3.6.2.2 Memory map

Figure 3.6.2 shows the memory map for the screen-related areas.

VRAM 1 is the VRAM for the user screen and located in the shared system area (at ØEØØØH and above). It holds data in bit image; that is, each bit in VRAM 1 corresponds to a single dot on the LCD.

VRAM 2 is used by the system screen. When the screen mode is switched from user to system screen mode, the bytes in VRAM 2 have one-to-one correspondence with dots on the LCD.

The user-defined character area is used to store the fonts of the user-defined characters.

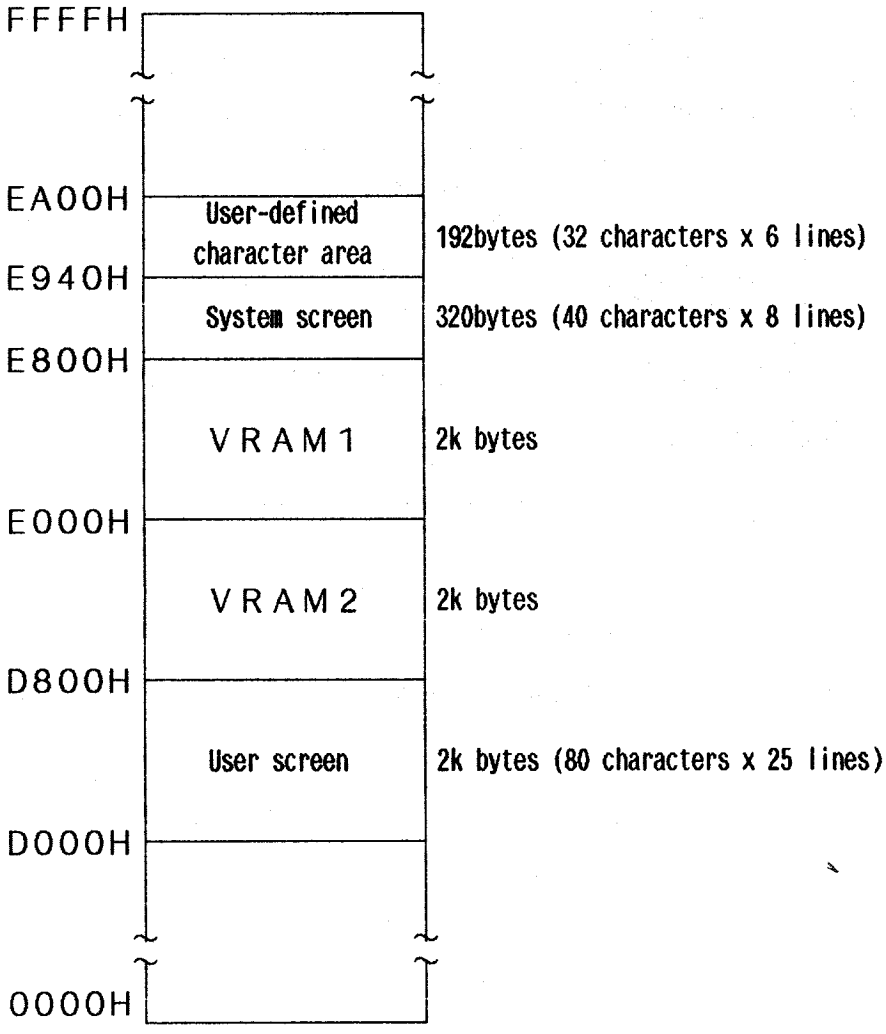


Fig. 3.6.2 Screen-related Area Memory Map

3.6.2.3 VRAM structure

On the following pages are descriptions of the structure and the use of VRAM.

Each bit in VRAM corresponds to a single dot on the LCD panel. The relationship between the LCD panel and VRAM is shown in Figure 3.6.3.

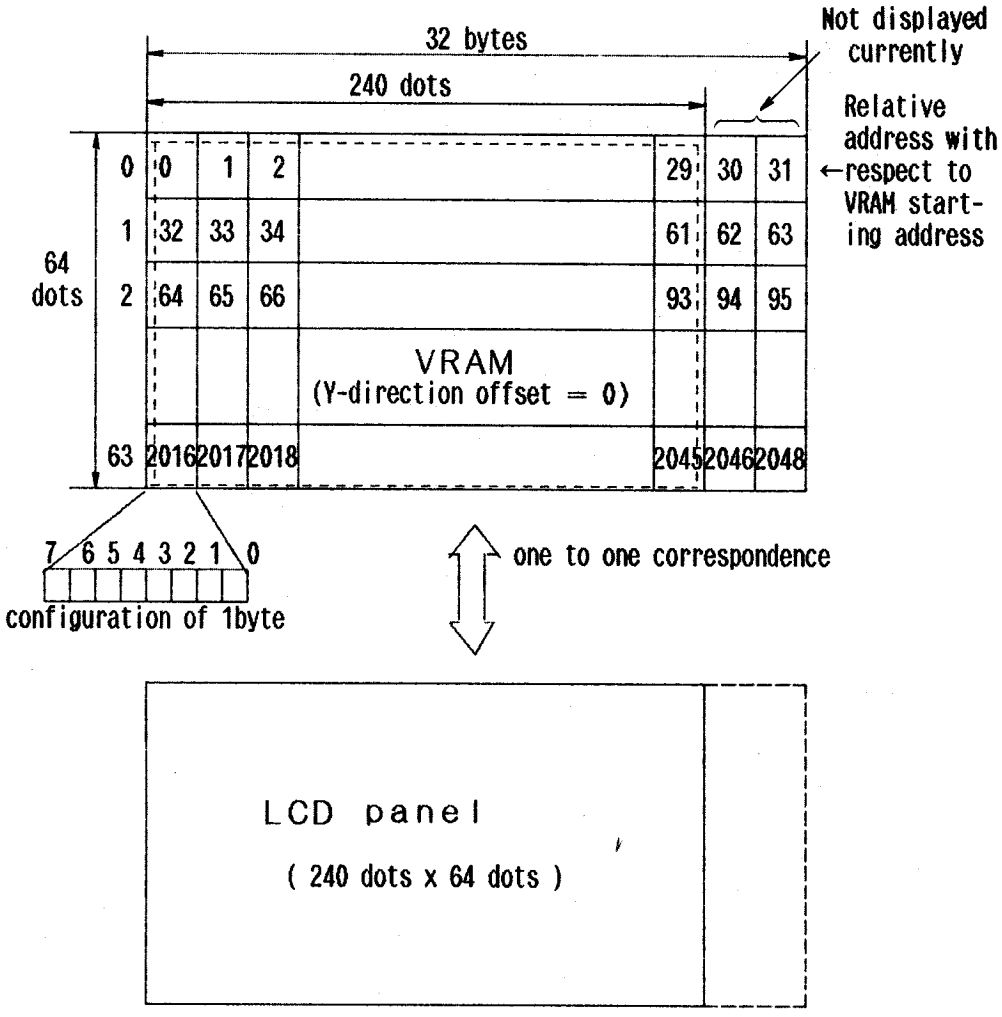


Fig. 3.6.3 Relationship Between VRAM and LCD Panel

The address at which display of VRAM data on the LCD panel begins is determined by:

- VRAM starting address (LSCRVRAM)
- Y direction offset (LVRAMYOF)

The Y-direction offset indicates the vertical correspondence between VRAM and the LCD panel. Display begins at the location offset dots away from the VRAM bottom and wraps around to the VRAM top when it reaches the VRAM bottom. The system uses this feature when performing a vertical scroll.

Figure 3.6.4 illustrates the relationship between the VRAM relative addresses and the LCD panel when the offset is set to two dots.

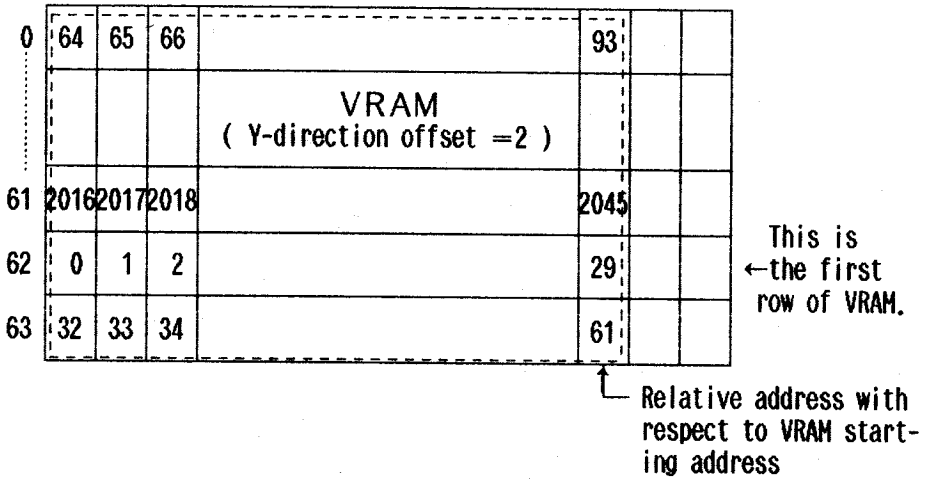


Fig. 3.6.4 Relationship Between VRAM and LCD Panel (When Y-direction Offset = 2)

The following system areas are used to control VRAM display:

LSCRVRAM (ØF294H) 2 bytes
- Contains the starting address of the VRAM (VRAM 1 or VRAM 2) whose contents are currently being displayed on the LCD display. The address must be 8ØØØH or higher and must be on a 2K-byte boundary.

LVRAMYOF (ØF2ADH) 1 byte
- Contains the VRAM Y-direction offset.
Ø ≤ LVRAMYOF ≤ 63
The offset value is usually a multiple of 8.

3.6.2.4 Switching the screen mode

The PINE OS supports two screen modes: the system screen mode used by the system and the user screen mode used by the user. Mode switching is automatically carried out by the OS whenever required. The OS has subroutines for this purpose.

(1) System screen mode
The system screen mode is the one in which the system screen is used with VRAM2. This mode is used mainly by the system. The system handles the switching to this mode as a kind of interrupt generated in the user screen mode. This mode is used by system display, alarm screen, and power fail screen functions.

(2) User screen mode
The user screen mode is the one in which the user screen is used with VRAM1. This mode is available to the user. Normal screen display is performed in this mode.

(3) Mode switching
Screen modes can be switched using the XUSRSCRN and XSYSSCRN subroutines. The entry addresses of these subroutines are found in the OS jump table (see Section 4.2, "Jump Tables").

XUSRSCRN -- Switches the screen mode to user screen.
XSYSSCRN -- Switches the screen mode to system screen.

(4) Note
The user may also use the system screen in the system screen mode. When the system screen is used by the user, the contents of the system screen will be lost if 1) the system display is made by means of the CTRL and HELP keys, 2) the alarm screen is displayed at an alarm/wake time, or 3) the power fail screen is displayed to signal a voltage drop condition. The screen mode is switched back to user screen when the screens is exited.

The screen is always in the user screen mode when the POWER is powered on whether the power-off state is in the continue mode or restart mode.

3.6.2.5 System screen

The system screen is used mainly by the system to display the system display, alarm screen, and power fail screen. It consists of 8 lines of 40 characters. This size is fixed.

The system screen is shown in Figure 3.6.5.

The system screen is functionally equivalent to a user screen whose size is fixed at 40 characters x 8 lines.

In the system screen mode, the cursor always moves in the tracking mode and thus does not move out of the screen.

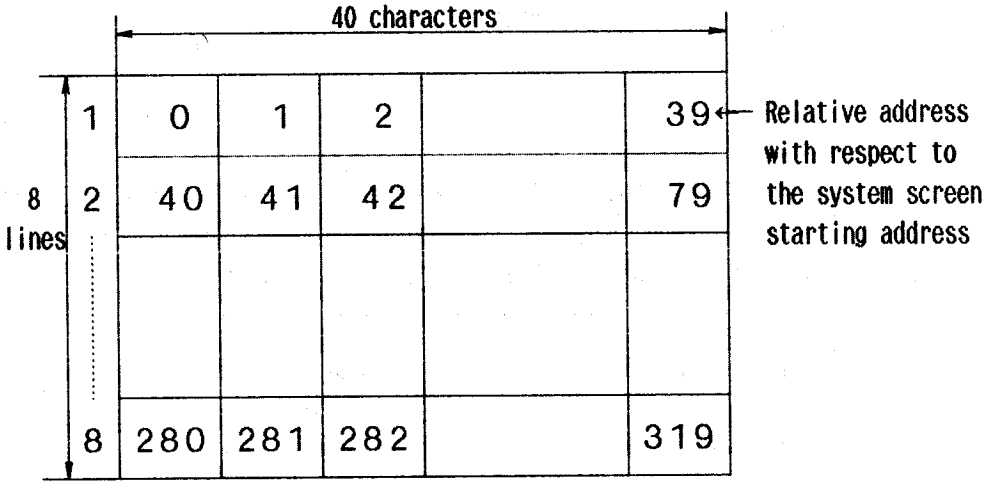


Fig. 3.6.5 System Screen Structure

System screen data is stored in the form of character generator codes.

3.6.2.6 User screen

Although the PINE has a physical display screen of 40 characters by 8 lines, it provides larger virtual screens to meet the needs of the applications programs which require larger screens. See 3.5.3 for the virtual screen.

The user screen size may be in the range from 40 characters by 8 lines to 80 characters by 25 lines or 40 characters by 50 lines.

The structure of the system screen is shown in Figure 3.6.6.

User screen data can be read using the BIOS RDVRAM routine.

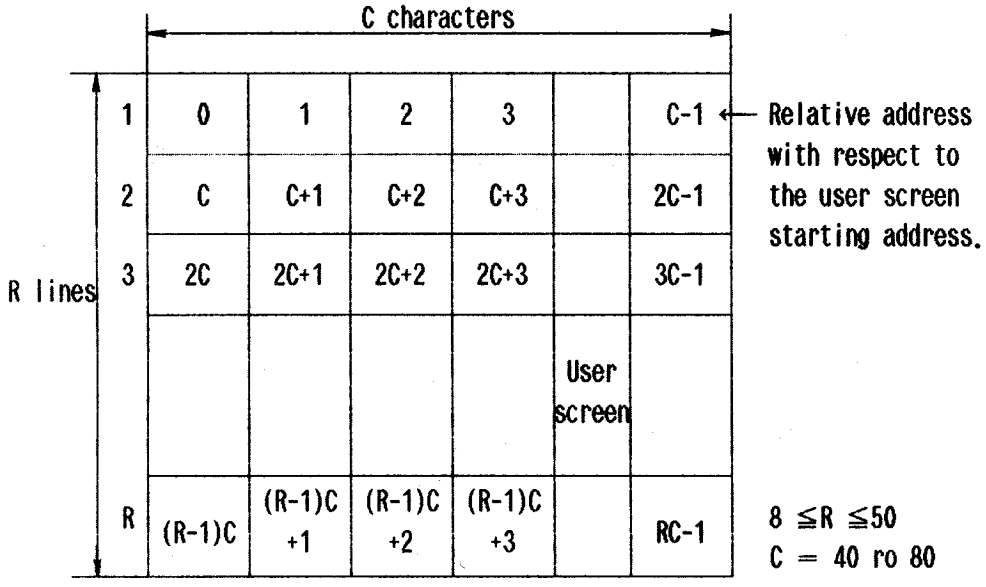


Fig. 3.6.6 User Screen Structure

User screen data is stored in the form of character generator codes.

3.6.3 Virtual Screen

3.6.3.1 Outline

The PINE uses the concept of virtual screen which allows the user to use a screen larger than it actually is (40 characters x 8 liens). The virtual screen may as large as 80 characters by 25 lines or 40 characters by 50 lines.

The portion of the virtual screen that is actually displayed is referred to as a window (40 characters x 8 lines) through which the part of the virtual screen can be viewed.

3.6.3.2 Virtual screen structure

The structure of the virtual screen is shown in Figure 3.6.7.

