## CHAPTER 3 CP/M I (BDOS, BIOS) CONTENTS

# CHAPTER 3 CP/M I (BDOS, BIOS)

## 3.1 PINE CP/M

### 3.1.1 General

The PINE operating system is an extended version of Standard CP/M Version 2.2. The new system modules of the PINE CP/M have been introduced in Chapter 2. In this chapter, the extended functions are described in full depth.

The structure of extended CP/M is shown in Figure 3.1.1.

PINE extended CP/M is provided with the following features:

1. Extended BIOS has thirty BIOS entries in addition to the standard BIOS entries to support expansion devices such as RS-232C interface, SIO, cartridge SIO, clock, buzzer, RAM disk, and ROM and RAM cartridges.
2. Most of OS programs are executed in ROM, leaving a large user area.
3. Frequently used application programs are executed in ROM, leaving a large user area.
4. Various types of memory-related devices (main RAM, ROM capsules, ROM and RAM cartridges, microcassette, etc.) are available as disk drives. This provides the PINE user with great operational ease.

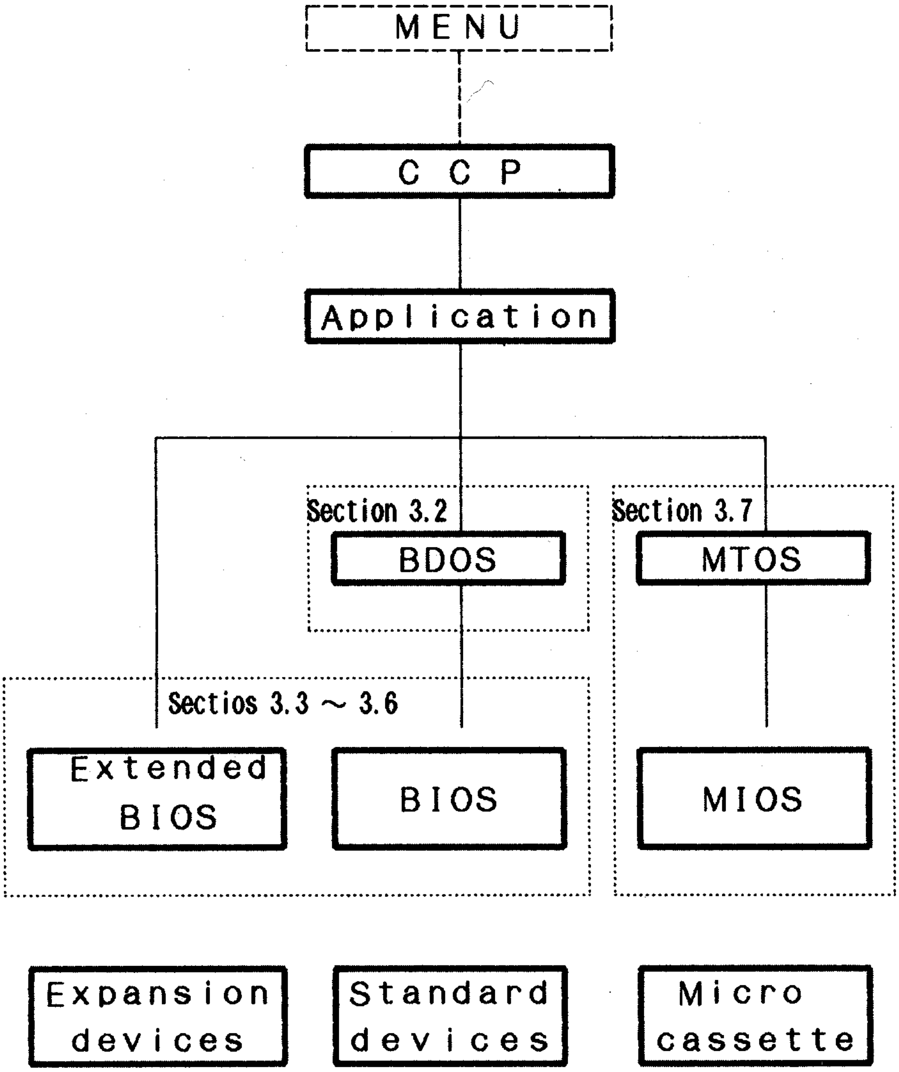


Fig. 3.1.1 CP/M Organization

### 3.1.2 Memory Addressing

The PINE supports four types of banks shown in Figure 3.1.2.

The four banks are:

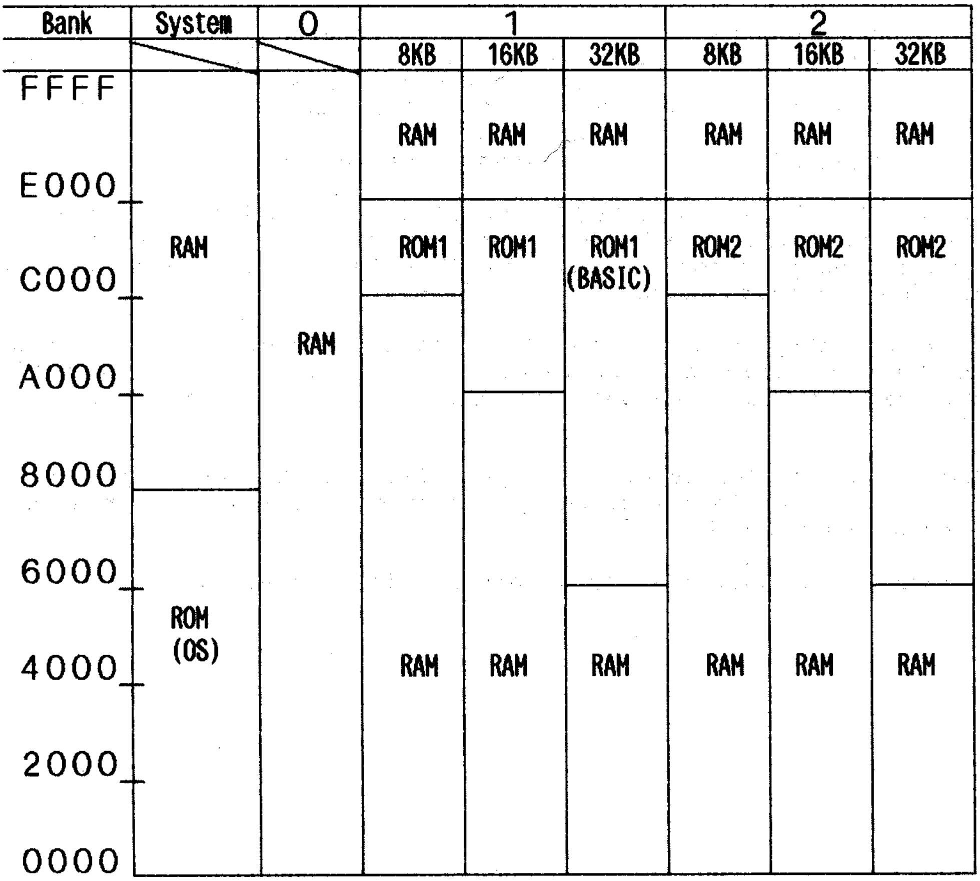
- System bank -- Contains OS ROM and the first half of RAM.
- Bank Ø -- Contains the entire RAM.
- Bank 1 -- Contains BASIC ROM (as shipped from the factory) and part of RAM.
- Bank 2 -- Contains application ROM and part of RAM.

The CP/M modules (CCP, BDOS, and BIOS) are loaded on bank Ø (all RAM) as they are on ordinary CP/M machines. This means that PINE application programs can use the CP/M functions in the same way as under standard CP/M.

Only the entry points to the BDOS and BIOS functions are loaded in RAM to reserve as large a RAM area as possible for application programs. Actual BDOS and BIOS operations are performed in OS ROM.

Each BDOS and BIOS function is provided with two entry points for convenience in executing application programs which run on both banks 1 and 2.

Figure 3.1.3 gives the RAM memory map for the PINE.



Note: The bank 1 of the standard PINE contains BASIC.

