABEL: NBFMK ADDRESS: B1D2H (45522)

Purpose: Creates BASIC file directory, expands file area by one byte, and writes 00H.

Input parameters: WORK1 ~ WORK8 contents written to directory as filename, WORK9 ~ WORK11 contents as extension.
WORKn address given by 68F4H + (n−1).

Output parameters: IX......Address where 00H was written.
Contents of following registers altered:
$0 ~ $18, $25 ~ $28, IZ

LABEL: NDFMK ADDRESS: B1C2H (45506)

Purpose: Creates sequential file directory, expands file area by one byte, and writes 1AH.

Input parameters: WORK1 ~ WORK8 contents written to directory as filename, WORK9 ~ WORK11 contents as extension.

Output parameters: IX......Address where 1AH was written.
Contents of following registers altered:
$0 ~ $18, $25 ~ $28, IZ

LABEL: **LNSCH** ADDRESS: **B4D1H (46289)**

Purpose: Searches for line specified by \$15 and \$16 and returns address where located.
Returns address of line with number greater than specified line when specified line does not exist. Returns program end address when line with number greater than specified line does not exist.

Input parameters: \$15, \$16....Line number (binary)
Directory address for BASIC program to be searched must be stored in NOWFL (6F54H).

Output parameters: IX.............Start address of line found
\$19, \$20...Start address of line found
FLG ....Flag register carry set (to 1) when specified line exists.
Contents of following registers altered:
\$0, \$1

LABEL: **RSOPN** ADDRESS: **BE11H (48657)**

Purpose: Initializes RS-232C I/O chip (LSI).
Sets baud rate, etc.
DTR, RTS switched ON.

Input parameters: \$00......Open mode
01H = Send open
02H = Receive open
03H = Send/receive open
\$11......Value entered at RS1 (6BFCH)
\$13......Value entered at RS3 (6BFEH)
Proper operation is not possible unless work area (RS1 ~ RS4, 6BFCH ~ 6BFFH) set before calling this routine.

Output parameters: None
Contents of following registers altered:
\$0 ~ \$6, IZ

LABEL: **RSCLO** ADDRESS: **BE5DH (48733)**

Purpose: Resets RS-232C I/O chip (LSI).

Input parameters: None

Output parameters: None
Contents of following registers altered:
\$0 ~ \$3, IZ

LABEL: RSGET ADDRESS: BE9CH (48796)

Purpose: Fetches one character from RS-232C receive buffer. Stands by until data are received if buffer is empty.
Also specifies XON/XOFF. In XOFF mode, fetches one character from buffer, and then sends XON when number of characters remaining in buffer is less than 33.
Jump performed to respective error when detected.

Input parameters: None

Output parameters: $00......Receive data
Contents of following registers altered:
$1 ~ $4, IZ

LABEL: PRTRS ADDRESS: BEFFH (48895)

Purpose: Sends $16 data to RS-232C.
Also specifies XON/XOFF and stands by for XON when in XOFF status. Corresponding control performed when SI/SO specified.
0EH and 20H sent when A0H sent during SI.

Input parameters: $16......Send data

Output parameters: None
Contents of following registers altered:
$0 ~ $4, IZ

LABEL: NTX0 ADDRESS: BF60H (48992)

Purpose: Sends $0 contents to RS-232C regardless of XON/XOFF and SI/SO specifications.

Input parameters: $0......Send data

Output parameters: None

LABEL: FDCLO ADDRESS: D8E6H (55526)

Purpose: Closes FDD. All files closed when $3 contents are FFH.

Input parameters: $3......Device number

Output parameters: This routine uses the user stack.
Contents of following registers altered:
$0 ~ $2

LABEL: **FDOPN** ADDRESS: **D9A0H (55712)**

Purpose: Opens FDD. For append, last block data written after address in IX register.

Input parameters:
$00......Device number
$01......Access type
$03......File attribute
IZ.........File work

Output parameters:
$26, $27...Number of characters for append
This routine uses the user stack.
Contents of following registers altered:
$0 ~ $6, IZ

LABEL: **FDWRB** ADDRESS: **DA05H (55813)**

Purpose: Writes sequential file to device specified by $2, consisting of number of characters specified by $7 and $8, starting from address specified by IX.

Input parameters:
$2.........Device number
$7, $8...Number of output characters
IX..........Start address of output data

Output parameters:
IX..........Start address of next output data
$0 ~ $8

LABEL: **FDWRR** ADDRESS: **DA10H (55824)**

Purpose: Writes random file to record number specified by $3 and $4 on device specified by $2, consisting of number of characters specified by $7 and $8, starting from address specified by IX.

Input parameters:
$2.........Device number
$3, $4...Random file record number
$7, $8...Number of output characters (256_{10} fixed)
IX..........Start address of output data

Output parameters:
IX..........Start address of next output data
This routine uses the user stack.
Contents of following registers altered:
$0 ~ $8

LABEL: FDRDB ADDRESS: DA3EH (55870)

Purpose: Writes data from device specified by $2 to area starting from address specified by IX (sequential open). Finished when $26 and $27 < 256_{10}

Input parameters: IX......Data write start address

Output parameters: IX......Next data write start address
$26, $27...Number of characters written
Contents of following registers altered:
$0 ~ $6

LABEL: FDRDR ADDRESS: DA49H (55881)

Purpose: Writes contents of record number specified by $3 and $4 on device specified by $2 to area starting from address specified by IX (random open).

Input parameters: $2.........Device number
$3, $4...Record number
IX..........Data write start address

Output parameters: IX..........Next data write start address
This routine uses the user stack.
Contents of following registers altered:
$0 ~ $6

LABEL: DOTMK ADDRESS: E737H (59191)

Purpose: Creates a dot pattern in LEDTP (6201H ~ 6800H) for the character in EDTOP (6100H ~ 6200H) specified by $10 and $11.

Input parameters: $10......Start cursor address
$11......End cursor address

Output parameters: None
Contents of following registers altered:
$0 ~ $11, IX, IZ

LABEL: FNSCH ADDRESS: E818H (59416)

Purpose: Searches for filename expressed in WORK1 (68F4H ~ 690BH) and enters it in $6 and $7 when found.

Input parameters: $4......File classification
Entering filename in WORK1 (68F4H ~ 690BH) and extension in (690CH ~ 690EH) are required.

Output parameters: $6, $7...Directory address
FLG......Flag register carry set (to 1) when file exists, and reset (to 0) when it does not exist.
Contents of following registers altered:
$0 ~ $5, IX, IZ

LABEL: KILL ADDRESS: E842H (59458)

Purpose: Deletes files specified by $6 and $7.

Input parameters: $6, $7...File directory address

Output parameters: None
Contents of following registers altered:
$0 ~ $21, $25 ~ $28, IX, IZ

LABEL: KYIN ADDRESS: E8B9H (59577)

Purpose: Samples keys and enters key code to $0. Executed after input of NEWALL, CALC, MEMOIN and CAL. Becomes FFH for one-key commands and one-key functions.

Input parameters: None

Output parameters: $0......Key code
Key codes are given in the following table.
Contents of following registers altered:
$1 ~ $11, IX, IZ

PRINT CODE TABLE

HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		(ROLL DOWN)	SPACE	0	@	P	`	p					タ	ミ		MEMO
1	(ROLL UP)	(DEL)	!	1	A	Q	a	q				ア	チ	ム		
2	(LINE TOP)	(INS)	"	2	B	R	b	r				イ	ツ	メ		ANS
3			#	3	C	S	c	s				ウ	テ	モ		OUT
4			$	4	D	T	d	t				エ	ト	ヤ		IN
5	(LINE DEL)		%	5	E	U	e	u				オ	ナ	ユ		LC
6	(LINE END)		&	6	F	V	f	v			ヲ	カ	ニ	ヨ		STOP
7	(SEL)		'	7	G	W	g	w	■		ァ	キ	ヌ	ラ		CONT
8	(BS)	(LINE C)	(	8	H	X	h	x			ィ	ク	ネ	リ	♠	ENG
9	(TAB)		)	9	I	Y	i	y			ゥ	ケ	ノ	ル	♥	$\overleftarrow{\text{ENG}}$
A			*	:	J	Z	j	z			ェ	コ	ハ	レ	♦	
B	(HOME)		+	;	K	[	k	{			ォ	サ	ヒ	ロ	♣	
C	(CLS)	→	,	<	L	¥	l	|			ャ	シ	フ	ワ	●	TK13
D	(CR LF)	←	−	=	M	]	m	}			ュ	ス	ヘ	ン	○	TK14
E		↑	.	>	N	^	n	~			ョ	セ	ホ	゛		TK15
F		↓	/	?	O	_	o				ッ	ソ	マ	゜	＼	TK16

Blanks indicate no output.
Indefinite character output by FF code when power is ON.
Configuration of characters output by E0 ~ FF are defined by the DEFCHR$ statement.