The virtual screen may be 40 or 80 characters wide. Its length may be any value in the range from 8 to 50 lines provided that (width x length) does not exceed 2000.

The window size is fixed at 40 characters by 8 lines. The entire contents of the virtual screen can be viewed by scrolling the window up and down or right and left over the virtual screen.

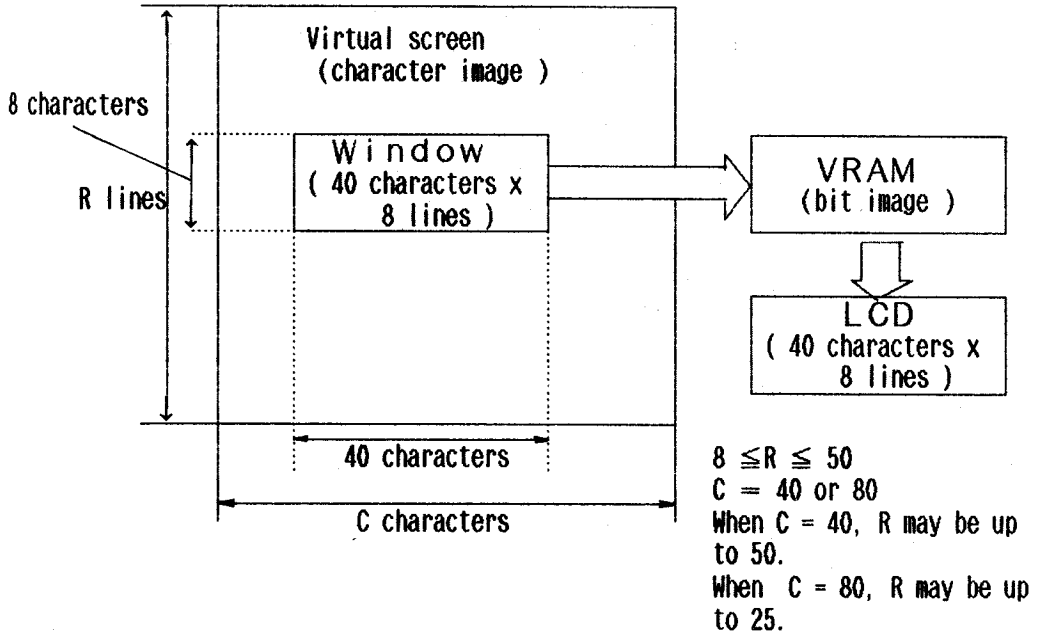


Fig. 3.6.7 Virtual Screen Structure

3.6.3.3 Scroll modes

The window scrolls over the virtual screen in the following three modes:

- Tracking mode
- Nontracking mode
- Horizontally nontracking mode

The window moves, as the cursor moves, in different manners depending on the scroll mode.

The scroll mode can be specified from the keyboard or using commands (ESC sequences).

(1) Tracking mode
In this mode, the window scrolls following the cursor. If the mode is switched from nontracking to tracking when the cursor is out of the window, the window automatically moves up to the point where the cursor is stationed. The tracking mode is valid in both vertical and horizontally directions.

(2) Nontracking mode
In this mode, the cursor does not follow the cursor movement. The window remains in the current position even when data is written into the virtual screen.

The display changes when the cursor moves out of the virtual screen, which triggers an automatic window scroll. However, the cursor in the virtual screen stays in the original position.

(3) Horizontally nontracking mode
This mode is valid only when the virtual screen is 80 characters wide. In this mode, the window follows the cursor movement only vertically and does not follow horizontally.

(4) Specifying the scroll mode
The scroll mode can be changed by pressing the SHIFT and INS keys simultaneously or sending an ESC sequence to the LCD through the BIOS CONOUT routine.

The mode changes cyclically from tracking to nontracking, from nontracking to horizontally nontracking, from horizontally nontracking to tracking, and so on as the SHIFT and INS keys are pressed. See Section 3.5, "Keyboard" for the SHIFT and INS keys.

The scroll mode can also be set by sending an ESC sequence (ESC + 95H) via the BIOS CONOUT routine. See the descriptions of the ESC sequences for how to set the scroll mode.

The system area shown below indicates the current scroll mode.

LSCROLMD (0F2A3H) 1 byte
- Current scroll mode
  = 00H: Tracking mode
  = 01H: Nontracking mode
  = 02H: Horizontally nontracking mode

3.6.3.4 Scrolling the window

The window can be scrolled by 1) moving the cursor, 2) using commands (ESC sequences), or 3) manipulating keys.

(1) Horizontal cursor movement
When the virtual screen is 80 characters wide and in the tracking mode, the window scrolls right and left following the horizontal movement of the cursor.

The horizontal scroll step (w) may be 20 or 40 columns. It can be specified using the ESC sequence "SET SCROLL STEP" (ESC + 94H).
The window scroll margin (m) may be in the range from 0 to 10. This can be specified using the ESC sequence "SET SCROLL MARGIN" (ESC + 98H). The the scroll margin is valid only when the horizontal scroll step is set to 20 columns.

Figures 3.6.8 and 3.6.9 show the relationships between the cursor and the window.

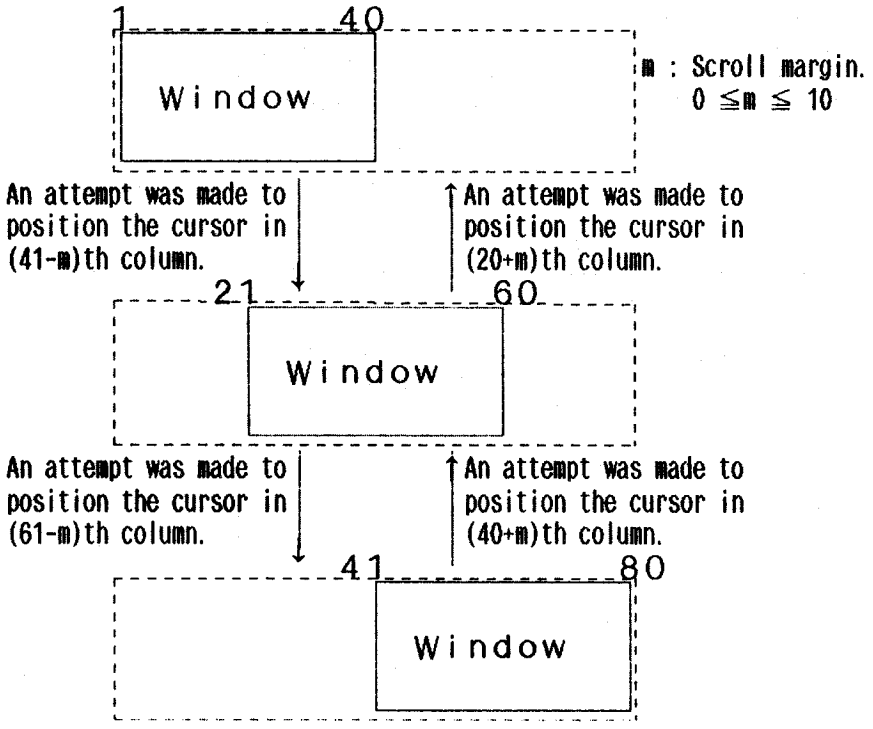


Fig. 3.6.8 Cursor Movements and the Window (When w = 20)

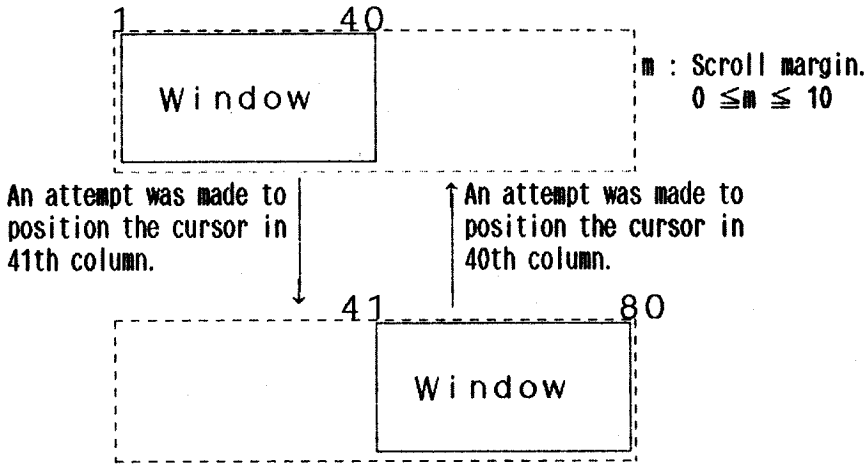


Fig. 3.6.9 Cursor Movements and the Window (When w = 40)

(2) Vertical cursor movement
The vertical scroll step for cursor movement is always set to 1. Whenever the cursor moves beyond the window vertically, the window scrolls one line up or down. The window remains in the current position in the nontracking mode, however.

The vertical scroll step specified by "SET SCROLL STEP" has no effect when the cursor is moving. It is valid only when the window scrolls vertically.

(3) Scrolling under command control
The window can be scrolled vertically or horizontally using commands (ESC sequences).

Horizontal scroll is valid only when the virtual screen is 80 characters wide. In other cases, nothing takes place when horizontal scroll is specified.

Horizontal scroll can be initiated by sending the ESC sequence "SCROLL RIGHT (LEFT) N CHAR" (ESC + 92H or 93H). The scroll step can be specified using the ESC sequence "SET SCROLL STEP" (ESC + 94H).

In the horizontal scroll mode, the cursor remains in the current position and only the window moves over the virtual screen.

Vertical scroll is valid when the virtual screen length is other than 8 lines. Nothing will take place if its length is set to 8 lines.

Vertical scroll can be started by sending the ESC sequence "SCROLL UP (DOWN) M LINE" (ESC + 96H or 97H). The scroll step can be specified using the ESC sequence "SET SCROLL STEP" (ESC + 94H).

In the vertical scroll mode, the cursor remains in the current position and only the window moves over the virtual screen.

(4) Scrolling under key control

In PINE OS, scrolling of the window in the horizontal and vertical directions can be controlled using special keys.

SHIFT/→ Scrolls the window 20 columns to the right.
SHIFT/← Scrolls the window 20 columns to the left.
SHIFT/↑ Scrolls the window one line up.
SHIFT/↓ Scrolls the window one line down.
CTRL/→ Scrolls the window 40 columns to the right.
CTRL/← Scrolls the window 40 columns to the left.
CTRL/↑ Scrolls the window eight lines up.
CTRL/↓ Scrolls the window eight lines down.

Any combination of these keys causes the window to scroll without moving the cursor. See Section 3.5, "Keyboard" for details.

(5) System areas used for controlling window scrolling

LSCROLX (0F2A4H) 1 byte
- Indicates the horizontal scroll step. The scroll step must be 20 or 40 columns. The default value is 20 columns.

LSCROLY (0F2A5H) 1 byte
- Indicates the vertical scroll step. The scroll step must be 1 to 8 lines. The default value is one line.

LLMARGIN (0F2DFH) 1 byte
- Indicates the left margin. The left margin must be 0 to 10 columns. The default value is 5 columns.

LRMARGIN (0F2E0H) 1 byte
- Indicates the right margin. The right margin must be 0 to 10 columns. The default value is 5 columns.

3.6.4 Graphics Display

3.6.4.1 Outline

Graphics data can be written directly into VRAM either in the system screen mode or in the user screen mode.

Graphics data may be mixed with character data in the same screen. If a portion of the currently displayed graphics data goes off the window as the window scrolls, that portion of data is deleted. That is, graphics data will not be redisplayed when the window scrolls back to the original position.

3.6.4.2 Graphics coordinates

The graphics screen is made up of 239 dots by 63 dots with coordinates (Ø,Ø) or origin assigned to the upper left corner of the LCD display screen.

As described in 3.6.2., the Y-direction offset value (in LVRAMYOF) must be taken into consideration when establishing the correspondence between VRAM and LCD..

Figure 3.6.4 shows the correspondence between VRAM and LCD viewed from the LCD's standpoint and Figure 3.6.1Ø shows that viewed from the VRAM's standpoint.

The address of coordinates (X, Y) in VRAM can be obtained using the formula

$$\{(y + Y) \bmod 64\} * 32 + (X / 8)$$

where $Ø \leq X \leq 239$, $Ø \leq Y \leq 63$, and y is a Y-direction offset. The integer part of the result is the relative address with respect to the top of VRAM. The fractional part designates the bit position of the coordinates in the relative address.

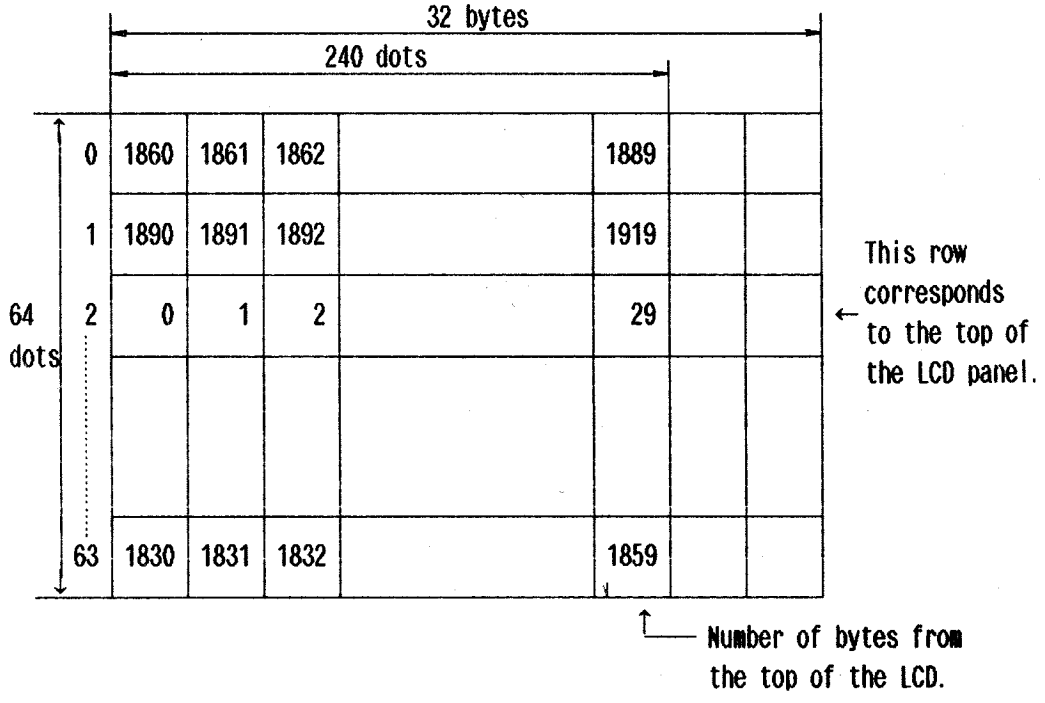


Fig. 3.6.1Ø Relationship Between VRAM and LCD Panel (When Y-direction Offset = 2)

3.6.4.3 Writing graphics data

Graphics data can be displayed on the screen by calling the BIOS PSET routine or by writing the data directly into VRAM.

(1) BIOS PSET
The BIOS PSET routine performs a specified logical operation (AND, OR, XOR) on the VRAM data and the specified data on a byte basis. See Section 3.3, "BIOS Operations" for further information.

(2) Writing data directly into VRAM
Data can be written into VRAM directly with ease when the user screen is used because VRAM is located in the shared system area (RAM area at location ØEØØØH and above).

The formula given on the previous page is used to find the VRAM address into which the data is to be written. Its absolute RAM address can be obtained by adding the VRAM starting address (LSCRVRAM) to the VRAM address.

3.6.5 CONOUT (BIOS)

3.6.5.1 Outline

The PINE CONOUT BIOS routine controls the keyboard, LEDs, and buzzer as well as the LCD.

The description that start at the next page hold when the CON: field of the I/O byte is set to LCD.

The pages that follow lists the functions of the PINE CONOUT BIOS routine. A full descriptions of the CONOUT functions are found in 3.6.5.3, "CONOUT specifications."