Fig. 3.1.2 Bank Structure

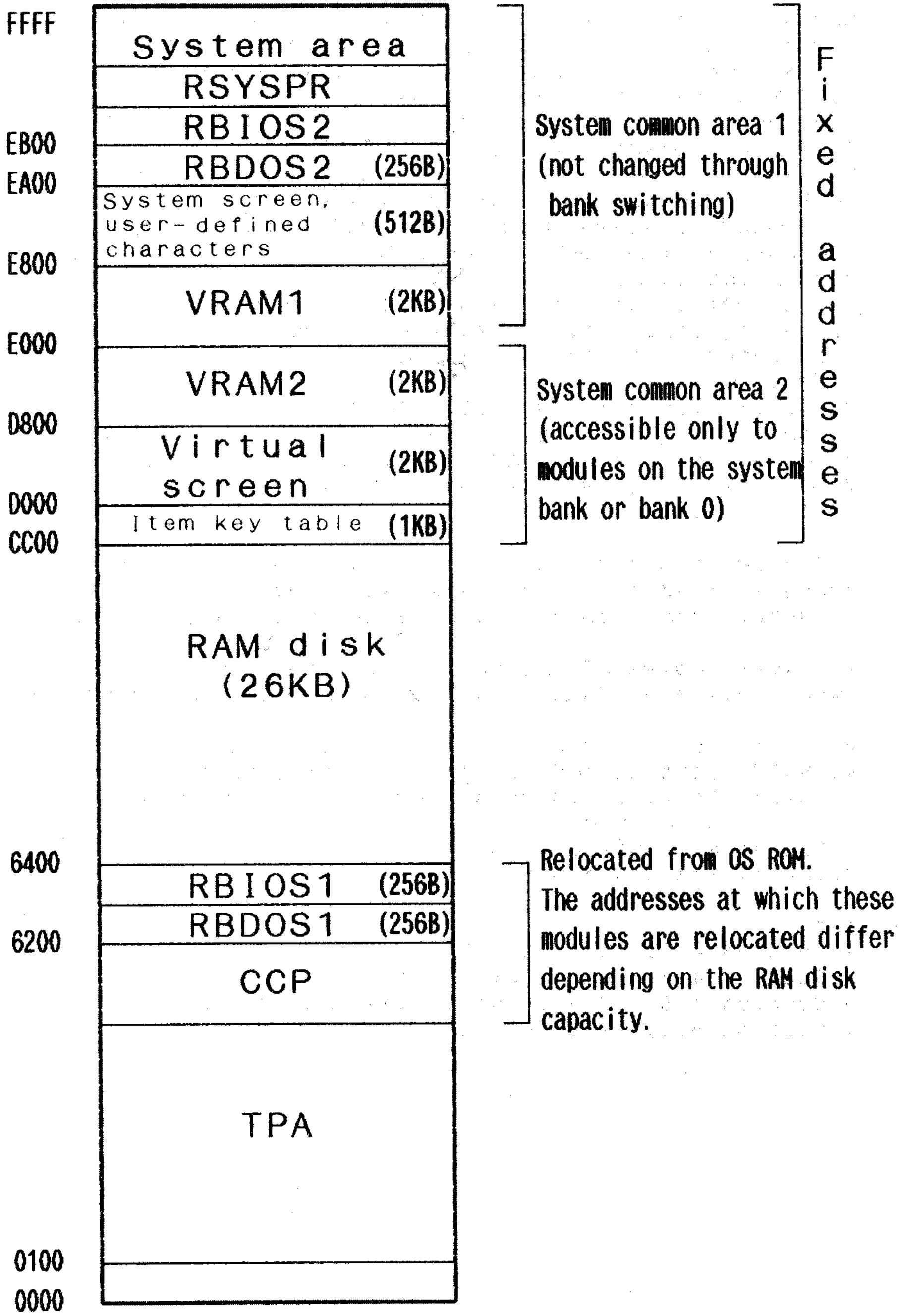

FFFF
System area
RSYSPR
RBIOS2
EB00
RBDOS2 (256B)
EA00
System screen, user-defined characters (512B)
E800
VRAM1 (2KB)
E000
VRAM2 (2KB)
D800
Virtual screen (2KB)
D000
Item key table (1KB)
CC00
RAM disk (26KB)
6400
RBIOS1 (256B)
6200
RBDOS1 (256B)
CCP
TPA
0100
0000
System common area 1 (not changed through bank switching)
System common area 2 (accessible only to modules on the system bank or bank 0)
Fixed addresses
Relocated from OS ROM. The addresses at which these modules are relocated differ depending on the RAM disk capacity.

Fig. 3.1.3 RAM Memory Map

(RAM disk = 26KB, user BIOS area = ØKB)

Notes:

1. Either bank 1 or 2 has three different organizations depending on the ROM capacity (8KB, 16KB, or 32KB). When power is turned on, the system determines the organization of each bank by reading from the header area the capacity of ROM on the corresponding bank.
2. The contents of RBDOS1 and RBDOS2, and those of RBIOS1 and RBIOS2 are exactly the same.
   Load-and-go programs must call RBDOS1 or RBIOS1 (ordinary BDOS or BIOS call) to ask for system services.
   ROM-based programs must call RBDOS2 or RBIOS2 (at a fixed address) because RBDOS1 or RBIOS1 may be located on the background bank.
3. The amount of memory reserved for the virtual screen is fixed regardless of the virtual screen size. This means that changes in the virtual screen size do not affect the CP/M size.
4. The user BIOS size is initially set to Ø. If the user BIOS is specified, however, the area for the user BIOS is reserved between the RAM disk area and the item table area.
5. The system area used by the PINE is divided into the following two types:

   - System area for which initial values are set only when predetermined conditions are satisfied.
   - System area simply used as a temporary work area.

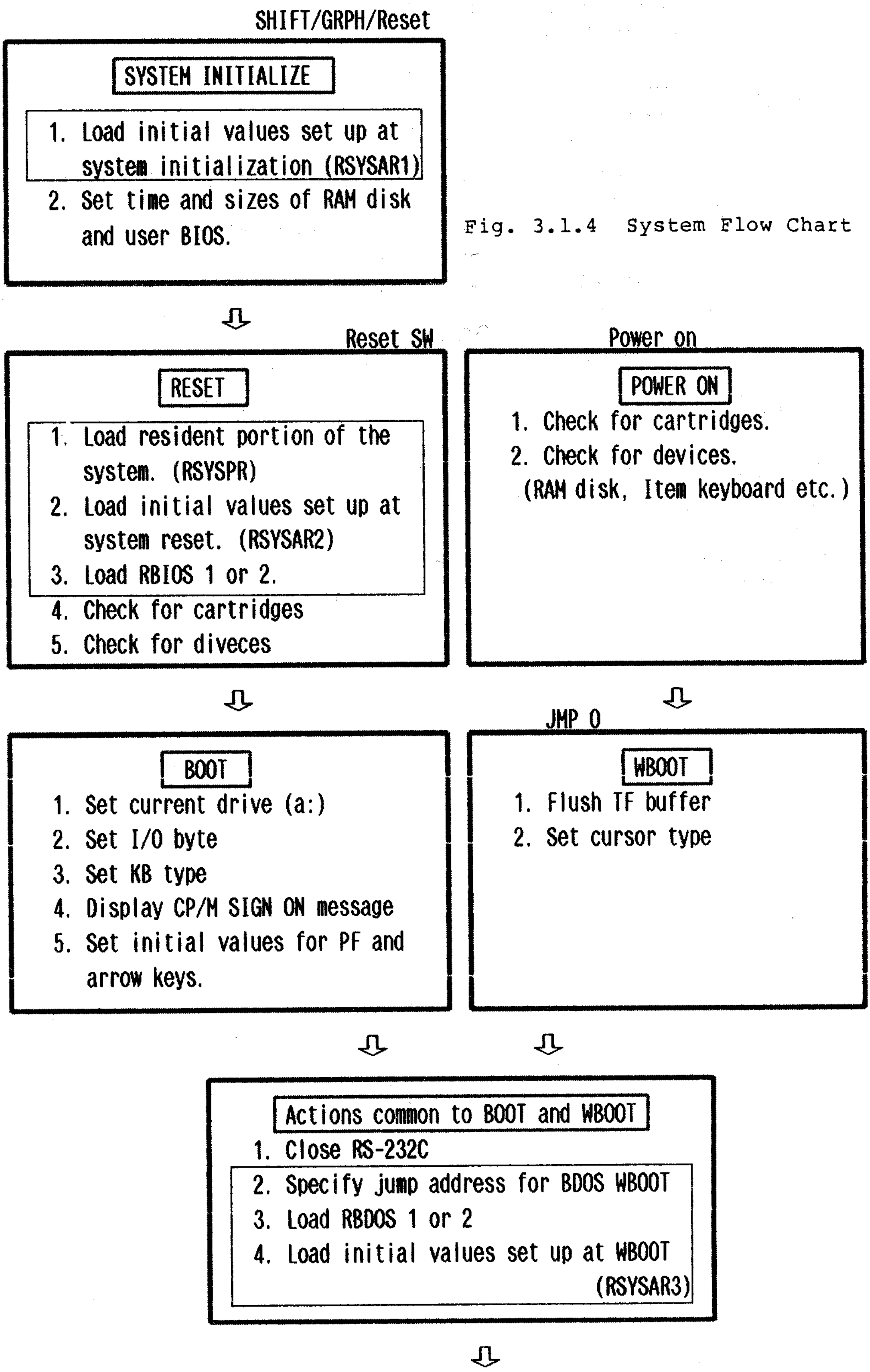
   The system area of the first type is subdivided into three types called RSYSAR1, RSYSAR2, and RSYSAR3, respectively.

   RSYSAR1: Initialized at system initialize time.
   RSYSAR2: Initialized at system initialize or reset time.
   RSYSAR3: Initialized at system initialize, reset, or WBOOT time.
   RSYSAR4: Not initialized.
   RSYSAR5: Stack and buffer.

See also:

- Section 4.1, "User BIOS"
- Section 4.4, "Bank Switching"
- Section 4.6, "Executing a ROM-based Program"
- Section 6.1, "Memory Map"

### 3.1.3 Constructing a CP/M System



SHIFT/GRPH/Reset

SYSTEM INITIALIZE

1. Load initial values set up at system initialization (RSYSAR1)
2. Set time and sizes of RAM disk and user BIOS.

Fig. 3.1.4 System Flow Chart

Reset SW

RESET

1. Load resident portion of the system. (RSYSPR)
2. Load initial values set up at system reset. (RSYSAR2)
3. Load RBIOS 1 or 2.
4. Check for cartridges
5. Check for diveces

Power on

POWER ON

1. Check for cartridges.
2. Check for devices.
   (RAM disk, Item keyboard etc.)

BOOT

1. Set current drive (a:)
2. Set I/O byte
3. Set KB type
4. Display CP/M SIGN ON message
5. Set initial values for PF and arrow keys.

JMP 0

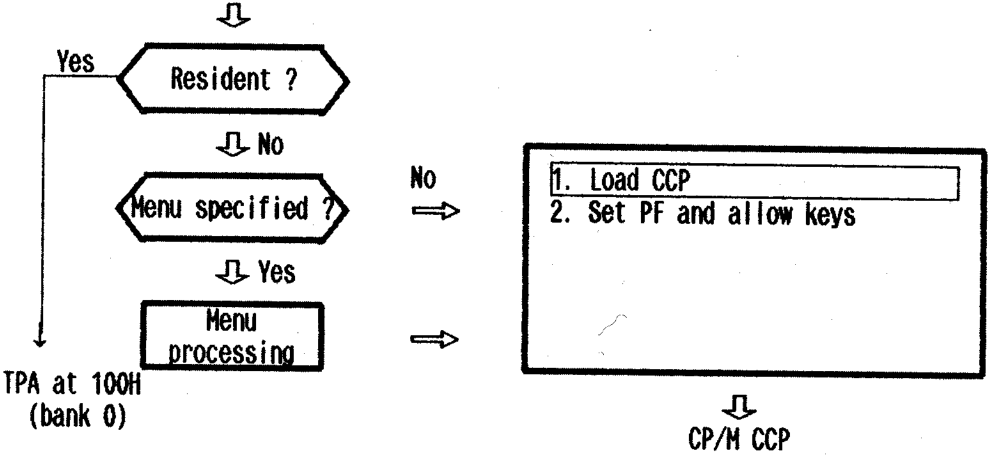
WBOOT

1. Flush TF buffer
2. Set cursor type

Actions common to BOOT and WBOOT

1. Close RS-232C
2. Specify jump address for BDOS WBOOT
3. Load RBDOS 1 or 2
4. Load initial values set up at WBOOT (RSYSAR3)

Continued from the previous page
(Actions common to BOOT and WBOOT)



* CP/M is loaded into RAM through the operations in the boxes enclosed in the larger boxes.