TK1	TK2	TK3	TK4
TK5	TK6	TK7	TK8
TK9	TK10	TK11	TK12
TK13	TK14	TK15	TK16

DISP

LABEL: **OUTAC** ADDRESS: **FF9EH (65438)**

Purpose: Outputs contents of $16 to device specified by NOWFC (690EH).

Input parameters: $16......Output data
NOWFC (690EH) = 0...Display
2...Printer
3...FCB device

Output parameters: None
Contents of following registers altered:
$0 ~ $13, IZ

LABEL: **BEEP** ADDRESS: **FFE5H (65509)**

Purpose: Outputs beep tone when key is pressed.

Input parameters: None

Output parameters: None
Contents of following registers altered:
$0 ~ $3

LABEL: **BUPDN** ADDRESS: **FFF7H (65527)**

Purpose: Transfers number of bytes specified by $4 and $5 from address specified by $2 and $3 to address specified by $0 and $1.

Input parameters: $0, $1....Transfer destination address
$2, $3....Transfer origin address
$4, $5....Number of bytes to be transferred
(Transfer not performed when both registers contain zero.)

Output parameters: None
Contents of following registers altered:
$0 ~ $14, IX, IZ

CHAPTER 13

APPENDICES

13-1 WORK AREA TABLE

	LABEL (OLD)	LABEL	ADDRESS		BYTE	NOTE
			Hex	Dec		
Screen data	INTOP	INTOP	6000	24576	256	Intermediate code buffer
	EDTOP	EDTOP	6100	24832	257	Input buffer
	* LEDTOP	LEDTP	6201	25089	1536	Display dot buffer
	* LPFTOP	LPFTP	6801	26625	192	4th line display dot buffer
	* CSRDAT	CSRDT	68C1	26817	6	Cursor flash data buffer
	* LCDSTS	LCDST	68C7	26823	1	76543210 (bit) At key input/PRINT Cursor display enable Virtual screen/Actual screen Cursor movement range setting Cursor ON/OFF Reverse field display
	EDCSR	EDCSR	68C8	26824	1	Cursor position
	SCTOP	SCTOP	68C9	26825	1	Actual screen top (High-order 3 bits, low-order 5 bits 0)
	TOEDB	TOEDB	68CA	26826	1	Logical line top (High-order 3 bits, low-order 5 bits 0)
	BOEDB	BOEDB	68CB	26827	1	Logical line bottom (High-order 3 bits, low-order 5 bits 1)
	MOEDB	MODEB	68CC	26828	1	Logical line top (at INPUT)
	* TOAREA	TOARE	68CD	26829	1	Cursor movement range top
	* BOAREA	BOARE	68CE	26830	1	Cursor movement range bottom
	EDCNT	EDCNT	68CF	26831	1	
	* DSPMODE	DSPMD	68D0	26832	1	00H=Normal display, 01H=PF display 03H=MENU display
	SCROL	SCROL	68D1	26833	1	80H=4-line scroll, 60H=3-line scroll
Key data	* KYSTAT	KYSTA	68D2	26834	1	76543210 (bit) 0 BEEP REPEAT ON/OFF REPEAT enable Contrast APO disable OFF AC
	CHATA	CHATA	68D3	26835	1	Time count while chattering in progress
	* KEYCOM	KEYCM	68D4	26836	1	KO
	KEYIN	KEYIN	68D5	26837	2	KI
	* KEYMODE	KEYMD	68D7	26839	1	76543210 (bit) CAPS
	* KYREPT	KYREP	68D8	26840	1	Key repeat count time
	KECNT	KECNT	68D9	26841	22	Key buffer
	KANA1	KANA1	68FF	26863	2	
	* APOCNT	APOCN	68F1	26865	1	APO counter(-1 for TMINT)
Calculation BASIC Data	* BANKNO	BNKNO	68F2	26866	1	Not used
	* ANGLEFLG	ANGFL	68F3	26867	1	Angle mode(0:DEG, 1:RAD, 2:GRAD)
	WORK1	WORK1	68F4	26868	24	Work buffer
	OUTDV	OUTDV	690C	26892	1	Output device (00H:display, 02H:printer, 04H:FCB)
	PTABC	PTABC	690D	26893	1	Number of printer output characters
	* NOWFCB	NOWFC	690E	26894	2	Start address of current object FCB
	VAR1	VAR1	6910	26896	1	Variable work
	VAR2	VAR2	6911	26897	1	Variable work
	VAR3	VAR3	6912	26898	1	Variable work

	LABEL (OLD)	LABEL	ADDRESS Hex	ADDRESS Dec	BYTE	NOTE
	VAR4	VAR4	6913	26899	2	Variable work
	* CONTADRS	CONTA	6915	26901	2	Execution restart pointer for CONT
	DATPA	DATPA	6917	26903	2	DATA statement pointer
	NB_X1	NB_X1	6919	26905	1	DRAW statement work area
	NB_Y1	NB_Y1	691A	26906	1	DRAW statement work area
	NB_X2	NB_X2	691B	26907	1	DRAW statement work area
	NB_Y2	NB_Y2	691C	26908	1	DRAW statement work area
	* ERRFILE	ERRFL	691D	26909	2	Dir address for ON ERROR enable file
	EJPDE	EJPDE	691F	26911	2	ON ERROR jump destination pointer
	ERRLN	ERRLN	6921	26913	2	Line number containing error generation
	ERRDE	ERRDE	6923	26915	2	Address of statement containing error generation
	ERRN	ERRN	6925	26917	1	Error number
	* EJPFLG	EJPFG	6926	26918	1	00H=Normal processing 01H=ON ERROR handling routine
	* TRACEFLG	TRAFG	6927	26919	1	00H=TROFF, 01H=TRON
	* INP_ERR	INPER	6928	26920	2	Return address for INPUT error
	* MACCNT	MACCN	692A	26922	1	Not used
Memo data	* MEMOPTR	MEMO	692B	26923	4	Not used
Main data	IOBF	IOBF	692F	26927	2	
	SSTOP	SSTOP	6931	26929	2	
	SBOT	SBOT	6933	26931	2	
	FORSK	FORSK	6935	26933	2	
	GOSSK	GOSSK	6937	26935	2	
	TONDT	TONDT	6939	26937	2	
	DTTB	DTTB	693B	26939	2	
	TOSDT	TOSDT	693D	26941	2	
	PTSDT	PTSDT	693F	26943	2	
	HIMEM	HIMEM	6941	26945	2	
	* BASEND	BASEN	6943	26947	2	
	* MEMEND	MEMEN	6945	26949	2	
	* DATDIR	DATDI	6947	26951	2	
	* BASDIR	BASDI	6949	26953	2	
	* DIREND	DIREN	694B	26955	2	
	ACJMP	ACJMP	694D	26957	2	Address of jump destination for BREAK
	CHARA	CHARA	694F	26959	1	
	* ANSFLG	ANSFG	6950	26960	1	Post calculation flag (1=After calculation, 0=Other)
	* CSRCNT	CSRCN	6951	26961	1	Cursor flash counter(-1 when ONINT)
Stack	* USPBTM	USPBT	6952	26962	249	User stack area
	* USPTOP	USPTP	6A4B	27211	0	
	* SSPBTM	SSPBT	6A4B	27211	255	System stack area
	* SSPTOP	SSPTP	6B4A	27466	0	
CHR$	CGRAM	CGRAM	6B4A	27466	96	F0H～FFH display dot pattern
	* ELVOLADR	ELVAD	6BAA	27562	2	Contrast data
	* BEEPFLAG	BEEPF	6BAC	27564	1	Key beep flag(ON=00, OFF=01)
Calcula-tion variables	* DATEAD	DATE	6BAD	27565	3	DATE$ data
	* TIMEAD	TIME	6BB0	27568	2	TIME$ data
	* STATVAR	STAT	6BB2	27570	72	STAT data
RS232C	* OPTIONAD	OPTCD	6BFA	27642	1	Option code
LCD	IOSTS	IOSTS	6BFB	27643	1	76543210 Send open Receive open