Code	Function	code	Function
02H	Screen left	ESC+92H	Scroll right n char
05H	Erase end of screen	ESC+93H	Scroll left n char
06H	Screen right	ESC+94H	Set scroll step
07H	Bell	ESC+95H	Set scroll mode
08H	Back space	ESC+96H	Scroll up m line
09H	Tab	ESC+97H	Scroll down m line
0AH	Line feed	ESC+98H	Set scroll margin
0BH	Home	ESC+A0H	INS LED on
0CH	Clear screen & Home	ESC+A1H	INS LED off
0DH	Carriage return	ESC+A2H	CAPS LOCK LED on
10H	Screen up	ESC+A3H	CAPS LOCK LED off
11H	Screen down	ESC+A4H	NUM LED on
1AH	Erase end of screen	ESC+A5H	NUM LED off
1BH	Escape (ESC)	ESC+B0H	Function key check mode on
1CH	Cursor right	ESC+B1H	Function key check mode off
1DH	Cursor left	ESC+D0H	Set screen size
1EH	Cursor up	ESC+D2H	Direct display
1FH	Cursor down	ESC+D4H	Locate top of screen
		ESC+D5H	Locate bottom of screen
ESC+'%'	Access CGROM directly	ESC+D6H	Select cursor kind
ESC+'('	Block reverse	ESC+D7H	Find cursor
ESC+'*'	Clear screen & Home	ESC+E0H	Set download character
ESC+'0'	Reverse on	ESC+F0H	Keyboard repeat on/off
ESC+'1'	Reverse off	ESC+F1H	Set keyboard repeat start time
ESC+'2'	Cursor off		
ESC+'3'	Cursor on	ESC+F2H	Set keyboard repeat interval time
ESC+'='	Set cursor position		
ESC+'C'	Set character-set table	ESC+F3H	Set arrow key code
ESC+'P'	Screen dump	ESC+F4H	Set scroll key code
ESC+'T'	Erase end of line	ESC+F5H	Set control key code
ESC+'Y'	Erase end of screen	ESC+F6H	Clear key buffer
ESC+7BH	Secret	ESC+F7H	Set key shift
ESC+7DH	Non secret		
ESC+90H	Partial scroll up		
ESC+91H	Partial scroll down		

3.6.5.2 How to use the CONOUT function

(1) Entry address: WBOOT + 09H or 0EB0CH
(2) Entry parameter: C = Output data
(3) Return parameter: None.
(4) How to use:
The BIOS CONOUT routine inputs a function code and data. The function code and data are distinguished by their numeric values as shown below:
- Function code: 00H - 1FH
1BH (ESC) + n1 + n2 + --- + nk
- Data: 20H - 0FFH

The CONOUT routine does nothing but preserves the original state when it detects an invalid function code or parameter in the ESC sequence received. The routine does not check the ESC sequence for parameter errors until it receives all parameters.

3.6.5.3 CONOUT specifications

This large subsection describes the operations that the PINE OS performs when it receives commands through the CONOUT BIOS function.

The word "screen" here refers to either the user screen if the screen is currently in the user mode and to the system screen if the screen is in the system mode.

"The cursor goes out of the window" here means that the cursor is located literally outside the window when the cursor moved in the vertical direction and, when the cursor moved in the horizontal direction, that the cursor moves beyond the window whose sides are delineated by the scroll margins.

- 02H Screen left
Moves the window one screenful of columns to the left over the screen.

The left edge of the window is aligned with that of the screen when an attempt is made to move the window beyond the left end of the screen. The cursor remains in the original position on the screen.

- 05H Erase end of line
Clears with spaces to the end of the line from the current cursor position (inclusive) on the screen. The cursor remains in the original position on the line.

- 06H Screen right
Moves the window one screenful of columns to the right over the screen.

The right edge of the window is aligned with that of the screen when an attempt is made to move the window beyond the right end of the screen. The cursor remains in the original position on the screen.

- 07H Bell
Sounds the buzzer at 880 Hz for one second. CONOUT is not exited until the buzzer stops.

- ØBH Backspace
Moves the cursor one column to the left on the screen.

The cursor moves to the end of the previous line when it is at the beginning of the line when this function is executed. The cursor does not move when it is in the home position (upper left corner) of the screen. How the window behaves when the cursor goes out of the window depends on the scroll mode.

- Ø9H Tab
Searches for the next tab position starting at the current cursor position on the screen forward and positions the cursor in the first tab position encountered.

If no tab position is found on the line, the function moves the cursor to the beginning of the next line. If the cursor is at the bottom of the screen when this function is executed, the function positions the cursor at the beginning of the line. How the window behaves when the cursor goes out of the window depends on the scroll mode.
Tab position = (1 + 8*n) column n = Ø, 1, 2, ---

- ØAH Line feed
Moves the cursor down one line on the screen.

If the cursor is on the bottom line of the screen when this function is executed, the function scrolls the screen one line down (deletes the first line on the screen and puts a blank line at the bottom of the screen) and moves the cursor one line down. How the window behaves when the cursor goes out of the window depends on the scroll mode.

- ØBH Home
Positions the cursor to the home position (upper left corner) of the screen.

How the window behaves when the cursor goes out of the window depends on the scroll mode.

- ØCH Clear screen & home
Clears the entire screen with spaces and performs the Home function (ØBH).

- ØDH Carriage return
Moves the cursor to the first column on the current line.

How the window behaves when the cursor goes out of the window depends on the scroll mode.

When this function is executed immediately after the cursor is moved from the last column into the first column on the next line as the result of displaying a character (2ØH - ØFFH), the function positions the cursor in the first column on the previous line.

- 1ØH Screen up
Moves the window one screenful of columns down over the screen.

The last line of the window is aligned with the bottom of the screen when an attempt is made to moves the window across the bottom of the screen. The cursor is held in the original position on the screen.

- 11H Screen down
Moves the window one screenful of columns the screen.

The first line of the window is aligned with the top of the screen when an attempt is made to moves the window across the top of the screen. The cursor is held in the original position on the screen.

- 1AH Erase end of screen
Clears to the end of the screen from the current cursor position (inclusive) on the screen. The cursor stays in the original position on the screen.

- 1BH Escape
Initiates an ESC sequence. The commands entered in the form of an ESC sequence are described later.

- 1CH Cursor right
Moves the cursor one column to the right on the screen.

If the cursor is in the last column on a line when this function is executed, the function positions the cursor in the first column on the next line. The function does nothing if the cursor is in the last column on the last line. How the window behaves when the cursor goes out of the window depends on the scroll mode.

- 1DH Cursor left
Performs the same function as Back space (08H).

- 1EH Cursor up
Moves the cursor one line up on the screen.

This function does nothing if it is executed when the cursor is positioned on the first line of the screen. How the window behaves when the cursor goes out of the window depends on the scroll mode.

- 1FH Cursor down
Moves the cursor one line down on the screen.

This function does nothing if it is executed when the cursor is positioned on the last line of the screen. How the window behaves when the cursor goes out of the window depends on the scroll mode.

- 20H - 0FFH Character display
Displays the character associated with the given character code in the current cursor position and then moves the cursor one column to the right.

If the cursor is in the last column on a line, this function displays a character and positions the cursor in the first column on the next line. If the cursor is in the last column on the last line when the function is executed, the display automatically scrolls one line down and the cursor is placed at the beginning of the next line. How the window behaves when the cursor goes out of the window depends on the current scroll mode.

- ESC '%' Access CGROM directly

Reads the character specified by the given code from the character generator and displays it in the cursor position on the screen.

The cursor is positioned in the same way as when the character display function (20H - 0FFH) is executed.

<Command sequence>

1st byte: ESC
2nd byte: '%' n: Code
3rd byte: n 00H ≤ n ≤ 0FFH

- ESC 'C' Block reverse

Displays the specified length of data in reverse video starting at the specified position on the screen.

The reverse video function is cancelled when the data displayed in reverse video goes out of the window during scrolling.

<Command sequence>

1st byte: ESC
2nd byte: 'C' Y: Y-coordinate 1 ≤ Y ≤ 8
3rd byte: Y X: X-coordinate 1 ≤ X ≤ 40
4th byte: X n: Number of characters displayed in
5th byte: n (H) reverse video.
6th byte: n (L) 1 ≤ n ≤ 320

- ESC '*' Clear screen & home

Performs the same function as Clear screen & home (0CH).

<Command sequence>

1st byte: ESC
2nd byte: '*'

- ESC '0' Reverse on

Turns on the reverse display mode and displays the subsequent output characters in reverse video.

The reverse video function is cancelled when the data displayed in reverse video goes out of the window during scrolling.

<Command sequence>

1st byte: ESC
2nd byte: '0'

- ESC '1' Reverse off

Turns off the reverse display mode.

<Command sequence>

1st byte: ESC
2nd byte: '1'

- ESC '2' Cursor off

Suppresses the cursor display. The cursor can subsequently move around, though invisible.

<Command sequence>

1st byte: ESC
2nd byte: '2'

- ESC '3' Cursor on
Displays the cursor.

<Command sequence>
1st byte: ESC
2nd byte: '3'

- ESC '=' Set cursor position
Specifies the cursor position on the screen and positions it in that position.

When the cursor goes out of the window in the tracking mode:
  - If the cursor moves in vertical direction: This function scrolls the window so that the cursor is positioned on the fourth line of the window. If the window goes off the screen, the function aligns the bottom or the top of the window with that of the screen.
  - If the cursor moves in the horizontal direction: This function does not move the window if the cursor position after the move will be within the window. If the cursor goes off the window, however, the function scrolls the window in increments of the scroll step specified.
When the cursor goes out of the window in the nontracking mode:
  The function does not scroll the window.

<Command sequence>
1st byte: ESC
2nd byte: '=' m: Row position
3rd byte: m+1FH 1 ≤ m ≤ Number of lines on the screen
4th byte: n+1FH n: Column position
                 1 ≤ n ≤ Number of columns on the screen

- ESC 'C' Set character set table
Sets up the character set for the given language.

The default language is set by DIP switches. This function does not affect the characters already displayed on the screen.

<Command sequence>
1st byte: ESC ID: Character set identification character
2nd byte: 'C' 'U': USA 'W': Sweden
3rd byte: ID 'F': France 'I': Italy
              'G': Germany 'S': Spain
              'E': England 'N': Norway
              'D': Denmark

- ESC 'P' Screen dump
Outputs the currently displayed VRAM data to the printer.

See BIOS SCRNDUMP for details.

<Command sequence>
1st byte: ESC
2nd byte: 'P'

- ESC 'T' Erase end of line
Performs the same function as Erase end of line (Ø5H).

<Command sequence>
1st byte: ESC
2nd byte: 'T'

- ESC 'Y' Erase end of screen
Performs the same function as Erase end of screen (1AH).

<Command sequence>
1st byte: ESC
2nd byte: 'Y'

- ESC 7BH Secret
Displays characters in the secret mode.

In this mode, characters are converted to spaces when directed to the screen for display.

<Command sequence>
1st byte: ESC
2nd byte: 7BH

- ESC 7DH Nonsecret
Cancels the screen mode.

<Command sequence>
1st byte: ESC
2nd byte: 7DH

- ESC 90H Partial scroll up
Scrolls m lines starting at the nth line up by one line.

This function erases the data on the nth line and changes the data on the (n + m - 1)th line to spaces. If (n + m - 1) is larger than the number of lines on the screen, the function automatically adjusts the value of m so that (n + m - 1) matches the maximum line number.

The cursor is held in the original position on the screen.

<Command sequence>
1st byte: ESC n: Number of the line at which scroll starts.
2nd byte: 90H 1 ≤ n ≤ Number of lines on the screen
3rd byte: n - 1 m: Scroll range
4th byte: m 1 ≤ m ≤ Number of lines on the screen

- ESC 91H Partial scroll down
Scrolls m lines starting at the nth line down by one line.

This function erases the data on the (n+m-1)th line and changes the data on the nth line to spaces. If (n + m - 1) is larger than the number of lines on the screen, the function automatically adjusts the value of m so that (n + m - 1) matches the maximum line number.

The cursor is held in the original position on the screen.

<Command sequence>
1st byte: ESC n: Number of the line at which scroll starts.
2nd byte: 91H 1 ≤ n ≤ Number of lines on the screen
3rd byte: n - 1 m: Scroll range
4th byte: m 1 ≤ m ≤ Number of lines on the screen

- ESC 93H Scroll right n char
Moves the window to the right in the increment of the horizontal scroll step specified.

The right edge of the window is aligned with that of the screen if an attempt is made to move the window beyond the right end of the screen. The cursor remains in the original position on the screen.

<Command sequence>
1st byte: ESC
2nd byte: 92H

- ESC 93H Scroll left n char
Moves the window to the left in the increment of the horizontal scroll step specified.

The left edge of the window is aligned with that of the screen if an attempt is made to move the window beyond the left end of the screen. The cursor remains in the original position on the screen.

<Command sequence>
1st byte: ESC
2nd byte: 93H

- ESC 94H Set scroll step
Specifies the number of lines or columns the window is to be moved in a single scroll operation.

<Command sequence>
1st byte: ESC n: Number of columns to scroll
2nd byte: 94H n = 20 or 40
3rd byte: n m: Number of lines to scroll
4th byte: m 1 ≤ m ≤ 8

The default values of n and m are 20 and 1, respectively.

- ESC 95H Set scroll mode
Specifies the scroll mode.

<Command sequence>
1st byte: ESC M: Scroll mode
2nd byte: 95H = 00H: Tracking mode
3rd byte: M = 01H: Nontracking mode
= 02H: Horizontally nontracking mode

The default mode is horizontally nontracking mode.

- ESC 96H Scroll up m lines
Moves the window up in the increment of the vertical scroll step specified.

The top of the window is aligned with that of the screen if an attempt is made to move the window beyond the screen. The cursor is held in the original position on the screen.

<Command sequence>
1st byte: ESC
2nd byte: 96H

- ESC 97H Scroll down m lines
Moves the window down in the increment of the vertical scroll step specified.

The bottom of the window is aligned with that of the screen if an attempt is made to move the window beyond the screen. The cursor is held in the original position on the screen.

<Command sequence>
1st byte: ESC
2nd byte: 97H

- ESC 98H Set scroll margin
Specifies the horizontal scroll margin.

<Command sequence>
1st byte: ESC
2nd byte: 98H n: Scroll margin
3rd byte: n Ø ≤ n ≤ 1Ø

The default values of n is 5.

- ESC AØH INS LED on
Turns on the INS LED.

The INS LED is the third LED from the top on the standard keyboard and the third LED from the left on the item keyboard.

<Command sequence>
1st byte: ESC
2nd byte: ØAØH

- ESC ØA1H INS LED off
Turns off the INS LED.

<Command sequence>
1st byte: ESC
2nd byte: ØA1H

- ESC A2H CAPS LOCK LED on
Turns on the CAPS LOCK LED.

The CAPS LOCK LED is on the top of the standard keyboard and in the leftmost position on the item keyboard.

<Command sequence>
1st byte: ESC
2nd byte: ØA2H

- ESC A3H CAPS LOCK LED off
Turns off the CAPS LOCK LED.

<Command sequence>
1st byte: ESC
2nd byte: ØA3H

- ESC A4H NUM LED on
Turns on the NUM LED.

The NUM LED is in the next to top position on the standard keyboard and in the next to leftmost position on the item keyboard.

<Command sequence>
1st byte: ESC
2nd byte: ØA4H

- ESC A5H NUM LED off
Turns off the NUM LED.

<Command sequence>
1st byte: ESC
2nd byte: ØA5H

- ESC BØH Function key check mode on
Turns on the mode (PF key check mode) in which PF keys return their inherent codes instead of the defined strings.

See BIOS CONIN for details.

<Command sequence>
1st byte: ESC
2nd byte: ØBØH

- ESC B1H Function key check mode off
Cancels athe PF key check mode.

When a defined PF key is pressed after this function is executed, the associated string is returned.