### 3.1.4 Terminating CP/M

CP/M is interrupted during processing by:

1. Pressing the STOP key.
   1) Purpose:
   To inform the system of the depression of the STOP key.
   2) System action:
   Clears the buffers for the keyboard (75Ø8 slave CPU) and BIOS, sets the stop flag BRKFLG (ØFØ19H) to on, and returns Ø3H.
2. Pressing the CTRL and STOP keys.
   1) Purpose:
   To terminate the current I/O operation immediately.
   2) System action:
   Clears the keyboard buffer, sets the CTRL/STOP flag CSTOPFLG (ØFØ1AH) to on, and returns Ø3H. The I/O operation is terminated immediately through the CTRL/STOP flag.
3. Turning off the power switch.
   1) Purpose:
   To turn off power in the restart mode (when a standard keyboard is installed).
   2) System action:
   Continues the I/O operation till the end of the current unit of work, terminates processing, and turns off power. Stops the buzzer (beep processing) immediately. The system performs a warm boot when power is turned on the next time.
4. Turning off the power switch while holding down the CTRL key.
   1) Purpose:
   To turn off power in the continue mode.
   2) System action:
   Continues the I/O operation till the end of the current unit of work, reserves all parameters necessary to continue the processing, and turns off power in the continue mode. The system continues processing at the next power on starting at the point where power was turned off.
5. The arrival of the auto power off time.
   1) Purpose:
   This function is provided to turn off power when the system has been waited for entry from the keyboard for a specified period of time. The function saves power by automatically turning off power when the user forgets to do so.
   2) System action:
   Preserves all parameters necessary to continue the processing and turns off power in the continue mode. The system continues processing at the next power on starting at the point where power was turned off.
6. The detection of a power failure.
   1) Cause:
   The battery voltage drops below a predetermined level.
   2) System action:
   Turns off power in the continue mode as in the action for 4.
7. Pressing the reset switch.
   1) Purpose:
   To cold start the system when the program hangs up. If the CP/M size or the RAM disk is destroyed, however, the system does not start normally.
   2) System action:

Performs a boot operation. The system reserves the RAM disk and user BIOS areas but restores the BIOS entries to initial values. The system also resets the 75Ø8 slave CPU (keyboard parameters are set to default values).

8. Pressing the reset switch while holding down both the right SHIFT and GRPH keys (when a standard keyboard is installed) or the STOP and INIT keys (when an ITEM keyboard is installed).
   1) Purpose:
   To initialize the system. The system starts unless the slave CPU is not in a hang-up state.
   2) System action:
   Initializes the system and resets the slave CPU (keyboard parameters are set to default values). The system starts I/O operations from the beginning.
9. Resetting the slave CPU.
   1) Purpose:
   To start the system when the slave CPU is in a hang-up state.
   2) System action:
   Initializes the entire system including the slave CPU.

Reference:
The following system areas are reserved for the STOP and CTRL/STOP keys and used as flags:

- BRKFLG (ØFØ19H) 1 byte

The STOP flag indicating whether the STOP key has been pressed. This flag is set when a keyboard interrupt occurs and reset by CONIN or CONST (if no entry has been made from the keyboard).
ØØH: STOP key not pressed.
Nonzero: STOP key pressed.

- CSTOPFLG (ØFØ1AH) 1 byte

The CTRL/STOP flag indicating whether the CTRL/STOP keys havebeen pressed. This flag is set by a keyboard interrupt and reset by CONIN or CONST (if no entry has been made from the keyboard).
ØØH: CTRL/STOP keys not pressed.
Nonzero: CTRL/STOP keys pressed.

## 3.2 BDOS Operations

### 3.2.1 General

As mentioned in Section 3.1, the operating system for the PINE is derived from the standard CP/M Version 2.2.

PINE BDOS is located in two places in RAM. This feature offers the following advantages:

- The upper limit of available RAM is indicated so that ordinary CP/M applications can execute without modification.
- The user can call BDOS from a ROM-based program without being aware of bank switching.

PINE expansion devices, especially microcassette, have several operational restrictions.

This section discusses BDOS operations focusing on the items unique to the PINE.

### 3.2.2 BDOS Function Operation Flow

When BDOS is called by a PINE application program, control is first transferred to the entry point to the BDOS in RAM. Then the OS switches banks and calls the real BDOS in OS ROM. Upon completion of processing, the OS switches the bank back to the original bank that was used when BDOS was called and returns control to the application program with return information loaded in registers.

The BIOS used by the BDOS in ROM calls the BIOS in OS ROM.

The procedure for calling BDOS differs depending on the type of the application program.

- Load-and-go programs call "JMP RBDOS1" at address ØØØ5H.
- ROM-based programs call "JMP RBDOS2" at address ØFF9ØH.

Both Load-and-go and ROM-based programs use BDOS functions exactly the same way.

Figure 3.2.1 illustrates the operation flow from a BDOS call to the return of control to the application program.

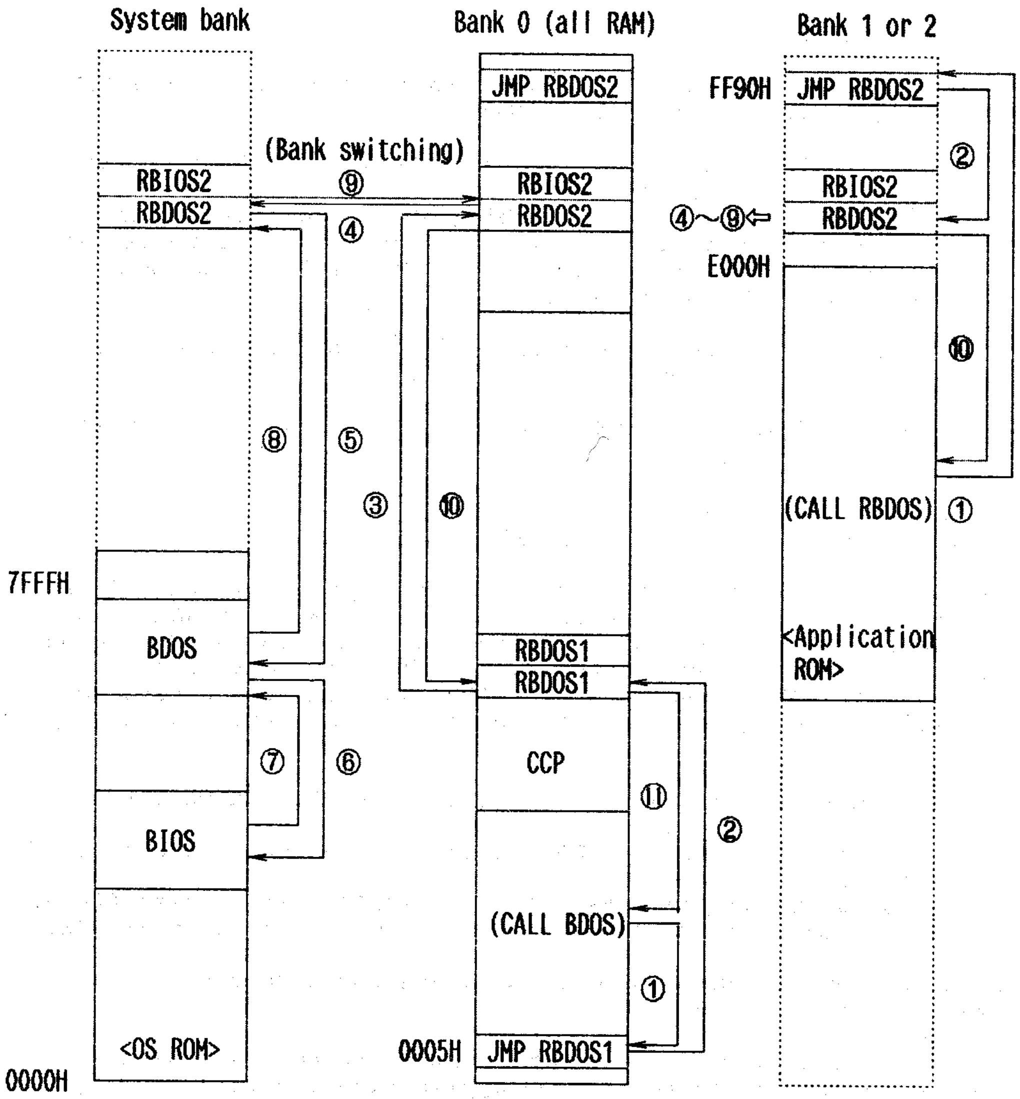


Fig. 3.2.1 BDOS Call Operation Flow

### 3.2.3 BDOS Functions

PINE application programs use the same interface as ordinary CP/M application programs when calling BDOS.

This subsection lists the BDOS functions with a brief description. Refer to "CP/M 2.2 Interface Guide" for detailed information about the use of BDOS functions.