	LABEL (OLD)	LABEL	ADDRESS Hex	ADDRESS Dec	BYTE	NOTE
Buffer	※ RS232C1	RS1	6BFC	27644	1	76543210 (bit) MT/RS-232C Odd/Even parity OFF/ON parity 7/8 data length 1/2 stop bit 000 9600 baud 001 4800 baud 010 2400 baud 011 1200 baud 100 600 baud 101 300 baud 110 150 baud 111 75 baud
	※ RS232C2	RS2	6BFD	27645	1	76543210 (bit) Output while XOFF in progress Input while SO in progress
	※ RS232C3	RS3	6BFE	27646	1	76543210 (bit) SI/SO control XON/XOFF control CTS control DSR control CD control Output SO in progress Input XOFF in progress
	※ RS232C4	RS4	6BFF	27647	1	76543210 (bit) Buffer Not Ready Over run Parity Framing
	※ RS232CFG	RSFG	6C00	27648	2	RS-232C default value(RS232C1, RS232C3 data)
	※ PRINTFLG	PRNFG	6C02	27650	2	76543210 (bit) printer BUSY check BUSY ERROR ACK (During input: 1=check, 0=no check) 3=initial data) (During RTN: 5=initial data)
	※ DISPFLG	DSPFG	6C04	27652	1	76543210 (bit) 00H=Full screen display 11 Last display line(0～3) First display line(0～3)
	※ INTCKF	INTCK	6C05	27653	1	01H=Data receive
	RXCNT	RXCNT	6C06	27654	258	RS-232C, MT receive buffer 1 byte...number of receive bytes 1 byte...input pointer 256 bytes..buffer
	ANSAD	ANSAD	6D08	27912	9	ANS data
	※ RANDAD	RND	6D11	27921	9	Random data
	※ CALCBF	CALC	6D1A	27930	258	Calc buffer
	※ MTFILE1	MTFL	6E1C	28188	1	MT work
	IOBF0	IOBF0	6E1D	28189	35	SAVE/LOAD FCB
	※ IOBUFFER	IOBUF	6E40	28224	258	SAVE/LOAD I/O buffer

	LABEL (OLD)	LABEL	ADDRESS Hex	Dec	BYTE	NOTE
Key data	* TIMEAUTO	TIMAT	6F42	28482	5	Clock boot flag
	* ONFILE0	ONFL0	6F47	28487	2	Clock boot file address
	* ONFILE1	ONFL1	6F49	28489	2	Preset file address
	* ONFILE2	ONFL2	6F4B	28491	2	Preset file address
	* ONFILE3	ONFL3	6F4D	28493	2	Preset file address
	* ONFILE4	ONFL4	6F4F	28495	2	Preset file address
Mode work	MODE	MODE	6F51	28497	3	76543210 (bit)
						BASIC
						EDIT
						MEMO
						INPUT
						CALC
						SAVE/LOAD
						MENU
						ASSEMBLER
						During BASIC execution
						RUN
						STOP
	* NOWFILE	NOWFL	6F54	28500	2	Address of current file
	NOWLN	NOWLN	6F56	28502	2	Current execution line number
	EXEDE	EXEDE	6F58	28504	2	Work
	* FDDFILNA	FDDFL	6F5A	28506	16	Work
	* LOADBYTE	LOADB	6F6A	28522	2	Work
	* MTFILE2	MTFL2	6F6C	28524	1	Work
	* MTFILE3	MTFL3	6F6D	28525	1	Work
	* MTFILE5	MTFL5	6F6E	28526	1	Work
	* MTFILE6	MTFL6	6F6F	28527	1	Work
	* MTFILE7	MTFL7	6F70	28528	2	Work
	* MTFILE8	MTFL8	6F72	28530	2	Work
	* BSAVEWRK	BSVWK	6F74	28532	32	Work
	* MENUCSR	MENUC	6F94	28564	15	Work
	* INPUTSTR	INPST	6FA3	28579	12	Work
	* FILEWORK	FLWK	6FAF	28591	40	Work