<Command sequence>
1st byte: ESC
2nd byte: ØB1H

- ESC DØH Set screen size
Defines the user screen size. This function does nothing in the system screen mode.

When the screen size is specified, the function reserves that size of area in memory and executes Clear screen & home (ØCH). The CP/M size remains unchanged even when the screen size is changed.

<Command sequence>
1st byte: ESC n: Number of screen lines
2nd byte: ØDØH 8 ≤ n ≤ 5Ø
3rd byte: n m: Number of screen columns
4th byte: m m = 4Ø or 8Ø
m * n ≤ 2Ø48

The default setting is 8Ø columns by 25 lines.

- ESC D2H Direct display
Displays the specified character in the specified position in the window.

The portion of the display data that is scrolled out of the window is deleted. The character to be displayed must be specified in character generator code. The cursor is held in the original position on the screen.

<Command sequence>
1st byte: ESC
2nd byte: ØD2H
3rd byte: Y Y: Line number 1 ≤ Y ≤ 8
4th byte: X X: Column number 1 ≤ X ≤ 4Ø
5th byte: n n: Character code ØØH ≤ n ≤ ØFFH

- ESC D4H Locate top of screen
Positions the window at the beginning of the screen.

The cursor is held in the original position on the screen.

<Command sequence>
1st byte: ESC
2nd byte: ØD4H

- ESC D5H Locate bottom of screen
Moves the window to the end of the screen.

The cursor is held in the original position on the screen.

<Command sequence>
1st byte: ESC
2nd byte: ØD5H

- ESC D6H Select cursor kind
Selects the type of the cursor.

<Command sequence>
1st byte: ESC
2nd byte: ØD6H
3rd byte: n

n: Type of the cursor
= ØØH: Block and blink
= Ø1H: Block and nonblink
= Ø2H: Underline and blink
= Ø3H: Underline and nonblink

The default is block and blink.

- ESC D7H Find cursor
Moves the window over the screen so that the cursor line will appear in the window.

This function does nothing when the cursor is already in the current window. It moves the window in the same way as Set cursor position (ESC '=').

<Command sequence>
1st byte: ESC
2nd byte: ØD7H

- ESC EØH Set download character
Defines user-defined characters with the codes ØE0H to ØFFH.

<Command sequence>
1st byte: ESC
2nd byte: ØEØH
3rd byte: n
4th byte: P(1)
5th byte: P(2)
6th byte: P(3)
7th byte: P(4)
8th byte: P(5)
9th byte: P(6)
10the byte: P(7)
11th byte: P(8)

n: Character code
ØEØH ≤ n ≤ ØFFH
P(1) - P(8): Character pattern

	7	6	5	4	3	2	1	0
4			0	0	1	0	0	0
5			0	1	0	1	0	0
6			1	0	0	0	1	0
7			1	0	0	0	1	0
8			1	1	1	1	1	0
9			1	0	0	0	1	0
10			1	0	0	0	1	0
11			0	0	0	0	0	0

6 x 8 dot pattern (pattern for 'A')

- ESC FØH Keyboard repeat ON/OFF
Controls the keyboard repeat function.

<Command sequence>
1st byte: ESC n: Repeat switch status
2nd byte: ØFØH = ØØH: Repeat ON
3rd byte: n = Ø1H: Repeat OFF

The default setting is Repeat ON for the standard keyboard and Repeat OFF for the item keyboard.

- ESC F1H Set keyboard repeat starting time
Specifies the keyboard repeat starting time

<Command sequence>
1st byte: ESC S: Repeat starting time (in 1/64 sec increments)
2nd byte: ØF1H Ø ≤ S ≤ 127
3rd byte: S

The default value is approximately 656 msec (S = 42).

- ESC F2H Set keyboard repeat interval time
Specifies the keyboard auto repeat interval.

<Command sequence>
1st byte: ESC S: Repeat interval (in 1/256 sec increments)
2nd byte: ØF2H Ø ≤ S ≤ 127
3rd byte: S

The default value is approximately 7Ø msec (S = 18).

- ESC F3H Set arrow key code
Defines the arrow key codes.

When an arrow key is pressed after this function is executed, the corresponding code is returned. See Section 3.5, "Keyboard" for detailed information.

<Command sequence>
1st byte: ESC
2nd byte: ØF3H n1 - n4: Arrow key code
3rd byte: n1 ØØH ≤ n1 - n4 ≤ ØFFH
4th byte: n2 n1: Code of →
5th byte: n3 n2: Code of ←
6th byte: n4 n3: Code of ↑
  n4: Code of ↓
The default setting is n1 = 1CH, n2 = 1DH, n3 = 1EH, and n4 = 1FH.

For the item keyboard, the arrow key code cannot be changed.

- ESC F4H Set scroll key code
Defines codes for the SHIFT + arrow key combinations.

When a SHIFT/arrow key sequence is entered after this function is executed, the corresponding code is returned. See Section 3.5, "Keyboard" for detailed information.

<Command sequence>
1st byte: ESC
2nd byte: ØF4H
3rd byte: n1
4th byte: n2
5th byte: n3
6th byte: n4

n1 - n4: SHIFT/arrow key code
ØØH ≤ n1 - n4 ≤ ØFFH
n1: Code of SHIFT/→
n2: Code of SHIFT/←
n3: Code of SHIFT/↑
n4: Code of SHIFT/↓

The default settings are n1 = ØF8H, n2 = ØF9H, n3 = ØFAH, and n4 = ØFBH.

- ESC F5H Set CTRL key code

Defines codes for the CTRL + arrow key combinations.

When a CTRL key sequence is entered after this function is executed, the corresponding code is returned. See Section 3.5, "Keyboard" for details.

<Command sequence>
1st byte: ESC
2nd byte: ØF5H
3rd byte: n1
4th byte: n2
5th byte: n3
6th byte: n4

n1 - n4: CTRL/arrow key code
ØØH ≤ n1 - n4 ≤ ØFFH
n1: Code of CTRL/→
n2: Code of CTRL/←
n3: Code of CTRL/↑
n4: Code of CTRL/↓

The default settings are n1 = ØFCH, n2 = ØFDH, n3 = ØFEH, and n4 = ØFFH.

- ESC F6H Clear key buffer

Clears the keyboard buffer to delete the type ahead key codes. The data in the 7508 buffer remains unaffected.

<Command sequence>
1st byte: ESC
2nd byte: ØF6H

- ESC F7H Set key shift

Defines the key shift modes.

<Command sequence>
1st byte: ESC
2nd byte: ØF7H
3rd byte: n

n: Shift mode
ØØH ≤ n ≤ ØFFH

The keyboard shift modes are defined on a bit basis. A 1 in a bit turns on the shift mode for the corresponding shift key.

Bit	7 (MSB)	CTRL
	6	GRPH
	5	---
	4	NUM
	3	---
	2	CAPS
	1	SHIFT (L)
	Ø (LSB)	SHIFT (R)

For the item keyboard, nothing takes place when a shift mode is specified.

3.6.6 Character Generator

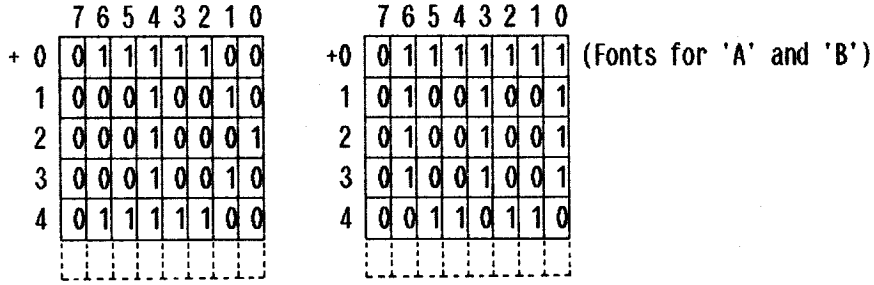
The PINE has a character generator in OS ROM.

```
ØØH - 7FH     5 x 8 dot character fonts
8ØH - 9FH     6 x 8 dot character fonts
ØAØH - ØDFH   5 x 8 dot character fonts
ØEØH - ØFFH   6 x 8 dot character fonts
```

ØEØH through ØFFH are used by the user to define user-defined characters. The fonts for the user-defined characters are stored in the user-defined character definition area. Initially, the fonts for ØEØH and ØElH are defined by the system. Other codes are initialized to spaces.

3.6.6.1 Font format

A font is made up of 5 or 6 bytes. Fonts are stored sideways as shown below:



Since characters are displayed on the LCD in 6 x 8 dot font, a ØØH code is appended to the end of each 5-byte font data to form a 6 x 8 dot font.

3.6.6.2 Character sets

The PINE supports the character sets for the languages of the following countries.

USA ASCII
France
Germany
England
Denmark
Sweden
Italy
Spain
Norway

The user can change the character set using the ESC sequence "Set character set table." The initial value is set by DIP switches. See Section 7.2, "DIP Switches" for further information.

3.6.6.3 Character generator codes arranged by country

The BIOS CONOUT routine converts character codes into character generator codes of the specified country and displays it on the LCD. The routine also places data in the user or system screen in the form of character generator codes.

Table 3.6.11 shows the relationships between the character codes and the character generator codes arranged by country.

Code	U.S.A.	France	Germany	England	Denmark	Sweden	Italy	Spain	Norway
23H	23H (#)	23H (#)	23H (#)	0FH (£)	23H (#)	23H (#)	23H (#)	18H (Pt)	23H (#)
24H	24H ($)	24H ($)	24H ($)	24H ($)	24H ($)	18H (¤)	24H ($)	24H ($)	18H (¤)
40H	40H (@)	00H (à)	03H (§)	40H (@)	40H (@)	16H (É)	40H (@)	40H (@)	16H (É)
5BH	5BH ([)	01H (°)	08H (Ä)	5BH ([)	10H (Æ)	08H (Ä)	01H (°)	1CH (¡)	10H (Æ)
5CH	5CH (\)	02H (Ç)	09H (Ö)	5CH (\)	11H (Ø)	09H (Ö)	5CH (\)	1DH (Ñ)	11H (Ø)
5DH	5DH (])	03H (§)	0AH (Ü)	5DH (])	12H (Å)	12H (Å)	04H (é)	1EH (¿)	12H (Å)
5EH	5EH (^)	5EH (^)	5EH (^)	5EH (^)	5EH (Ü)	0AH (Ü)	5EH (^)	5EH (^)	0AH (Ü)
60H	60H (`)	60H (`)	60H (`)	60H (`)	60H (`)	04H (é)	05H (ù)	60H (`)	04H (é)
7BH	7BH ({)	04H (é)	0BH (ä)	7BH ({)	13H (æ)	0BH (ä)	00H (à)	07H (¨)	13H (æ)
7CH	7CH (¦)	05H (ù)	0CH (ö)	7CH (¦)	14H (ø)	0CH (ö)	19H (ò)	1FH (ñ)	14H (ø)
7DH	7DH (})	06H (è)	0DH (ü)	7DH (})	15H (å)	15H (å)	06H (è)	7DH (})	15H (å)
7EH	7EH (~)	07H (¨)	0EH (ß)	7EH (~)	7EH (~)	0DH (ü)	1AH (ì)	7EH (~)	0DH (ü)