Table 3.2.1 BDOS Calls

| Number | Function Name | Input | Output |
|---|---|---|---|
| 0 | System Reset | C : 00H | None |
| 1 | Console Input | C : 01H | A : Input char |
| 2 | Console Output | C : 02H<br>E : Output char | None |
| 3 | Reader Input | C : 03H | A : Input char |
| 4 | Punch Output | C : 04H<br>E : Output char | None |
| 5 | List Output | C : 05H<br>E : Outpur char | None |
| 6 | Direct Console I/O | C : 06H<br>E : 0FFH (input)<br>: Output char (output) | A : Input char (input)<br>: None |
| 7 | Get IOBYTE | C : 07H | A : IOBYTE |
| 8 | Set IOBYTE | C : 08H<br>E : IOBYTE | None |
| 9 | Print String | C : 09H<br>DE : Address at which the string is stored. | None |
| 10 | Read Console Buffer | C : 0AH<br>DE : Buffer address | Loads the buffer with entry from the console. |
| 11 | Get Console Status | C : 0BH | A : Console Status |
| 12 | Get Version Number | C : 0CH | HL : Version Number |
| 13 | Reset Disk System | C : 0DH | None |
| 14 | Select Disk | C : 0EH<br>E : Disk number | None |
| 15 | Open File | C : 0FH<br>DE : FCB address | A : Directory code |

| Number | Function Name | Input | Output |
|---|---|---|---|
| 16 | Close File | C : 10H<br>DE : FCB address | A : Directory code |
| 17 | Search for First | C : 11H<br>DE : FCB address | A : Directory code |
| 18 | Search for Next | C : 12H | A : Directory code |
| 19 | Delete File | C : 13H<br>DE : FCB address | A : Directory code |
| 20 | Read Sequential | C : 14H<br>DE : FCB address | A : Return code |
| 21 | Write Sequential | C : 15H<br>DE : FCB address | A : Return code |
| 22 | Create File | C : 16H<br>DE : FCB address | A : Directory code |
| 23 | Rename File | C : 17H<br>DE : FCB address | A : Directory code |
| 24 | Get Login Vector | C : 18H | HL : Login vector |
| 25 | Get Disk Number | C : 19H | A : Disk Number |
| 26 | Set DMA Address | C : 1AH<br>DE : DMA address | None |
| 27 | Get Allocation Address | C : 1BH | HL : Allocation address |
| 28 | Write Protect Disk | C : 1CH | None |
| 29 | Get R/O Vector | C : 1CH | HL : R/O vector |
| 30 | Set File Attributes | C : 1EH<br>DE : FCB address | A : Directory code |
| 31 | Get DPB Address | C : 1FH | HL : DPB address |
| 32 | Set/get User Code | C : 20H<br>E : 0FFH (Get)<br>: User code (Set) | A : None (Set)<br>: User code (Get) |
| 33 | Read Random | C : 21H<br>DE : FCB address | A : Return code |
| 34 | Write Random | C : 22H<br>DE : FCB address | A : Return code |
| 35 | Compute File Size | C : 23H<br>DE : FCB address | FCB r0, r1, r2 |
| 36 | Set Random Number | C : 24H<br>DE : FCB address | FCB r0, r1, r2 |

| Number | Function Name | Input | Output |
|---|---|---|---|
| 37 | Reset Disk Drive | C : 25H<br>DE : Drive vector | None |
| 38 | | | |
| 39 | | | |
| 40 | Random Write with Zero File | C : 28H<br>DE : FCB address | A : Return code |
| 251 | Verify File | C : 0FBH<br>DE : T~FCB | |
| 252 | Remove Tape | C : 0FCH | |
| 253 | Mount Tape | C : 0FDH | |
| 254 | Read Tape ID | C : 0FEH | |
| 255 | Create Tape Directory | C : 0FFH<br>DE : T~FCB | |

Note: Functions 251 through 255 comprises Extended BDOS (MTOS) for microcassette. See Section 3.7, "MTOS/MIOS Operations" for details.

### 3.2.4 BDOS Errors

BDOS errors are divided into the following categories:

1. Bad Sector error
   1) Cause:
   An error was found while reading or writing a disk.
   2) Action:
   Terminate processing by pressing STOP or CTRL/C. If any other key is hit, the system continues processing, ignoring the BAD SECTOR error.

2. Select error
   1) Cause:
   An attempt was made to address a drive beyond the specified range or a drive that was not ready.
   2) Action:
   Press any key. The system sets the drive to the currently logged in drive.

3. R/O error
   1) Cause:
   An attempt was made to write a read-only disk.
   2) Action:
   Press any key. The system performs a warm boot.

4. File R/O error
   1) Cause:
   An attempt was made to write a read-only file.
   2) Action:
   Press any key. The system performs a warm boot.

A drive is reported as not ready when:

1. Floppy disk drive power is off.
2. The floppy disk drive cable is not connected.
3. No disk is inserted.
4. No cartridge is installed.
5. No ROM capsule is installed.

Table 3.2.2 shows the relationship between disk devices and BDOS errors.

| Cause | RAM disk<br>RAM cartridge | ROM capsule<br>ROM cartridge | External<br>RAM disk | Micro-cassette | BDOS action |
|---|---|---|---|---|---|
| Checksum error | ○ | – | – | – | Bad Sector |
| Directory full | ○ | – | ○ | ○ | Returns with A = 0FFH |
| Disk full | ○ | – | ○ | ○ | Returns with A = 0FFH |
| Write processing | – | ○ | – | – | R/O |
| Attempt to write write-protected device | ○ | – | ○ | ○ | R/O |
| End of tape encountered during write | – | – | – | ○ | Bad Sector |
| File not found in directory | ○ | ○ | ○ | ○ | Returns with A = 0FFH |
| File not found | – | – | – | ○ | Bad Sector |
| Drive not ready | ○ | ○ | ○ | ○ | Select |
| No cassette found during read or mount | – | – | – | ○ | Bad Sector |
| No cassette found during write or remove | – | – | – | ○ | Bad Sector |

○ ··· probable

– ··· Not probable

Table 3.2.2 BDOS Error Recovery Actions

As described above, BDOS can indicate four types of error conditions. Since these errors are handled totally under BDOS control, they may destroy the current screen image or initiate a warm boot on receipt of user response from the keyboard after displaying the error.

One of the countermeasures to avoid this is to make the application program report and handle error conditions by itself.

The application program can achieve this by taking the following two measures against the error:

1. Receiving BDOS error information in a return code.
2. Rewriting the jump vector for BDOS error processing and performing use-supplied error processing.

(1) Receiving BDOS error information in a return code

1) Procedure

The application program can receive any BDOS error information in registers by calling location ØØ12H (SETERR) in OS ROM (system bank). It can also have BDOS return any error information by calling location ØØ15H (RSTERR) in OS ROM.

The application program must use BIOS CALLX (WBOOT + 66H) to directly call the routine in OS ROM.

2) Return codes

The A and H registers are loaded with the following return codes when SETERR is executed:

| Register<br>Error | A | H | |
|---|---|---|---|
| BAD SECTOR | 0FFH | 01H | Standard CP/M BDOS errors |
| BAD SELECT | 0FFH | 02H | |
| R/O DISK | 0FFH | 03H | |
| R/O FILE | 0FFH | 04H | |
| MCT ERROR | 0FFH | 05H | MCT only |
| | | 00H | |

When the H register is loaded with ØØH, the return code corresponding to the CP/M return information is also loaded in the A register.

For Bad Sector errors, BDOS stores more detailed error information in system area BIOSERROR (ØF52BH).

BIOSERROR (ØF52BH) 1 byte

Loaded with one of the following information at the end of a BIOS disk read or write:

= ØØH: Normal termination
= Ø1H: Read error
= Ø2H: Write error
= Ø3H: Write protect error
= Ø4H: Time-out error
= Ø5H: Seek error
= Ø6H: Break error
= Ø7H: Power off error
= Ø8H: Mount error
= ØFEH: Other errors

3) Programming notes

a. Once SETERR is executed, BDOS only returns error status and performs no error processing until RSTERR or WBOOT is executed.

b. When SETERR is executed, the results are not guaranteed unless the application program performs its own error checking and recovery processing.

(2) Rewriting the jump vector for processing BDOS errors

1) Procedure

The jump vector for BDOS error processing is located at the beginning of BDOS in RAM. The application program can perform its own error processing by changing the contents of the jump vector.

The address and contents of the jump vector are shown below. (The address of RBDOS1 can be obtained from the contents of ØØ6H and ØØ7H.)

| Address | Data | Contents |
|---|---|---|
| RBDOS1+03H | DW PERERR | Address of parameter error processing routine (Bad Sector) |
| RBDOS1+05H | DW SELERR | Address of Select error processing routine (Bad Select) |
| RBDOS1+07H | DW RODERR | Address of R/O Disk error processing routine (R/O Disk) |
| RBDOS1+09H | DW ROFERR | Address of R/O File error processing routine (R/O File) |

2) Programming notes

a. Since the stack area for the BDOS is used during BDOS processing, it is necessary to restore the stack area for the application program when returning control directly to the application program.

b. Note that bank Ø (all RAM) is selected during BDOS processing when rewriting the jump vector from a ROM-based program.

c. When rewriting the jump vector from a ROM-based program, keep in mind which bank the program is using because the jump vector may be located in the background bank of the bank on which the ROM-based program resides.

d. The user error processing routine must not contain any BDOS call if it is to return control to the system (BDOS) after error processing.