13-2 COMMAND TABLE

OP NAME	MNEMONIC	OP-CODE	HEX	BYTE	CYCLE		STATE		FLAG Z, C, LZ, UZ				Bit	Type
					FETCH	EXE	FETCH	EXE	Z	C	LZ	UZ		
LD	LD $, $	00000010	02	3	2	2	6	6	※	※	※	※	8	Send
	LD $, ($)	00010001	11	3	2	4	6	11	※	※	※	※	8	Send
	LD $, (IX, ±$)	00101000	28	3	2	4	6	11	※	※	※	※	8	Send
	LD $, (IZ, ±$)	00101001	29	3	2	4	6	11	※	※	※	※	8	Send
	LD $, n	01000010	42	3	2	2	6	6	※	※	※	※	8	Send
	LD $, (IX, ±n)	01101000	68	3	2	4	6	11	※	※	※	※	8	Send
	LD $, (IZ, ±n)	01101001	69	3	2	4	6	11	※	※	※	※	8	Send
LDI	LDI $, (IX±$)	00101010	2A	3	2	4	6	11	※	※	※	※	8	Send
	LDI $, (IZ±$)	00101011	2B	3	2	4	6	11	※	※	※	※	8	Send
	LDI $, (IX±n)	01101010	6A	3	2	4	6	11	※	※	※	※	8	Send
	LDI $, (IZ±n)	01101011	6B	3	2	4	6	11	※	※	※	※	8	Send
ST	ST $, ($)	00010000	10	3	2	4	6	11	※	※	※	※	8	Send
	ST $, (IX±$)	00100000	20	3	2	4	6	11	※	※	※	※	8	Send
	ST $, (IZ±$)	00100001	21	3	2	4	6	11	※	※	※	※	8	Send
	ST $, (IX±n)	01100000	60	3	2	4	6	11	※	※	※	※	8	Send
	ST $, (IZ±n)	01100001	61	3	2	4	6	11	※	※	※	※	8	Send
STI	STI $, (IX±$)	00100010	22	3	2	4	6	11	※	※	※	※	8	Send
	STI $, (IZ±$)	00100011	23	3	2	4	6	11	※	※	※	※	8	Send
	STI $, (IX±n)	01100010	62	3	2	4	6	11	※	※	※	※	8	Send
	STI $, (IZ±n)	01100011	63	3	2	4	6	11	※	※	※	※	8	Send
PHS	PHS $	00100110	26	2	1	3	3	9	※	※	※	※	8	Send
PHU	PHU $	00100111	27	2	1	3	3	9	※	※	※	※	8	Send
PPS	PPS $	00101110	2E	2	1	4	3	11	※	※	※	※	8	Send
PPU	PPU $	00101111	2F	2	1	4	3	11	※	※	※	※	8	Send
GFL	GFL $	00011100	1C	2	1	2	3	6	※	※	※	※	8	Send
GPO	GPO $	00011100	1C	2	1	2	3	6	※	※	※	※	8	Send
GST	GST PE, $	00011110	1E	2	1	2	3	6	※	※	※	※	8	Send
	GST PD, $	00011110	1E	2	1	2	3	6	※	※	※	※	8	Send
	GST UA, $	00011110	1E	2	1	2	3	6	※	※	※	※	8	Send
	GST IA, $	00011111	1F	2	1	2	3	6	※	※	※	※	8	Send
	GST IE, $	00011111	1F	2	1	2	3	6	※	※	※	※	8	Send
	GST TM, $	00011111	1F	2	1	2	3	6	※	※	※	※	8	Send
PFL	PFL $	00010100	14	2	1	2	3	6	M	M	M	M	8	Send
PST	PST PE, $	00010110	16	2	1	2	3	6	※	※	※	※	8	Send
	PST PD, $	00010110	16	2	1	2	3	6	※	※	※	※	8	Send
	PST UA, $	00010110	16	2	1	2	3	6	※	※	※	※	8	Send
	PST IA, $	00010111	17	2	1	2	3	6	※	※	※	※	8	Send
	PST IE, $	00010111	17	2	1	2	3	6	※	※	※	※	8	Send
	PST PE, n	01010110	56	3	2	2	6	6	※	※	※	※	8	Send
	PST PD, n	01010110	56	3	2	2	6	6	※	※	※	※	8	Send
	PST UA, n	01010110	56	3	2	2	6	6	※	※	※	※	8	Send
	PST IA, n	01010111	57	3	2	2	6	6	※	※	※	※	8	Send
	PST IE, n	01010111	57	3	2	2	6	6	※	※	※	※	8	Send
LDW	LDW $, $	10000010	82	3	2	4	6	11	※	※	※	※	16	Send
	LDW $, ($)	10010001	91	3	2	5	6	14	※	※	※	※	16	Send
	LDW $, (IX±$)	10101000	A8	3	2	5	6	14	※	※	※	※	16	Send
	LDW $, (IZ±$)	10101001	A9	3	2	5	6	14	※	※	※	※	16	Send
	LDW $, m	11010001	D1	4	3	5	9	14	※	※	※	※	16	Send
LDIW	LDIW $, (IX±$)	10101010	AA	3	2	5	6	14	※	※	※	※	16	Send
	LDIW $, (IZ±$)	10101011	AB	3	2	5	6	14	※	※	※	※	16	Send
STW	STW $, (IX±$)	10100000	A0	3	2	5	6	14	※	※	※	※	16	Send
	STW $, (IZ±$)	10100001	A1	3	2	5	6	14	※	※	※	※	16	Send
	STW $, ($)	10010000	90	3	2	5	6	14	※	※	※	※	16	Send
STIW	STIW $, (IX±$)	10100010	A2	3	2	4	6	14	※	※	※	※	16	Send
	STIW $, (IZ±$)	10100011	A3	3	2	4	6	14	※	※	※	※	16	Send
PHSW	PHSW $	10100110	A6	2	1	4	3	12	※	※	※	※	16	Send
PHUW	PHUW $	10100111	A7	2	1	4	3	12	※	※	※	※	16	Send

OP NAME	MNEMONIC	OP-CODE HEX		BYTE	CYCLE		STATE		FLAG				Bit	Type
					FETCH	EXE	FETCH	EXE	Z	C	LZ	UZ		
PPSW	PPSW $	10101110	AE	2	1	5	3	14	*	*	*	*	16	Send
PPUW	PPUW $	10101111	AF	2	1	5	3	14	*	*	*	*	16	Send
GRE	GRE IX, $	10011110	9E	2	1	4	3	11	*	*	*	*	16	Send
	GRE IY, $	10011110	9E	2	1	4	3	11	*	*	*	*	16	Send
	GRE IZ, $	10011110	9E	2	1	4	3	11	*	*	*	*	16	Send
	GRE US, $	10011110	9E	2	1	4	3	11	*	*	*	*	16	Send
	GRE SS, $	10011111	9F	2	1	4	3	11	*	*	*	*	16	Send
	GRE KY, $	10011111	9F	2	1	4	3	11	*	*	*	*	16	Send
PRE	PRE IX, $	10010110	96	2	1	4	3	11	*	*	*	*	16	Send
	PRE IY, $	10010110	96	2	1	4	3	11	*	*	*	*	16	Send
	PRE IZ, $	10010110	96	2	1	4	3	11	*	*	*	*	16	Send
	PRE US, $	10010110	96	2	1	4	3	11	*	*	*	*	16	Send
	PRE SS, $	10010111	97	2	1	4	3	11	*	*	*	*	16	Send
	PRE IX, m	11010110	D6	4	3	4	9	11	*	*	*	*	16	Send
	PRE IY, m	11010110	D6	4	3	4	9	11	*	*	*	*	16	Send
	PRE IZ, m	11010110	D6	4	3	4	9	11	*	*	*	*	16	Send
	PRE US, m	11010110	D6	4	3	4	9	11	*	*	*	*	16	Send
	PRE SS, m	11010111	D7	4	3	4	9	11	*	*	*	*	16	Send
AD	AD $, $	00001000	08	3	2	2	6	6	M	M	M	M	8	*1
	AD (IX±$), $	00111100	3C	3	2	4	6	12	M	M	M	M	8	*1
	AD (IZ±$), $	00111101	3D	3	2	4	6	12	M	M	M	M	8	*1
	AD $, n	01001000	48	3	2	2	6	6	M	M	M	M	8	*1
	AD (IX±n), $	01111100	7C	3	2	4	6	12	M	M	M	M	8	*1
	AD (IZ±n), $	01111101	7D	3	2	4	6	12	M	M	M	M	8	*1
ADB	ADB $, $	00001010	0A	3	2	2	6	6	M	M	M	M	8	*1
	ADB $, n	01001010	4A	3	2	2	6	6	M	M	M	M	8	*1
ADC	ADC $, $	00000000	00	3	2	2	6	6	M	M	M	M	8	*1
	ADC (IX±$), $	00111000	38	3	2	4	6	12	M	M	M	M	8	*1
	ADC (IZ±$), $	00111001	39	3	2	4	6	12	M	M	M	M	8	*1
	ADC $, n	01000000	40	3	2	2	6	6	M	M	M	M	8	*1
	ADC (IX±n), $	01111000	78	3	2	4	6	12	M	M	M	M	8	*1
	ADC (IZ±n), $	01111001	79	3	2	4	6	12	M	M	M	M	8	*1
AN	AN $, $	00001100	0C	3	2	2	6	6	M	0	M	M	8	*1
	AN $, n	01001100	4C	3	2	2	6	6	M	0	M	M	8	*1
ANC	ANC $, $	00000100	04	3	2	2	6	6	M	0	M	M	8	*1
	ANC $, n	01000100	44	3	2	2	6	6	M	0	M	M	8	*1
NA	NA $, $	00001101	0D	3	2	2	6	6	M	1	M	M	8	*1
	NA $, n	01001101	4D	3	2	2	6	6	M	1	M	M	8	*1
NAC	NAC $, $	00000101	05	3	2	2	6	6	M	1	M	M	8	*1
	NAC $, n	01000101	45	3	2	2	6	6	M	1	M	M	8	*1
OR	OR $, $	00001110	0E	3	2	2	6	6	M	1	M	M	8	*1
	OR $, n	01001110	4E	3	2	2	6	6	M	1	M	M	8	*1
ORC	ORC $, $	00000110	06	3	2	2	6	6	M	1	M	M	8	*1
	ORC $, n	01000110	46	3	2	2	6	6	M	1	M	M	8	*1
SB	SB $, $	00001001	09	3	2	2	6	6	M	M	M	M	8	*1
	SB (IX±$), $	00111110	3E	3	2	4	6	12	M	M	M	M	8	*1
	SB (IZ±$), $	00111111	3F	3	2	4	6	12	M	M	M	M	8	*1
	SB $, n	01001001	49	3	2	2	6	6	M	M	M	M	8	*1
	SB (IX±n), $	01111110	7E	3	2	4	6	12	M	M	M	M	8	*1
	SB (IZ±n), $	01111111	7F	3	2	4	6	12	M	M	M	M	8	*1
SBB	SBB $, $	00001011	0B	3	2	2	6	6	M	M	M	M	8	*1
	SBB $, n	01001011	4B	3	2	2	6	6	M	M	M	M	8	*1
SBC	SBC $, $	00000001	01	3	2	2	6	6	M	M	M	M	8	*1
	SBC (IX±$), $	00111110	3E	3	2	4	6	12	M	M	M	M	8	*1
	SBC (IZ±$), $	00111111	3F	3	2	4	6	12	M	M	M	M	8	*1
	SBC $, n	01000001	41	3	2	2	6	6	M	M	M	M	8	*1
	SBC (IX±n), $	01111010	7A	3	2	4	6	12	M	M	M	M	8	*1