Table 3.6.11 Relationships between ASCII Codes and Character Generator Codes

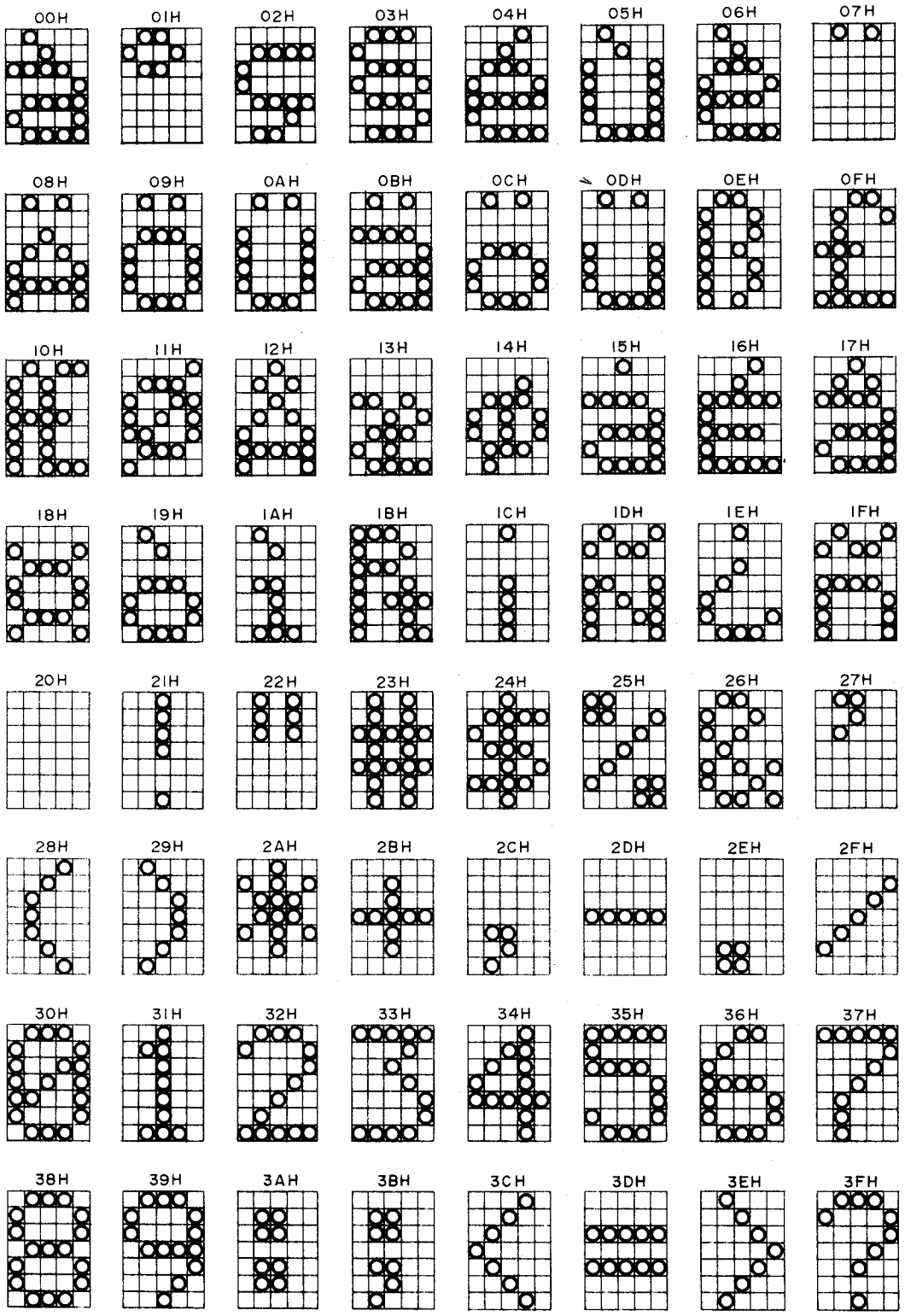
3.6.6.4 Character generator code charts

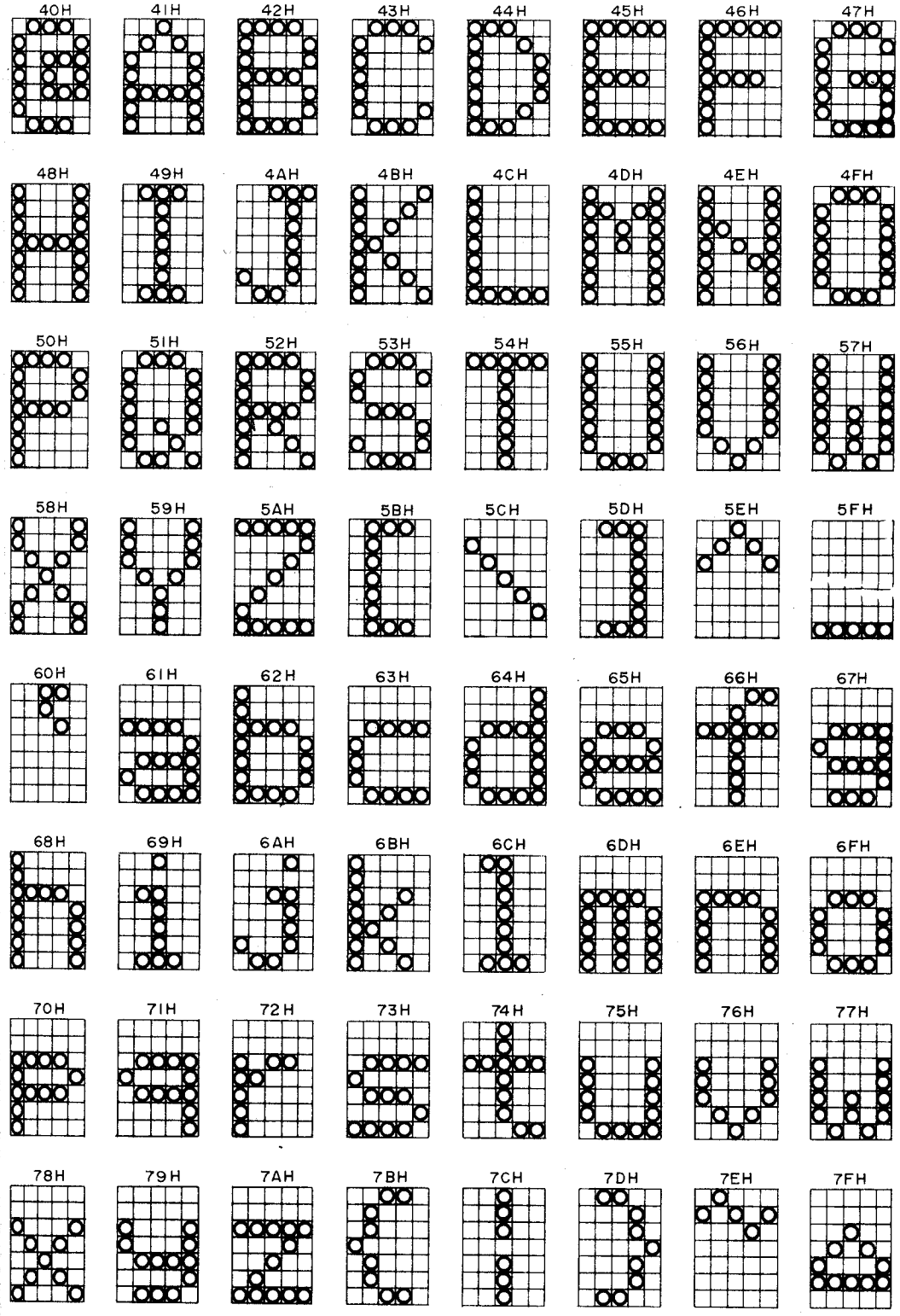
The pages that follow list charts of the character generator data (fonts).

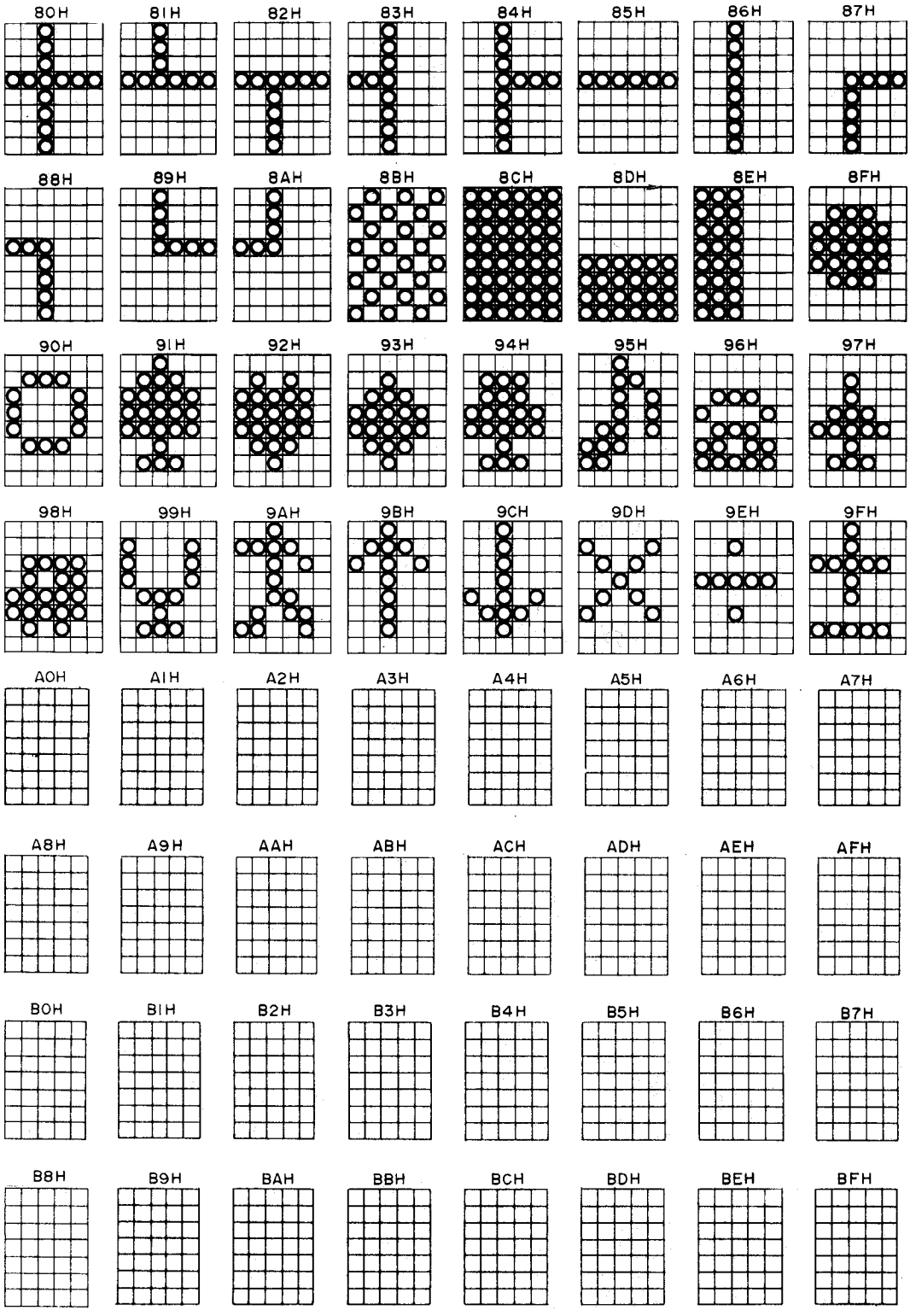
Font data can be read using the OS utility XFONTGET. For more information, see Section 4.2, "Jump Tables."

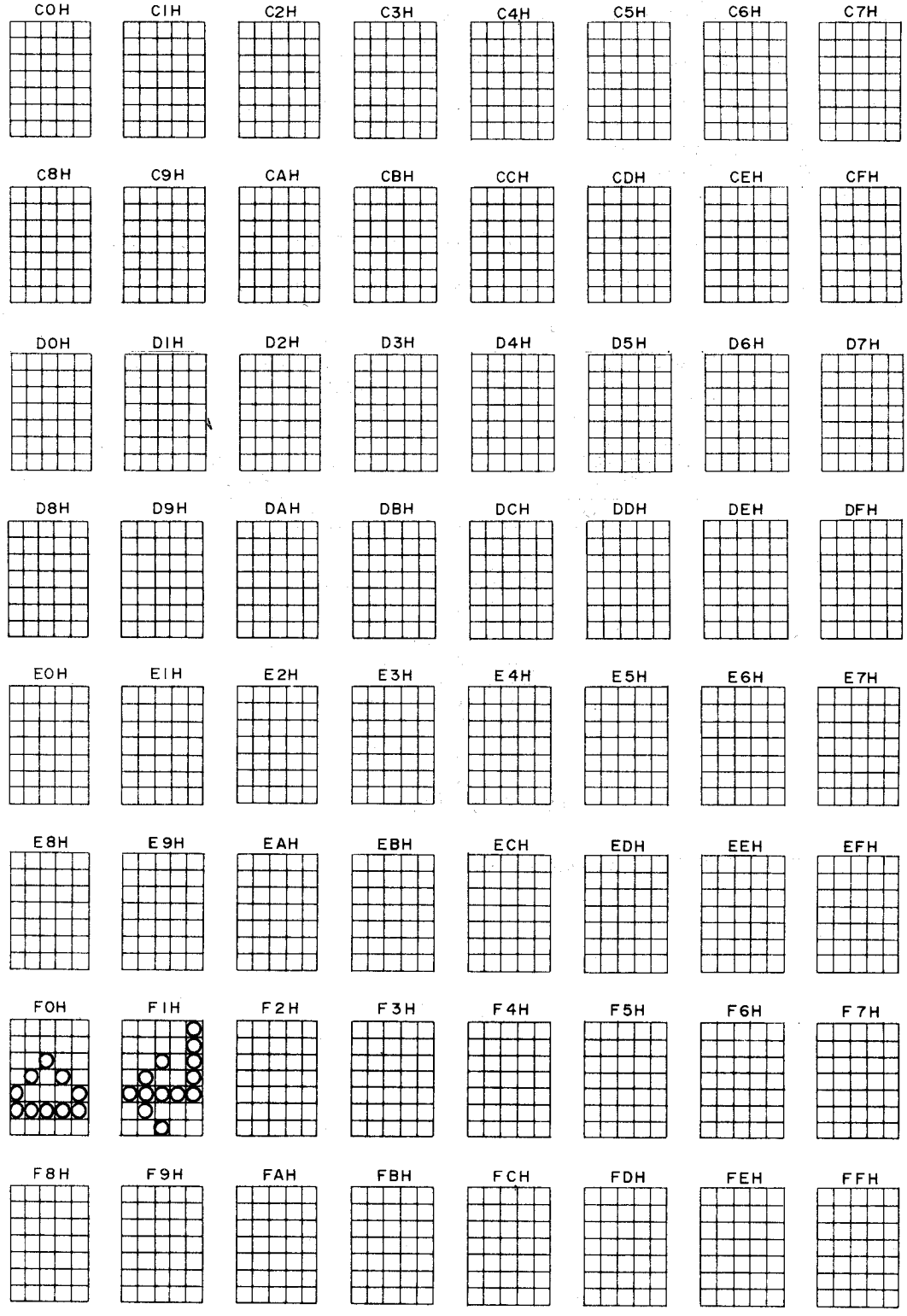
Character generator code chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	à	Æ	sp	0	@	P	`	p	┼	○					△	
1	°	ø	!	1	A	Q	a	q	┴	♠					↵	
2	ç	Å	"	2	B	R	b	r	┬	♥						
3	§	æ	#	3	C	S	c	s	┤	♦						
4	é	ø	$	4	D	T	d	t	├	♣						
5	ù	å	%	5	E	U	e	u	─	♪						
6	è	É	&	6	F	V	f	v	│	☎						
7	¨	â	'	7	G	W	g	w	┌	▲						
8	Ä	☒	(	8	H	X	h	x	┐	🚚						
9	Ö	ò	)	9	I	Y	i	y	└	Ψ						
A	Ü	ì	*	:	J	Z	j	z	┘	🚶						
B	ä	Pt	+	;	K	[	k	{	▒	↑						
C	ö	ì	,	<	L	\	l	:	■	↓						
D	ü	Ñ	−	=	M	]	m	}	▬	×						
E	ß	¿	.	>	N	^	n	~	▙	÷						
F	£	ñ	/	?	O	_	o	△	●	±						









3.6.7 Miscellaneous Considerations

3.6.7.1 Initializing screen-related parameters

The screen-related parameters include the following:

(1) User screen size
(2) Window scroll mode
(3) Cursor display on/off
(4) Cursor type

These parameters are initialized at reset and warm boot times.

The timing of parameter initialization is illustrated in Figure 3.6.12.

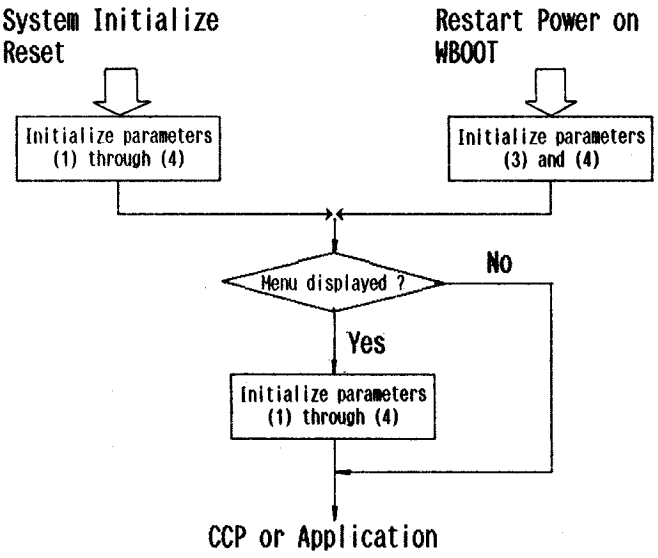


Fig. 3.6.12 Parameter Initialization Timing Diagram

The screen-related parameters are reserved in the system area in such a format that they can be directly set up using the CONOUT BIOS function.

CONSCRN1 (ØEFF3H) 8 bytes
- Screen parameter 1
This area is used to specify the user screen size and the window scroll mode.

1st byte: ESC (1BH)
2nd byte: ØDØH
3rd byte: Number of screen lines
4th byte: Number of screen columns
5th byte: ESC (1BH)
6th byte: 95H
7th byte: Scroll mode
8th byte: ØØH

The first four bytes specify the screen size. The initial value is 8Ø columns by 25 lines.

The fifth to seventh bytes specify the window scroll mode. The initial value is horizontally nontracking mode.

See the descriptions of the ESC sequences for details.

CONSCRN2 (ØEFFBH) 6 bytes
- Screen parameter 2
This area is used to specify cursor display on/off and the cursor type.

1st byte: ESC (1BH)
2nd byte: Cursor display on/off
3rd byte: ESC (1BH)
4th byte: ØD6H
5th byte: Cursor type
6th byte: ØFFH

The first two bytes specify whether the cursor is to be turned on or off. The initial value is cursor display on (33H, '3').

The third to fifth bytes specify the cursor type. The initial value is block and blink.

ØFFH identifies the end of the parameters.

3.6.7.2 Cursor display during execution of BIOS CONOUT

When BIOS CONOUT receives a command such as displaying a character, moving the screen, or changing the cursor mode, it turns off the cursor before processing the command and turns it on after completing the execution of the command.

Consequently, the cursor would flicker if such a command is controlled by CONOUT. This can be avoided in two ways:
- Turns off the cursor while CONOUT is processing.
- Obtain the address of the subroutine in CONOUT that actually processes the command and call that subroutine directly.

(1) Turning off the cursor
Turn off the cursor using ESC + '2' before displaying a character or moving the screen.

(2) Obtaining the actual command handling subroutine address
Execute the pertinent command using CONOUT and look into the function address area LFKADDR (ØF2A8H) to get the address of the actual command processing subroutine.

Once the address is obtained, cursor turning on/off processing can be bypassed by directly calling the address in OS ROM via BIOS CALLX.

Note that the command processing subroutine does not control the cursor. That is, the cursor does not move forward as characters are displayed on the screen. (Use this technique when displaying characters in one part of the screen while having the cursor blink in another part of the screen.)

In practice, direct execution is carried out by calling OS ROM procedures as explained below.
(a) When an ESC sequence has parameters (when including ESC there are more than 3 bytes of data), LESCPRM (F2ACH) is called after setting the parameters. Furthermore, at this time the last parameter is put into the C register.
(b) When an ESC sequence does not have parameters (for example ESC + 'P'), it is sufficient to simply make a call.
(c) To display a character (2ØH - FFH), the call is made after putting the parameter into the C register.
(d) In the case of a control codes (ØØH - 1FH), it is sufficient to simply make a call.