e. Switch the active bank to the system bank before returning control to the system.

```
                         ;       *************************************************
                         ;               BDOS ERROR RECOVERY SAMPLE PROGRAM
                         ;       *************************************************
                         ;
                         ;       NOTE :
                         ;       <> assemble condition <>
                         ;
                                 .Z80
                         ;
                         ;       <> loading address <>
                         ;
                                 .PHASE  100H
                         ;
                         ;       <> constant values <>
                         ;
0012                     SETERR          EQU     00012H          ; SETERR routine address
0015                     RSTERR          EQU     00015H          ; RSTERR routine address
                         ;
EB03                     WBOOT           EQU     0EB03H          ; WBOOT entry address
EB09                     CONIN           EQU     0EB09H          ; CONIN entry address
EB0C                     CONOUT          EQU     0EB0CH          ; CONOUT entry address
EB63                     LDIRX           EQU     0EB63H          ; LDIRX entry address
EB69                     CALLX           EQU     0EB69H          ; CALLX entry address
                         ;
0005                     RBDOS1          EQU     00005H          ; RBDOS1 entry address
FF90                     RBDOS2          EQU     0FF90H          ; RBDOS2 entry address
                         ;
F52B                     BIOSERROR       EQU     0F52BH          ; BIOS error information
F52E                     DISBNK          EQU     0F52EH          ; Distination bank
                         ;
1000                     MAINSP          EQU     1000H           ; Stack pointer
1000                     BDOSSP          EQU     1000H           ; Stack pointer
                         ;
                         ;       *************************************************
                         ;               SELECT BDOS ERROR RECOVER
                         ;       *************************************************
0100                     START:
0100     31 1000                 LD      SP,MAINSP       ; Set stack pointer.
                         ;
0103     CD 0109                 CALL    SELERR          ; Select error recover type.
0106                     LOOP:
                         ;
                         ;       *************************************************
                         ;               USER PROGRAM
                         ;       *************************************************
                         ;
                         ;       NOTE :
                         ;               This part is user program.
                         ;
0106     C3 0106                 JP      LOOP            ; Loop permanent.
                         ;
                         ;       *************************************************
                         ;               SELECT BDOS ERROR RECOVERY
                         ;       *************************************************
                         ;
                         ;       NOTE :
                         ;               Select BDOS error recovery type.
                         ;                1. Using SETERR and RSTERR
                         ;                2. Replacing BDOS error vector
                         ;
                         ;       <> entry parameter <>
                         ;               NON
                         ;       <> return parameter <>
                         ;               NON
                         ;       <> preserved registers <>
                         ;               NON
                         ;
                         ;       CAUTION :
                         ;               If BREAK key is pressed, then WBOOT
                         ;
0109                     SELERR:
0109     21 01E2                 LD      HL,MSG01        ; Select error recover message.
010C     CD 013B                 CALL    DSPMSG          ; Display message.
                         ;
010F     CD 0147                 CALL    KEYIN           ; Get input key code.
0112     D6 31                   SUB     31H             ; Using SETERR?
0114     28 05                   JR      Z,ERR010        ; Yes.
0116     3D                      DEC     A               ; Using error vector?
0117     28 0F                   JR      Z,ERR100        ; Yes.
0119     18 EE                   JR      SELERR          ; Others, then retry.
                         ;
                         ;       CALL SETERR ROUTINE.
                         ;
011B                     ERR010:
011B     DD 21 0012              LD      IX,SETERR       ; Set calling address.
011F     3E FF                   LD      A,0FFH          ; Select system bank.
0121     32 F52E                 LD      (DISBNK),A
0124     CD EB69                 CALL    CALLX           ; Call SETERR.
0127     C9                      RET                     ;
                         ;
                         ;       CHANGE ERROR VECTOR.
                         ;
0128                     ERR100:
0128     21 01DA                 LD      HL,VECTOR       ; New error vector address.
012B     ED 5B 0006              LD      DE,(RBDOS1+1)   ; Get error vector address.
012F     13                      INC     DE              ;  RBDOS1 top addr. + 3
0130     13                      INC     DE              ;
0131     13                      INC     DE              ;
0132     01 0008                 LD      BC,0008H        ; Transmite byte no.
0135     3E 00                   LD      A,00H           ; Select bank 0 (RAM bank).
0137     CD EB63                 CALL    LDIRX           ; Change error vector.
013A     C9                      RET                     ;
                         ;
```

```
                        ;       *************************************************
                        ;               DISPLAY MESSAGE
                        ;       *************************************************
                        ;
                        ;       NOTE :
                        ;               Display message until fine 00H.
                        ;
                        ;       <> entry parameter <>
                        ;               HL : Message data top address
                        ;       <> return parameter <>
                        ;               NON
                        ;       <> preserved registers <>
                        ;               NON
                        ;
                        ;       CAUTION :
                        ;
013B                    DSPMSG:
013B    7E                      LD      A,(HL)          ; Get data 1 byte.
013C    B7                      OR      A               ; End of data?
013D    C8                      RET     Z               ; Yes.
                        ;
013E    4F                      LD      C,A             ; Set display data.
013F    E5                      PUSH    HL              ; Save pointer.
0140    CD EB0C                 CALL    CONOUT          ; Display message 1 byte.
0143    E1                      POP     HL              ; Restore pointer.
0144    23                      INC     HL              ; Update pointer.
0145    18 F4                   JR      DSPMSG          ; Loop until find 0.
                        ;
                        ;       *************************************************
                        ;               INPUT A KEY DATA
                        ;       *************************************************
                        ;
                        ;       NOTE :
                        ;               Get inputed key data.
                        ;
                        ;       <> entry parameter <>
                        ;               NON
                        ;       <> return parameter <>
                        ;               NON
                        ;       <> preserved registers <>
                        ;               NON
                        ;
                        ;       CAUTION :
                        ;               If BREAK key is pressed, then WBOOT
                        ;
0147                    KEYIN:
0147    CD EB09                 CALL    CONIN           ; Get inputed key code.
014A    FE 03                   CP      03H             ; Break code?
014C    CA EB03                 JP      Z,WBOOT         ; Yes, then WBOOT.
014F    C9                      RET
                        ;
                        ;       *************************************************
                        ;               BDOS ERROR RECOVERY
                        ;       *************************************************
                        ;
                        ;       NOTE :
                        ;               BDOS error recovery
                        ;
                        ;       <> entry parameter <>
                        ;               H   : Error type 1
                        ;               A   : Error type 2
                        ;       <> return parameter <>
                        ;               NON
                        ;       <> preserved registers <>
                        ;               NON
                        ;
                        ;       CAUTION :
                        ;               If BREAK key is pressed, then WBOOT
                        ;
0150                    ERRCHK:
0150    4F                      LD      C,A             ; Save return code.
0151    7C                      LD      A,H             ; Error type.
0152    B7                      OR      A               ; Normal end?
0153    28 10                   JR      Z,BDOSER        ; Yes.
                        ;
0155    3D                      DEC     A               ; Bad sector?
0156    28 20                   JR      Z,BADSEC        ; Yes.
0158    3D                      DEC     A               ; Bad select?
0159    28 33                   JR      Z,BADSEL        ; Yes.
015B    3D                      DEC     A               ; Read only disk?
015C    28 3A                   JR      Z,RODISK        ; Yes.
015E    3D                      DEC     A               ; Read only file?
015F    28 41                   JR      Z,ROFILE        ; Yes.
0161    3D                      DEC     A               ; Micro cassette error?
0162    28 48                   JR      Z,MCTERR        ; Yes.
0164    C9                                              ;
                        ;
                        ;       BDOS ERROR INFORMATION.
                        ;
0165                    BDOSER:
0165    C5                      PUSH    BC              ; Save return code.
0166    21 0239                 LD      HL,MSG04        ; BDOS error code message.
0169    CD 013B                 CALL    DSPMSG          ; Display message.
016C    C1                      POP     BC              ; Restore return code.
                        ;
016D    3E 30                   LD      A,30H           ; Change error code to ASCII.
016F    81                      ADD     A,C             ;  Return code + 30H
0170    4F                      LD      C,A             ;
0171    CD EB0C                 CALL    CONOUT          ; Display return code.
0174    CD 0147                 CALL    KEYIN           ; Input any key.
0177    C9                      RET                     ;
                        ;
                        ;       BAD SECTOR
                        ;
0178                    BADSEC:
0178    3A F52B                 LD      A,(BIOSERROR)   ; BIOS error type.
017B    87                      ADD     A,A             ; Get message address.
017C    21 0250                 LD      HL,MSG05        ;  Message table top address.
```

```
017F   06 00                              LD      B,00H           ;  Get target message pointer.
0181   4F                                 LD      C,A             ;
0182   09                                 ADD     HL,BC           ;
0183   5E                                 LD      E,(HL)          ;  Get target message address.
0184   23                                 INC     HL              ;
0185   56                                 LD      D,(HL)          ;
0186   EB                                 EX      DE,HL           ; Set message address to HL.
0187   CD 013B                            CALL    DSPMSG          ; Display message.
018A   CD 0147                            CALL    KEYIN           ; Input any key.
018D   C9                                 RET                     ;
                            ;
                            ;             BAD SELECT.
                            ;
018E                        BADSEL:
018E   21 02F6                            LD      HL,MSG06        ; Bad select message.
0191   CD 013B                            CALL    DSPMSG          ; Display message.
0194   CD 0147                            CALL    KEYIN           ; Input any key.
0197   C9                                 RET                     ;
                            ;
                            ;             READ ONLY DISK.
                            ;
0198                        RODISK:
0198   21 0308                            LD      HL,MSG07        ; Read only disk message.
019B   CD 013B                            CALL    DSPMSG          ; Display message.
019E   CD 0147                            CALL    KEYIN           ; Input any key.
01A1   C9                                 RET                     ;
                            ;
                            ;             READ ONLY FILE.
                            ;
01A2                        ROFILE:
01A2   21 031C                            LD      HL,MSG08        ; Read only file message.
01A5   CD 013B                            CALL    DSPMSG          ; Display message.
01A8   CD 0147                            CALL    KEYIN           ; Input any key.
01AB   C9                                 RET                     ;
                            ;
                            ;             MICRO CASSETTE ERROR
                            ;
01AC                        MCTERR:
01AC   21 0330                            LD      HL,MSG09        ; Micro cassette error message.
01AF   CD 013B                            CALL    DSPMSG          ; Display message.
01B2   CD 0147                            CALL    KEYIN           ; Input any key.
01B5   C9                                 RET                     ;
                            ;
                            ;
                            ;
01B6                        XBADSEC:
01B6   31 1000                            LD      SP,BDOSSP       ; Set stack pointer.
01B9   CD 0178                            CALL    BADSEC          ; Bad sector error.
01BC   C3 0106                            JP      LOOP            ; Return to user program.
01BF                        XBADSEL:
01BF   31 1000                            LD      SP,BDOSSP       ; Set stack pointer.
01C2   CD 018E                            CALL    BADSEL          ; Bad select error.
01C5   C3 0106                            JP      LOOP            ; Return to user program.
01C8                        XRODISK:
01C8   31 1000                            LD      SP,BDOSSP       ; Set stack pointer.
01CB   CD 0198                            CALL    RODISK          ; Read only disk error.
01CE   C3 0106                            JP      LOOP            ; Return to user program.
01D1                        XROFILE:
01D1   31 1000                            LD      SP,BDOSSP       ; Set stack pointer.
01D4   CD 01A2                            CALL    ROFILE          ; Read only file error.
01D7   C3 0106                            JP      LOOP            ; Return to user program.
                            ;
                            ;             NEW ERROR VECTOR
                            ;
01DA                        VECTOR:
01DA   01B6                               DW      XBADSEC         ; Bad sector
01DC   01BF                               DW      XBADSEL         ; Bad select
01DE   01C8                               DW      XRODISK         ; Read only disk
01E0   01D1                               DW      XROFILE         ; Read only file
                            ;
                            ;             MESSAGE
                            ;
01E2                        MSG01:
01E2   0C                                 DB      0CH
01E3   53 65 6C 65                        DB      'Select BDOS error recover type.',0DH,0AH
01E7   63 74 20 42
01EB   44 4F 53 20
01EF   65 72 72 6F
01F3   72 20 72 65
01F7   63 6F 76 65
01FB   72 20 74 79
01FF   70 65 2E 0D
0203   0A
0204   20 20 31 20                        DB      '  1 -- Using SETERR',0DH,0AH
0208   2D 2D 20 55
020C   73 69 6E 67
0210   20 53 45 54
0214   45 52 52 0D
0218   0A
0219   20 20 32 20                        DB      '  2 -- Replacing error vector',0DH,0AH
021D   2D 2D 20 52
0221   65 70 6C 61
0225   63 69 6E 67
0229   20 65 72 72
022D   6F 72 20 76
0231   65 63 74 6F
0235   72 0D 0A
0238   00                                 DB      00H
0239                        MSG04:
0239   0D 0A                              DB      0DH,0AH
023B   42 44 4F 53                        DB      'BDOS return code is '
023F   20 72 65 74
0243   75 72 6E 20
0247   63 6F 64 65
024B   20 69 73 20
024F   00                                 DB      00H
0250                        MSG05:
0250   0260                               DW      MSG050
```

```
0252   0273                          DW     MSG051
0254   0283                          DW     MSG052
0256   0294                          DW     MSG053
0258   02AD                          DW     MSG054
025A   02C2                          DW     MSG055
025C   02D2                          DW     MSG056
025E   02E3                          DW     MSG057
0260                        MSG050:
0260   0D 0A                         DB     0DH,0AH
0262   4E 6F 72 6D                   DB     'Normal return.',0DH,0AH
0266   61 6C 20 72
026A   65 74 75 72
026E   6E 2E 0D 0A
0272   00                            DB     00H
0273                        MSG051:
0273   0D 0A                         DB     0DH,0AH
0275   52 65 61 64                   DB     'Read error.',0DH,0AH
0279   20 65 72 72
027D   6F 72 2E 0D
0281   0A
0282   00                            DB     00H
0283                        MSG052:
0283   0D 0A                         DB     0DH,0AH
0285   57 72 69 74                   DB     'Write error.',0DH,0AH
0289   65 20 65 72
028D   72 6F 72 2E
0291   0D 0A
0293   00                            DB     00H
0294                        MSG053:
0294   0D 0A                         DB     0DH,0AH
0296   57 72 69 74                   DB     'Write protect error.',0DH,0AH
029A   65 20 70 72
029E   6F 74 65 63
02A2   74 20 65 72
02A6   72 6F 72 2E
02AA   0D 0A
02AC   00                            DB     00H
02AD                        MSG054
02AD   0D 0A                         DB     0DH,0AH
02AF   54 69 6D 65                   DB     'Time over error.',0DH,0AH
02B3   20 6F 76 65
02B7   72 20 65 72
02BB   72 6F 72 2E
02BF   0D 0A
02C1   00                            DB     00H
02C2                        MSG055:
02C2   0D 0A                         DB     0DH,0AH
02C4   53 65 65 6B                   DB     'Seek error.',0DH,0AH
02C8   20 65 72 72
02CC   6F 72 2E 0D
02D0   0A
02D1   00                            DB     00H
02D2                        MSG056:
02D2   0D 0A                         DB     0DH,0AH
02D4   42 72 65 61                   DB     'Break error.',0DH,0AH
02D8   6B 20 65 72
02DC   72 6F 72 2E
02E0   0D 0A
02E2   00                            DB     00H
02E3                        MSG057:
02E3   0D 0A                         DB     0DH,0AH
02E5   50 6F 77 65                   DB     'Power off error.',0DH,0AH
02E9   72 20 6F 66
02ED   66 20 65 72
02F1   72 6F 72 2E
02F5   0D 0A
02F7   00                            DB     00H
02F8                        MSG06:
02F8   0D 0A                         DB     0DH,0AH
02FA   42 61 64 20                   DB     'Bad select.',0DH,0AH
02FE   73 65 6C 65
0302   63 74 2E 0D
0306   0A
0307   00                            DB     00H
0308                        MSG07:
0308   0D 0A                         DB     0DH,0AH
030A   52 65 61 64                   DB     'Read only disk.',0DH,0AH
030E   20 6F 6E 6C
0312   79 20 64 69
0316   73 6B 2E 0D
031A   0A
031B   00                            DB     00H
031C                        MSG08:
031C   0D 0A                         DB     0DH,0AH
031E   52 65 61 64                   DB     'Read only file.',0DH,0AH
0322   20 6F 6E 6C
0326   79 20 66 69
032A   6C 65 2E 0D
032E   0A
032F   00                            DB     00H
0330                        MSG09:
0330   0D 0A                         DB     0DH,0AH
0332   4D 69 63 72                   DB     'Micro cassette error.',0DH,0AH
0336   6F 20 63 61
033A   73 73 65 74
033E   74 65 20 65
0342   72 72 6F 72
0346   2E 0D 0A
0349   00                            DB     00H

                                     END
```