*1: Arithmetic calculation

OP NAME	MNEMONIC	OP-CODE	HEX	BYTE	CYCLE		STATE		FLAG				Bit	Type
					FETCH	EXE	FETCH	EXE	Z	C	LZ	UZ		
	SBC (IZ±n), $	01111011	7B	3	2	4	6	12	M	M	M	M		*1
XR	XR $, $	00001111	0F	3	2	2	6	6	M	0	M	M		*1
	XR $, n	01001111	4F	3	2	2	6	6	M	0	M	M		*1
XRC	XRC $, $	00000111	07	3	2	2	6	6	M	0	M	M		*1
	XRC $, n	01000111	47	3	2	2	6	6	M	0	M	M		*1
ADW	ADW $, $	10001000	88	3	2	4	6	11	M	M	M	M		*1
	ADW (IX±$), $	10111100	BC	3	2	6	6	18	M	M	M	M		*1
	ADW (IZ±$), $	10111101	BD	3	2	6	6	18	M	M	M	M		*1
ADBW	ADBW $, $	10001010	8A	3	2	4	6	11	M	M	M	M		*1
ADCW	ADCW $, $	10000000	80	3	2	4	6	11	M	M	M	M		*1
	ADCW (IX±$), $	10111000	B8	3	2	4	6	11	M	M	M	M		*1
	ADCW (IZ±$), $	10111001	B9	3	2	4	6	11	M	M	M	M		*1
ANW	ANW $, $	10001100	8C	3	2	4	6	11	M	0	M	M		*1
ANCW	ANCW $, $	10000100	84	3	2	4	6	11	M	0	M	M		*1
NAW	NAW $, $	10001101	8D	3	2	4	6	11	M	1	M	M		*1
NACW	NACW $, $	10000101	85	3	2	4	6	11	M	1	M	M		*1
ORW	ORW $, $	10001110	8E	3	2	4	6	11	M	1	M	M		*1
ORCW	ORCW $, $	10000110	86	3	2	4	6	11	M	1	M	M		*1
SBW	SBW $, $	10001001	89	3	2	4	6	11	M	M	M	M		*1
	SBW (IX±$), $	10111110	BE	3	2	6	6	18	M	M	M	M		*1
	SBW (IZ±$), $	10111111	BF	3	2	6	6	18	M	M	M	M		*1
SBBW	SBBW $, $	10001011	8B	3	2	6	6	18	M	M	M	M		*1
SBCW	SBCW $, $	10000001	81	3	2	4	6	11	M	M	M	M		*1
	SBCW (IX±$), $	10111010	BA	3	2	6	6	18	M	M	M	M		*1
	SBCW (IZ±$), $	10111011	BB	3	2	6	6	18	M	M	M	M		*1
XRW	XRW $, $	10001111	8F	3	2	4	6	11	M	0	M	M		*1
XRCW	XRCW $, $	10000111	87	3	2	4	6	11	M	0	M	M		*1
BID	BID $	00011000	18	2	1	2	3	6	M	M	M	M	8	Rotate/Shift
BIU	BIU $	00011000	18	2	1	2	3	6	M	M	M	M	8	Rotate/Shift
ROD	ROD $	00011000	18	2	1	2	3	6	M	M	M	M	8	Rotate/Shift
ROU	ROU $	00011000	18	2	1	2	3	6	M	M	M	M	8	Rotate/Shift
DID	DID $	00011010	1A	2	1	2	3	6	M	0	M	0	8	Rotate/Shift
DIU	DIU $	00011010	1A	2	1	2	3	6	M	0	0	M	8	Rotate/Shift
CMP	CMP $	00011011	1B	2	1	2	3	6	M	M	M	M	8	Rotate/Shift
INV	INV $	00011011	1B	2	1	2	3	6	M	1	M	M	8	Rotate/Shift
BIDW	BIDW $	10011000	98	2	1	4	3	11	M	M	M	M	16	Rotate/Shift
BIUW	BIUW $	10011000	98	2	1	4	3	11	M	M	M	M	16	Rotate/Shift
RODW	RODW $	10011000	98	2	1	4	3	11	M	M	M	M	16	Rotate/Shift
ROUW	ROUW $	10011000	98	2	1	4	3	11	M	M	M	M	16	Rotate/Shift
DIDW	DIDW $	10011010	9A	2	1	4	3	11	M	0	M	M	16	Rotate/Shift
DIUW	DIUW $	10011010	9A	2	1	4	3	11	M	0	M	M	16	Rotate/Shift
BYDW	BYDW $	10011010	9A	2	1	4	3	11	M	0	M	M	16	Rotate/Shift
BYUW	BYUW $	10011010	9A	2	1	4	3	11	M	0	M	M	16	Rotate/Shift
CMPW	CMPW $	10011011	9B	2	1	4	3	11	M	M	M	M	16	Rotate/Shift
INVW	INVW $	10011011	9B	2	1	4	3	11	M	1	M	M	16	Rotate/Shift
JP	JP Z, m	00110000	30	3	2	2	6	6	*	*	*	*		*2
	JP NC, m	00110001	31	3	2	2	6	6	*	*	*	*		*2
	JP LZ, m	00110010	32	3	2	2	6	6	*	*	*	*		*2
	JP UZ, m	00110011	33	3	2	2	6	6	*	*	*	*		*2
	JP NZ, m	00110100	34	3	2	2	6	6	*	*	*	*		*2
	JP C, m	00110101	35	3	2	2	6	6	*	*	*	*		*2
	JP m	00110111	37	3	2	2	6	6	*	*	*	*		*2
JR	JR Z, ±p	10110000	B0	2	1	2	3	6	*	*	*	*		*3
	JR NC, ±p	10110001	B1	2	1	2	3	6	*	*	*	*		*3
	JR LZ, ±p	10110010	B2	2	1	2	3	6	*	*	*	*		*3
	JR UZ, ±p	10110011	B3	2	1	2	3	6	*	*	*	*		*3
	JR NZ, ±p	10110100	B4	2	1	2	3	6	*	*	*	*		*3