3.6.7.3 Multiple Use of Virtual Screens
As mentioned earlier, in principle it is possible to use only two types of screen, the user screen and system screen, however for simple applications it is also possible to use a number of

additional screens.
Here we will explain methods for multiple use of virtual screens, and some points to note.

(1) Concepts
The operating system maps a window in the virtual screen onto the LCD. Accordingly, it is possible to use different virtual screens within an application program, by providing buffers for use by the virtual screens, and specifying the buffers with the system area display parameters. In Figure 3.6.12 mapping is executed by the operating system, and switching is controlled by the application program.

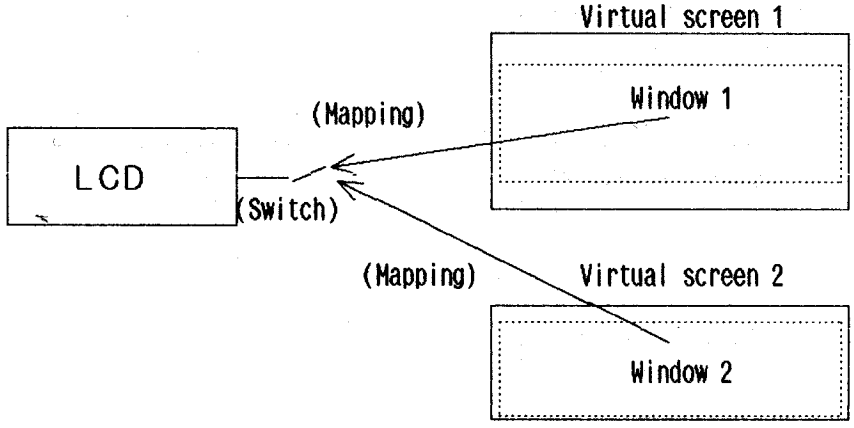


Fig. 3.6.12 Multiple Use of Virtual Screens

(2) Procedure

For multiple use of virtual screens, the following two methods are available.

(a) Multiple use of virtual screens only.
(b) Multiple use of virtual screens and VRAM.

The main difference between (a) and (b) above, is whether or not graphic data and inverse displays etc. can be preserved when the screen is switched.

(a) Multiple use of virtual screens only.
This procedure is used when only character data is to be used.
The procedure is illustrated in Figure 3.6.13.

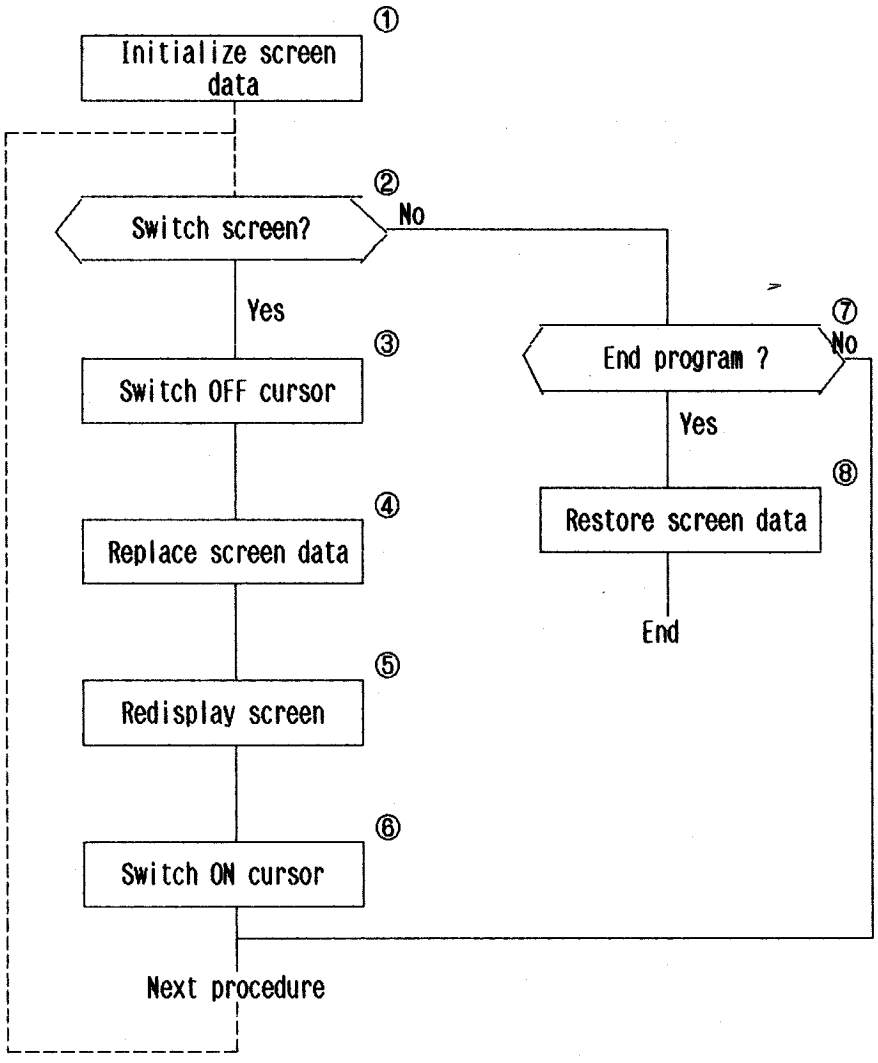


Fig. 3.6.13 Procedure for Switching Screen

(b) Multiple use of virtual screens and VRAM.
This procedure is used when character data and graphic data are used mixed together.
The procedure is illustrated in Figure 3.6.14

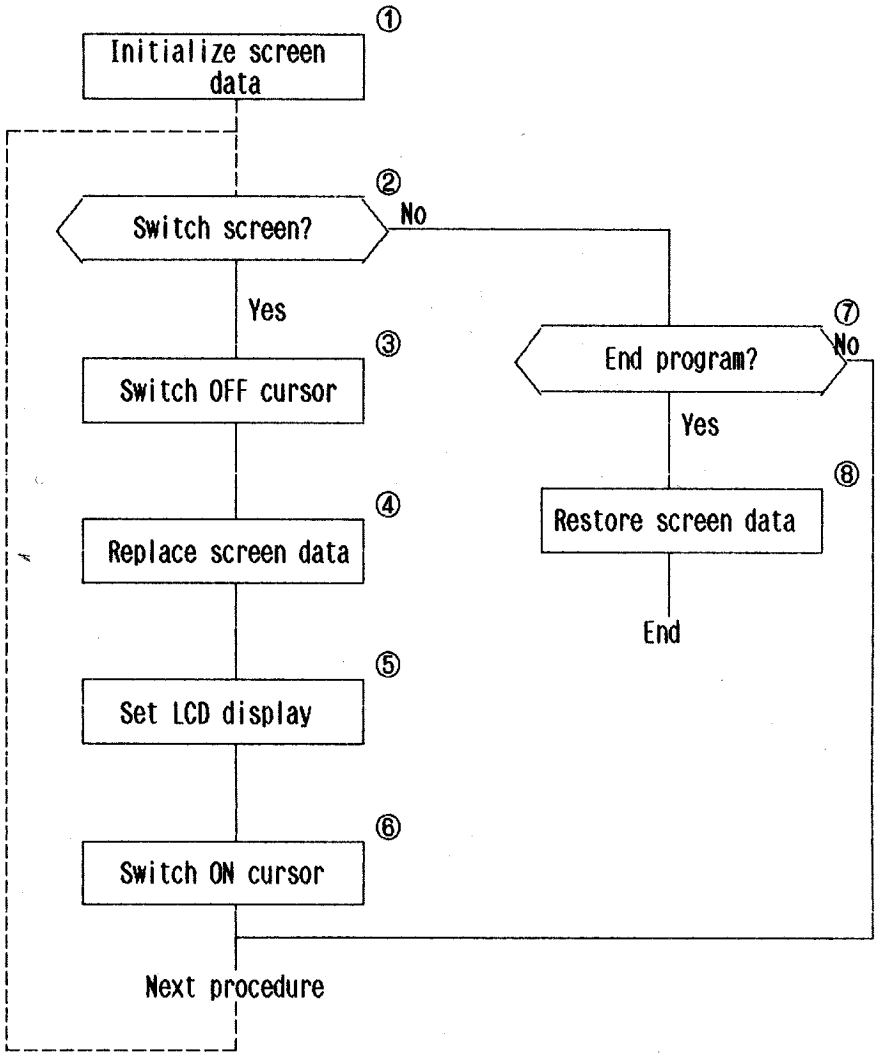


Fig. 3.6.14 Procedure for Switching VRAM

Step	Process	Content
1	Initialize screen data	* When use is to be made of VRAM or virtual screen areas other than areas previously prepared by the system, the areas to be used are initialized by this process beforehand. * The data area for replacement of system area display parameters is initialized beforehand. The display parameters are 39 bytes of data from LSCADDR(F290H) to LESCPRM(F2B6H).
2	Switch screen ?	* Tests if condition requiring screen mode to be changed is satisfied or not.
3	Switch OFF cursor	* A display screen operation is to be carried out, so this process switches OFF the cursor beforehand.
4	Replace screen data	* Replaces system area display parameters(39 bytes from F290H) with previously prepared data.
5	Redisplay screen	* In the case of virtual screen replacement only, a call is made to XREDSP (redisplay) in the OS jump table.. * In the case of VRAM replacemt, the VRAM address and offset value in the Y direction are output through a port.
6	Switch ON cursor	* Switches ON the cursor when the display screen operation has been completed. * The cursor is switched ON using the BIOS CONOUT.

Step	Process	Content
7	End program ?	* Tests if application program should finish or not.
8	Restore screen data	* When the application program is terminated, LSCADDR(F290H) is restored to D000H and LSRVRAM (F294H) is restored to E000H. * Termination of the application program with the virtual screen and VRAM still in the state set by the application program must be avoided.

(3) Points to Note
When processing involving switching of screens is carried out, attention must be given to the following points.

(a) In cases in which the position of the virtual screen or position of the VRAM is changed, care must be taken not to input Restart/Power off.
This is to avoid misoperation when the application program which will next be executed modifies screen size etc.
During switching of screens, it is necessary for either Continue mode to have been previously set, or for Power off to have been forbidden at the application level.

(b) When the position of the virtual screen or position of the VRAM is changed, RAM locations must be reserved from 8000H upward.
Also, the leading VRAM address must be specified on an 800H (2k) byte boundary.

3.6.8 Screen-related Work Areas

DSPFLAG (ØEFB6H) 1 byte
- LCD display flag
  = ØØH: LCD display off
  = 8ØH: LCD display on

BLNKSTAT (ØEFB7H) 1 byte
- Blink state flag
  = ØØH: Blink disabled
  = 8ØH: Blink enabled

This flag is examined by the blink processing routine.

BLNKCNT (ØEFB8H) 1 byte
- Blink counter

This counter is incremented by one when an overflow interrupt is generated. The cursor is displayed in reverse video when BLNKTIME is reached.

BLNKRVRS (ØEFB9H) 1 byte
- Indicates the cursor status during blinking.
  = ØØH: Cursor display on
  = ØFFH: Cursor display off

BLNKTIME (ØEFBAH) 1 byte
- Specifies the blink interval time.

The initial value is Ø4H. With this setting, cursor display is switched between normal and reverse video every 5ØØ ms or so.

LSCADDR (ØF29ØH) 2 bytes
- Starting address of the buffer for the currently displayed screen

The screen buffer size is stored in LSCSIZE.

8ØØØH ≤ LSCADDR ≤ ØFFFFH

LSCSIZE (ØF292H) 2 bytes
- Screen buffer size

Ø ≤ LSCSIZE ≤ 2Ø48

LSCSIZE is equal to LSCSIZEX * LSCSIZEY.

LSCRVRAM (ØF294H) 2 bytes
- Starting address of the currently displayed VRAM

8ØØØH ≤ LSCRVRAM ≤ ØFFFFH

2K bytes of memory must be allocated for VRAM.

LCURSOR (ØF296H) 2 bytes
- Cursor state flag

Bit 7: Cursor mode
  Ø: Display on 1: Display off
Bits 6 - 2: Don't care.
Bit 1: Cursor type
  Ø: Block 1: Underline
Bit Ø: Cursor blink
  Ø: On 1: Off

LCRVRSW (ØF297H) 1 byte
- Reverse display flag
     Bits 7 - 1: Don't care.
     Bit Ø: Reverse state
            Ø: Reverse display mode off
            1: Reverse display mode on

LSCCPOSX (ØF298H) 1 byte
- X-coordinate of the cursor on the screen
     1 ≤ X ≤ Number of columns on the screen

LSCCPOSY (ØF299H) 1 byte
- Y-coordinate of the cursor on the screen
     1 ≤ Y ≤ Number of lines on the screen

LSCSIZEX (ØF29AH) 1 byte
- Screen width
     LSCSIZEX = 4Ø or 8Ø

LSCSIZEY (ØF29BH) 1 byte
- Screen length
     8 ≤ LSCSIZEY ≤ 5Ø

LWDXMIN (ØF29CH) 1 byte
- X-coordinate of the upper left corner of the window
This variable indicates where the current window is located on the screen.

LWDYMIN (ØF29DH) 1 byte
- Y-coordinate of the upper left corner of the window
This variable indicates where the current window is located on the screen.

LWDCPOSX (ØF29EH) 1 byte
- X-coordinate of the cursor on the screen
     = ØFDH: Inside the right margin
     = ØFEH: Inside the left margin
     = ØFFH: Outside the window
     Others: Inside the window (Ø ≤ X ≤ 39)

LWDCPOSY (ØF29EH) 1 byte
- Y-coordinate of the cursor on the screen
     = ØFFH: Outside the window
     Others: Inside the window (Ø ≤ Y ≤ 7)

LVRAMYOF (ØF2AØH) 1 byte
- Y-direction offset of VRAM
     Ø ≤ LVRAMYOF ≤ 63

LWDTYPE (ØF2A1H) 1 byte
- Window type
     = ØFEH: Right screen (scroll step = 4Ø)
     = ØFFH: Left screen (scroll step = 4Ø)
     = ØØH: Screen width is 4Ø
     = Ø1H: Left screen (scroll step = 2Ø)
     = Ø2H: Center screen (scroll step = 2Ø)
     = Ø3H: Right screen (scroll step = 2Ø)

LSECRETW (ØF2A2H) 1 byte
- Secret mode on/off state
     = ØØH: Secret mode off
     = Ø1H: Secret mode on

LSCROLMD (ØF2A3H) 1 byte
- Scroll mode
     = ØØH: Tracking mode
     = Ø1H: Nontracking mode
     = Ø2H: Horizontally nontracking mode

LSCROLX (ØF2A4H) 1 byte
- Horizontal scroll step
     LSCROLX = 2Ø or 4Ø

LSCROLY (ØF2A5H) 1 byte
- Vertical scroll step
     1 ≤ LSCROLY ≤ 8

LCRWAIT (ØF2A6H) 1 byte
- Carriage return check flag
     = ØØH: Carriage return check mode off
     = Ø1H: Carriage return check mode on
The carriage return check mode is turned on when the cursor moves from the end of a line to the beginning of the next line after a character is displayed.

LFKSTAT (ØF2A7H) 1 byte
- CONOUT function status
     Bit 7: Cursor off flag
          Ø: Cursor off not required 1: Cursor off required
     Bit 6: Indicates whether the CR check flag is to be cleared.
          Ø: Not cleared 1: Cleared
     Bits 5, 4: Don't care
     Bits 3 - 1: Number of function parameters

LFKADDR (ØF2A8H) 2 bytes
- Address of a CONOUT function processing routine
When a CONOUT function is specified, this area is loaded with the address of its processing routine. Control is transferred to this address after all parameters have been received.