## 3.3 BIOS Operations

### 3.3.1 General

The major BIOS operations are carried out in OS ROM of the system bank as BDOS operations are.

The PINE OS provides two entry points to BIOS in RAM to enable ROM-based programs to use BIOS without being aware of the banks.

PINE extended BIOS allows the user to have easy access to PINE-unique peripheral devices such as serial interfaces, a clock, buzzer, or expansion I/O units (ROM/RAM cartridge, MCT, etc.).

PINE BIOS is provided with user BIOS and a hook to user BIOS to facilitate system extension by the user.

### 3.3.2 BIOS Function Operation Flow

#### 3.3.2.1 Outline

When a call to BIOS is made from an application program, the PINE OS takes the following actions as when a BDOS call is made:

1. Switches the active bank to the system bank within the BIOS in RAM.
2. Calls the real BIOS in OS ROM.
3. Upon completion of BIOS processing, switches the bank to the one that was active when the BIOS call was made, and returns control to the application program with return information and data loaded in registers.

The procedure for calling BIOS differs depending on the type of the application program.

1. The load-and-go program calls a BIOS function by specifying its address obtained by adding the function offset to the JMP WBOOT addres (ØØØØH).
2. The ROM-based program directly calls the entry in the BIOS jump table in RBIOS2 (ØEBØØH - ØEBFFH).

In both cases, the called function operates exactly the same way.

### 3.3.2.2 PREBIOS and PSTBIOS

The PINE introduces PREBIOS and PSTBIOS for processing BIOS calls without disruption due to interrupts and thus increasing system reliability.

There are some cases when the PINE cannot successfully resume the execution of a program, which is interrupted by an interrupt occurring during execution of a BIOS call, when control is returned from the interrupt servicing program if the interrupt is processed immediately.

This problem can be avoided if interrupts are disabled before the execution of any BIOS call and processing of any interrupts occurring during the execution of the BIOS call is performed after the BIOS call has been terminated. This is controlled by PREBIOS and PSTBIOS.

PREBIOS and PSTBIOS are automatically executed whenever a call to BIOS is made via the BIOS vector in RAM.

(1) PREBIOS
PREBIOS sets on three flags indicating that BIOS processing is in execution, that alarm processing is disabled, and that power off processing is disabled.

(2) PSTBIOS
PSTBIOS resets the flags that are set by PREBIOS and performs any alarm or power off processing that is held pending during BIOS processing.

See Sections 2.5, 2.9, and 4.7 for detailed information.