*2: Unconditional Jump
*3: Conditional Jump

OP NAME	MNEMONIC	OP-CODE HEX	BYTE	CYCLE FETCH	CYCLE EXE	STATE FETCH	STATE EXE	FLAG Z, C, LZ, UZ	Bit	Type
	JR C, ±p	10110101 B5	2	1	2	3	6	* * * *		*3
	JR ±p	10110111 B7	2	1	2	3	6	* * * *		*3
CAL	CAL Z, m	01110000 70	3	2	4	6	12	* * * *		Call
	CAL NC, m	01110001 71	3	2	4	6	12	* * * *		Call
	CAL LZ, m	01110010 72	3	2	4	6	12	* * * *		Call
	CAL UZ, m	01110011 73	3	2	4	6	12	* * * *		Call
	CAL NZ, m	01110100 74	3	2	4	6	12	* * * *		Call
	CAL C, m	01110101 75	3	2	4	6	12	* * * *		Call
	CAL m	01110111 77	3	2	4	6	12	* * * *		Call
RTN	RTN Z	11110000 F0	1	0	5	0	14	* * * *		Return
	RTN NC	11110001 F1	1	0	5	0	14	* * * *		Return
	RTN LZ	11110010 F2	1	0	5	0	14	* * * *		Return
	RTN UZ	11110011 F3	1	0	5	0	14	* * * *		Return
	RTN NZ	11110100 F4	1	0	5	0	14	* * * *		Return
	RTN C	11110101 F5	1	0	5	0	14	* * * *		Return
	RTN	11110111 F7	1	0	5	0	14	* * * *		Return
BDN	BDN	11011001 D9	1	0	2a+2	0	6a+6	* * * *		Block Move
BUP	BUP	11011000 D8	1	0	2a+2	0	6a+6	* * * *		Block Move
SDN	SDN $	11011101 DD	2	1	2a+2	3	6a+6	M M M M		Search
	SDN n	01011101 5D	2	1	2a+2	3	6a+6	M M M M		Search
SUP	SUP $	11011100 DC	2	1	2a+2	3	6a+6	M M M M		Search
	SUP n	01011100 5C	2	1	2a+2	3	6a+6	M M M M		Search
NOP	NOP	11111000 F8	1	0	2	0	6	* * * *		Special
CLT	CLT	11111001 F9	1	0	2	0	6	* * * *		Special
FST	FST	11111010 FA	1	0	2	0	6	* * * *		Special
SLW	SLW	11111011 FB	1	0	2	0	6	* * * *		Special
CANI	CANI	11111100 FC	1	0	2	0	6	* * * *		Special
RTNI	RTNI	11111101 FD	1	0	5	0	14	* * * *		Special
OFF	OFF	11111110 FE	1	0	2	0	6	* * * *		Special
TRP	TRP	11111111 FF	1	0	4	0	12	* * * *		Special

13-3 8-BIT COMMAND TABLE

MNEMONIC	OP-CODE	HEX	MNEMONIC	OP-CODE	HEX
AD $,$	00001000	08	PPU $	00101111	2F
AD $,n	01001000	48	PST IA,$	00010111	17
AD (IX±$),$	00111100	3C	PST IA,n	01010111	57
AD (IX±n),$	01111100	7C	PST IE,$	00010111	17
AD (IZ±$),$	00111101	3D	PST IE,n	01010111	57
AD (IZ±n),$	01111101	7D	PST PD,$	00010110	16
ADB $,$	00001010	0A	PST PD,n	01010110	56
ADB $,n	01001010	4A	PST PE,$	00010110	16
ADC $,$	00000000	00	PST PE,n	01010110	56
ADC $,n	01000000	40	PST UA,$	00010110	16
ADC (IX±$),$	00111000	38	PST UA,n	01010110	56
ADC (IX±n),$	01111000	78	ROD $	00011000	18
ADC (IZ±$),$	00111001	39	ROU $	00011000	18
ADC (IZ±n),$	01111001	79	SB $,$	00001001	09
AN $,$	00001100	0C	SB $,n	01001001	49
AN $,n	01001100	4C	SB (IX±$),$	00111110	3E
ANC $,$	00000100	04	SB (IX±n),$	01111110	7E
ANC $,n	01000100	44	SB (IZ±$),$	00111111	3F
BID $	00011000	18	SB (IZ±n),$	01111111	7F
BIU $	00011000	18	SBB $,$	00001011	0B
CMP $	00011011	1B	SBB $,n	01001011	4B
DID $	00011010	1A	SBC $,$	00000001	01
DIU $	00011010	1A	SBC $,n	01000001	41
GFL $	00011100	1C	SBC (IX±$),$	00111110	3E
GPO $	00011100	1C	SBC (IX±n),$	01111010	7A
GST IA,$	00011111	1F	SBC (IZ±$),$	00111111	3F
GST IE,$	00011111	1F	SBC (IZ±n),$	01111011	7B
GST PD,$	00011110	1E	ST $,($)	00010000	10
GST PE,$	00011110	1E	ST $,(IX±$)	00100000	20
GST TM,$	00011111	1F	ST $,(IX±n)	01100000	60
GST UA,$	00011110	1E	ST $,(IZ±$)	00100001	21
INV $	00011011	1B	ST $,(IZ±n)	01100001	61
LD $,$	00000010	02	STI $,(IX±$)	00100010	22
LD $,($)	00010001	11	STI $,(IX±n)	01100010	62
LD $,(IX,±$)	00101000	28	STI $,(IZ±$)	00100011	23
LD $,(IX,±n)	01101000	68	STI $,(IZ±n)	01100011	63
LD $,(IZ,±$)	00101001	29	XR $,$	00001111	0F
LD $,(IZ,±n)	01101001	69	XR $,n	01001111	4F
LD $,n	01000010	42	XRC $,$	00000111	07
LDI $,(IX±$)	00101010	2A	XRC $,n	01000111	47
LDI $,(IX±n)	01101010	6A			
LDI $,(IZ±$)	00101011	2B			
LDI $,(IZ±n)	01101011	6B			
NA $,$	00001101	0D			
NA $,n	01001101	4D			
NAC $,$	00000101	05			
NAC $,n	01000101	45			
OR $,$	00001110	0E			
OR $,n	01001110	4E			
ORC $,$	00000110	06			
ORC $,n	01000110	46			
PFL $	00010100	14			
PHS $	00100110	26			
PHU $	00100111	27			
PPS $	00101110	2E			

13-4 16-BIT COMMAND TABLE

MNEMONIC	OP-CODE	HEX	MNEMONIC	OP-CODE	HEX
ADBW $,$	10001010	8A	ORCW $,$	10000110	86
ADCW $,$	10000000	80	ORW $,$	10001110	8E
ADCW (IX±$),$	10111000	B8	PHSW $	10100110	A6
ADCW (IZ±$),$	10111001	B9	PHUW $	10100111	A7
ADW $,$	10001000	88	PPSW $	10101110	AE
ADW (IX±$),$	10111100	BC	PPUW $	10101111	AF
ADW (IZ±$),$	10111101	BD	PRE IX,$	10010110	96
ANCW $,$	10000100	84	PRE IX,m	11010110	D6
ANW $,$	10001100	8C	PRE IY,$	10010110	96
BIDW $	10011000	98	PRE IY,m	11010110	D6
BIUW $	10011000	98	PRE IZ,$	10010110	96
BYDW $	10011010	9A	PRE IZ,m	11010110	D6
BYUW $	10011010	9A	PRE SS,$	10010111	97
CMPW $	10011011	9B	PRE SS,m	11010111	D7
DIDW $	10011010	9A	PRE US,$	10010110	96
DIUW $	10011010	9A	PRE US,m	11010110	D6
GRE IX,$	10011110	9E	RODW $	10011000	98
GRE IY,$	10011110	9E	ROUW $	10011000	98
GRE IZ,$	10011110	9E	SBBW $,$	10001011	8B
GRE KY,$	10011111	9F	SBCW $,$	10000001	81
GRE SS,$	10011111	9F	SBCW (IX±$),$	10111010	BA
GRE US,$	10011110	9E	SBCW (IZ±$),$	10111011	BB
INVW $	10011011	9B	SBW $,$	10001001	89
LDIW $,(IX±$)	10101010	AA	SBW (IX±$),$	10111110	BE
LDIW $,(IZ±$)	10101011	AB	SBW (IZ±$),$	10111111	BF
LDW $,$	10000010	82	STIW $,(IX±$)	10100010	A2
LDW $,($)	10010001	91	STIW $,(IZ±$)	10100011	A3
LDW $,(IX±$)	10101000	A8	STW $,($)	10010000	90
LDW $,(IZ±$)	10101001	A9	STW $,(IX±$)	10100000	A0
LDW $,m	11010001	D1	STW $,(IZ±$)	10100001	A1
NACW $,$	10000101	85	XRCW $,$	10000111	87
NAW $,$	10001101	8D	XRW $,$	10001111	8F