LESCFLG (ØF2AAH) 1 byte
- ESC sequence receive flag
     = ØØH: ESC code not received
     = Ø1H: ESC code received
     = Ø2H: ESC code and function received

LESCCNT (ØF2ABH) 1 byte
- Number of ESC sequence parameters received.

LESCPRM (ØF2ACH) 11 bytes
- The area for storing ESC sequence receive parameters.

LWORKBF (ØF2B7H) 39 bytes
- The area for holding the screen parameters saved when the screen mode is changed.
When the screen mode is changed, the 39 bytes of data from LSCADDR to LESCPRM are exchanged with the 39 bytes of data in this area.

LSCMODE (ØF2DEH) 1 byte
- Screen mode
     = ØØH: System screen mode
     = Ø1H: User screen mode

LLMARGIN (ØF2DFH) 1 byte
- Left margin size
    Ø ≤ LLMARGIN ≤ 1Ø

LRMARGIN (ØF2EØH) 1 byte
- Right margin size
    Ø ≤ LRMARGIN ≤ 1Ø

RLCGENX (ØF35CH) 2 bytes
- Reserved for system

RLCGENN (ØF35EH) 2 bytes
- Reserved for system

RLCGENG (ØF36ØH) 2 bytes
- Reserved for system

RLCGENK (ØF362H) 2 bytes
- Reserved for system

LCHRFONT (ØF7C6H) 8 bytes
- Character font data conversion area

	7 6 5 4 3 2	1 0
+0		0 0
1		0 0
2	for write in VRAM	0 0
3	6 x 8 dot	0 0
4	font data	0 0
5		0 0
6		0 0
7		0 0

RWUVSCTOP (ØD ØØØH) 8ØØH bytes
- User screen buffer
Loaded with character codes.

RWVRAM2TOP (ØD8ØØH) 8ØØH bytes
- VRAM2 area
VRAM for the system screen.

RWVRAM1TOP (ØE ØØØH) 8ØØH bytes
- VRAM 1 area
VRAM for the user screen.

RWSYSCTOP (ØE8ØØH) 24Ø bytes
- System screen buffer
Loaded with character codes.

RWEXCHRTOP (ØE94ØH) 192 bytes
- User-defined character area
This area can contain up to 32 user-defined characters in 6 x 8 dot font.

```
                        ;       ************************************************
                        ;               CHANGE SCREEN AREA PROGRAM
                        ;       ************************************************
                        ;
                        ;       NOTE :
                        ;               This sample program is changing screen
                        ;               area.
                        ;               So, you can use screen more than only one.
                        ;
                        ;       <> assemble condition <>
                        ;
                        ;       .Z80
                        ;
                        ;       <> loading address <>
                        ;
                        ;       .PHASE  100H
                        ;
                        ;       <> constant values <>
                        ;
                        ;       BIOS entry address.
                        ;
EB03                    WBOOT           EQU     0EB03H
EB09                    CONIN           EQU     WBOOT   +06H
EB0C                    CONOUT          EQU     WBOOT   +09H
EB69                    CALLX           EQU     WBOOT   +66H
                        ;
                        ;       System area
                        ;
F290                    LSCADDR         EQU     0F290H  ; Screen buffer top addr.
F294                    LSCRVRAM        EQU     0F294H  ; VRAM area top addr.
F2A0                    LVRAMYOF        EQU     0F2A0H  ; VRAM Y-offset value.
EF94                    TOPRAM          EQU     0EF94H  ; User BIOS area top addr.
                        ;
                        ;       OS ROM jump table
                        ;
0036                    XREDSP          EQU     00036H  ; Re-display window
                        ;
                        ;
                        ;
001B                    ESC             EQU     1BH     ; ESC code
0003                    STOP            EQU     03H     ; STOP code
0000                    HELP            EQU     00H     ; HELP code
                        ;
C000                    VADDR           EQU     0C000H  ; New VRAM address.
                        ;
                        ;       IO register address
                        ;
0009                    ZYOFF           EQU     09H     ;
0008                    ZVADR           EQU     08H     ;
                        ;
                        ;
                        ;
                        ;       ************************************************
                        ;               MAIN PROGRAM
                        ;       ************************************************
                        ;
                        ;       NOTE ;
                        ;               This routine sets new screen data.
                        ;
                        ;       CAUTION :
                        ;               This program uses User BIOS area for VRAM.
                        ;               But this routine doesn't check that
                        ;               other program already User BIOS area.
                        ;
                        ;               If you stop program, you must restore
                        ;               old screen status.
                        ;               If you forget this, other system area
                        ;               will be destroyed.
                        ;
0100                    START:
0100    31 1000                 LD      SP,1000H        ; Set stack pointer.
                        ;
                        ;       Set initial data
                        ;
0103    2A EF94                 LD      HL,(TOPRAM)     ; User BIOS area check.
0106    11 C001                 LD      DE,0C001H       ; User BIOS area top addr <= C000H?
0109    AF                      XOR     A               ;
010A    ED 52                   SBC     HL,DE           ;
010C    D2 EB03                 JP      NC,WBOOT        ; No. then WBOOT
                        ;
010F    01 3228                 LD      BC,50*256+40    ; Set default screen size
0112    CD 01AF                 CALL    SETSCR          ;
                        ;
0115    21 F290                 LD      HL,LSCADDR      ; Save current screen data.
0118    11 0224                 LD      DE,SCRSAVE      ;
011B    01 0027                 LD      BC,27H          ;
011E    ED B0                   LDIR                    ;
                        ;
0120    01 2228                 LD      BC,34*256+40    ; Set new screen size.
0123    CD 01AF                 CALL    SETSCR          ;
                        ;
0126    21 C000                 LD      HL,VADDR        ; Clear new VRAM area
0129    36 00                   LD      (HL),0          ;
012B    54                      LD      D,H             ;
012C    5D                      LD      E,L             ;
012D    13                      INC     DE              ;
012E    01 07FF                 LD      BC,2048-1       ;
0131    ED B0                   LDIR                    ;
                        ;
                        ;       Main loop
                        ;
0133                    LOOP:
0133    CD EB09                 CALL    CONIN           ; Get inputed key code.
0136    FE 03                   CP      STOP            ; If STOP,
0138    CA 01C5                 JP      Z,PEND          ;  then end
013B    FE 1B                   CP      ESC             ; If ESC,
```

```
013D  28 0A                     JR      Z,CHNGSCR       ;  then change screen.
013F  FE 00                     CP      HELP            ; If HELP,
0141  28 2E                     JR      Z,CHNGVRAM      ;  then change VRAM.
0143  4F                        LD      C,A             ; Console out inpued data.
0144  CD EB0C                   CALL    CONOUT          ;
0147  18 EA                     JR      LOOP            ; Loop
0149                    CHNGSCR:
0149  0E 1B                     LD      C,ESC           ; Erase cursor
014B  CD EB0C                   CALL    CONOUT          ;
014E  0E 32                     LD      C,'2'           ;
0150  CD EB0C                   CALL    CONOUT          ;
                        ;
0153  21 01D6                   LD      HL,WORKBF1      ; Source addr.
0156  CD 0190                   CALL    WKCHNG          ; Change screen
                        ;
0159  3E FF                     LD      A,0FFH          ; Set destination bank
015B  32 F52E                   LD      (0F52EH),A      ;
015E  DD 21 0036                LD      IX,XREDSP       ; Re-display window
0162  CD EB69                   CALL    CALLX           ;
                        ;
0165  0E 1B                     LD      C,ESC           ; Cursor on.
0167  CD EB0C                   CALL    CONOUT          ;
016A  0E 33                     LD      C,'3'           ;
016C  CD EB0C                   CALL    CONOUT          ;
016F  18 C2                     JR      LOOP            ;
                        ;
                        ;
0171                    CHNGVRAM:
0171  0E 1B                     LD      C,ESC           ;Erase cursor
0173  CD EB0C                   CALL    CONOUT
0176  0E 32                     LD      C,'2'
0178  CD EB0C                   CALL    CONOUT
                        ;
017B  21 01FD                   LD      HL,WORKBF2      ;Source addr.
017E  CD 0190                   CALL    WKCHNG          ;Change screen
                        ;
0181  CD 019F                   CALL    SETVRAM         ;VRAM data set.
                        ;
0184  0E 1B                     LD      C,ESC           ;Cursor on.
0186  CD EB0C                   CALL    CONOUT
0189  0E 33                     LD      C,'3'
018B  CD EB0C                   CALL    CONOUT
018E  18 A3                     JR      LOOP
                        ;
                        ;
0190                    WKCHNG:
0190  11 F290                   LD      DE,LSCADDR      ;Destination addr.
0193  06 27                     LD      B,27H           ;Exchange byte no.
                        ;
0195                    WKC10:
0195  4E                        LD      C,(HL)          ;Exchange data
0196  1A                        LD      A,(DE)          ; (DE) <--> (HL)
0197  77                        LD      (HL),A
0198  79                        LD      A,C
0199  12                        LD      (DE),A
                        ;
019A  23                        INC     HL              ;Pointer update
019B  13                        INC     DE
019C  10 F7                     DJNZ    WKC10           ;Loop until b=0
                        ;
019E  C9                        RET
                        ;
                        ;
019F                    SETVRAM:
019F  AF                        XOR     A               ;Display off
01A0  D3 09                     OUT     (ZYOFF),A
01A2  3A F295                   LD      A,(LSCRVRAM+1)  ;Set VRAM addr.
01A5  D3 08                     OUT     (ZVADR),A
01A7  3A F2A0                   LD      A,(LVRAMYOF)    ;Set Y-offset
01AA  F6 80                     OR      10000000B
01AC  D3 09                     OUT     (ZYOFF),A
01AE  C9                        RET
                        ;       ************************************************
                        ;               CHANGE SCREEN SIZE ROUTINE
                        ;       ************************************************
                        ;
                        ;       NOTE :
                        ;               This routine is changing screen size.
                        ;
                        ;       <> entry parameter <>
                        ;               C    : New screen size for vertical
                        ;               B    : New screen size for horizontal
                        ;       <> return parameter <>
                        ;               NON
                        ;       <> preserved registers <>
                        ;               NON
                        ;
                        ;       CAUTION :
                        ;
01AF                    SETSCR:
01AF  C5                        PUSH    BC              ; Change screen size.
01B0  0E 1B                     LD      C,ESC           ;
01B2  CD EB0C                   CALL    CONOUT          ;
01B5  0E D0                     LD      C,0D0H          ;
01B7  CD EB0C                   CALL    CONOUT          ;
01BA  C1                        POP     BC              ;
01BB  C5                        PUSH    BC              ;
01BC  48                        LD      C,B             ; Size of Y
01BD  CD EB0C                   CALL    CONOUT          ;
01C0  C1                        POP     BC              ; Size of X
01C1  CD EB0C                   CALL    CONOUT          ;
01C4  C9                        RET                     ;
                        ;
01C5                    PEND:
01C5  21 0224                   LD      HL,SCRSAVE      ;Restore screen data
```

```
01C8    CD 0190                 CALL    WKCHNG
                        ;
01CB    0E 0C                   LD      C,0CH           ;Clear screen
01CD    CD E80C                 CALL    CONOUT
                        ;
01D0    CD 019F                 CALL    SETVRAM
01D3    C3 0000                 JP      0000H           ;WBOOT
                        ;
                        ;
01D6                    WORKBF1:
01D6    D550                    DW      0D000H+40*34    ;Screen addr.
01D8    0140                    DW      40*8            ;Screen size
01DA    E000                    DW      0E000H          ;VRAM top addr
01DC    00                      DB      0               ;Cursor status
01DD    00                      DB      0               ;Reverse status
01DE    0101                    DW      0101H           ;Cursor position in screen
01E0    28                      DB      40              ;Screen size X
01E1    08                      DB      8               ;Screen size Y
01E2    0101                    DW      0101H           ;Window left-upper position
01E4    0000                    DW      0000H           ;Cursor position in window
01E6    00                      DB      0               ;VRAM Y-offset
01E7    00                      DB      0               ;Window type
01E8    00                      DB      0               ;Secret mode
01E9    00                      DB      0               ;Scroll mode
01EA    0000                    DW      0               ;Scroll step
01EC    00                      DB      0               ;Carriage return wait flag
01ED    00                      DB      0               ;Function status
01EE    0000                    DW      0               ;Function addr
01F0    00                      DB      0               ;ESC flag
01F1    00                      DB      0               ;ESC count
01F2    00 00 00 00             DB      0,0,0,0         ;Parameter store area
01F6    00 00 00 00             DB      0,0,0,0
01FA    00 00 00                DB      0,0,0
                        ;
01FD                    WORKBF2:
01FD    D690                    DW      0D000H+40*42    ;Screen addr.
01FF    0140                    DW      40*8            ;Screen size
0201    C000                    DW      VADDR           ;VRAM top addr
0203    00                      DB      0               ;Cursor status
0204    00                      DB      0               ;Reverse status
0205    0101                    DW      0101H           ;Cursor position in screen
0207    28                      DB      40              ;Screen size X
0208    08                      DB      8               ;Screen size Y
0209    0101                    DW      0101H           ;Window left-upper position
020B    0000                    DW      0000H           ;Cursor position in window
020D    00                      DB      0               ;VRAM Y-offset
020E    00                      DB      0               ;Window type
020F    00                      DB      0               ;Secret mode
0210    00                      DB      0               ;Scroll mode
0211    0000                    DW      0               ;Scroll step
0213    00                      DB      0               ;Carriage return wait flag
0214    00                      DB      0               ;Function status
0215    0000                    DW      0               ;Function addr
0217    00                      DB      0               ;ESC flag
0218    00                      DB      0               ;ESC count
0219    00 00 00 00             DB      0,0,0,0         ;Parameter store area
021D    00 00 00 00             DB      0,0,0,0
0221    00 00 00                DB      0,0,0
                        ;
0224                    SCRSAVE:
0224                            DS      27H             ;Screen data save area
                                END
```

```
                        ;       ************************************************
                        ;               CONSOLE DIRECT DISPLAY SAMPLE
                        ;       ************************************************
                        ;
                        ;       NOTE :
                        ;               This sample program is using console
                        ;               out direct display.
                        ;
                        ;               This is same as TIMDAT sample program.
                        ;
                        ;       <> assemble condition <>
                        ;
                                .Z80
                        ;
                        ;       <> loading address <>
                        ;
                                .PHASE  100H
                        ;
                        ;       <> constant values <>
                        ;
EB03                    WBOOT           EQU     0EB03H  ; WBOOT entry address.
EB06                    CONST           EQU     0EB06H  ; CONST entry address.
EB09                    CONIN           EQU     0EB09H  ; CONIN entry address.
EB0C                    CONOUT          EQU     0EB0CH  ; CONOUT entry address.
EB4E                    TIMDAT          EQU     0EB4EH  ; TIMDAT entry address.
EB69                    CALLX           EQU     0EB69H  ; CALLX entry address.
                        ;
                        ;       Bank value
                        ;
00FF                    SYSBANK         EQU     0FFH
0000                    BANK0           EQU     000H
0001                    BANK1           EQU     001H
0002                    BANK2           EQU     002H
                        ;
                        ;
1000                    MAINSP          EQU     01000H  ; Stack pointer.
                        ;
                        ;       System area
                        ;
F2AC                    LESCPRM         EQU     0F2ACH  ; ESC sequence parameter area.
F2A8                    LFKADDR         EQU     0F2A8H  ; CONOUT execute addr.
F52E                    DISBNK          EQU     0F52EH  ; Bank data.
                        ;
                        ;
000D                    CR              EQU     0DH
000A                    LF              EQU     0AH
0012                    CLS             EQU     12H
001B                    ESC             EQU     1BH
                        ;
                        ;       ************************************************
                        ;               MAIN PROGRAM
                        ;       ************************************************
                        ;
                        ;       NOTE :
                        ;               Display time until press BREAK key.
                        ;               And key input any key.
                        ;
0100                    START:
0100    31 1000                 LD      SP,MAINSP       ; Set stack pointer.
                        ;
0103    CD 0136                 CALL    GETFKAD         ; Get direct display function addr.
                        ;
0106    21 01C6                 LD      HL,MSG01        ; Date & time message.
0109    CD 0143                 CALL    DSPMSG          ; Dispay message.
                        ;
010C                    LOOP:
010C    76                      HALT
                        ;
010D    CD EB06                 CALL    CONST           ; Key in check.
0110    3C                      INC     A               ; Input any key?
0111    20 0B                   JR      NZ,SKIP         ; No.
0113    CD EB09                 CALL    CONIN           ; Get inputed key.
0116    FE 03                   CP      03H             ; BREAK key?
0118    28 19                   JR      Z,TIMEEND       ; Yes.
                        ;
011A    4F                      LD      C,A             ; Display inputed character.
011B    CD EB0C                 CALL    CONOUT          ;
                        ;
011E                    SKIP:
011E    11 0225                 LD      DE,NTIME        ; Time discrepter.
0121    0E 00                   LD      C,00H           ; Read time function.
0123    CD EB4E                 CALL    TIMDAT          ; Read time.
                        ;
0126    CD 017E                 CALL    TIMECHK         ; New & old time compare.
0129    28 E1                   JR      Z,LOOP          ; If same, then loop.
                        ;
012B    CD 018E                 CALL    TIMESET         ; Set new time data.
012E    CD 014F                 CALL    DSPTIME         ; Display time data
0131    18 D9                   JR      LOOP            ; Loop
                        ;
0133                    TIMEEND:
0133    C3 EB03                 JP      WBOOT           ; Jump WBOOT.
                        ;
                        ;       ************************************************
                        ;               GET DIRECT DISPLAY FUNCTION ADDRESS.
                        ;       ************************************************
                        ;
                        ;       NOTE :
                        ;               This routine sends dummy console out function.
                        ;               And get the function execute address in
                        ;               OS ROM.
                        ;
                        ;       <> entry parameter <>
                        ;               NON
                        ;       <> return parameter <>
                        ;               NON
```

```
                        ;       <> preserved registers <>
                        ;               NON
                        ;
                        ;       CAUTION :
                        ;
0136                    GETFKAD:
0136    21 021F                 LD      HL,DIRECT       ; Dummy console out data.
0139    CD 0143                 CALL    DSPMSG          ;
                        ;
013C    2A F2A8                 LD      HL,(LFKADDR)    ; Get the function execute addr.
013F    22 0233                 LD      (FKDIRECT),HL   ; Save the address.
0142    C9                      RET                     ;
                        ;
                        ;       ************************************************
                        ;               DISPLAY MESSAGE UNTIL FIND 0
                        ;       ************************************************
                        ;
                        ;       NOTE :
                        ;
                        ;       <> entry parameter <>
                        ;               HL : Message data top address.
                        ;       <> return parameter <>
                        ;               NON
                        ;       <> preserved registers <>
                        ;               NON
                        ;
0143                    DSPMSG:
0143    7E                      LD      A,(HL)          ; Get message data.
0144    B7                      OR      A               ; End mark?
0145    C8                      RET     Z               ; Yes, then return.
                        ;
0146    4F                      LD      C,A             ; Set display data to c reg.
0147    E5                      PUSH    HL              ; Save message pointer.
0148    CD EB0C                 CALL    CONOUT          ; Display message.
014B    E1                      POP     HL              ; Restore message pointer.
014C    23                      INC     HL              ; Pointer update.
014D    18 F4                   JR      DSPMSG          ; Loop until find 0.
                        ;
                        ;       ************************************************
                        ;               DISPLAY TIME DATA
                        ;       ************************************************
                        ;
                        ;       NOTE :
                        ;               Display time data by calling OS ROM directly.
                        ;
                        ;       <> entry parameter <>
                        ;               NON
                        ;       <> return parameter <>
                        ;               NON
                        ;       <> preserved registers <>
                        ;               NON
                        ;
                        ;       CAUTION :
                        ;
014F                    DSPTIME:
014F    21 020B                 LD      HL,MSG02        ; Date message.
0152    CD 0158                 CALL    DSPT10          ; Display the message.
                        ;
0155    21 0215                 LD      HL,MSG03        ; Time message.
0158                    DSPT10:
0158    11 F2AC                 LD      DE,LESCPRM      ; CONOUT parameter area.
015B    7E                      LD      A,(HL)          ; Set Y-coordinate.
015C    12                      LD      (DE),A          ;
015D    23                      INC     HL              ;
015E    13                      INC     DE              ;
                        ;
015F    7E                      LD      A,(HL)          ; Set X-coordinate.
0160    12                      LD      (DE),A          ;
0161    23                      INC     HL              ;
                        ;
0162    06 08                   LD      B,08H           ; Loop counter.
0164                    DSPT20:
0164    4E                      LD      C,(HL)          ; Display character.
0165    3E FF                   LD      A,SYSBANK       ; Set system bank value.
0167    32 F52E                 LD      (DISBNK),A      ;
016A    DD 2A 0233              LD      IX,(FKDIRECT)   ; OS ROM call address.
016E    C5                      PUSH    BC              ; Save registers.
016F    D5                      PUSH    DE              ;
0170    E5                      PUSH    HL              ;
0171    CD EB69                 CALL    CALLX           ; Go !!
0174    E1                      POP     HL              ; Restore registers.
0175    D1                      POP     DE              ;
0176    C1                      POP     BC              ;
                        ;
0177    1A                      LD      A,(DE)          ; Increment X-coordinate.
0178    3C                      INC     A               ;
0179    12                      LD      (DE),A          ;
017A    23                      INC     HL              ; Message pointer update.
017B    10 E7                   DJNZ    DSPT20          ; Loop.
                        ;
017D    C9                      RET                     ;
                        ;
                        ;       ************************************************
                        ;               CHECK OLD & NEW TIME
                        ;       ************************************************
                        ;
                        ;       NOTE :
                        ;
                        ;         entry parameter <>
                        ;               NON
                        ;       <> return parameter <>
                        ;               ZF : Return ifomation
                        ;                    =1 : New time is same as old one.
                        ;                    =0 : New time is different from old one.
                        ;       <> preserved registers <>
                        ;               NON
                        ;
```

```
017E                    TIMECHK:
017E    21 0225         LD      HL,NTIME        ; New time data.
0181    11 022C         LD      DE,OTIME        ; Old time data.
0184    06 06           LD      B,06H           ; Data counter.
                        ;
0186                    TLOOP:
0186    1A              LD      A,(DE)          ; Get old time data.
0187    BE              CP      (HL)            ; Compare it with new one.
0188    C0              RET     NZ              ; If disagree, then return.
0189    13              INC     DE              ; Poninters update.
018A    23              INC     HL              ;
018B    10 F9           DJNZ    TLOOP           ; Loop 6 times.
018D    C9              RET                     ;

                        ;***********************************************
                        ;       SET TIME DATA
                        ;***********************************************
                        ;
                        ;       NOTE :
                        ;
                        ;       <> entry parameter <>
                        ;               NON
                        ;       <> return parameter <>
                        ;               NON
                        ;       <> preserved registers <>
                        ;               NON
                        ;
018E                    TIMESET:
018E    21 0225         LD      HL,NTIME        ; Set time data to old time area.
0191    11 022C         LD      DE,OTIME        ;
0194    01 0006         LD      BC,6            ; Year/month/date/hour/minuite/second
0197    ED B0           LDIR                    ; Move new data to old area.
                        ;
0199    21 0225         LD      HL,NTIME        ; Set BCD data to message area with ASCII.
019C    11 020D         LD      DE,DATE         ; HL is source, DE is destination.
019F    06 03           LD      B,03H           ; B is counter.
01A1                    SET10:
01A1    CD 01B5         CALL    SETASCII        ; Convert BCD to ASCII.
01A4    23              INC     HL              ; Pointer update.
01A5    13              INC     DE              ;
01A6    10 F9           DJNZ    SET10           ; Loop 3 times. (Year/month/date)
                        ;
01A8    11 0217         LD      DE,TIME         ; Time date setting area.
01AB    06 03           LD      B,03H           ;
01AD                    SET20:
01AD    CD 01B5         CALL    SETASCII        ; Convert BCD to ASCII.
01B0    23              INC     HL              ; Pointer update.
01B1    13              INC     DE              ;
01B2    10 F9           DJNZ    SET20           ; Loop 3 times. (Hour/minuite/second)
01B4    C9              RET                     ;

                        ;***********************************************
                        ;       SET ASCII DATA FROM BCD DATA
                        ;***********************************************
                        ;
                        ;       NOTE :
                        ;
                        ;       <> entry parameter <>
                        ;               HL : BCD data address.
                        ;               DE : ASCII data setting address.
                        ;       <> return parameter <>
                        ;               DE ; Entry DE + 2
                        ;       <> preserved registers <>
                        ;               HL
                        ;
01B5                    SETASCII:
01B5    7E              LD      A,(HL)          ; Get BCD data.
01B6    F5              PUSH    AF              ; Save BCD data.
01B7    0F              RRCA                    ; Move MSB 4 bit to LSB 4bit.
01B8    0F              RRCA                    ;
01B9    0F              RRCA                    ;
01BA    0F              RRCA                    ;
01BB    CD 01BF         CALL    NEXT            ; Set ASCII data by 1 byte.
01BE    F1              POP     AF              ; Restore BCD data.
01BF                    NEXT:
01BF    E6 0F           AND     0FH             ; Check LSB 4 bit.
01C1    C6 30           ADD     A,30H           ; Change to ASCII data.
01C3    12              LD      (DE),A          ; Set ASCII data.
01C4    13              INC     DE              ; Setting pointer update.
01C5    C9              RET                     ;
                        ;
                        ;
01C6                    MSG01:
01C6    0C              DB      0CH
01C7    50 72 65 73     DB      'Present date is          .',CR,LF
01CB    65 6E 74 20
01CF    64 61 74 65
01D3    20 69 73 20
01D7    20 20 20 20
01DB    20 20 20 20
01DF    2E 0D 0A
01E2    50 72 65 73     DB      'Present time is          .',CR,LF
01E6    65 6E 74 20
01EA    74 69 6D 65
01EE    20 69 73 20
01F2    20 20 20 20
01F6    20 20 20 20
01FA    2E 0D 0A
01FD    49 6E 70 75     DB      'Input line = '
0201    74 20 6C 69
0205    6E 65 20 3D
0209    20
020A    00              DB      00H
020B                    MSG02:
020B    01 11           DB      01H,11H                 ; Direct display
020D                    DATE:
```

```
020D  30 30 2F 30                     DB      '00/00/00'
0211  30 2F 30 30
0215                            MSG03:
0215  02 11                           DB      02H,11H                 ; Direct display
0217                            TIME:
0217  30 30 3A 30                     DB      '00:00:00'
021B  30 3A 30 30
                                ;
021F                            DIRECT:
021F  1B D2 01 01                     DB      ESC,0D2H,1,1,20H        ; Direct display dummy
0223  20
0224  00                              DB      00H
                                ;
                                ;
                                ;
0225                            NTIME:
0225                                  DS      7
022C                            OTIME:
022C                                  DS      7
                                ;
0233                            FKDIRECT:
0233                                  DS      2
                                      END
```