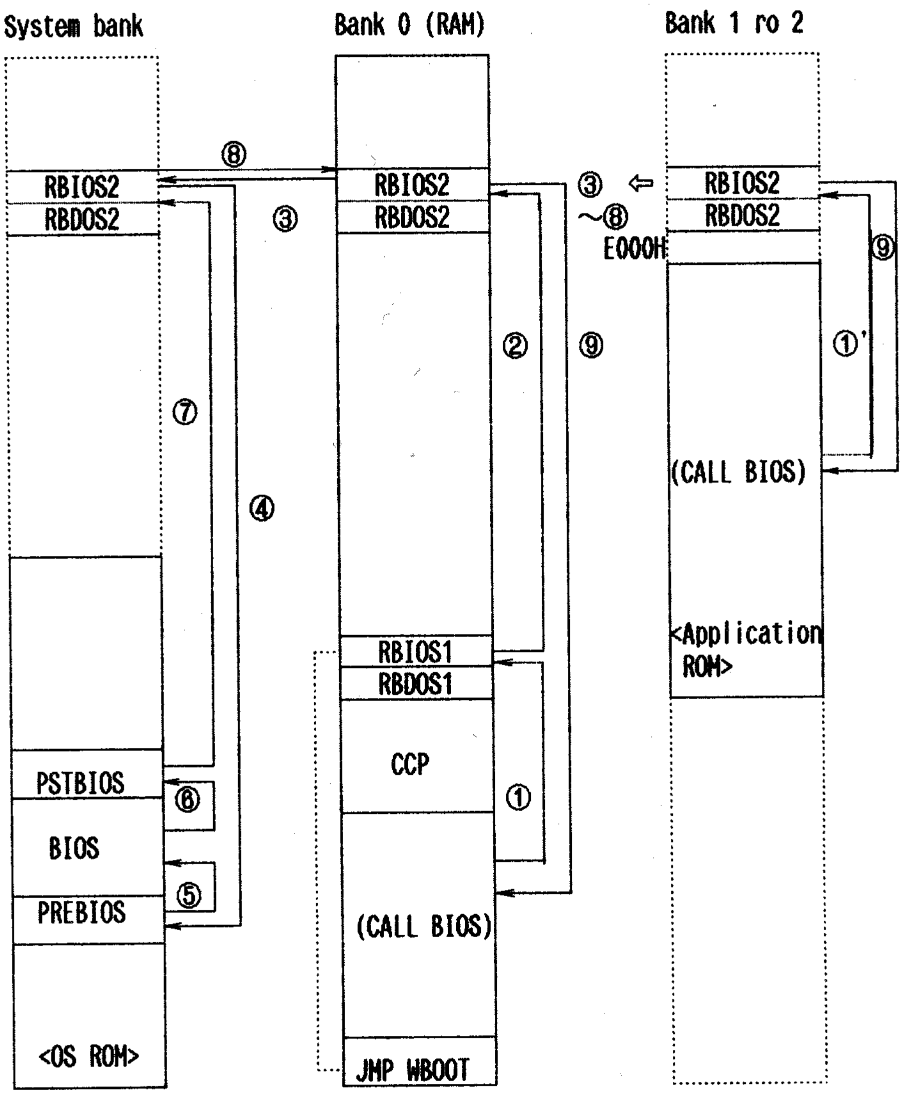


Fig. 3.3.1 BIOS Call Operation Flow

### 3.3.3 BIOS Hook

The PINE extended BIOS contains a hook to BIOS in addition to user BIOS. The BIOS hook permits the user to update or modify existing BIOS processing routines. This subsection explains how to use the BIOS hook.

#### 3.3.3.1 Relationship of the BIOS hook to BIOS

The BIOS hook is referenced immediately before control is transferred from OS ROM on the system bank to a BIOS function. Figure 3.3.2 shows the relationship of the BIOS hoot to BIOS in a flowchart form. Note that only steps 3 through 8 in Figure 3.3.2 are taken when BIOS is called from BDOS.

a
.2

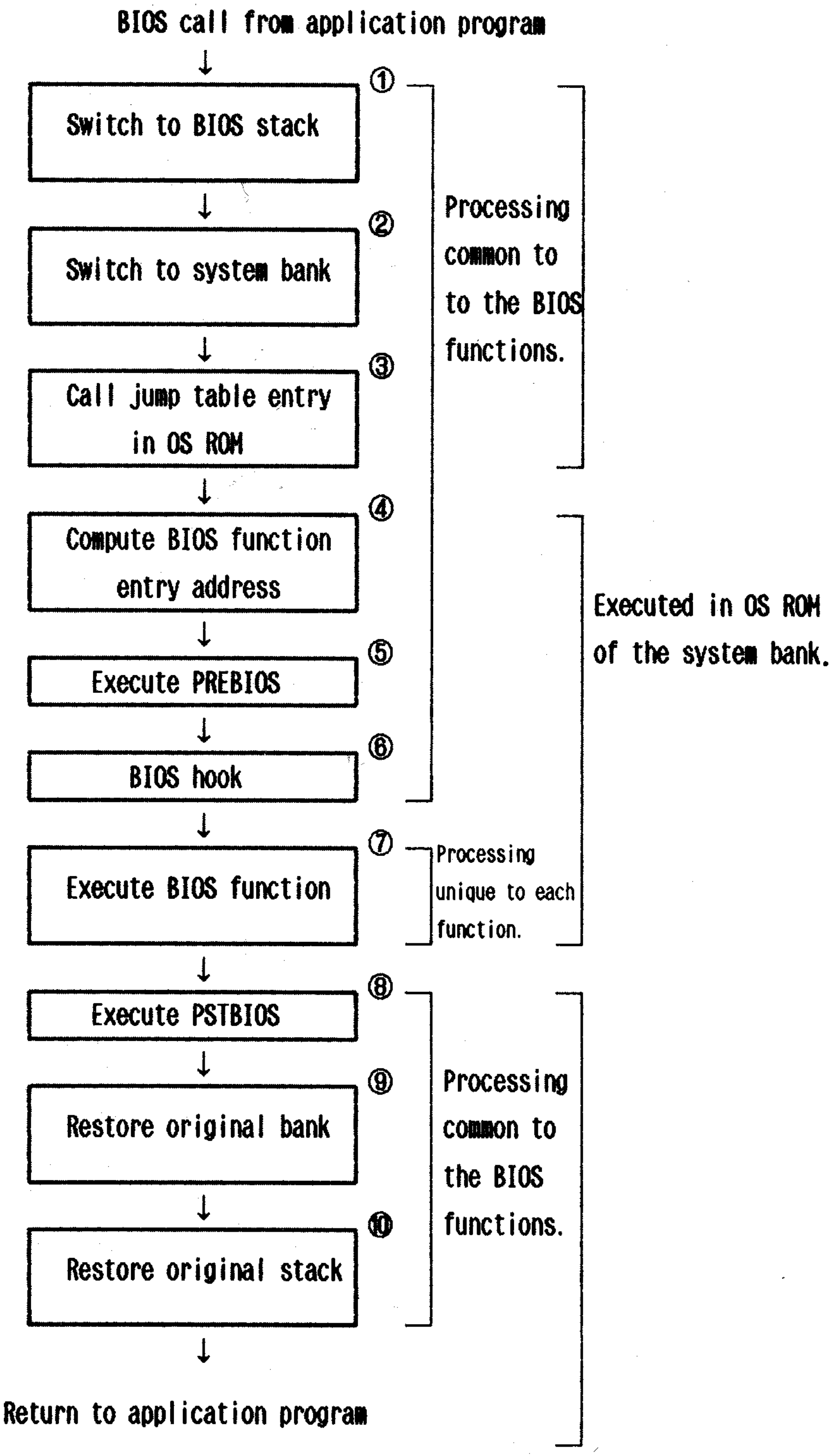

BIOS call from application program
Switch to BIOS stack
①
Switch to system bank
②
Call jump table entry in OS ROM
③
Compute BIOS function entry address
④
Execute PREBIOS
⑤
BIOS hook
⑥
Execute BIOS function
⑦
Execute PSTBIOS
⑧
Restore original bank
⑨
Restore original stack
⑩
Return to application program
Processing common to to the BIOS functions.
Executed in OS ROM of the system bank.
Processing unique to each function.
Processing common to the BIOS functions.

Fig. 3.3.2 BIOS Processing Flow

| Step | Action | Description |
|---|---|---|
| 1 | Switch to BIOS stack | - Save the the current stack pointer to USRSBI.<br>- Switch the stack pointer to BIOS stack. |
| 2 | Switch to system bank | - Check the BIOS function number.<br>- Translate the DMA address to the system DMA address.<br>- Check the current I/O byte and place its contents in RIOBYTE.<br>- Switch the active bank to the system bank after saving original bank information. |
| 3 | Call jump table entry in OS ROM | - Call the corresponding jump table entry in OS ROM determined by the BIOS function number. |
| 4 | Compute BIOS function entry address | - Obtain the address at which the BIOS function is to be started based on the address of the called jump table entry. |
| 5 | Execute PREBIOS | - Carry out the PREBIOS processing described in 3.3.2.1. |
| 6 | BIOS hook | - Call the BIOS hook entry (ØFFE7H).<br>- The register contents remain the same as when the BIOS function was called. |
| 7 | Execute BIOS function | - Call the corresponding BIOS function processing routine based on the BIOS function entry address obtained in step 4. |
| 8 | Execute PSTBIOS | - Carry out the PSTBIOS processing described in 3.3.2.1. |
| 9 | Restore original bank | - Restore the bank information saved in step 2 and switch the bank to the original bank.<br>- Copy the data at the DMA address when the called function is read. |
| 1Ø | Restore original stack | - Restore the stack pointer saved in step 1. |

The following system areas are shared by the BIOS functions:

RIOBYTE (ØF529H) 1 byte
- I/O byte save area.
- The I/O byte format is presented in Section 3.9, "I/O Byte."

OLDBNK (ØF52CH) 1 byte
- Bank information save area.
    = ØFFH: System bank
    = ØØH: Bank Ø
    = Ø1H: Bank 1
    = Ø2H: Bank 2

USRSBI (ØF535H) 2 bytes
- Area for saving the user stack for BIOS.

BIOSFN (ØF537H) 1 byte
- Area for storing a BIOS function number
    = ØØH: BOOT
    = Ø3H: WBOOT
        .
        .
        .
    = 8AH: CONTINUE

SAVEIX (ØF54ØH) 2 bytes
- Area for storing the contents of the IX register pair.

SAVEIY (ØF542H) 2 bytes
- Area for storing the contents of the IY register pair.