13-5 OTHER COMMANDS

MNEMONIC	OP-CODE	HEX	MNEMONIC	OP-CODE	HEX
BDN	11011001	D9	JR NC,±p	10110001	B1
BUP	11011000	D8	JR NZ,±p	10110100	B4
CAL C,m	01110101	75	JR UZ,±p	10110011	B3
CAL LZ,m	01110010	72	JR Z,±p	10110000	B0
CAL NC,m	01110001	71	JR ±p	10110111	B7
CAL NZ,m	01110100	74	NOP	11111000	F8
CAL UZ,m	01110011	73	OFF	11111110	FE
CAL Z,m	01110000	70	RTN	11110111	F7
CAL m	01110111	77	RTN C	11110101	F5
CANI	11111100	FC	RTN LZ	11110010	F2
CLT	11111001	F9	RTN NC	11110001	F1
FST	11111010	FA	RTN NZ	11110100	F4
JP C,m	00110101	35	RTN UZ	11110011	F3
JP LZ,m	00110010	32	RTN Z	11110000	F0
JP NC,m	00110001	31	RTNI	11111101	FD
JP NZ,m	00110100	34	SDN $	11011101	DD
JP UZ,m	00110011	33	SDN n	01011101	5D
JP Z,m	00110000	30	SLW	11111011	FB
JP m	00110111	37	SUP $	11011100	DC
JR C,±p	10110101	B5	SUP n	01011100	5C
JR LZ,±p	10110010	B2	TRP	11111111	FF

13-6 OPERATION CODE COMMANDS

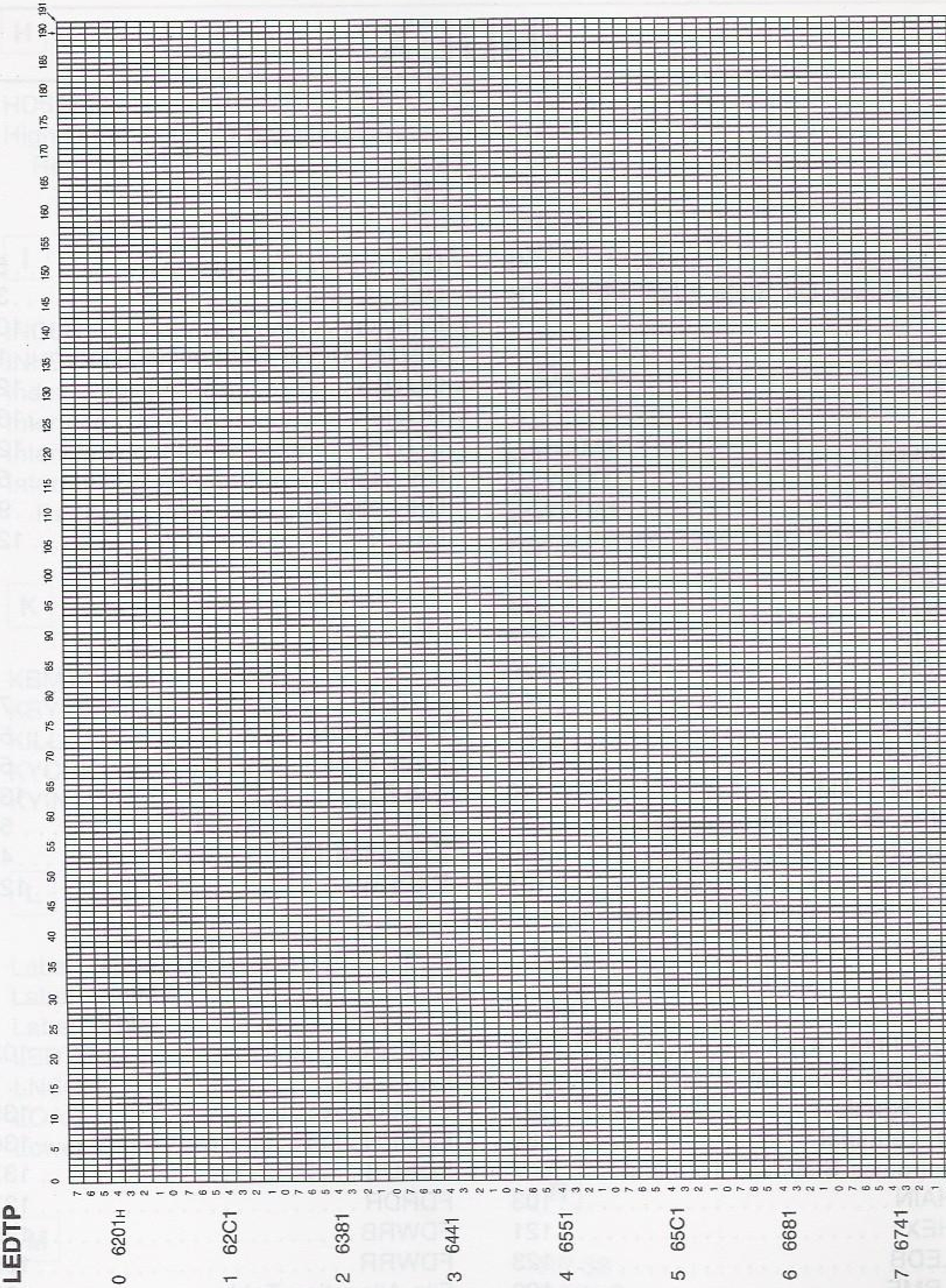
MNEMONIC	OP-CODE HEX		MNEMONIC	OP-CODE HEX	
ADC $, $	00000000	00	JP NZ, m	00110100	34
SBC $, $	00000001	01	JP C, m	00110101	35
LD $, $	00000010	02	JP m	00110111	37
ANC $, $	00000100	04	ADC (IX±$), $	00111000	38
NAC $, $	00000101	05	ADC (IZ±$), $	00111001	39
ORC $, $	00000110	06	AD (IX±$), $	00111100	3C
XRC $, $	00000111	07	AD (IZ±$), $	00111101	3D
AD $, $	00001000	08	SB (IX±$), $	00111110	3E
SB $, $	00001001	09	SBC (IX±$), $	00111110	3E
ADB $, $	00001010	0A	SB (IZ±$), $	00111111	3F
SBB $, $	00001011	0B	SBC (IZ±$), $	00111111	3F
AN $, $	00001100	0C	ADC $, n	01000000	40
NA $, $	00001101	0D	SBC $, n	01000001	41
OR $, $	00001110	0E	LD $, n	01000010	42
XR $, $	00001111	0F	ANC $, n	01000100	44
ST $, ($)	00010000	10	NAC $, n	01000101	45
LD $, ($)	00010001	11	ORC $, n	01000110	46
PFL $	00010100	14	XRC $, n	01000111	47
PST PD, $	00010110	16	AD $, n	01001000	48
PST PE, $	00010110	16	SB $, n	01001001	49
PST UA, $	00010110	16	ADB $, n	01001010	4A
PST IA, $	00010111	17	SBB $, n	01001011	4B
PST IE, $	00010111	17	AN $, n	01001100	4C
BID $	00011000	18	NA $, n	01001101	4D
BIU $	00011000	18	OR $, n	01001110	4E
ROD $	00011000	18	XR $, n	01001111	4F
ROU $	00011000	18	PST PD, n	01010110	56
DID $	00011010	1A	PST PE, n	01010110	56
DIU $	00011010	1A	PST UA, n	01010110	56
CMP $	00011011	1B	PST IA, n	01010111	57
INV $	00011011	1B	PST IE, n	01010111	57
GFL $	00011100	1C	SUP n	01011100	5C
GPO $	00011100	1C	SDN n	01011101	5D
GST PD, $	00011110	1E	ST $, (IX±n)	01100000	60
GST PE, $	00011110	1E	ST $, (IZ±n)	01100001	61
GST UA, $	00011110	1E	STI $, (IX±n)	01100010	62
GST IA, $	00011111	1F	STI $, (IZ±n)	01100011	63
GST IE, $	00011111	1F	LD $, (IX, ±n)	01101000	68
GST TM, $	00011111	1F	LD $, (IZ, ±n)	01101001	69
ST $, (IX±$)	00100000	20	LDI $, (IX±n)	01101010	6A
ST $, (IZ±$)	00100001	21	LDI $, (IZ±n)	01101011	6B
STI $, (IX±$)	00100010	22	CAL Z, m	01110000	70
STI $, (IZ±$)	00100011	23	CAL NC, m	01110001	71
PHS $	00100110	26	CAL LZ, m	01110010	72
PHU $	00100111	27	CAL UZ, m	01110011	73
LD $, (IX, ±$)	00101000	28	CAL NZ, m	01110100	74
LD $, (IZ, ±$)	00101001	29	CAL C, m	01110101	75
LDI $, (IX±$)	00101010	2A	CAL m	01110111	77
LDI $, (IZ±$)	00101011	2B	ADC (IX±n), $	01111000	78
PPS $	00101110	2E	ADC (IZ±n), $	01111001	79
PPU $	00101111	2F	SBC (IX±n), $	01111010	7A
JP Z, m	00110000	30	SBC (IZ±n), $	01111011	7B
JP NC, m	00110001	31	AD (IX±n), $	01111100	7C
JP LZ, m	00110010	32	AD (IZ±n), $	01111101	7D
JP UZ, m	00110011	33	SB (IX±n), $	01111110	7E