3.7 MTOS/MIOS Operations

The PINE is provided with an optional microcassette drive. It is connected to the PINE main unit via a cartridge interface (MS mode). Tape data is also directed to the buzzer in the main unit or an external loudspeaker. The microcassette drive can be controlled either manually or by software. The user can handle microcassettes in the same way as ordinary disk drives.

A microcassette can store only sequential files. It contains a directory file at its beginning so that the system can handle microcassette files on a file basis.

3.7.1 General

The PINE controls I/O operations on microcassette tape (referred to simply as tape from now on) at two levels of control programs (MTOS and MIOS). MTOS is an operating system which corresponds to CP/M BDOS and manages files on tape with unique interface modules. MIOS corresponds to CP/M BIOS and controls the microcassette drive motor and processes tape data on a record basis.

In PINE CP/M, the microcassette drive (MCT) is assigned to drive H:. The application program can handle it as an ordinary external drive without being aware that it is actually an MCT drive.

Each MCT tape has a directory file at its beginning. This file is used to control accesses to the files on the MCT. Whenever an access is made to an MCT file, the directory file is loaded into the directory area in RAM (RAM directory). The results of operations on MCT files are all managed in the RAM directory. Accordingly, when an MCT file is updated, the contents of the RAM directory must also be rewritten into the directory file on the MCT. Reading in this directory file into memory is called "MOUNT" and writing it onto MCT is called "REMOVE".

- MOUNT
The MOUNT function loads the tape directory into the RAM directory. After a mount, any directory update is processed in the RAM directory. This function must be executed before accessing an MCT file. (The OS supports the Auto Mount function.)

- REMOVE
The REMOVE function writes the RAM directory onto the tape directory. This function must be executed after manipulating an MCT file. If the tape is removed without the execution of this function, the tape contents are not guaranteed. Or, in the worst case, the contents of the microcassette that is mounted next may be destroyed.

The PINE MCT is furnished with an LED which indicates whether tape MOUNT/REMOVE can be executed. The LED is turned off by a mount and turned on by a remove. Tape can be mounted or removed while the LED is on.

Figure 3.7.1 shows the relationship of MTOS and MIOS to CP/M.

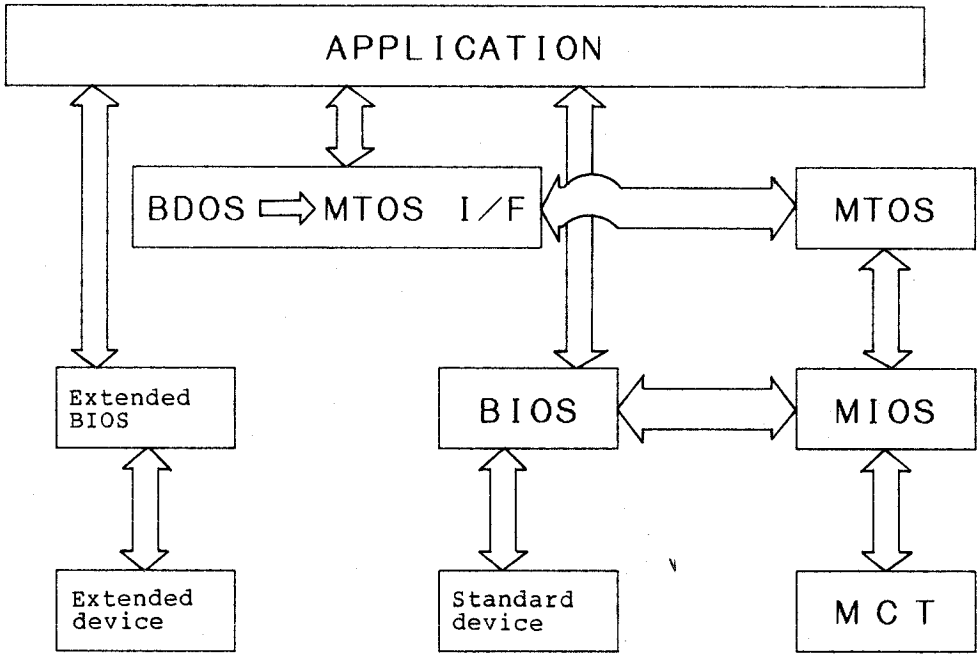


Fig. 3.7.1 MTOS/MIOS Control Flow

3.7.2 File Control

This subsection explains the MCT file structure and file control method.

The structure of an MCT file is shown in Figure 3.7.2.

The PINE directory file can contain a maximum of 12 entries. Its contents are loaded into the RAM directory for control of file accesses.

Each file consists of the header, data, and EOF sections.

Header area:	Contains the information pertaining to the organization of the file.
Data area:	Contains actual file data. The data area is normally made of one or more blocks. The number of blocks depends on the file size.
EOF area:	Is the last block identifying the end of the file.

The PINE can write a block several times to increase data reliability (normally, a block is written twice). Each block has an ID field which stores the block type, block number, and the ordinal number of writes. The ID field is referenced during subsequent read operations.