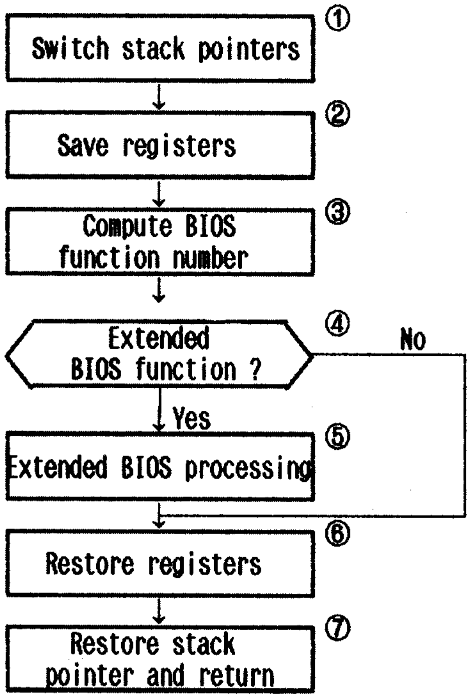
### 3.3.3.2 Using the BIOS hook

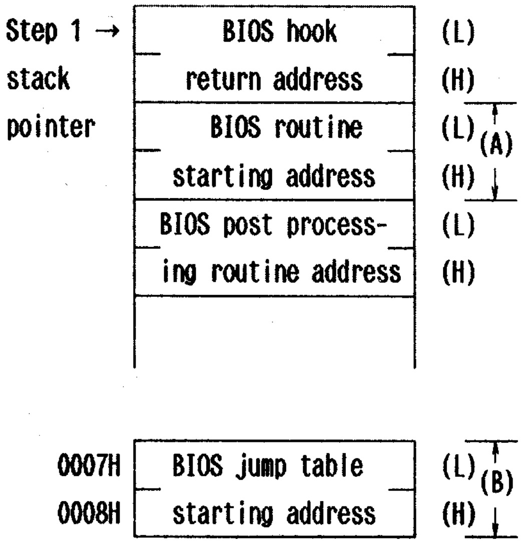
The procedure given below shows how to call BIOS through the BIOS hook.

(1) Hook processing routine logic
BIOS functions are always called through the BIOS hook.
The hook processing routine, therefore, must check for extended BIOS functions.

Sample hook processing routine:

Sample hook processing routine



| Step | Action | Description |
|---|---|---|
| 1 | Switch stack pointers | - The stack pointer currently points to the system BIOS stack area.<br>If the routine is to use a large stack area, it must reserve its own stack area. |
| 2 | Save registers | - Registers are currently loaded with parameters. The registers that are to be used by the routine must be saved. |
| 3 | Compute BIOS function number | - Compute the BIOS function number of the called routine.<br>- How to compute:<br>Find the BIOS function number from the starting addresses of the called BIOS routine and the BIOS jump table.<br><br> Step 1 → stack pointer<br>BIOS hook return address<br>(L)<br>(H)<br>BIOS routine starting address<br>(L)<br>(H)<br>(A)<br>BIOS post processing routine address<br>(L)<br>(H)<br>0007H<br>0008H<br>BIOS jump table starting address<br>(L)<br>(H)<br>(B)<br>Subtracting the value of (B) from (A) results in the offset of a BIOS function (ØØH, Ø3H, ...) with respect to the BOOT entry. |

| Step | Action | Description |
|---|---|---|
| 4 | Check for BIOS function | Compare the specified function number with the function number computed in step 3 to determine whether the given BIOS function is an extended one. The specified BIOS function is found to be a standard BIOS function if the following condition is met:<br>(A) - (B) = 3n<br>where n is the specified function number.<br>n = ØØH ... BOOT<br>n = Ø1H ... WBOOT<br>n = Ø2H ... CONST<br>n = 2DH ... CONTINUE |
| 5 | Extended BIOS processing | - Perform the extended BIOS function.<br>- The user-supplied routine must be placed here. |
| 6 | Restore registers | - Restore the registers saved in step 2. |
| 7 | Restore stack pointer and return | - Restore the stack pointer saved in step 1.<br>- Return. On return, control is passed to the main OS BIOS section.<br>- When you don't want to utilize the BIOS function in the OS, pop the stack level two levels (4 bytes) and return. You will find that item 7 in Fig. 3.3.2 is skipped, and operation goes directly to item 8. |

### 3.3.3.2 Rewriting the BIOS hook

The user can rewrite the BIOS hook by modifying its jump address.

```
FFE8H  |_  BIOS hook     _|  (L)
FFE9H  |   jump address   |  (H)
```

The BIOS hook jump address is initialized to EF1FH. Address EF1FH contains the RET instruction.

Rewrite addresses FFE8H and FFE9H with the starting address of the new hook processing routine. Subsequently, any BIOS calls will be routed through the new BIOS hook processing routine.

### 3.3.3.3 Programming notes

(1) Since the system bank is selected when control is transferred to the BIOS hook, the hook processing routine must be placed at location 8ØØØH or higher. It is desirable that a separate user BIOS area be reserved and the hook processing routine be implemented in that area.
(2) The BIOS hook is also given control by BIOS calls that are invoked by BDOS.
(3) The hook processing routine cannot call BIOS or BDOS functions. To use a BIOS function, directly call the BIOS function on OS ROM. No BDOS call can be made from the hook processing routine.
(4) When placing return information in IX and IY, save the contents of the IX and IY registers into SAVEIX (ØF54ØH) and SAVEIY (ØF542H), respectively.

```
                        ;       ******************************************
                        ;               BIOS HOOK SAMPLE PROGRAM
                        ;       ******************************************
                        ;
                        ;       NOTE :
                        ;
                        ;       <> assemble condition <>
                        ;
                        ;               .Z80
                        ;
                        ;
                        ;       <> loading address <>
                        ;
                        ;               .PHASE  100H
                        ;
                        ;       <> constant values <>
                        ;
FFE8                    BIOSHK          EQU     0FFE8H  ; BIOS hook address
                        ;
CB00                    LOADADDR        EQU     0CB00H  ; Extend BIOS load address
                        ;
0000                    WBOOT           EQU     00000H  ; Warm boot address
                        ;
                        ;       ******************************************
                        ;               BIOS HOOK DATA WRITE
                        ;       ******************************************
0100                    START:
0100    31 016D                 LD      SP,MAINSP       ; Set stack pointer.
                        ;
0103    CD 0120                 CALL    UBSZCHECK       ; Check User-BIOS size.
0106    DA 0000                 JP      C,WBOOT         ; Size error, then WBOOT.
                        ;
0109    21 0126                 LD      HL,LOADDATA     ; Extend BIOS routine load.
010C    11 CB00                 LD      DE,LOADADDR     ;
010F    01 0027                 LD      BC,LOADSIZE     ;
0112    ED B0                   LDIR                    ;
                        ;
0114    21 FFE8                 LD      HL,BIOSHK       ; Change BIOS hook data.
0117    11 CB00                 LD      DE,LOADADDR     ;
011A    73                      LD      (HL),E          ; Set low address.
011B    23                      INC     HL              ;
011C    72                      LD      (HL),D          ; Set high address.
                        ;
011D    C3 0000                 JP      WBOOT           ;
                        ;
                        ;       ******************************************
                        ;               USER-BIOS SIZE CHECK
                        ;       ******************************************
                        ;
                        ;       NOTE :
                        ;
                        ;       <> entry parameter <>
                        ;               NON
                        ;       <> return parameter <>
                        ;               CY : return information
                        ;                     = 0 : size O.K.
                        ;                     = 1 : size N.G.
                        ;       <> preserved registers <>
                        ;               NON
                        ;
                        ;       <> constant values <>
EF2D                    USERBIOS        EQU     0EF2DH
0001                    UBSIZE          EQU     001H
                        ;
                        ;
0120                    UBSZCHECK:
                        ;
0120    3A EF2D                 LD      A,(USERBIOS)    ; USER-BIOS size --> A
0123    FE 01                   CP      UBSIZE          ; Check USER-BIOS size.
0125    C9                      RET
                        ;
                        ;
                        ;       ******************************************
                        ;               EXTEND BIOS ROUTINE
                        ;       ******************************************
                        ;
                        ;       NOTE : This routine must be loaded to 0CB00H
                        ;
                        ;       <> entry parameter <>
                        ;               Depend on each BIOS parameters
                        ;       <> return parameter <>
                        ;               NON
                        ;       <> preserved registers <>
                        ;               ALL
                        ;
                        ;       <> constant values <>
CC00                    EXBIOSSP        EQU     0CC00H          ; Extend BIOS stack area (20H)
CBE0                    SAVESP          EQU     EXBIOSSP-20H    ; BIOS stack save area (02H)
                        ;
0003                    CONINF          EQU     03H             ; CONIN function number
0009                    TARGETBIOS      EQU     CONINF*3        ; Target BIOS function number
                        ;
0007                    BIOSJPTB        EQU     00007H          ; BIOS jump table address
                        ;
CB1F                    EXBIOSE         EQU     EXBIOSR-EXBIOS+LOADADDR
                                                                ; EXBIOSR addr in USER-BIOS area
                        ;
                        ;
0126                    LOADDATA:
0126                    EXBIOS:
                        ;
0126    ED 73 CBE0              LD      (SAVESP),SP     ; Save BIOS stack pointer.
012A    31 CC00                 LD      SP,EXBIOSSP     ; Set new stack pointer.
012D    E5                      PUSH    HL              ; Save registers to new stack.
012E    D5                      PUSH    DE              ;
012F    F5                      PUSH    AF              ;
                        ;
0130    2A CBE0                 LD      HL,(SAVESP)     ; Get default BIOS JUMP address.
```

```
0133    23                      INC     HL              ;
0134    23                      INC     HL              ;
0135    5E                      LD      E,(HL)          ;
0136    23                      INC     HL              ;
0137    56                      LD      D,(HL)          ;
                        ;
0138    2A 0007                 LD      HL,(BIOSJPTB)   ; Get BIOS jump table top addr.
013B    EB                      EX      DE,HL           ;
013C    B7                      OR      A               ; Carry clear.
013D    ED 52                   SBC     HL,DE           ; Calculate offset value.
013F    7D                      LD      A,L
                        ;
0140    FE 09                   CP      TARGETBIOS      ; Target BIOS call ?
0142    C2 CB1F                 JP      NZ,EXBIOSE      ; No.
                        ;
                        ;
                        ;       You can insert your own extend-BIOS routine
                        ;       in this part.
                        ;
                        ;
0145                    EXBIOSR:
0145    F1                      POP     AF              ; Register restore.
0146    D1                      POP     DE              ;
0147    E1                      POP     HL              ;
0148    ED 7B CBE0              LD      SP,(SAVESP)     ; Recover stack pointer.
014C    C9                      RET                     ; Return to OS-BIOS process.
                        ;
0027                    LOADSIZE        EQU     $-LOADDATA      ; Extend-BIOS loading size
                        ;
014D                                    DS      20H             ; Stack area for main routone
016D                    MAINSP          EQU     $
                        ;
                                END
```