MNEMONIC	OP-CODE	HEX
SB (IZ±n),$	01111111	7F
ADCW $,$	10000000	80
SBCW $,$	10000001	81
LDW $,$	10000010	82
ANCW $,$	10000100	84
NACW $,$	10000101	85
ORCW $,$	10000110	86
XRCW $,$	10000111	87
ADW $,$	10001000	88
SBW $,$	10001001	89
ADBW $,$	10001010	8A
SBBW $,$	10001011	8B
ANW $,$	10001100	8C
NAW $,$	10001101	8D
ORW $,$	10001110	8E
XRW $,$	10001111	8F
STW $,($)	10010000	90
LDW $,($)	10010001	91
PRE IX,$	10010110	96
PRE IY,$	10010110	96
PRE IZ,$	10010110	96
PRE US,$	10010110	96
PRE SS,$	10010111	97
BIDW $	10011000	98
BIUW $	10011000	98
RODW $	10011000	98
ROUW $	10011000	98
BYDW $	10011010	9A
BYUW $	10011010	9A
DIDW $	10011010	9A
DIUW $	10011010	9A
CMPW $	10011011	9B
INVW $	10011011	9B
GRE IX,$	10011110	9E
GRE IY,$	10011110	9E
GRE IZ,$	10011110	9E
GRE US,$	10011110	9E
GRE KY,$	10011111	9F
GRE SS,$	10011111	9F
STW $,(IX±$)	10100000	A0
STW $,(IZ±$)	10100001	A1
STIW $,(IX±$)	10100010	A2
STIW $,(IZ±$)	10100011	A3
PHSW $	10100110	A6
PHUW $	10100111	A7
LDW $,(IX±$)	10101000	A8
LDW $,(IZ±$)	10101001	A9
LDIW $,(IX±$)	10101010	AA
LDIW $,(IZ±$)	10101011	AB
PPSW $	10101110	AE
PPUW $	10101111	AF
JR Z,±p	10110000	B0
JR NC,±p	10110001	B1
JR LZ,±p	10110010	B2
JR UZ,±p	10110011	B3
JR NZ,±p	10110100	B4
JR C,±p	10110101	B5
JR ±p	10110111	B7
ADCW (IX±$),$	10111000	B8
ADCW (IZ±$),$	10111001	B9
SBCW (IX±$),$	10111010	BA
SBCW (IZ±$),$	10111011	BB
ADW (IX±$),$	10111100	BC
ADW (IZ±$),$	10111101	BD
SBW (IX±$),$	10111110	BE
SBW (IZ±$),$	10111111	BF
LDW $,m	11010001	D1
PRE IX,m	11010110	D6
PRE IY,m	11010110	D6
PRE IZ,m	11010110	D6
PRE US,m	11010110	D6
PRE SS,m	11010111	D7
BUP	11011000	D8
BDN	11011001	D9
SUP $	11011100	DC
SDN $	11011101	DD
RTN Z	11110000	F0
RTN NC	11110001	F1
RTN LZ	11110010	F2
RTN UZ	11110011	F3
RTN NZ	11110100	F4
RTN C	11110101	F5
RTN	11110111	F7
NOP	11111000	F8
CLT	11111001	F9
FST	11111010	FA
SLW	11111011	FB
CANI	11111100	FC
RTNI	11111101	FD
OFF	11111110	FE
TRP	11111111	FF

13-7 LAYOUT FORM

EDTOP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	6100																															611F
1	6120																															613F
2	6140																															615F
3	6160																															617F
4	6180																															619F
5	61A0																															61BF
6	61C0																															61DF
7	61E0																															61FF


LEDTP
0 6201H
1 62C1
2 6381
3 6441
4 6551
5 65C1
6 6681
7 6741

INDEX

S

T

U

V

Z

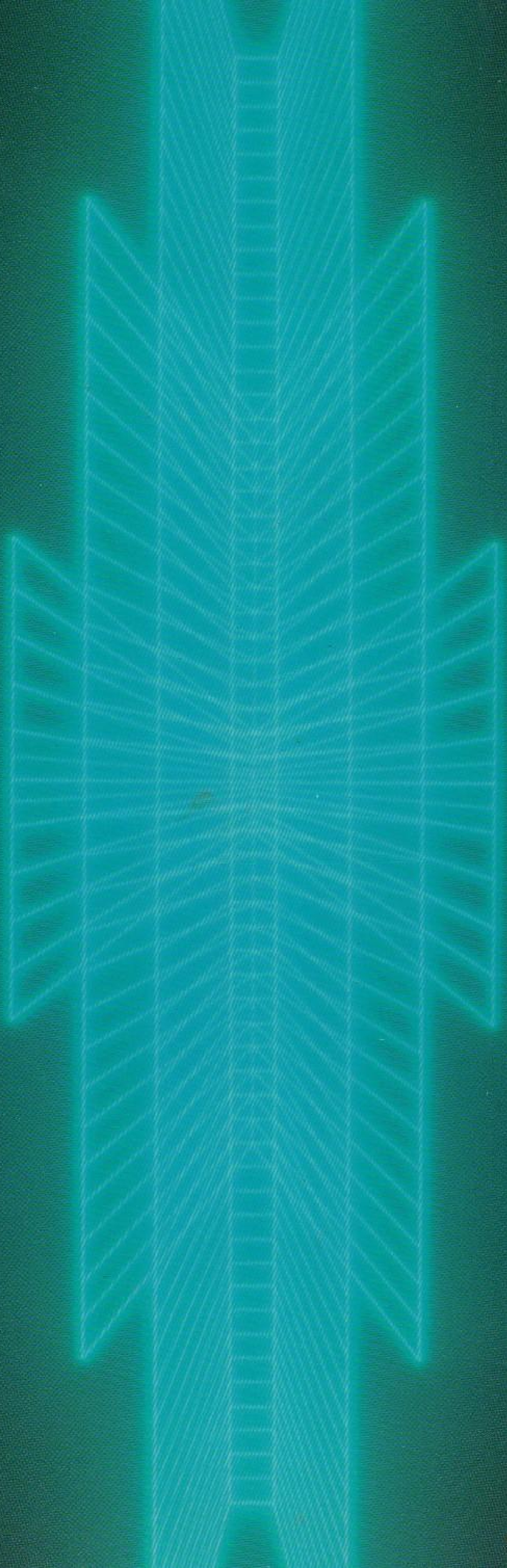
S

T

U

V

Z



CASIO®

MO0200572A Printed in Japan